

Distributed Shared Agent Representations

Frances Brazier¹, Maarten van Steen², and Niek Wijngaards¹

¹ Vrije Universiteit Amsterdam, Faculty of Sciences,
Department of Artificial Intelligence
Intelligent Interactive Distributed Systems Group
de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{frances, niek}@cs.vu.nl
<http://www.iids.org>

² Vrije Universiteit Amsterdam, Faculty of Sciences,
Department of Computer Science, Computer Systems Group
de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
steen@cs.vu.nl
<http://www.cs.vu.nl/vakgroepen/cs>

Abstract. The external representation of an agent is (part of) the embodiment of an agent: other agents may observe this information. The public representation of an agent usually contains at least the identity of an agent, and may include profiles of the agent, profiles of the user of an agent, an avatar, etc. In large-scale agent systems, scalability is an important issue. Replication is a scaling technique for distributing information over a number of locations. Replication of the external representation of an agent results in distributed shared agent representations. This paper addresses a number of issues involved in the realisation of such distributed shared agent representations, and briefly discusses middleware that is being devised to support such developments.

1 Introduction

Different definitions of agents [1][12][20][26][32] use concepts such as autonomy, pro-activity, reactivity, social abilities, and intentional models [2][5]. Very few (if any), however, refer to an agent's external representation: the part of an agent that can be observed.

This representation is the visible part of an agent: (part of) its embodiment. This external representation may be limited to an agent's identity (e.g. name of owner or an IP address of the host on which it resides). It may also contain extensive public information about an agent (e.g. its profile), or it may even be a graphical figure (e.g. an avatar) representing an agent. An agent's external representation is not necessarily a representation of the agent's internal processes. The environment of an agent includes objects, and representations of itself and other agents. It's obvious that an agent's environment influences the thoughts and actions of an agent [9][14], but it may also influence its own representation.

The distinction between an agent's external representation and an agent's internal processes and knowledge makes it possible to consider new ways to im-

plement large scale systems. Section 2 discusses the problem of scalability. Section 3 addresses the option of agent replication in this context. Section 4 presents a short overview of middleware that is currently being designed to support such replication. Section 5 summarizes the new areas of research involved.

2 The Problem of Scalability

In the multi-agent system community, large multi-agent systems are considered to consist of hundreds of agents, not thousands nor millions. As an example, consider the claim that Auctionbot is scalable, which is supported by an experiment with only 90 agents [33]. In the near future, however, we expect that multi-agent systems will need to be able to scale (in terms of the number of agents and available resources) to much larger populations. This almost immediately without noticeable loss of performance, or considerable increase in administrative complexity [18].

The current support for scalability in multi-agent frameworks is discussed in Section 2.1. A number of scaling techniques are described in Section 2.2.

2.1 Scalability in Multi-Agent Frameworks

The term *scalability* is not always used to refer to architecture, services and performance of systems. In some cases it is used to refer to scalable functionality. For example, the SAIRE approach [21], claims to be scalable because it supports heterogeneous agents. Shopbot [6] claims to be scalable because its agents can adapt to understand new websites. In both cases, the term *extensible functionality* would seem to be more appropriate.

Researchers and developers of multi-agent frameworks are beginning to realize that scalability in the sense of architecture, services and performance is an issue. Most multi-agent frameworks (e.g. DECAF [13], InfoSleuth [19], April [17], AgentTcl [10], JAFMAS [4], Plangent [22], DESIRE [3]) do not seem to address the problem of scalability at all.

Other multi-agent frameworks rely on another framework to solve the problem of scalability. For example, scalability in the CoABS (DARPA Control of Agent Based Systems) approach [29] is based on adequate support from computational grids in providing a plug-in backplane for agents [8].

There are, however, frameworks that clearly address one or more aspects of scalability. In ZEUS [31] scalability is defined to be the growth rate of the maximum communication load (as a function of the number of agents). Their conclusions are that the maximum communication load grows at worst linearly with the number of agents. This addresses a loss of performance problem, and is a step towards developing scalable multi-agent frameworks. In OAA (Open Agent Architecture) [16] matchmaking agents are described which can handle larger number of agents. The RETSINA MAS infrastructure [28] is designed to support multi-agent systems that run on a number of LANs and to avoid single-point of failures (e.g., in agent name services). The DARX framework [15] aims for

fault-tolerance in distributed multi-agent systems by incorporating replication of (tasks of) agents.

Turner and Jennings [30] propose to (automatically) change the organization of agents in the multi-agent system to handle an increase in the population of a multi-agent system. For example, more middle agents or matchmakers are introduced to reduce overhead. Their approach is a possible step towards addressing administrative problems related to scalability.

None of the aforementioned approaches addresses minimizing the loss of performance as well as minimizing administrative overhead.

Research on specific services in multi-agent systems such as directory services also address scalability. The approach taken by Shehory [25] is an example in which agents locate agents based on each agent's own caching lists of agents they know. The theoretical analysis is based on a population of size 10,000; no experiments have yet been conducted.

Although agents have an identity in all of these multi-agent frameworks, none have explicitly distinguished explicit agent representations. Some multi-agent frameworks offer matchmaking services - in which information about agents is made public to some extent. None have considered using agent-based replication: replication of agents' external representation.

2.2 Scaling Techniques

Three scaling techniques can be distinguished to minimise loss of performance: (1) hiding communication latencies, (2) distribution, and (3) replication.

Hiding communication latencies is applicable in the case of geographical scalability, that is, when an agent system needs to span a wide-area network. To avoid waiting for responses to requests that have been issued to remote agents or services the requesting agent is programmed to do other useful work. This approach does require that an agent can be interrupted when the expected response (if any) is to be delivered.

Distribution generally involves partitioning a (large) set of data into parts that can be handled by separate servers. A well-known example of distribution is the natural partitioning of the set of Web pages across the approximately 25 million Web servers that are currently connected through the Internet. Other examples of distribution include the vertical or horizontal partitioning of tables in distributed databases [23].

When considering large-scale networks like the Internet it becomes crucial to combine distribution with latency hiding. Unfortunately, this is not always possible, for example when an agent simply needs an immediate response.

A third, and widely applied technique is to place multiple copies of data sets across a network, also referred to as *replication*. The underlying idea is that by placing data close to where they are used, communication latency is no longer an issue, so that agent-perceived performance is high. Having multiple copies means that such performance is good for all agents, no matter where they are located.

Keeping replicas consistent introduces a consistency problem that can be solved only by means of global synchronization. However, global synchronization in a multi-agent system is not a realistic option. Scalable multi-agent systems will need to support configurable and perhaps even adaptive replication strategies. No single strategy will show to be optimal under all conditions. Even for relatively simple systems such as the Web, differentiating strategies can make a lot of difference [24].

3 Replicating Agents

As stated above: one approach to handling scalability in a multi-agent system is to replicate agents. The distinction between an agent's external representation and an agent's internal representation makes replication possible. Section 3.1 discusses what is to be replicated of an agent, Section 3.2 discusses issues that play a role in distributed shared agent representations.

3.1 Replicating Agents or Public Representations

An agent can be seen as a (multi-threaded) process with internal knowledge, and an external representation. Replication of an entire agent is not an obvious option for the realisation of scalable systems (running processes in parallel on different machines will seldom be synchronous, e.g., see [11] for an approach on replicating entire agents). In some cases cloning an agent may be a viable alternative to replication, but this is clearly application dependent and outside the scope of this paper.

Replicating an agent on a number of hosts makes it possible to decrease the load on each of the individual hosts. Consider, for example, an auction room. Having an auction hall replicated on different hosts, makes it possible to decrease the load on each individual machine. Instead of having any number of agents having to reside on the same machine in order to participate in a given auction, each agent's internal state and processes need only to reside on one of the hosts on which the auction room is replicated. Each agent's external representation is, however, replicated on each of the machines individually: this representation becomes a *distributed shared agent representation*. The internal process and data (or knowledge) of each agent resides on one machine, their public information is replicated. The agents do not need to know the location of the other agents: their presence is obvious, as is the information which needs to be shared.

Another example is that of information acquisition. An agent may wish to be informed of interesting publications by a number of bookstores at a time: as if it were actually in each of these bookstores at the same time. One way to accomplish this is to replicate the agent's external representation on each of the bookshops sites. This representation would include a profile of an agent's interests, book collection, and any other relevant information. If an agent's state changes, e.g. new books have been acquired, the agent's profile changes thus changing the profile in the agent's external representation. It is as if the agent was actually residing on each of the bookstores sites.

3.2 Issues in Distributed Shared Agent Representations

Replication of public representations of agents, i.e. distributed shared agent representations, raises a number of issues regarding agents, and supportive middleware. These issues are related to policies (and/or strategies) for replication, accessibility, authority, and awareness.

- A number of issues concern *replication strategies*. The first issue is whether the middleware enforces a specific replication strategy, or whether each agent or system is able to specify its own replication strategy.
- A number of issues concern *accessibility policies*. Which agent may access which part of a public representation?
- A number of issues concern *authorisation policies*. Which agents are allowed to change part of the public representation of an agent? Only the agent itself? Other (authorized) agents? All agents? A human agent?
- A number of issues concern *awareness policies*. Does an agent notice modifications to its public representation? Does the agent know that replicas exist? Their location? Which agent observes its replica?

By combining choices, more and more complex situations may arise. Consider for example the combination of:

- public representation is replicated as soon as another agent wishes to observe it (i.e., many replicas)
- all agents may change a specific part of the public information of an agent
- an agent is aware of all changes to its public information.

The middleware needed to support this combination of policies, involves more expensive mechanisms (in terms of communication between entities in the multi-agent system) than a situation in which only the agent itself may modify its public representation. The question may even arise whether the more complex case scales well.

4 AgentScape: a Scalable Agent Framework

AgentScape is a currently being designed to support the design and development of worldwide distributed, scalable, secure, and extensible agent systems. It aims to provide support in two ways. First, support is provided on the level of a basic agent operating system. Second, support is provided by services, such as location and directory services, automated creation of agents, and management of agents, objects, locations and groups. AgentScape provides basic building blocks needed to build such systems.

AgentScape is a basic agent middleware system, intended to be usable for a wide range of multi-agent applications. As middleware, it offers primitives on the level of agents, shielding application developers from details at lower levels. In a sense, AgentScape is similar to UNIX. Within UNIX, everything is a file,

on which operations are defined. Within AgentScape, two main concepts are distinguished: agents and objects. An agent is an active process, while an object is passive. Operations are defined on agents, akin to file operations in UNIX: move (`mv`), change owner (`chown`), change group (`chgrp`), change security modus (`chmod`), create, remove (`rm`), etc. Similar operations are defined on objects. Unique to AgentScape is the use of objects that are physically distributed across multiple machines, and that encapsulate their own distribution strategy. These objects are adopted from the Globe wide-area distributed system [27].

An important issue for AgentScape is that its model of agents and objects enable scalable solutions. In our approach, agents are expected to be mobile and can be implemented in different ways. This approach allows for implementing applications that require a high degree of interoperability across heterogeneous platforms. For a similar reason, our objects have self-managing capabilities. In contrast, most distributed-object models are based on remote objects in which the object state is not distributed, and is managed by the server the object is located [7]. Clients are only provided transparent access to an object through a proxy.

An agent in AgentScape consists of an external, visible part and an internal, invisible part. The external visible part of an agent may be observed by other agents, and contains the public representation of the agent. The internal (invisible) part of an agent includes local information, its process, data and/or knowledge. The visible part of an agent may be replicated, the invisible part of an agent is not replicated. The visible part of an agent may be (nearly) empty. Any amount of information may be included in the public representation of an agent. The public representation of an agent is implemented as a distributed shared GLOBE object [27].

An agent operating system intended to be used in a worldwide setting needs services to enable retrieval of, for example, agents. Specific directory services are being developed, with which agents, distributed (and possibly replicated) objects, and groups of agents or objects can be found. Another service is a multi-agent factory with makes automated agent creation and modification possible. Finally, management services are being developed to reactively and pro-actively control agents, objects, locations, and groups in AgentScape. The challenge for these services is that they too need to be scalable across a worldwide network and they need to be able to support vast numbers of agents and objects.

5 Discussion

Distinguishing an external representation of an agent from its internal processes and knowledge, makes it possible to consider replication as an option with which large scale multi-agent systems can be devised. The challenge is to design an environment in which replication of an agent's public information is possible and effective. Which replication strategies are most useful and applicable, which accessibility, authorisation and awareness policies are needed, are clearly, as yet, unanswered. Scalable services to support such environments, is another chal-

lenge. These services address the notion of finding "agents" without necessarily having to contact a home address (not all agents necessarily have a home address), and finding objects, building new agents and adapting existing agents. Further research is clearly required!

6 Acknowledgements

This work was supported in part by NLnet Foundation. The authors wish to thank Guido van 't Noordende and Andy Tanenbaum for discussions on agents and agent platforms.

References

1. J.M. Bradshaw, (ed). "Software Agents." AAAI Press / MIT Press, 1997.
2. F.M.T. Brazier, B. Dunin-Keplicz, J. Treur and L.C. Verbrugge. "Modelling Internal Dynamic Behaviour of BDI agents." In: J-J. Meyer and P. Schobbes, (eds.), *Formal Models of Agents, Selected papers from final ModelAge Workshop*, Springer Verlag, Lecture Notes in AI, Vol. 1760, pp. 36-56, 1996.
3. F.M.T. Brazier, B. Dunin-Keplicz, N. Jennings, and J. Treur. "DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework.." *International Journal of Cooperative Information Systems, special issue on Formal Methods in Cooperative Information Systems*, 6:67-94, 1997.
4. D. Chauhan. "Developing coherent multiagent systems using jafmas." In *Proc. International Conference on Multi Agent Systems, ICMAS98*, Cite des Sciences - La Villette, Paris, France, July 1998.
5. D.C. Dennett. "The Intentional Stance." Cambridge: MIT Press (1987).
6. R. B. Doorenbos, O. Etzioni, and D. S. Weld. "A Scalable Comparison-Shopping Agent for the World-Wide Web." In W. L. Johnson and B. Hayes-Roth, (eds.), *Proc. Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pp. 39-48, Marina del Rey, CA, USA, 1997. ACM Press.
7. W. Emmerich. *Engineering Distributed Objects*. John Wiley, New York, 2000.
8. I. Foster and C. Kesselman, (eds.). *Computational Grids: The Future of High Performance Distributed Computing*. Morgan Kaufman, San Mateo, CA., 1998.
9. J.S. Gero. "Conceptual Designing as a Sequence of Situated Acts." In: I. Smith, (ed.), *Artificial Intelligence in Structural Engineering*. Berlin: Springer, pp. 165-177, 1998.
10. R. Gray, D. Kotz, G. Cybenko, and D. Rus. "Agent Tcl." In W. Cockayne and M. Zyda, (eds.), *Proc. Mobile Agents: Explanations and Examples*. Manning Publishing, 1997.
11. Z. Guessoum, M. Quenault, and R. Durand. "An Adaptive Agent Model." In *Proc. AISB-2001*, pp. 100-116, York, 20-24 mar, 2001, ISBN 1 902956 17 0.
12. N.R. Jennings and M.J. Wooldridge, (eds.). "Agent Technology; Foundations, Application, and Markets." Springer Verlag, Berlin (1998).
13. J. G. Keith. "Towards a Distributed, Environment-Centered Agent Framework.
14. M.L. Maher, S. Simoff and A. Cicognani. "Understanding Virtual Design Studios." London: Springer-Verlag, 2000.

15. O. Marin, P. Sens, J-P. Briot, and Z. Guessoum. "Towards Adaptive Fault Tolerance for Distributed Multi-Agent Systems." In: *Proceedings of ERSADS'2001*, Bertinoro, Italy, May 14-18, 2001.
16. D. Martin, A. Cheyer, and D. Moran. "The Open Agent Architecture: a framework for building distributed software systems." *Applied Artificial Intelligence*, 13(1/2):91-128, 1999.
17. F. McCabe and K. Clark. "April: Agent Process Interaction Language." In N. Jennings and M. Wooldridge, (eds.), *Proc. Intelligent Agents*, volume 890 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.
18. B. Neuman. "Scale in Distributed Systems." In T. Casavant and M. Singhal, (eds.), *Readings in Distributed Computing Systems*, pp. 463-489. IEEE Computer Society Press, Los Alamitos, CA., 1994.
19. M. Nodine, B. Perry, and A. Unruh. "Experience with the InfoSleuth agent architecture." In *Proc. Proceedings of the AAAI-98 Workshop on Software Tools for Developing Agents*, 1998.
20. HS. Nwana. "Software agents: an overview." *The Knowledge Engineering Review*, 11(3):205-244, 1996.
21. J. B. Odubiyi, D. J. Kocur, S. M. Weinstein, N. Wakim, S. Srivastava, C. Gokey, and J. Graham. "SAIRE—a scalable agent-based information retrieval engine." In *Proc. Proceedings of the first international conference on Autonomous agents*, pp. 292-299, Marina del Rey, CA USA, Feb. 1997.
22. A. Ohsuga, Y. Nagai, Y. Irie, M. Hattori, and S. Honiden. "Plangent: An Approach to Making Mobile Agents Intelligent." *IEEE Internet Computing*, 1(4), July 1997.
23. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, Upper Saddle River, N.J., 2nd edition, 1999.
24. G. Pierre, I. Kuz, M. van Steen, and A. Tanenbaum. "Differentiated Strategies for Replicating Web Documents." *Comp. Comm.*, 24(2):232-240, Feb. 2001.
25. O. Shehory. "A Scalable Agent Location Mechanism." In *Proc. Lecture Notes in Artificial Intelligence, Intelligent Agents VI*, 1999.
26. Y. Shoham. "Agent-oriented programming." *Artificial Intelligence*, 60:51-92, 1993.
27. M. van Steen, P. Homburg, and A. Tanenbaum. "Globe: A Wide-Area Distributed System." *IEEE Concurrency*, 7(1):70-78, Jan. 1999.
28. K. Sycara, M. Paolucci, M. van Velsen, and J. Giampapa. "The RETSINA MAS Infrastructure." Technical Report CMU-RI-TR-01-05, Robotics Institute Technical Report, Carnegie Mellon, 2001.
29. C. Thompson, T. Bannon, P. Pazandak, and V. Vasudevan. "Agents for the Masses." In *Proc. Agent-Based High Performance Computing - Problem Solving Applications and Practical Deployment at Autonomous Agents 1999*, Seattle, Washington, USA, May 1999.
30. P. J. Turner and N. R. Jennings. "Improving the Scalability of Multi-agent Systems." In *Proc. Proc. 1st International Workshop on Infrastructure for Scalable Multi-Agent Systems*, 2000.
31. P.D. Wilde. "Stability, Fairness and Scalability of Multi-Agent Systems."
32. M. Wooldridge and N. Jennings. "Intelligent agents: theory and practice." *The Knowledge Engineering Review*, 10(2):115-152, 1995.
33. P. R. Wurman, M. P. Wellman, and W. E. Walsh. "The Michigan Internet AuctionBot: A configurable auction server for human and software agents." In K. P. Sycara and M. Wooldridge, (eds.), *Proc. Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pp. 301-308, New York, 9-13, 1998. ACM Press.