Managing Agent Life Cycles in Open Distributed Systems

D.G.A. Mobach, B.J. Overeinder, N.J.E. Wijngaards, and F.M.T. Brazier
IIDS Group, Department of Artificial Intelligence, Faculty of Sciences,
Vrije Universiteit Amsterdam, de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{mobach,bjo,niek,frances}@cs.vu.nl

ABSTRACT

Large scale open, heterogeneous, distributed environments such as the Internet, are the environments in which (intelligent) agents need to be able to function and survive. These environments need to provide distributed support, including management services, for such agent systems. In this paper a local management architecture, implemented in AgentScape, is provided together with a management-oriented life cycle model. A major feature of this model is the central role of one of the states of the life cycle model, namely the "suspended" state: the state in which an agent is manageable. A prototype implementation of the management system based on the life cycle model is described.

Keywords

Agent management, agent life cycle, distributed systems, middleware

1. INTRODUCTION

The Internet can be described as an open distributed system, that provides a large-scale environment for (intelligent) software agents. Agents are autonomous (mobile) processes, capable of communication with other agents, interaction with the world, and adaptation to changes in their environment. Agents are defined by their ability to function autonomously. This autonomy enables agents to coordinate their behavior, e.g., to tune their actions and interactions to those of other agents, increasing the overall problem solving capability of a multi-agent system.

Large-scale multi-agent systems are very diverse, dynamic, and unpredictable. Their requirements with respect to performance, security, and fault-tolerance are not always known in advance. These requirements make multi-agent system management different from regular process management. Traditional process management methods [2] provide little support with respect to the management of autonomous entities that set their own goals. An agent management system needs to be able to manage large scale, heterogeneous, open environments without infringing on individual agents' autonomy. Agents are namely autonomous (mobile) processes capable of communication with other agents, interaction with their environment, and adaptation to these same environments.

Besides autonomy, a management system for open distributed agent systems needs to be designed to manage heterogeneous agents (heterogeneous with respect to architecture, code languages, resource requirements). One necessary condition is that a management system needs to be able to understand the implications of the life cycle model for which an agent has been designed.

Current life cycle models emphasize an agent's active state, but for management purposes the agent's suspended state is of more importance: only a suspended agent can be manipulated and managed (e.g., migrated, stored on disk, deleted, or created).

Agent platforms are systems that support (a number of) multi-agent systems by hosting large numbers of individual agents. In large-scale open distributed systems, the resources within the different administrative domains need to be managed locally, to ensure that resource owners keep control over their resources, and to ensure that the resources are used effectively by the agents.

In this paper a management architecture, including a management-oriented agent life cycle model for AgentScape, is described. Section 2 discusses current approaches to agent platform management systems. Section 3 discusses agent life cycle models, and presents the agent life cycle model chosen for the AgentScape management system. Section 4 describes the architecture of AgentScape's management system and its prototypes. Section 5 discusses some conclusions that can be drawn from the first results, and indicates some future research directions.

2. AGENT MANAGEMENT

The management of agents is recognized to be an important part of an agent platform. This certainly holds for large-scale multi-agent systems deployed on heterogeneous and open systems. In multi-agent system literature, different aspects of management are addressed.

In Grasshopper [3], the concept *management service* is mentioned in the range of functionalities in the Core Agency. Core Agencies represent the minimal functionality required by an agency in order to support the execution of agents. Support for human interaction is provided by the management services, to monitor and control agents and places.

Abeck *et al.* [1] presented a framework for managing agents, which applies traditional systems management concepts for monitoring

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SAC 2003 Melbourne, Florida, USA
©2003 ACM 1-58113-624-2/03/03 ...$5.00.

and controlling agents. Agents are equipped with a Management Information Base (from the OSI management model), and agents are either managed by their home base, or their current base. In his design and implementation prototype, Abeck recognizes the multiplicity of the problems that management is related to (resources, security, debugging).

The FIPA agent management reference model provides a normative framework within which FIPA agents exist and operate [6]. It establishes the logical reference model for the creation, registration, location, communication, migration and retirements of agents. The agent management reference model consists of the following logical components: an agent, a directory facilitator, an agent management system, a message transport service, an agent platform, and software (services). The directory facilitator provides yellow pages services to other agents. The agent management system exerts supervisory control over access to and use of the agent platform. An agent platform provides the physical infrastructure in which agents can be deployed.

The MASIF standard for allowing different multi-agent systems to inter-operate, also addresses the management of agents [11]. In the MASIF standard, the focus of agent management is placed on interoperability issues. Management is interpreted as a method to allow agent systems to control agents of other agent systems. Resource management is not defined in the MASIF specification, only existing CORBA support can be used for resource management.

The design goals of the Mobile Objects and Agents (MOA) [12] project are support for migration, communication, and control of agents, and provide extensive resource control and interoperability. Resource management is deeply ingrained in the design decisions of many MOA layers and components. Agents negotiate for MOA resources about which and how many it can utilize at the target MOA system. This is achieved by calculating local policy from the agent policy and the host policy. The agent local policy is enforced during its lifetime at the visiting MOA system. The resource management system can also protect hosts from overly demanding agents, by putting limitations on resource consumption.

AgentSNMP [15] applies the SNMP network management solution to managing agent platforms. Utilizing the flexibility of industry-standard SNMP techniques, a formal interface is defined for management of a FIPA-compliant agent platform. A specific proxy agent implementation of this interface for the JADE (FIPA-compliant) agent platform was also developed. The result is an efficient, flexible means of managing agent platforms.

Agent-based monitoring and visualization tools such as discussed in [5] and [8] provide human users/system administrators support, but are not used by the systems themselves.

The monitoring and management systems discussed in this section all have (implicitly or explicitly) agent life cycle models which define the states and state transitions that are allowed. In the next section, some of the underlying life cycle models are presented.

3. LIFE CYCLE MODELS

Agent life cycle models are used by management systems to both monitor the state of agents, as well as control agent-state changes. Management systems are not concerned with internal states of an agent, or agent-specific tasks within an multi-agent application. The "observable" states of an agent, from the perspective of the management system, include states such as:

1. *activated*: an agent can actively participate in a multi-agent application,
2. *suspended*: an agent is temporarily not available, but may become activated in the future,
3. *migrating*: an agent is moving to another location.

Section 3.1 briefly describes a number of existing life cycle models. An alternative life cycle model, with a management-oriented perspective on the states of an agent, is presented in Section 3.2.

3.1 Life Cycle Models in Literature

An agent life cycle can be defined as [10]: "A series of stages through which an agent passes during its lifetime." Life cycle models describe both the states of an agent, and the (allowed) transitions between states.

The FIPA Agent Management Specification [6, 13] explicitly defines an agent life cycle with a state diagram. FIPA agents can be in one of the following states: initiated, active, transit, or suspended. After creation, the agent is said to be in the initiated state. After invoking the agent, it is active. The central state of a FIPA agent is the so-called active state. After moving into another state, agents always return to the active state before additional state transitions are possible.

In [9], two life cycle models are discussed: a task based agent life cycle model, which is useful for coordination between agents, and a persistent process based model which is more suitable for management of agents. The latter model consists of four states: start, running, frozen, and death; "frozen" is the state in which an agent can be transported to another host.

The life cycle model in Pathfinder [4] does not consider migration to be part of a mobile agent's life cycle. The model consists of the following states and transitions between them: stopped, running, suspended, aborted, and completed. Agent life cycle models without migration as an explicit state, look similar to Service life cycle models (e.g., [7]).

Some descriptions of agent platform management systems do not explicitly include life cycle models, but do recognize agent states. For example, the AMS in [3] (Agent Management System) recognizes the following states: creation, suspension, resumption, termination, migration and localization. Other descriptions of management systems recognize "actions" on agents, which seem to be derived from a (hidden) life cycle model. For example, MASIF [11] includes the following functions: agent creation, termination, suspension, transfer, and resumption.

3.2 Management-Oriented Life Cycle Model

The aforementioned agent life cycle models had "activated" as their central state, although an agent cannot be managed by the system when it is active. From a management perspective, the suspended state is the central state of an agent. A suspended agent is manageable, that is, can make the necessary transitions to other defined states. In Figure 1, the management-oriented life cycle is shown.

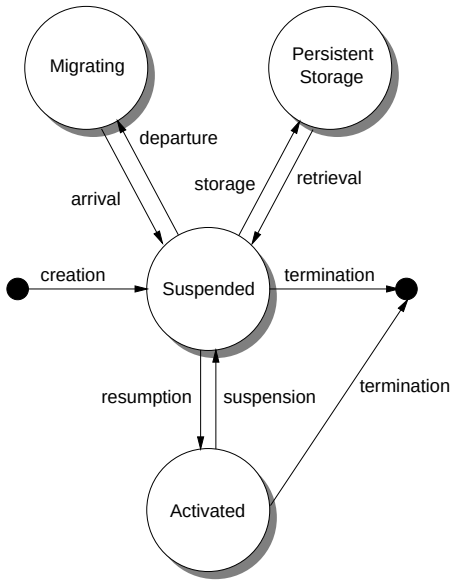

Figure 1: The management-oriented agent life cycle model.

Our management-oriented life cycle model consists of four states, and a number of transitions between these states. The four basic states are:

1. the *activated state*, in which the agent is running and able to perform actions and pursue its goals;
2. the *suspended state*, in which an agent is still within the agent platform;
3. the *migrating state*, during which the agent is traveling between two platform instances;
4. the *persistent storage state*, in which an agent is removed from the agent platform and stored elsewhere.

In this model the initial and final state are not explicitly distinguished. Instead, only the possible transitions *from* the initial state and *to* the final state are distinguished of importance, as the two states themselves are not important from a management point of view.

To allow agents to migrate from one agent platform to another, and not be rejected because of life cycle differences between the management systems at these agent platforms, agreement on (or transformations between) life cycle models is required. The life cycle model presented in this section is a basic model, which provides a minimal life cycle for (mobile) agents. The model may be extended, e.g., by defining additional states and transitions, if so required. Furthermore, the basic model provides a basis for interoperability, as it clearly defines the states and transitions that are supported. The next section illustrates the use of a life cycle model in a management system.

4. AGENTSCAPE'S MANAGEMENT ARCHITECTURE

The agent management system in AgentScape [17] employs the management-oriented life cycle model described above. Section 4.1 briefly describes AgentScape, after which Section 4.2 describes the architecture of the current management system.

4.1 AgentScape

The main objective of the open source AgentScape project [17] is to provide a framework to support the development of large-scale, secure, distributed multi-agent systems. AgentScape is a distributed middleware layer that supports these large-scale agent systems, effectively by providing a distributed virtual machine which hosts agents, objects and services. The rationale behind design decisions in AgentScape are (i) to provide a platform for large-scale agent systems, (ii) support multiple code bases and operating systems, and (iii) interoperability with other agent platforms.

Agents and objects are supported by *agent servers* and *object servers*. Services are made accessible via *service access providers*. *Agents* are the active entities (i.e., processes), *objects* are passive entities, which may be distributed over multiple object servers (currently as Globe distributed shared objects [16]). A *location* consists of a number of agent servers, object servers, and service access providers, running on one or more hosts. Each location has a kernel distributed over hosts in the location, and may interact with other locations, e.g., to enable agent communication, object interaction, agent migration, etc. Figure 2 shows an overview of the AgentScape architecture, in which agent and object servers are shown that host agents and objects, as well as some example services.

AgentScape is designed to be extensible; the kernel provides basic functionality on top of which application or platform-specific functionality can be realized (via specific agent-servers, libraries, and/or services). Also, the kernel itself is modularized, which allows for flexible adaptation of the kernel.

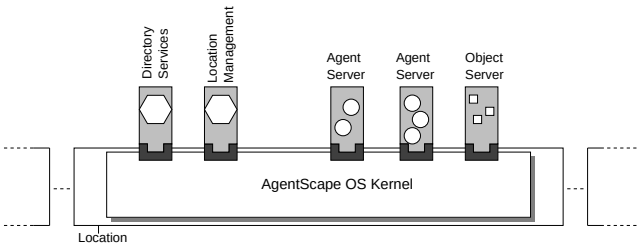

Figure 2: The AgentScape architecture.

4.2 Management Architecture

This section presents a management architecture for a single location within AgentScape. A number of design principles for a location management architecture are formulated. First the management system should not generate disproportionate communication and processing overhead within a location. Second, gathering data must not be disruptive, especially in the case of monitoring agents. Other important issues such as the security and reliability of the management system are recognized, but fall outside the scope of this paper.

Within a location in the AgentScape system, each host that needs to be managed is equipped with a monitoring module in the kernel. The monitoring modules gather data from the kernel and from agent

servers, object servers, and service access providers. The monitoring modules are controlled by a location manager, which runs as a service on the distributed kernel of the location (see Figure 2). Figure 3 shows an example of a location. The location consists of four hosts, which each run different components of the AgentScape system. Each host contains a monitoring module. Host B of the location runs the location manager service.

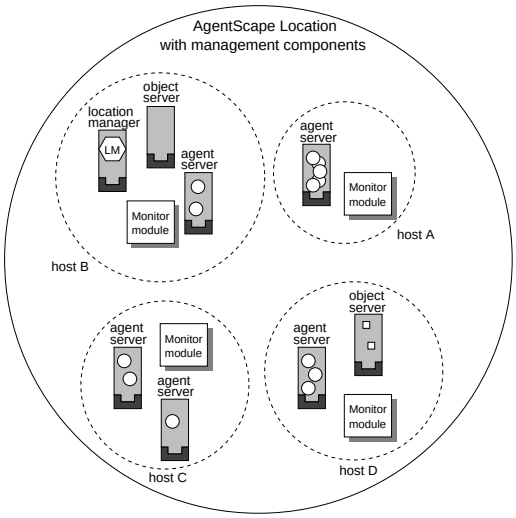

Figure 3: An AgentScape location with monitoring modules distributed over four hosts.

The location manager service issues requests to the monitoring modules to monitor specific information by defining information filters. Filters are used to selectively monitor information, or to monitor for specific events. Monitor operations are delegated to the monitoring modules which locally implement these filters. The amount of data processed by the monitoring modules and the location manager is minimized, as only that information is collected, which is specified by the location manager, and communication between the manager and the monitoring modules is performed only when necessary.

The location manager uses monitoring information to determine which actions need to be taken and by which component (e.g., agent servers, agent containers, kernel modules). In the example shown in Figure 4, interactions are shown between a location manager, a remote monitoring module on another host in the location, and an agent server on that remote host. Data is gathered from the agent server by the monitoring module, and processed and communicated to the location manager. The location manager reacts by issuing an instruction to the agent server.

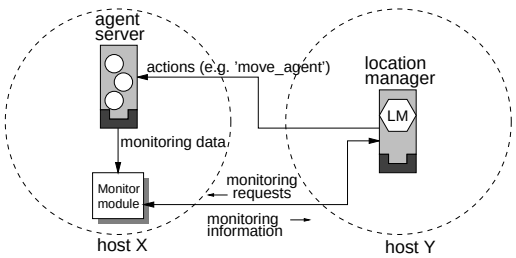

Figure 4: Example monitoring and control operations.

The management system uses the management-oriented life cycle model described in Section 3.2 to monitor an agent's states, enabling management to perform appropriate operations on these agents. For example, to migrate an agent it must first be *suspended*. To be "off-loaded", i.e. placed in *persistent storage*, it also must first be suspended before being stored for future activation. Using the life cycle model as a basis for management operations ensures that all agents that are under management control adhere to the same states and transitions, and can be managed by the location management system.

For example, consider a location manager that needs to monitor communication originating from a specific agent running on an agent server within the location. Depending on the resource management objectives, the location manager decides which information it requires and configures the monitoring filters to only return communication information that is relevant for the current objectives. In another situation, a location manager may choose to use a filter to instruct a monitoring module to report to the manager when the number of communicated messages at that kernel exceeds a specified limit. As long as the number of messages stays below the specified limit, the monitoring module does not need to alert the location manager, and no communication between the monitoring module and the location manager in necessary.

A prototype of the management system has been implemented and incorporated with the AgentScape Operating System. The prototype performs basic agent management tasks, such as load balancing agents over the available agent servers in a location, by migrating them from one server to another. Monitoring modules keep track of the number of agents on each agent server, and notify the location manager when this number deviates from the desired value. The location manager then responds by migrating the surplus agents to another agent server within the location.

The first experiments with the prototype present valuable insights into the flexibility and shortcomings of the proposed management architecture. The prototype allows for experimentation with different communication protocols between the components of the management system, as well as different management strategies, and various approaches for distributing management intelligence across the various components of the management system. The results of these experiments help in finding the optimal balance between the benefits and costs (i.e. overhead) of a management system.

5. DISCUSSION

An open distributed system requires a distributed management system. AgentScape's management system is aimed to be distributed: within a location, management components are distributed across the available system components. In addition, future research will involve resource management between locations in a distributed manner, possibly using a peer-to-peer approach [14].

AgentScape is designed to support and manage heterogeneous agents. To this purpose AgentScape's management system uses a management-oriented agent life cycle model to describe the state of heterogeneous agents. Within this model, the suspended state is viewed as the central state of an agent. Open questions that will be addressed in future research concern interoperability with life cycle models in other multi-agent system frameworks, as well as interoperability implications when multiple extensions of the life cycle model are used concurrently.

AgentScape's management architecture monitors and controls the resources within a location, in addition to the agents and objects it hosts. Our current research focuses on experiments with load-balancing within a location. These experiments provide insights into the process of integrating a management system within a multi-agent system framework. Future research will involve incorporating the location management model into a global AgentScape management model, which will address important management aspects including security, fault tolerance and performance.

Acknowledgments

This research is supported by the NLnet Foundation, http://www.nlnet.nl/. The authors would like to thank Etienne Posthumus, Maarten van Steen, and Andy Tanenbaum for their valuable discussions and comments.

6. REFERENCES

[1] S. Abeck, A. Koppel, and J. Seitz. A management architecture for multi-agent systems. In *3rd IEEE Workshop on Systems Management*, pages 133–138, Newport, 1998.

[2] G. Allen, T. Dramlitsch, I. Foster, N. T. Karonis, M. Ripeanu, E. Seidel, and B. Toonen. Supporting efficient execution in heterogeneous distributed computing environments with Cactus and Globus. In *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing*, pages 52–52, Denver, Colorado, Nov. 2001.

[3] M. Breugst and S. Choy. Management of mobile agent based services. In H. Z. et al, editor, *6th International Conference on Intelligence in Services and Networks*, volume 1597 of *Lecture Notes in Computer Science*, pages 143–154, Berlin, Germany, 1999. Springer-Verlag.

[4] W.-S. E. Chen, C. Lin, and Y.-N. Lien. A mobile agent infrastructure with mobility and management support. In *Proceedings of the 1999 International Workshops on Parallel Processing*, pages 508–515, Wakamatsu, Japan, 1999.

[5] Z. Cui, B. Odgers, and M. Schroeder. An in-service agent monitoring and analysis system. In *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, pages 237–244, Chicago, USA, 1998.

[6] Foundation for Intelligent Physical Agents. FIPA agent management specification, Aug. 2001.

[7] M. Garschhammer, R. Hauck, H.-G. Hegering, B. Kempter, M. Langer, M. Nerb, I. Radisic, H. Roelle, and H. Schmidt. Towards generic service management concepts – A service model based approach. In *Proceedings of the 7th International IFIP/IEEE Symposium on Integrated Management (IM 2001)*, pages 719–732, Seattle, Washington, USA, May 2001.

[8] J. R. Graham, D. McHugh, M. Mersic, F. McGeary, M. V. Windley, D. Cleaver, and K. S. Decker. Tools for developing and monitoring agents in distributed multi-agent systems. In T. Wagner and O. Rana, editors, *International Workshop on Infrastructure for Scalable Multi-Agent Systems*, volume 1887 of *Lecture Notes in Computer Science*, pages 12–27, Berlin, Germany, 2001. Springer-Verlag.

[9] S. Green, L. Hurst, B. Nangle, P. Cunningham, F. Somers, and R. Evans. Software agents: A review. Technical Report TCS-CS-1997-06, Department of Computer Science, Trinity College Dublin, Dublin, Ireland, May 1997.

[10] M.-W. Incorporated. Merriam-webster online. `http://www.m-w.com`, 2001.

[11] D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran, and J. White. MASIF: The OMG mobile agent system interoperability facility. In *Proceedings of Mobile Agents*, pages 50–67, Stuttgart, Germany, Sept. 1998.

[12] D. Milojicic, W. LaForge, and D. Chauhan. Mobile objects and agents (MOA). In *Proceedings of USENIX COOTS'98*, pages 179–194, Santa Fe, NM, Apr. 1998.

[13] P. D. O'Brien and R. Nicol. FIPA - Towards a standard for software agents. *BT Technology Journal*, 16(3):51–59, 1998.

[14] B. J. Overeinder, N. J. Wijngaards, M. van Steen, and F. M. T. Brazier. Multi-agent support for Internet-scale Grid management. In *Proceedings of the AISB'02 Symposium on AI and Grid Computing*, pages 18–22, London, UK, Apr. 2002.

[15] B. D. Remick and R. R. Kessler. Managing agent platforms with agentSNMP. In *Proceedings of the 1st International Workshop on Challenges in Open Agent Systems, at AAMAS'02*, Bologna, Italy, July 2002.

[16] M. van Steen, P. Homburg, and A. Tanenbaum. Globe: A wide-area distributed system. *IEEE Concurrency*, 7(1):70–78, Jan.–Mar. 1999.

[17] N. Wijngaards, B. Overeinder, M. v. Steen, and F. Brazier. Supporting internet-scale multi-agent systems. *Data and Knowledge Engineering*, 41(2-3):229–245, June 2002.