

Supporting Life Cycle Coordination in Open Agent Systems

F.M.T. Brazier, D.G.A. Mobach, B.J. Overeinder, and N.J.E. Wijngaards

IIDS Group, Department of Artificial Intelligence, Faculty of Sciences,

Vrije Universiteit Amsterdam, de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

{frances,mobach,bjo,niek}@cs.vu.nl

Abstract

Coordination mechanisms need the proper support if they are to function within large-scale, heterogeneous multi-agent systems. This paper presents a life cycle model for mobile agents in which a number of different states and transitions are defined. An agent's state can be used to provide coordination mechanisms with useful status information, enabling these mechanisms to deal with realistic situations, in which agents are not always directly available, and the agent population possibly consists of many different types of agents.

1 Introduction

Coordination between agents is an active topic within the multi-agent system research community. Within a multi-agent system, coordination between agents is needed to accomplish collective tasks. Coordination between agents implies the need for the effective usage of communication and allocation of resources. Within open systems, the availability of agents and other (external) resources is essential. The resource aspects of agents typically translate to agent life cycles: agents can be active, suspended, migrating or terminated. This paper presents a life cycle model for life cycle coordination in open multi-agent systems.

2 Coordination

Within the domain of multi-agent systems, coordination is needed to ensure that agents can perform their tasks in a coherent manner. Coordination allows agents to tune their actions and interactions to those of other agents, increasing the overall problem solving capability of a multi-agent system. In the following section, a number of reasons for coordination within a multi-agent system are discussed.

2.1 Coordination techniques

A number of techniques are currently being used for coordination [1, 5, 6], ranging from techniques based on organizational models that impose coordination structures on agents,

to coordination solutions based more on negotiation between agents or coalition formation [8]. Other techniques are based on the well-known contract net protocol [9]. This protocol regulates coordination with contracts. In general, the focus of discussion on these coordination techniques is on the communication issues involved. Resource availability issues related to realistic multi-agent system environments are not often addressed. The following section discusses these issues.

2.2 Coordination in large-scale multi-agent systems

As the focus of the multi-agent system community shifts towards larger and more heterogeneous agent systems, coordination mechanisms are needed that can deal with large agent systems running many different types of mobile agents. In the literature, attempts are made to determine the suitability of current coordination solutions for these new domains. Cabri et al. [3] and Omicini and Zambonelli [7] address the issue of mobility, to determine which coordination mechanisms are suitable for coordinating mobile agents. Durfee [4] discusses the issue of scalability and scalability properties of current coordination solutions.

Within large scale multi-agent systems, a number of issues can be identified that have an impact on the functioning of coordination mechanisms. First, traditional views on coordination often assume that agents are always available, and can take part in the coordination process. This is not the case in realistic, large-scale distributed agent systems. In a large-scale, open agent system, agents migrate from host to host, agents are suspended, agents are terminated suddenly or crash without warning. Coordination mechanisms need to be able to deal with agents in these situations.

Second, coordination mechanisms in multi-agent systems need to be aware that agents use external resources to perform their tasks, such as, for example, databases or directory services. In a realistic multi-agent environment, these services are not always directly available to all agents on demand. Coordination mechanisms need to take into account that resources can be occupied or off-line.

In traditional homogeneous multi-agent systems, this fundamental support is fairly straightforward, because all agents

are known to be based on the same architecture. However, in open agent systems, this assumption does not hold, as agents from different platforms may need to coordinate their actions with each other. A “common ground” between these agents needs to be available to enable agent coordination. In this paper, the life cycle of an agent provides this common ground, and life cycle management is presented as a means to support agents engaging in coordination activities with each other.

3 The Role of the Life Cycle Model

This section describes a basic life cycle model for mobile agents, and the possibilities this model offers for providing support for coordination in multi-agent systems.

3.1 Agent Life Cycle

The life cycle model in Figure 1 provides a model of the most basic states of a mobile agent and the transitions between these basic states. More sophisticated life cycle models can be defined by extending this model.

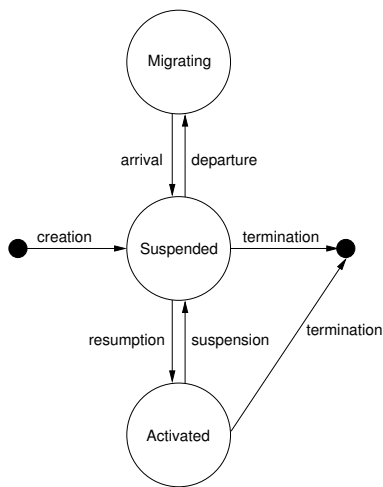


Figure 1: The agent life cycle model.

The model describes three states in which an agent can exist: Activated, Suspended and Migrating. In the active state, an agent is running and able to perform actions, observations, and pursue its goals. In the suspended state, an agent is not. The migrating state is the state in which an agent is traveling between two locations within a multi-agent system.

The model also defines the possible transitions between these states. From the activated state, agents are able to suspend or terminate. From the suspended state, agents are able to terminate, resume, or leave a host. From the migrating state, agents are only able to arrive at a host. Finally, a state transition is identified in the model for creation of the agent.

The model differs from other models with respect to its central state: the suspended state. From an agent’s perspective, the most important state is the active state, in which it can perform its tasks. However, from a management perspective, it can be argued that the suspended state is the most important state of an agent, as in this state, the entire agent, including its internal state, is stored, which allows for a management system to perform actions on the agent without interfering with its tasks.

Also, making the suspended state the central state in the life cycle of an agent ensures that an agent always passes through this state when it changes state: An agent that migrates to another location, first passes through the suspended state before the migrating state is reached. Similarly, an agent that arrives at a location, first passes through the suspended state before it is activated. This ensures that agents are always suspended before they begin their execution at a location, or before they leave a location. This could be beneficial from various perspectives: Security checks can be performed to see if agents are carrying stolen data; Performance checks can be performed to schedule the resumption of an agent; Accounting actions can be performed (such as agent registration and the determination of access privileges for an agent).

3.2 Life Cycle Model Implications for Coordination

Because the life cycle model described above is common to all agents within a multi-agent system, it can be used as a basis for coordination mechanisms to build upon. The life cycle model provides a clear view of the possible states of agents within a system. This information can aid in defining specific situations that coordination mechanisms will have to deal with. For example, coordination mechanisms may need to be aware of the different life cycle states of agents: agents that migrate could lose the ability to perform the tasks they were performing at their original location. Coordination mechanisms should be able to detect migration and adapt the coordination scheme accordingly. Similarly, coordination mechanisms may need to be aware of suspending agents: agents that are suspended for a longer period of time for a specific reason should perhaps be excluded from a coordination scheme.

The agent life cycle information described above needs to be made available by platforms to enable coordination mechanisms to make use of it. A life cycle management module could provide this information, together with additional life cycle management information. A life cycle management module monitors and influences agent life cycles. For example, coordination mechanisms could prevent agents from suspending when they perform an important role within a coordination scheme.

4 AgentScape

The agent life cycle model presented in Section 3.1 and the life cycle management module briefly discussed above have been adopted as part of the AgentScape framework.

The main objective of the AgentScape project [10] is to provide a framework to support the development of large-scale, distributed multi-agent systems. AgentScape is a middleware layer that supports these large-scale agent systems. The rationale behind the design decisions 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.

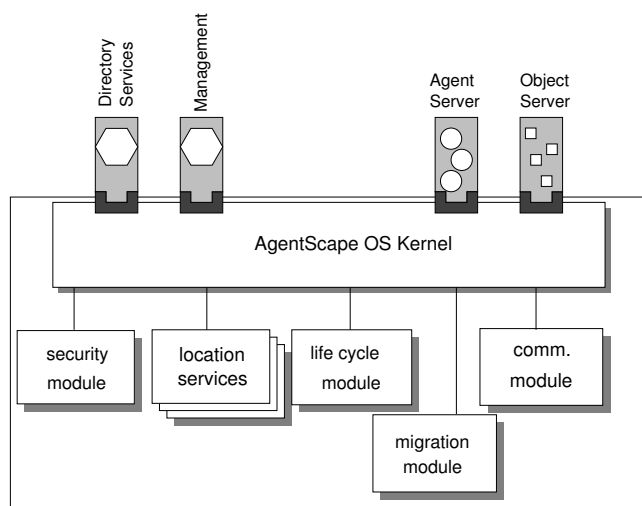


Figure 2: An AgentScape middleware architecture.

Within AgentScape, the life cycle model allows for a uniform approach to creating, starting, killing, suspending and migrating agents. To enable the management of agents, the agent life cycle model discussed in this paper is adopted within the AgentScape OS as shown in Figure 2. The figure also shows that both a life cycle module and a management service are part of the architecture. AgentScape services also (easily) adopt the model for specific purposes. For example, the agent factory [2] uses the model as a basis for uniform agent (re-)generation. The Agent Factory makes use of the life cycle management module to suspend agents, change their configuration, and resume the agents again.

The current AgentScape prototype uses the presented life cycle model for the basic agent creation, deletion, and migration operations. Future prototypes will incorporate a complete management service based on the OSI management functional areas: Fault-tolerance management, configuration management, account management, performance management, and security management.

5 Discussion

As multi-agent systems are becoming larger and more dynamic, coordination techniques will need to be able to handle these new conditions. Especially the availability of resources has to be considered, if coordination techniques are to be successful in these large-scale systems. The life cycle model described in this paper can be used as a starting point for coordination models to define the proper handling of dynamic agent populations and resource usage.

The life cycle model has already proven its versatility in the design and implementation of agent creation, deletion, and migration of agents in the AgentScape prototype. The life cycle model will also be a central notion in the management system that will be part of the AgentScape system.

Acknowledgments

This research is funded by the NLnet Foundation, <http://www.nlnet.nl/>. The authors acknowledge Maarten van Steen, Andrew Tanenbaum, Etienne Posthumus and Guido van 't Noordende for their contributions to the AgentScape framework.

References

- [1] R. A. Bourne, C. B. Excelente-Toledo, and N. R. Jennings. Run-time selection of coordination mechanisms in multi-agent systems. In *Proceedings of the 14th European Conference on Artificial Intelligence*, pages 348–352, Berlin, Germany, 2000.
- [2] F. M. T. Brazier, B. J. Overeinder, M. van Steen, and N. J. E. Wijnngaards. Agent factory: Generative migration of mobile agents in heterogeneous environments. In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC 2002)*, pages 101–106, Madrid, Spain, March 2002.
- [3] G. Cabri, L. Leonardi, and F. Zambonelli. XML datas-paces for mobile agent coordination. In *Proceedings of the 2000 ACM symposium on Applied computing 2000*, pages 181–188. ACM Press, 2000.
- [4] E. H. Durfee. Scaling up agent coordination strategies. *IEEE Computer*, 34(7):39–46, July 2001.
- [5] N. R. Jennings. *Coordination Techniques for Distributed Artificial Intelligence*, pages 187–210. Wiley, 1996.
- [6] H. S. Nwana, L. C. Lee, and N. R. Jennings. Coordination in software agent systems. *The British Telecom Technical Journal*, 14(4):79–88, 1996.

- [7] A. Omicini and F. Zambonelli. Tuple centres for the coordination of Internet agents. In *Proceedings of the 1999 ACM symposium on Applied computing*, pages 183–190. ACM Press, 1999.
- [8] O. Shehory, S. K. Sycara, and S. Jha. Multi-agent coordination through coalition formation. In *Intelligent Agents IV: Agent Theories, Architectures and Languages, Lecture Notes in Artificial Intelligence*, number 1365, pages 143–154. Springer, 1997.
- [9] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computing*, 29(12), December 1980.
- [10] N. J. E. Wijngaards, B. J. Overeinder, M. van Steen, and F. M. T. Brazier. Supporting Internet-scale multi-agent systems. *Data and Knowledge Engineering*, 2002. in press.