Hierarchal Traveling Design Pattern for Mobile Agents in JADE Framework

Mohammed Eshtay
Applied Science University
Faculty of IT, P.O.Box 11931, Amman-Jordan
M_eshtay@asu.edu.jo

ABSTRACT

Mobile agents are program that can migrate between different hosts, begin execution at some point and then continue their execution at another host from the point they stopped. Many design patterns have been proposed for different problems of mobile agents. Itinerary, Star-Shaped, Branching, Master-Slave, MoProxy, Meeting, Facilitator, and Mutual Itinerary Recording are examples of mobile agents design patterns, some of these patterns implemented and tested, some are partially implemented and some still need implementation .

This paper proposes a design pattern called (Hierarchal Traveling design pattern) which is concerned in the way that mobile agents will migrate between different host, this pattern can be classified under the traveling design patterns.

Key Words:
Mobile Software Agents, Mobile Agents Design Patterns, Hierarchal Design Pattern, Itinerary Pattern, Traveling Design Patterns, JADE Framework.

1. Introduction

Software agents are programs assist people and act on their behalf [1]. Mobile agents are software agents with an additional property which is the ability to transport them from one system to another system in the network; the mobile agents are not bound to the system where they begin execution, mobile agents can begin execution in some part of the network and then dispatch and travel to another part and continue execution.

Using mobile agents help to improve the creation of the distributed systems by reducing the network load, overcome network latency, execute asynchronously and they are robust and fault-tolerant.

Distributed Information Retrieval, parallel processing, and monitoring and notification are some of the applications that can get great benefit from using mobile agents.

In this paper the main concern is the design patterns of mobile agents, design patterns can help by capturing solutions to common problems in agent design.

The use of design patterns is an approach to improve the development process of applications and the quality of the final products.

Agent transfer:

The transfer process can be initiated by the agent itself, by another agent residing in the same place, or by an agent or no agent system outside the place [1]
When an agent wants to travel to another place, it must know its destination.

The following figure [figure 1] describes the process of agent transfer.

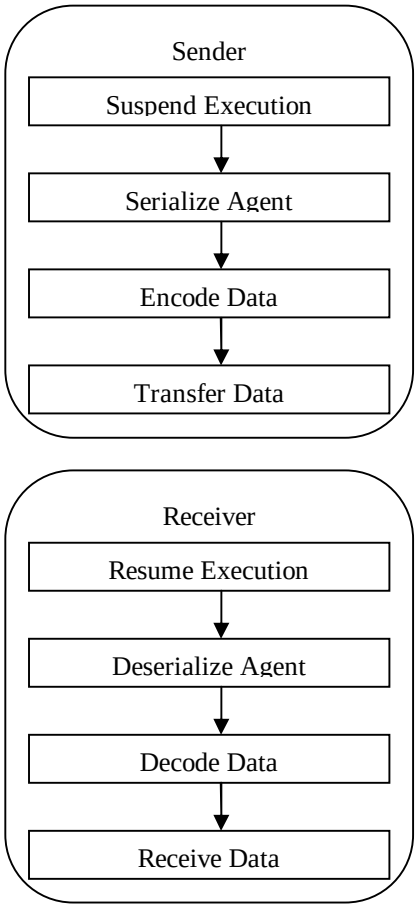

Figure 1 - Agent Transfer Process

2. Hierarchal Model

2.1. Design Patterns

Design patterns can be classified into three classes: traveling patterns, task patterns, and interaction patterns

1. Traveling patterns: traveling patterns deal with the movement of agents between different parts of the network, the Itinerary pattern is an example of traveling patterns, this pattern, proposed in [3], provides a way to execute the migration of an agent, which will be responsible for executing a given task in remote hosts. The agent receives an itinerary on the source agency, indicating the sequence of agencies it should visit. Once in an agency, the agent executes its task locally and then continues on its itinerary. After visiting the last agency, the agent goes back to its source agency [2]. Branching and Forwarding patterns are examples of traveling patterns.

2. Task Patterns: task patterns are concerned in breaking down the tasks and how to distribute these tasks among agents, single task can be performed by one agent or by more than one agent, master-slave pattern is a common example of this type of patterns, On the Master-Slave pattern [4], a master agent delegates a task to be done on a given agency to a slave agent, in order continue executing other tasks that cannot be interrupted. The slave agent visits the indicated agency, where it accomplishes the task and then returns to the source agency with the results. The master agent receives the results from the slave agent. Then, the slave one destroys itself. Another example is Plan pattern.

3. Interaction Patterns: are concerned with interaction and communication of agents, Meeting and Finder are examples of this pattern [1].

2.2. Model

The proposed model is related to traveling patterns, the model is an extension of Itinerary pattern. The hierarchal pattern is concerned with the management of the movement of mobile agents. This pattern will use the depth first algorithm in the process of routing, we can consider as a combination between Itinerary pattern and depth first algorithm.

On the hieratical pattern the agent receives a list of agencies that it has to migrate to. So, it migrates to the first destination agency beginning from the left most, where it executes a task, then going to the other most left agency, if the agent reaches a dead end it

will go back to the upper agency and then go right and so on. The agent repeats this cycle until it visits the last agency on its list. [Figure 2].

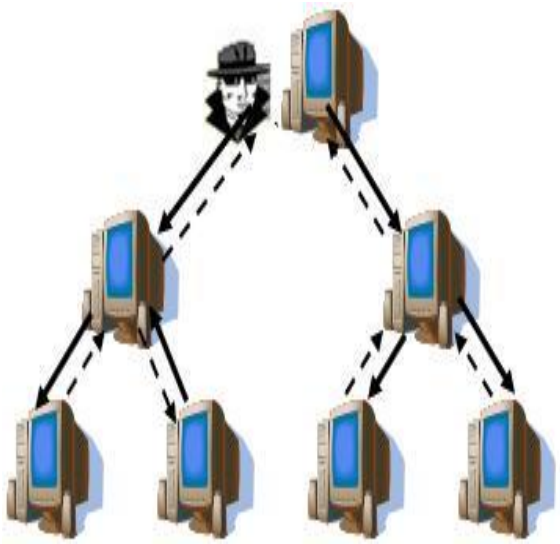

Figure 2 - Hierarchal Pattern

Another scenario of the hierarchal pattern is that the agent receives a list of agencies to visit and clones itself according to the proposed number of nodes (agencies) in the tree of agencies. Then all clones will visit an agency of the received list. Each clone has to execute its corresponding task and notify the source agency when the task is completed. The importance of this part of the pattern is that it splits the tasks that can be executed in parallel. [Figure 3]

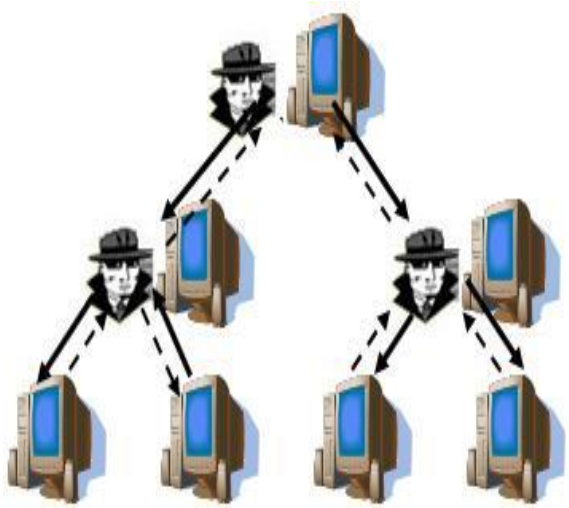

Figure 3 - Hierarchal Pattern with Clones

2.3. Relationship between Hierarchal Object and Agent Object

One of the main objectives of the hierarchal pattern is to move the responsibility of traveling from the agent itself to the associated hierarchal object, the Hierarchal class [Figure 4] should be associated with the Agent class and take the responsibility of navigation from one host to another, the Hierarchal class must implement an interface to communicate with the agent and to dispatch the agent to the new destination and it must define exceptions for the expected exceptions that might appear, for example if the Hierarchal class cannot dispatch to the required host.

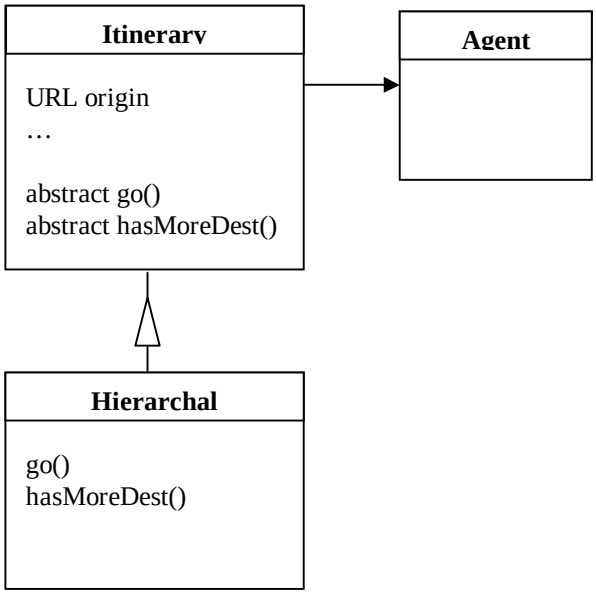

Figure 4 – Hierarchal Class

3. Implementation

For implementing hierarchal pattern we will adopt JADE frameworks and then implement the above classes and taking into consideration the interface between Hierarchal classes and JADE framework.

Implementing the hierarchal pattern is out of the scope of this paper.

4. Conclusions and Future Directions

In this paper I am introducing a model for a design pattern that can be used in application development. The use of design patterns has

been increased due to the advantages they can bring to applications development, like reuse and a better understanding of their project. In this work we proposed a mobile agent design pattern that can be implemented using JADE framework. Hypothetically Hierarchal pattern can be applied on problems that have the nature of trees and the problems that can be solved using parallel or distributed nature.
Using JADE for implementation eases this process, one of the main advantages of JADE is the possibility to check whether a destination is available before transferring agent to it or not, this is very useful for our pattern because the pattern proposed deals with the movement of mobile agents.
I will comment some future direction trends based on this model:

- Implement the pattern in JADE framework
- Compare the results of using Hierarchal pattern with other different patterns.
- applying some case studies using Hierarchal pattern

5. References

[1] Y. Aridor and D.B. Lange. "Agent design patterns: Elements of agent application design." In Proceedings of the Second International Conference on Autonomous Agents. ACM Press, 1998.

[2] Emerson Ferreira de Araújo Lima, "Implementing Mobile Agent Design Patterns in the JADE framework 2003"

[3] Y. Tahara, A. Ohsuga, and S. Honiden. "Agent system development method based on agent patterns." IEEE Computer Society Press, 1999.

[4] D.B. Lange and M. Oshima. "Programming and Deploying Java Mobile Agents with Aglets". Addison-Wesley, Reading, MA, 1998.

[5] K. Yasser, A.Hesham, N. Elmahdi, S. Allola, H. Ahmad. "Optimizing Mobile Agents Migration Based on Decision Tree Learnin". Proceedings of World Academy of Science Engineering and Technology. 2007,

[6] P. Braun, W. Rossak."Mobile Agents: Basic Concepts, Mobility Models and the Tracy Toolkit". Centre for Intelligent & Multi-Agent Systems, Morgan Kaufmann publishers, Australia, 2004.

[7] S. Frederick, H. Lieberman. "Introduction to Operations Research" 8th edition, McGraw-Hill Education, USA, 2005.

[8] Faiz Al-Shrouf , Mohd Eshtay and Khaled Abu Humaidan "Performance Optimization for Mobile Agent Message Broadcast Model Using V-Agent" IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.8, p.p 285-290. 2008.

[9] Faiz Al-Shrouf, Mohammed Eshtay and Ayman Turani, “Mobile Agent Optimization Analysis of Least Time Approach Versus V-Agent” EUROMEDIA'2009, April 2009.

[10] Munehiro Fukuda · Koichi Kashiwagi · Shinya Kobayashi “AgentTeamwork: Coordinating grid-computing jobs with mobile agents” Applied Intelligence, Springer Netherlands, Volume 25, Number 2 / October, 2006, 181-198.