

# Policy Based Approach to Enhance Task Execution Performance of Mobile Agents

**Sarmad Sadik, Arshad Ali**  
NUST Institute of Information Technology  
Chaklala Scheme 3  
Rawalpindi, Pakistan

**H. Farooq Ahmad, Hiroki Suguri**  
Communication Technologies  
2-15-28 Omachi Aoba-ku  
Sendai, Japan

**Abstract** - *Mobile Agents research is an active area in the agent community. Researchers are working to define efficient mobility architecture for mobile agents in multi agent systems but a limited progress has been made towards how to make mobile operations efficient and enhance task execution mechanisms at target machines. We have used Policy based approach in FIPA compliant multi agent system to improve the task setup and execution performance. Policies consist of rules, which are sets of preconditions, actions and post conditions, and each set is attached with a particular goal. The user of a mobile agent specifies its goal at run time and all the associated set of conditions and actions are loaded from policy resource. The administrator of multi agent system can also define various sets of policies for its task execution as well as movement control pattern of mobile agents. We have distributed and shared these policies to prevent increasing the size of mobile agents as agent can invoke the required set of operations at target machines according to its specified goal.*

**Keywords:** Mobile Agents, Multi Agent Systems, Policy Model

## 1. Introduction

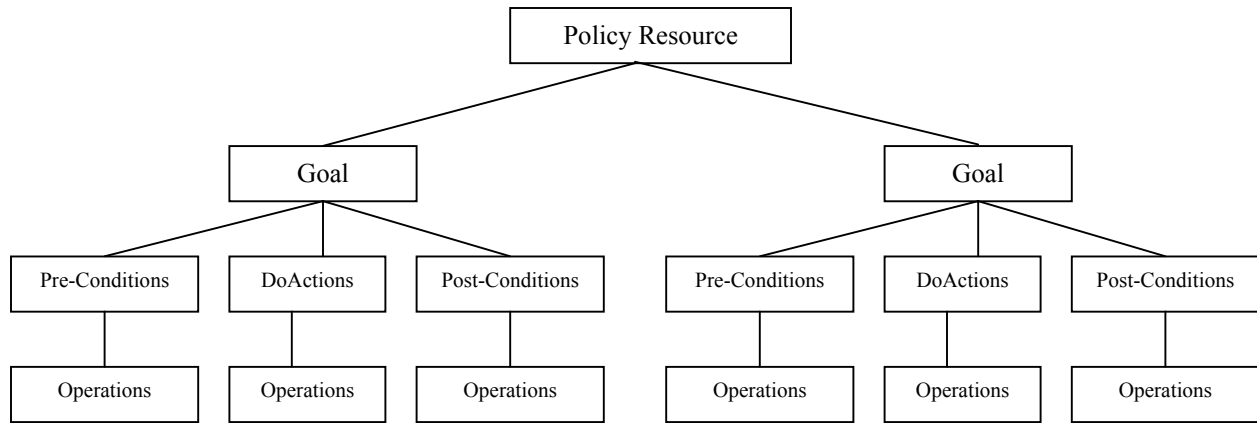
Mobile Agents is an active research area in the agent community. It has the potential to change the way distributed applications are developed and deployed [1]. It is still a developing research area, and a limited research progress has been made towards how to make mobility operations efficient by enhancing

mobile agent task execution mechanisms at target machines.

The existing problems in mobile agent lies in the areas of dynamic and runtime update of knowledge, assimilation and dissimilation of information from agent platform to mobile agents and vice versa. It also includes information about run time update of Goals and information related to movement and execution pattern of mobile agents. If preference of goals is changed or some goal is no more required, mobile agents should update their knowledge base. A mobile agent is not useful enough if it performs only the mobility operation by moving to other machines, rather it should be able to perform its desired task and achieve its goal effectively.

In order to make mobile agent more autonomous and efficient in terms of task execution, its structure is divided into two parts one includes its mobility mechanisms and other portion comprises of its task execution strategies including goals, operations implementation and knowledge base etc. This will make it more versatile and intelligent to simulate real life behavior. One way to achieve this is that all this information and skills are embedded in one mobile agent. This will make it too complex and heavy. Another better approach is only necessary information like network location and mobility capability is provided to the agent and all other expert skills are provided at run time depending on the particular scenario.

Current constraints in Mobile Agent Computing limit their effectiveness and



**Figure 1. General Architecture**

usefulness in application domains. Mobile Agents depend on their agent platform and users on whose behalf they are working.

There is a need to make them more autonomous [2] so that they decide themselves when and where to migrate. Also while in travel, there should be a mechanism available to send, receive and control information from their users or platforms to individual mobile agents. Runtime and dynamic update of information include the change of goals and target locations. Mobile Agents need to be adaptive so that information of state values like goals should be able to adapt and update. Also in heterogeneous environment conditions like different platform, operating system or network parameters, an agent has to adapt its destination environment for its execution [3]. Mobility enhances the need of flexible behavior [4] especially in multi agent platforms, there is a need of new components and protocols to adapt to evolving service and user requirements.

In this paper, we have discussed the approach and strategy to enhance mobility architecture for mobile agents including efficient task execution issues among such mobile agents. The next section provides the information about the related work especially in the context of policy based solutions to existing problems. In Section 3, the proposed architecture is explained with highlighting the policy sharing mechanism and implementation work. In Section 4,

discussion has been made about the proposed work and at the end conclusion and future work is presented.

## 2. Related Work

Researchers have used various reconfigurable approaches in applications for mobile service provisioning [5] but no specific adaptive policies have been used for reconfiguration. A policy based infrastructure has been discussed [6] for mobile code applications which separate the application programming from code mobility using event based models in which if some event happens, the system performs the related action. Policy based framework has also been used [7] for designing applications in ubiquitous environment in which a policy package has been used to describe rules, roles and set of contents. Similarly, Policies have been in use [8] for management of Mobile Adhoc Networks in which policies are deployed for specifying the desired behavior at high level of the system. A role based infrastructure has been proposed [9] where role is defined as behavior or set of capabilities expected for the agent that plays such roles. In [10], an approach is provided for building a mobile service. The basic building blocks have been discussed but no solution is provided for sharing of knowledge between mobile agents and their task handling infrastructure. Our work differs from these approaches as we have implemented policy

based rules which are set of pre-conditions, actions and post conditions and it is integrated with multi agent system for supporting task execution of mobile agents. It's a generic support for all mobile agents created on that particular agent platform.

### 3. System Architecture

#### 3.1 Policy Model

Currently in SAGE (Scalable fAult tolerant Agent Grooming Environment) multi agent system [11][12], a class loading mechanism is used for mobility operations in which on destination machine the related classes of agent are loaded and agent is operational. However, an agent in order to be adaptive and add semantics in its task execution mechanisms, it should add new characteristics in its program. We have used a policy-based approach in FIPA compliant multi agent system SAGE to address the issue of dynamic update of information and improvement of task execution mechanism. Policies consist of rules, which are set of preconditions, actions and post conditions, and each set is attached with a particular goal. The administrator of the system can define these conditions and actions with each goal while

setting the policy of the multi agent system. The general architecture of policy based rules is shown in Figure 1. A policy resource contains number of goals and each goal is attached with a set of conditions. Each of these conditions and actions contain at least a single or many operations which form the core of mobility and task execution strategy. These operations can be picked from a list of operations, placed and adjusted according to particular domain functionality.

The user of a mobile agent specifies his goal at run time and the associated set of conditions and actions are loaded from policy resource which is stored in a separate Policy class. Firstly, it matches the goal of user with the goals mentioned explicitly in the policy resource data. If user mentioned goal matches with the already defined goal, the associated conditions with the specific goal like its pre-conditions, actions and post conditions are loaded dynamically. Each Pre-condition has to be satisfied before moving to the actions list. To achieve this operation, each pre-condition is picked one by one and checked is it true or satisfying the condition. If it denotes some operation, its definition is loaded from list of

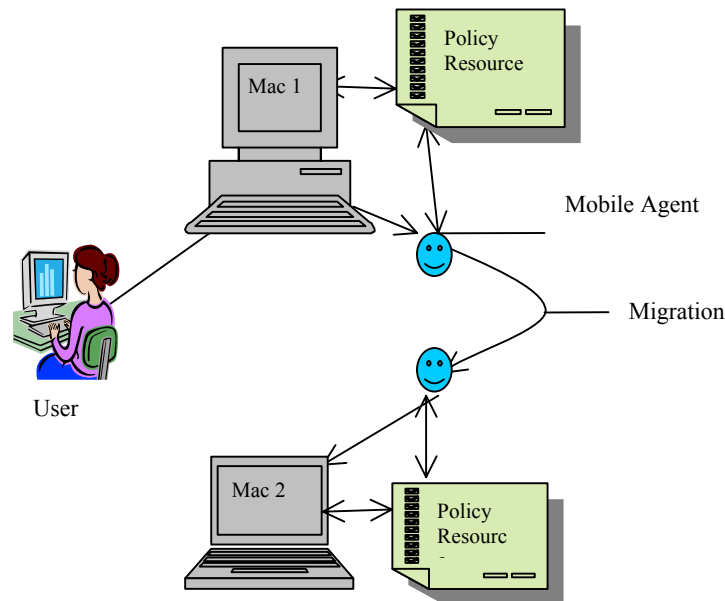


Figure 2. Sharing of Policy Resource

operations class, which includes the implementation details of each operation or entity. After loading its implementation detail, the operation is executed with the required parameters. Similarly all pre-conditions are checked and executed to satisfy true values and after that the control is passed to DoActions list. DoAction also consists of some operations that are run sequentially one by one and executed according to implementation defined in the list of operations class. This is actually the main task execution body, which forms the core of mobile agent functionality. After executing all operations, it goes to the post conditions list and checks out any task to be executed. Post conditions contain those operations, which need to be executed after the mobility operation on the target machines.

In case the goal entered by a user is a new goal, which is not defined, in existing resource, the user is asked to define its associated sequence of operations for task execution. The user can enter its new goal and customize its sets of conditions by bringing the operations from list to its automatically generated and associated lists. The user can also define new operations particular to some domain functionality and add it to the operations resource. This provides user with great flexibility and convenience to create and use mobile agents at run time.

### 3.2 Policy Sharing

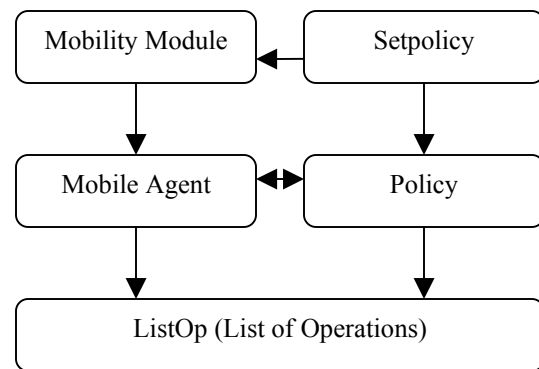
The policies are distributed and shared on the multi agent systems. The agent platforms which are connected with each other can dynamically share these policies. This helps in reducing the size of mobile agents. Initially when an agent has to carry its entire task related information and sets of conditions with it, it increases its size and affects its performance and execution. With the advantage of sharing and distributing such policies resource structure, the mobile agent just need information of its goal and it can invoke its associated actions and operations at runtime even after migrating to its target machines according to its specified goal. The Policy sharing and interaction with mobile agent mechanism is shown in Figure 2. At start of itinerary on machine 1, a mobile agent can

retrieve the necessary set of information relevant to its specified goal from policy resource and execute the related operations. After moving to Machine 2, it can invoke the remaining set of tasks or information which are part of that goal activity but targeted to be run on destination machine after migration.

This policy based framework can also be used to control the movement and interaction pattern of mobile agents. The administrator of multi agent system can use policies which are strictly implemented with the agent system and they control the movement of agents to specific locations. The administrator can either block certain locations or set of IP addresses for agents to migrate or he can create a mechanism to block the execution of certain activities or task at destination machines. These measures provide extra functionality to platform administrator for control and authenticating the mobility procedures.

### 3.3 Implementation

This work has been implemented using SAGE (Scalable fault tolerant Agent Grooming Environment) multi-agent system. In SAGE, mobility support is provided for Intra-platform and Inter-platform mobility but to enhance the existing work, a policy based mobility framework has been integrated with already existing mobility model. We have created separate classes for representing policy resources and for adding and administrating



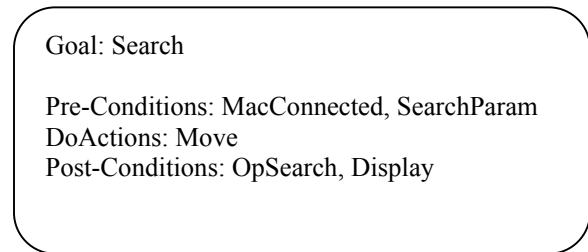
**Figure 3. Class Structure**

policies as well as implementing various operations.

This has been achieved by implementing a Setpolicy class which is executed after booting the multi agent system and before creation of mobile agents. We have created a mechanism to separately define the operations, which are actually executed and mentioned in sets of conditions and actions. Each particular operation “op” has its implementation in separate class of naming “ListOp”. These defined operations can be placed in pre-conditions, actions or post conditions. The administrator or user can dynamically place them to create working functional mobile agent. This reflects one of the advantages of policy based approach using sets of rules which helps in creating agents conveniently by separating the application programming from functionality defining and execution. Figure 3 shows the class level distribution of the current system.

As a case study, we have chosen an example of a search program which takes some parameter, searches at target machine and displays the results. The user while creating the mobile agent program enters his goal as “search”. This goal is matched with the already defined goal definitions in policy resource and if it matches the goal the user is informed of loading already defined sets of conditions and actions. We have defined the basic functionality of this search program, the list of operations which are defined in Pre-conditions are “MacConnected” and “SearchParam”. The MacConnected parameter represent that the destination machine is available and connected. The “SearchParam” operation checks for the search parameters availability and get it from user at runtime. After the criteria verification by successful execution and true values of pre-condition parameters, the control goes to DoAction lists. We have defined a single action in the DoAction list of search program which is “move”. This “move” action serves as the mobility request, it takes the target machine information and its platform name. The mobile agent migrates to the destination machine. In post conditions, we have included the parameters of OpSearch and Display results. Its

implementation is dynamically invoked from list of operations and these operations are executed one by one in a sequential way. An example instance is shown in Figure 4.



**Figure 4: Example instance**

At the booting time of multi agent system, the administrator may run the Setpolicy agent and configures the various policies and its associated conditions. At this moment, he may enter a single or range of IP addresses for which he want to restrict the movement of mobile agents. At the same time, administrator can enter operations which are restricted to be executed and it will be checked each time, some operation activity is entered by users.

## 4. Discussion

This policy based strategy and implementation framework provides user the capability to create mobile agents with ease and convenience. They can also tailor their functionality by adding and defining their own operations in the list and include them as parameters in pre-condition, post condition or actions. The implementation of this model in multi agent system makes it more user friendly and customizable as information can be added and updated at run time and dynamically. The distribution of such policies has provided efficient information sharing mechanism and reduced the over all size of mobile agents. This makes their operations more efficient and improves the overall performance. This framework also provides reusability to a great extent so that goals and its associated conditions once defined can be reloaded and

executed. This not only saves time in creation of mobile agents but rather it helps in making efficient functional agents in less time.

## 5. Conclusion

In this paper, a policy based approach has been defined which consists of rules for defining the list of operation, conditions and actions for mobile agents. Each rule consists of sets of pre-conditions, actions and post-conditions which are associated with a particular goal. These can be filled with parameters dynamically from operations whose implementation details are specified separately along with list of other implemented operations. The administrator or the user can define a goal and its associated set of conditions and actions. If user goal matches with already defined goals in policy resource, it is loaded dynamically and operations present in it are executed one by one to achieve the desired result. If user defined goal is not available in already existing resource, the user can add and customize its properties and related activities and operations. The model can also be used to control the movement and interaction pattern of mobile agents by restricting their movement to certain locations or controlling the execution of some specific activities on target and source machines.

## 6. Future Work

There is a need to enhance the policy model to control and simplify the execution of all agents in addition to mobile agents running on multi agent systems. This separates the abstraction of making efficient functional agents from agent programming at code level details. This policy based model may be extended to monitor and control different quality of service parameters in multi agent systems like terminating or limiting the execution of certain agents if system resources are degrading and to sustain the running of basic processes of multi agent systems.

## 7. References

1. A. Fuggetta, G. Picco, and G. Vigna, "Understanding Code Mobility", IEEE Trans. Software Engineering, May 1998

2. Michael Wooldridge, An Introduction to Multi-agent Systems, John Wiley & Sons Press, 2002.
3. Brazier, F.M.T., Overeinder, B.J., Steen, M. van and Wijngaards, N.J.E., "Generative Migration of Agents", Proceedings of the AISB'02 Symposium on Adaptive Agents and Multi-Agent Systems, 2002, pp. 116-119.
4. Paolo Bellavista, Antonio Corrad, Cesare Stefanelli, "Mobile Agent Middleware for Mobile Computing", IEEE Computer, March 2001.
5. Radu Litiu, Amgad Zeitoun, "Infrastructure Support for Mobile Collaboration", Proceedings of the 37th Hawaii International Conference on System Sciences - 2004
6. Rebecca Montanari, Emil Lupu and Cesare Stefanelli, "Policy-Based Dynamic Reconfiguration of Mobile-Code Applications", IEEE Computer, July 2004, pp. 73-80.
7. Ken'ichi Takahashi, Satoshi Amamiya, Tadashige Iwao, "An Agentbased Framework for Ubiquitous Systems", Challenges in Open Agent Systems '03 Workshop, Melbourne, Australia, 2003.
8. Ritu Chadha, Yuu-Heng Cheng, Jason Chiang, Gary Levin, Shih-Wei Li, Alexander Poylisher, "Policy-Based Mobile Ad Hoc Network Management For DRAMA", MILCOM 2004 USA.
9. Giacomo Cabri "Role based Infrastructure for Agents" , The 8th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2001), Bologna, 2001
10. Bernd Mrohs, Christian Rack, Stephan Steglich, "Basic Building Blocks for Mobile Service Provisioning" The 7th International Symposium on Autonomous Decentralized Systems (ISADS 2005), China, 2005.
11. Arshad Ali,H. Farooq Ahmad,Zaheer Abbas Khan, Abdul Ghafoor, Mujahid and Hiroki Suguri, "SAGE: Next Generation Multi-Agent System", in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, Nevada, USA 2004, pp. 139-145
12. Scalable fault tolerant Agent Grooming Environment (SAGE), In the Fourth International Joint Conference on Autonomous Agents and Multi agent Systems (AAMAS) Demo, Utrecht, Netherlands, 2005.