MOBILE AGENT ORIENTED ONTOLOGY BASED POLICIES FOR HONEY-BEE TEAMWORK ARCHITECTURE

By

Sarmad Sadik
2003-NUST-PhD-IT-45

A thesis submitted for the partial fulfillment of the requirements for the degree of

Doctor of Philosophy

to

School of Electrical Engineering and Computer Science

National University of Sciences and Technology

Pakistan

2010

It is certified that the contents and form of thesis entitled "**Mobile Agent Oriented Ontology Based Policies for Honey Bee Teamwork Architecture**" submitted by Sarmad Sadik, have been found satisfactory for the requirement of the degree.

Advisor: ____________________________

(Prof Dr. Arshad Ali)

Co-Advisor: ___________________________________

(Dr. H. Farooq Ahmad)

Committee Member 1: ___________________________

(Prof Dr. Muhammad Akbar)

Committee Member 2: ___________________________

(Dr. S.M. Hassan Zaidi)

Committee Member 3: ___________________________

(Dr. Hiroki Suguri)

CERTIFICATE OF ORIGNALITY

I hereby declare that the research thesis titled "Mobile Agent Oriented Ontology Based Policies for Honey Bee Teamwork Architecture" is my own work. It contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at School of Electrical Engineering and Computer Science (SEECS) or any other education institute, except where due acknowledgment, is made in the thesis. Any contribution made to the research by others, with whom I have worked at School of Electrical Engineering and Computer Science (SEECS) or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic is acknowledged. I also verified the originality of contents through plagiarism software.

Name: _Sarmad Sadik

Signature: _____________

In the name of Almighty Allah
the Most Beneficent and the Most Merciful

Dedicated to One and Only Allah Almighty

ACKNOWLEDGEMENTS

I humbly acknowledge my beloved God for all the blessing showered upon me. I profoundly acknowledge my adviser Dr. Arshad Ali and co-adviser Dr. H. Farooq Ahmad for their academic and moral support during the entire period. I am also grateful to my committee members Dr. Muhammad Akbar, Dr S.M. Hassan Zaidi, Dr. Hiroki Suguri for their guidance and assistance throughout the PhD phase.

I am also thankful to external reviewers including Dr Coskun Bayrak (USA), Dr Flavio Oquendo (France) and Dr Liaqat Ali for taking out their valuable time and reviewing the thesis. Lastly I am thankful to my family and friends for their boundless support.

Sarmad Sadik

ABSTRACT

With exponential growth in applications complexity, software agents' domain is making a remarkable progress. Complex and distributed applications need a group of software agents to perform the intended tasks, instead of standalone isolated agents. In software agents' domain, limited work has been done in case of teamwork among mobile agents. Mobile agents which have the capability to roam around on various machines need special communication and coordination strategies as well as goal sharing and task accomplishment mechanisms. In this thesis, a teamwork strategy is proposed to enhance the task execution performance as well as mechanism to efficiently address the communication and coordination issues among group of mobile agents. The proposed technique is developed by conceptualizing the Honey-Bee teamwork strategy and named after it as "Honey-Bee teamwork architecture". In association with it, the goal oriented ontology based policies concept is proposed for improving the mobility mechanisms of mobile agents. It also addresses the aspects of goal definition, task creation and execution on distributed machines by mobile agents. This goal oriented approach reinforces the teamwork architecture as well as its goal and task division characteristics. The ontology based policy technique facilitates in convenient assignment of goals and associated tasks setup information for mobile agents which were earlier difficult to create and execute as desired especially in dynamic environments. The proposed strategy is evaluated using combination of teamwork strategies on single and multiple machines scenarios. The proposed honey-bee technique was found more efficient in the distributed infrastructure especially for higher number of member agents in a team. It shows linear behavior as compared to exponential increase in task accomplishment time in case of conventional strategy. The combination of simplified task execution as well as reduced size of mobile agent requires less time in performing the desired functionality. Additionally, the earthquake management system (EMS) is proposed and discussed as proof of concept application. It highlights the usage of software agents in disaster management and expresses its teamwork capabilities to handle complex application scenarios. This leads to conception of major application domain and test bed where software agents may execute in combination of various teamwork fashions. The major structure and behavior of proposed application is expressed and analyzed using Pi Calculus and Pi ADL formal techniques.

Table of Contents

LIST OF FIGURES

FIGURE	CAPTION	PAGE

LIST OF TABLES

TABLE	CAPTION	PAGE

LIST OF ABBREVIATIONS

MA	Mobile Agent
MAS	Multi Agent System
FIPA	Foundation for Intelligent Physical Agents
KAoS	Knowledgeable Agent-oriented System
JADE	Java Agent Development Environment
OWL	Web Ontology Language
ORL	OWL Rule language
RDF	Resource Description Framework
EMS	Earthquake Management System
ANSS	Advanced National Seismic System
HLA	High Level Architecture
ADL	Architecture Description Language
TL	Team Leader
NTL	Non Team Leader
W3C	World Wide Web Consortium
SQL	Structured Query Language
SAGE	Scalable fault tolerant Agent Grooming Environment
ACL	Agent Communication Language
URI	Uniform Resource Identifier
ADS	Autonomous Decentralized Systems
ISA	Information Service Agent

FSA	Field Service Agent
PSA	Personalized Service Agent
ESA	Emergency Service Agent
PA	Personal Assistant
ABC	Another Bisimiliarity Checker
MWB	Mobility Work Bench

Chapter-1

INTRODUCTION

1.1 Motivation

The last decade has seen an exponential growth in computing and the use of Internet. The rapidly evolving network and computer technology, coupled with the expansion of services and information available, is moving towards a new era of mobile and ubiquitous computing. The devices are no longer isolated; rather they are distributed in nature. In the upcoming era, new paradigms are required for building distributed systems and applications, with autonomy and social ability, like software agents.

A software agent can be defined as a program or an independent module which works or executes in order to accomplish the goals assigned by its creator or user. Software agents need a supporting platform on which the agents can be created, managed and executed. This underlying middleware or platform is defined as Multi-Agent System.

The mobile agent technology is playing a key role in driving research activities in agent related research community. It possesses the capability to revolutionize the methodology in which distributed applications are designed and deployed [1]. The mobility feature highlights the flexible behavior [2] in multi agent systems, in which various interactive components as well as protocols are involved in order to address heterogeneous functionalities. The fundamental properties of mobile agents are capability to move across various hosts and autonomous operations at remote hosts.

Major research is being undertaken in domain of teamwork among software agents, highlighting various coordination and collaboration strategies. However, very limited work is being done for teamwork among mobile agents. Mobile agents are special types of software agents which demand extra capabilities and features in traditional design of communication and coordination as well as goal sharing and task accomplishment.

The ontology based policy technique is a strong candidate strategy to address major issues in teamwork among mobile agents. The term “Policy” is defined as per current environment or relevant context [3]. Policy is a rule based expression or statement which is linked to constituent conditions and actions [4]. Policy based systems have been actively used in domains of management and security related functions.

Ontology can be described as meta-data or domain concepts and its relationships. The ontologies are used for sharing of domain knowledge and expressing data in specified schema. Ontologies [5] can also be defined as formal specifications of domain knowledge. It is a constituent part of building semantics infrastructure in multi agent systems and semantic web. Ontologies provide support for interoperation and definition of domain data in multi-agent systems. Ontology based systems facilitate in runtime reconfiguration and information sharing among software agents in distributed architecture. Additionally, it simplifies the policy engineering problems [6] like authoring, conflict resolution and deployment.

Ontology based policy techniques can improve the task execution efficiency of mobile agents by providing a flexible approach for creation and execution of tasks. It adds reusability, customizability and flexibility. This technique can help to incorporate teamwork support in multi-agent systems by reducing the middleware support as teamwork will be the need of future especially for complex applications in the future arena.

1.2 Problem Statement

Mobile agent is a new paradigm for building distributed systems. The existing approaches are more oriented towards building inherent mobility mechanism in multi agent systems as compared to enhancing the individual characteristics of mobile agents. Additionally, there is no consensus on a conceptual framework for building mobility characteristics in multi-agent systems. As the applications are becoming more complex and distributed, mobile agents require more efficient teamwork mechanisms. More autonomous coordination and cooperation is required in order to execute the assigned tasks. These circumstances introduce new issues such as knowledge sharing, expression of domain data and tasks execution mechanisms in the distributed environment. There is a need to enhance the mobile agent basic structure and fundamental capabilities like intelligence, autonomy, proactive and social behavior [7] in addition to its inherent mobility potential.

Furthermore, there is no clear understanding of the new abstractions offered by this paradigm. Achieving mobility in software agents is a complex process which requires developer's extensive role in defining when and where to move which components under varying operating conditions. Additionally, there is a need to analyse possible techniques to carry preferences and goals by agents during mobility operations. Also there is a need to explore the possibility of making groups of agents with varying combination of core properties like mobility, rationality, behaviours etc. in order to find out the ways of achieving goals in collaboration with each other. In collaboration architecture, agents need to share with each other their specializations, knowledge, goals and dynamic parameters. The problems are: change in environment, change in goals to-be-achieved and their priorities. Also the approach needs to be formalized in order to enhance the modelling and reliability aspects.

1.3 Research Hypothesis and Questions

The aim of this research is to explore the limitations in domain of teamwork among mobile agents and highlight its potential capabilities through employing ontology based policy strategy. The hypothesis is stated as follows,

"Goal oriented ontology based policies technique can address the major issues of task definition and execution in dynamic teamwork architecture of mobile agents."

The following research questions originate from the hypothesis statement, which have been explored and addressed in this thesis.

- What are the limitations in execution as well as coordination and cooperation of mobile agent applications?
- Which teamwork strategy is more efficient among mobile agents when they are distributed on multiple machines?
- How interactions among mobile agents influence the performance of teamwork architecture especially in scenario of distributed infrastructure?
- Why mobile agent applications are tightly constrained with multi agent system?
- How can an ontology based policies approach address the limitations and improve the overall architecture?

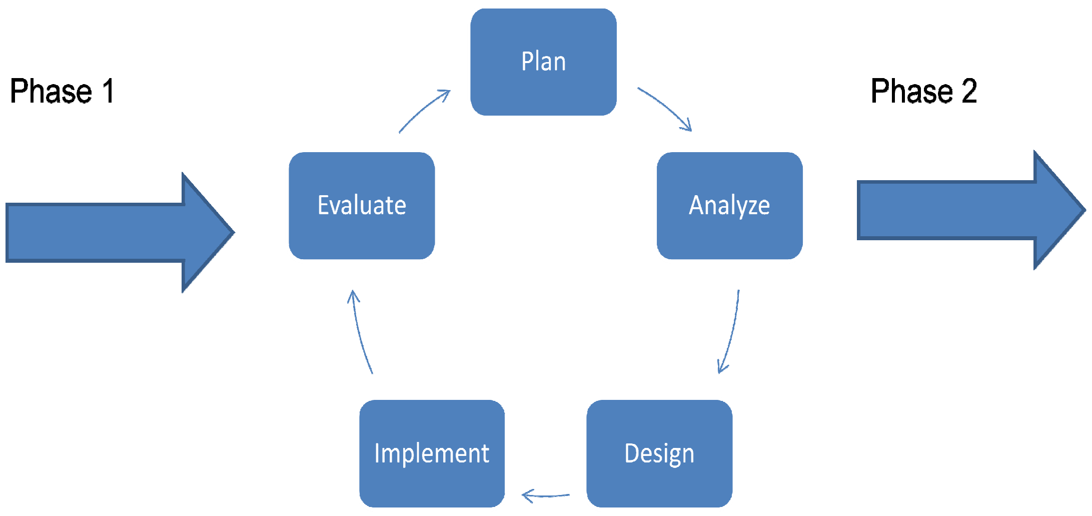


Figure 1.1 Action Research with Iterative Approach

1.4 Research Methodology

The methodology revolves around paradigm of action research in order to investigate the above mentioned research questions. As per action research methodology, the work is planned, designed and developed, evaluated and re-visited. The flow of activities is described in figure 1.1 and figure 1.2 which classifies the research work in five phases.

- In first phase, research domain and problem statement is analyzed in context of research questions. In addition, background study and state of the art literature review is done to familiarize with latest trends and techniques in research domain.

- In second phase, an efficient teamwork architecture has been proposed in the domain of multi-agent systems, which is based on honey-bee teamwork strategy especially for mobile agents. Two major teamwork paradigms are considered. In first case, the primary goal is shared through team leader approach while in second case members are assigned the goal and they perform their respective tasks in coordination and collaboration with each other. The evaluation is made for both paradigms when the teams are distributed on multiple machines and overhead of inter-machine communication is analyzed.

- In third phase, another contribution is proposal of ontology based goal oriented policies technique where policies are made of various tasks and conditions in a tree like structure. Each primary goal is divided to sub-goals which are associated

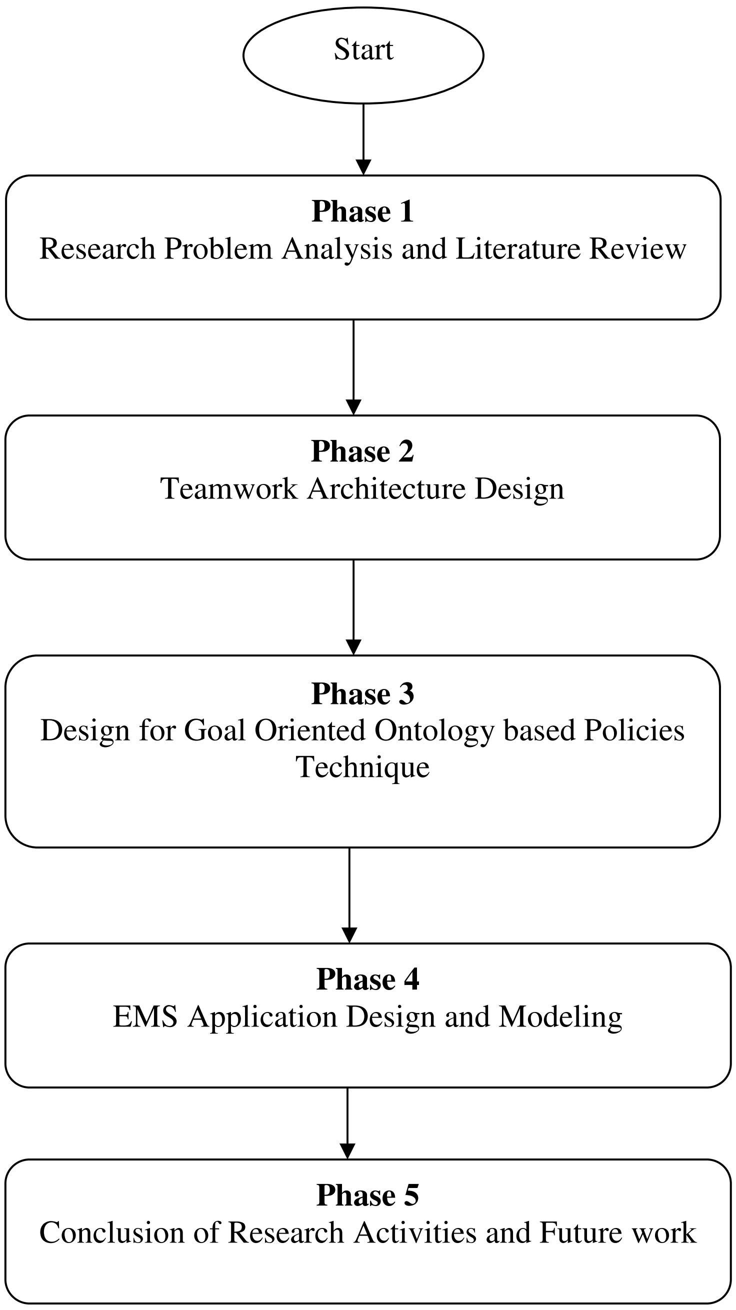

Start
Phase 1
Research Problem Analysis and Literature Review
Phase 2
Teamwork Architecture Design
Phase 3
Design for Goal Oriented Ontology based Policies Technique
Phase 4
EMS Application Design and Modeling
Phase 5
Conclusion of Research Activities and Future work

Figure 1.2 Research Methodology

with respective sub-tasks. These are joined together in policies form where a particular goal triggers the associated list of conditions and actions. These policies are represented in ontologies form using OWL [8] which is standardized by W3C. It provides higher flexibility as compared to tightly constrained traditional approaches. These ontology based policies are published and accessed as well as manipulated by mobile agents using URI through Protégé [9] and Jena [10] APIs.

- In fourth phase, the proposed work is discussed and analyzed by designing a novel application in domain of disaster management systems called as Earthquake Management System (EMS). This application is based on concept of utilizing the autonomous and intelligent nature of software agents in order to fulfill the demand of quick response activities from start of earthquake to other relief efforts. The major modules are used as the proof of concept application for teamwork and ontology based goal oriented policies research work. The proposed application is modeled using formal method techniques of Pi-Calculus [11-14] and Pi-ADL [15-16, 93] in order to analyze the specifications and working behavior.

- In last phase, research activities are concluded by discussing the research outcome and highlighting the future directions.

1.5 Thesis Outline

The thesis highlights the honey-bee teamwork architecture as well as ontology based policy framework for mobile agents in multi-agent systems. The literature review is presented in chapter 2 where a review of teamwork efforts is highlighted along with their limitations in context of mobile agents. Also, various generic policy based techniques are discussed and later relevant ontology based work is highlighted.

In chapter 3, an efficient teamwork strategy is proposed after discussing two major paradigms of team leader and non-team leader approaches. The Honey-Bee teamwork strategy is discussed and mapped with mobile agent operations. Chapter 4 describes the ontology based policy architecture where goal oriented task based policies concept is proposed. The classification of policies as well as its role in mobile agent operations is presented and later its representation in ontologies form is discussed.
Chapter 5 presents the formal approach towards design, modeling and analysis of agent based disaster management systems. An Earthquake Management System (EMS) is proposed which is composed of software agents. Additionally, the roles of agents along with major activities are analyzed using formal methods. In Chapter 6, the implementation of proof of concept application is discussed along with modeling and specification of earthquake management system in pi-calculus and pi-ADL.

In Chapter 7, the evaluation is presented firstly about the teamwork strategies and then by using the ontology based policies approach. The evaluation is highlighted in context of agents' execution on multiple machines and inter-machine communication aspect. Also the formal verification work is described in later part of the chapter.

Chapter 8 provides the discussion and critique about the results obtained as well as the overall analysis of proposed work in context of research domain. Conclusion and Future work is highlighted in Chapter 9.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, the background related to research problem and state of the art is highlighted. It starts with introduction about mobile agents and its brief history. Then review of teamwork is presented, followed by the policy definition and background in related work context. The ontology based approach is presented by highlighting the major work in this domain. Lastly formal approaches are reviewed which are applicable in the proposed domain.

2.2 Mobile Agents

As compared to existing techniques for information exchange, mobile agents exhibits to be better technology especially in e-commerce domain, distributed data searching and retrieval, and parallel processing etc. In distributed hosts environment such as varying operating system, data models and network parameters, a mobile agent needs to learn and adapt according to target host requirements [17]. The mobility property highlights the requirement of flexibility in multi agent systems [18] for addressing various interoperability issues. A new set of system components and interaction protocols are required to match the evolving service related requirements. Researchers have focused on the factors behind limited performance of mobile agents such as its tightly constrained structure with underlying system and highlighted comparative migration approaches in [19].

FIPA initially provided specifications for Agent mobility [23], however it was not adopted widely and no further work was accomplished. The specifications provide guideline framework for providing mobility in FIPA agent systems. The document highlights various features and mobility protocols for proposed mobile agents, however a lot of work needs to be designed and implemented on agent systems part. The document also describes possible agent states as a result of specific actions as well as brief description about concepts related to agent migration, cloning and invocation.

Researchers have highlighted the analysis of weak and strong mobility [24] and also compared the two approaches at system and application level. Mobile agent technique has been described as one of the promising technologies for distributed applications in heterogeneous environments. In addition, various aspects of Java for mobility support have been highlighted. An example of information retrieval is highlighted where mobile agent visits web servers for finding interested pages. It also explains the example application scenarios of mobile agents in electronic commerce, network management and load balancing domains. A comparison of three programming languages for developing mobile systems is highlighted in [20]. A unifying framework is discussed [21] where the proposed approach attempts to simplify the mobile agent development and addressing interoperability issues. Also, the problem of agent migration among incompatible platforms is highlighted in [22].

In [25], the importance of mobile agents is highlighted and their interoperability issues are discussed especially when they are designed by different vendors. The researchers have provided mechanism to support runtime interoperability of mobile agent systems. They have considered the application which is most motivating for this research domain i.e. peace keeping and disaster response systems. In such situations, there is a need to form joint coalition of organizations with their specific information sets to achieve joint objectives. Also, it is emphasized to adopt API translation mechanism among agent systems to support mobile agent execution.

Ara agent platform supports execution of mobile agents in a secure way [26]. It introduces the concept of virtual places and common security policy for mobile agents developed in various languages and executed over identical core system. It presents comprehensive security strategies for issues related to mobile agents. However, tradeoff needs to be realized between security and flexibility as well as performance. Other efforts in this regard include mobile object workbench [27] which provide location transparent migration services for mobile agents.

There are other related attempts, made for making applications mobile [28] as teleporting system where the objective is to move the system interfaces between the machines. Also mobile agents find their application in various scenarios of network management [29], telecare and teleassistance [30]. Researchers have also exploited the use of mobile agents on lightweight devices like PDAs [31] and discussed their properties especially from security point of view.

2.3 Teamwork among Agents

There is very limited work in domain of teamwork among mobile agents; however most of research revolves around teamwork issues among generic software agents. There is a requirement of generating team or joint action strategies by software agents having individual expertise or specialization like rationality or mobility etc.

The merger of software based agents' technology with teamwork domain highlights a critical direction in this particular research domain. The software agents, which are collaborating with each other in a teamwork fashion, will be part of future applications. Such agents will work to search, collect and analyze information, which leads to planning of goals and related task oriented activities. [32].

A Hybrid strategy has been presented in [33] for highlighting teamwork in software agents. Additionally, the existing techniques which are employed in multi agent systems have been discussed. The work highlights the need for cross cutting research involving proven concepts in various techniques. They have applied their approach on disaster rescue simulation project for verification. The major research issue is how to build teams and enable them to communicate, coordinate and adapt in complex and dynamic environments. The work also describes the infrastructure and algorithms as well as application domains for teamwork operations. Similarly, the task allocation strategies in teams are discussed in [34].

In [35], a software agent based system is exhibited in order to analyze and design the composition of Humans and Agents team. This teamwork exemplifies roles of joint operations by mobile robots and human users in future mission critical space missions. The work provides integration of Brahms and KAoS agent frameworks to model and simulate work scenarios in space in the context of human-agent teamwork. The work emphasizes that design of agents should be problem-driven, activity centered and context bound. Brahms is an agent based tool developed by NASA Ames for modeling and simulating real work situations in space. KAoS agent services are used to express teamwork models, mobility and resource controls for space operations. In this case, the models are represented in the form of policies. These policies are used to specify the limitations and capabilities of agents where they can control their autonomy depending upon the location and knowledge awareness. The policy based teamwork model defines the team and its collaboration activities. Also, a model of human-agent teamwork is highlighted in [36].

Various design patterns have been discussed in [37] to generate a variety of teamwork strategies. However, it lacks the definition and description of any specialized system architecture for coordination of static and mobile agents. The work primarily emphasizes the building of application in form of teams of agents with certain specific communication strategies among them. These properties are based on mathematical concepts and classified into seven major types called as inverse, compensation, commutative, idempotence independence, monotonic and multi monotonic. They have also discussed three example scenarios (where the proposed design pattern could be considered) which are information searching, mobile shopping and information filtering in wireless environments.

Teamwork capabilities have been discussed in [38] with reference to software agent based framework. In this particular case, the proposed system shows the specification of goals, tasks and related activities explicitly. This framework highlights the selection of decisions in an explicit form to address the issues of communication and coordination overheads. The framework also emphasizes about monitoring of team performance by using team operators. The focus is to develop integrated teamwork capabilities in order to achieve flexibility and reusability instead of pre-planned and domain specific strategies which are due to limitations in existing frameworks.

There is a unified framework proposed [39] for teamwork operations in order to address the problems of heterogeneous and distributed entities as well as uncertain and dynamic environments. They have also analyzed the computational complexities of development under various problem domain classes. Also, the strategies have been proposed in [40] for designing and analyzing teamwork models in which the model of building blocks is used to highlight the collaboration among group of software agents. The work primarily contributes towards techniques and tools for strategy analysis and performance modeling.

2.4 Semantic Policies

Policy terminology takes its meaning according to relevant context. According to Wikipedia [41], “a policy is a plan of action to guide decisions and actions. The term may apply to government, private sector organizations, groups, and individuals. The policy process includes the identification of different alternatives, such as programs or priorities, and choosing among them on the basis of the impact. Policies could be understood as political, management, financial, and administrative mechanisms arranged to reach explicit goals.” Policy is rule based declarative statement. It is usually represented in the form of event condition and action form where an event is triggered by state or execution changes.

Mobile code programming is a difficult task since its inception. Major techniques include remote evaluation, code on demand and mobile agent. The major work in domain of policy based systems was proposed in [4, 43]. In this case, policy based system is used to detach application level code from mobility related programming by using event driven architecture. The traditional approaches of embedding strategies in code make it more complex and enhance coupling. Policy based approach is being incorporated, which acts at higher levels of abstraction and separates mobility from application functionality. In this case, event driven models have been employed such that in case of any event occurrence, the underlying system executes the associated actions. The limitations of such systems include use of complex middleware and not suitable enough for web related ubiquitous as well as light weight services.

In [44], a policy based context aware service is presented for next generation networks where policies are used to represent context as well as services in networks. The service is enhanced by using mobile agents approach. Policies have been used to act as a guide in order to control and regulate the networks and distributed systems. Policies are acting as high level declarative language and help administrators to configure network related devices by making changes at system level policies. Additionally, mobile agents have been used to travel between machines running Grasshopper agent platform for service delivery as well as results collection through secure channels.

A variety of reconfigurable techniques have been employed in mobility related applications for mobile service, [45] however no specific adaptive policies have been defined for dynamic reconfiguration. Policy based framework has also been used [46] 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 used [47] 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 [48] where role is defined as behavior or set of capabilities expected from the agent that plays such roles. There is a policy and mechanism level introduced in architecture that controls the environment and provides mechanism among agents as well as between agents and resources. In [49], an approach is provided for building a mobile service in which the fundamental building blocks are highlighted however no particular solution is presented for information sharing among mobile agents and its task execution capability.

However, the major research in the field of mobile agents and policy based models is taking place in the domain of security in which researchers are working to define security policies for multi agent systems that support mobile agents [50, 51]. Also policy research forum is doing considerable work in this domain [52].

Although most multi-agent systems existing today like Jade [53], Aglets [54], Concordia [55] etc. support agent mobility, but policy based framework is not available in mobile agents operations for enhancing their flexibility and task handling capability. A performance analysis among multi-agent systems and evaluation is highlighted in [56-58].

The proposed work in this thesis differs from the above mentioned approaches as we have implemented goal oriented 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.

2.4.1 Ontology based Policies Approach

The ontology based approach is syntactically and semantically richer than common approaches for databases [5]. The information described by ontology consists of semistructured natural language texts and not tabular information. If it is to be used for information sharing and exchange, ontology must be a shared and consensual terminology. Various types of ontologies are classified as Domain Ontology, Common sense ontology, Meta data ontology, Representational ontology and Task ontology.

The ontology based policy approach has been used in pervasive and ubiquitous computing [59]. The prime goal is to achieve the seamless roaming of mobile devices and users in dynamic environment. In such approach, the devices can access services and data with identical access rights and behavioral preferences in various zones. A framework has been highlighted in which ontology based semantics is employed for expressing authorization related policies.

In [60], active policy based network management system is described where this policy based technique could be used to adapt as active node framework. They have used policy based management protocol to exchange policy information between policy server and its clients. The work also uses the Web Ontology Language (OWL) in order to enhance the semantic expressiveness and maintenance. Policy based approach provides more customized and flexible management solution which helps in configuring network elements dynamically. It also provides various arguments for assigning preference to OWL rather than other related technologies.

In [61] a framework has been discussed for applying semantic policy structure to robotic systems. The semantic policies have been considered in order to verify actions and conflicts as well as presenting automated recovery in extreme situations. KAoS and OWL have been used to create policies for controlling robotic systems. The policy domain is elaborated and classified into authorization and obligation policies. The work describes expressiveness, external nature, transparency and flexibility as the benefits of using policies. . They have further defined each of these properties in detail.

In [62], a dynamic policy based technique has been discussed to manage data exchange by incorporating mechanisms of semantic filtering and in-stream message transformation. Such approaches are based on policies infrastructure and facilitate in safeguarding transmission of critical information by software agents.

Policy based architecture has been discussed for service delivery in [63]. The policies have been modeled using formal techniques and the advantages of simple policy engine implementation are discussed. This approach facilitates in specification of policies for various domains in single declarative language.

The rule based policy infrastructure has been presented to specify adaptive service behavior as well as semantic representation of operational state [64]. Policy based management system is proposed where resources can be provided to devices which offer semantically annotated services. An approach is provided to combine semantic web services with management information semantics and policy rules to define adaptive behavior.

Researchers have proposed ontology based model for agents especially in pervasive environments [65]. The ontologies are proposed in OWL which describes the agent execution context and its composing elements. They have presented component based generic and adaptive architecture which facilitate modularity, reusability and extensibility of agents.

Researchers have specified the service constraints as policies [66] which are represented as combination of ontology and rules as metadata and schema. A concept of OWL Rule language (ORL) has been proposed in order to specify semantic web service constraints for peer to peer systems. Ontologies have been used to describe agent execution context and its functional components.

In the domain of autonomic computing, there are approaches specified [67] to address problems of semantic interoperability. It uses ontological reasoning for self-management systems. The autonomous system requires dynamic policies mapped with objectives for adapting elements which are addressed by modeling resources as composable services by using service-oriented approach. This strategy of ontology based semantics facilitates in heterogeneity and reasoning framework for policy refinement.

A constraint logic based policy specification language has been proposed [68]. It provides access to specified parts of ontologies while other contents are restricted. The major objective was safe sharing of ontologies through a policy based framework. Their work is primarily based on RDF technology. The work comprehensively describes the example scenarios which come under defense related applications.

Autonomic computing is an upcoming domain where semantic based policy work is gaining wide attention [69]. Researchers have identified that ontology based solution efficiently addresses the problems of heterogeneity in task requirements, resources and services. They have presented service oriented model for policy engineering as well as dynamic semantic queries. The policy engineering process has been used for proposing ontology based semantic models in order to provide policy resolution and interactions.

2.5 Formal Modeling and Specification

Pi-calculus and Pi-ADL provide major support for modeling and specification of concurrent and distributed applications [11-14, 93]. The formal methods technique facilitates in eliminating inconsistencies, errors in information flow as well as redundant information. Archware toolkit [15] is a European funded project which provides support for verifying and validating formal specifications. The recent advance in this domain is development of pi-ADL.NET compiler [91] for executing and verifying pi-ADL formal specifications in Windows operating system environment.

Researchers have worked on applying calculus for mobile agents [79] in earlier age of this technology but the work was more focused on migration mechanism specification. A formal approach was also exhibited in mapping design and implementation especially in context of agents in [80] but no effective mobility strategies have been discussed or analyzed using formal approaches.

Another Bisimiliarity Checker (ABC) [89] and Mobility Work Bench(MWB) [90] are pi calculus based tools which have been widely used by researchers in order to verify the syntax as well as analyze the semantics of the proposed system. However, these tools have very limited functionality and provides little expressive power for representation of formal specifications and verification .

A pattern language has been introduced for multi agent systems [92] where a form of pi-ADL is used to model and specify architecture especially in dynamic and self adaptive systems. However, there is no specific modeling or formal specification issues have been discussed in relation to mobile agents and its particular requirements of goals and task division characteristics.

Pi-ADL is more suitable for modeling and specifying mobile agents because such type of software agents represents mobility and dynamic characteristics. These requirements and characteristics can be addressed in view of mobile and dynamic architecture as highlighted in [93].

2.6 Agents in Disaster Management Systems

The disaster management systems domain is selected in order to build the prototype application due to its diverse and dynamic nature. Therefore, the formal design of agent based systems is highlighted especially related to disaster management domain in the teamwork context. The concept of software agents has been widely employed for highlighting the model of teamwork [33] in disaster management systems. The examples have been provided in domains of fire fighting and rescue related emergency situations.

The Advanced National Seismic System (ANSS) [70] is a giant stride towards monitoring of earthquake activity specially focused in United States. The work involves increasing number of base stations and communication networks for information monitoring and analysis. The major objective is to acquire information for earthquake related events and its impact analysis on buildings using the latest methods and techniques.

An open source tool named as SAHANA [71] has been developed with a focus on rehabilitation and aid assistance after disastrous events and natural calamities. It is a web based tool which addresses the problems of coordination especially in the cases of missing aid, managing people and volunteers. This system employs manual usage and autonomous features of the system are limited. Other approaches in the domain of agent based disaster management systems include emergency team formation for relief and evacuation as expressed in [72-74]. Also the multi-agent system based planning and communication has been used in [75, 76] for addressing relief operations.

Crisis Information Management system [77] as well as HLA based Multi-Agent System approaches have been used for emergency relief management and operations [78]. However, such techniques do not support proactive response mechanisms for environment variation. More importantly, one thing common with these works is that they are most effective in post earthquake operations and management.

2.7 Summary

In this chapter, an overview of mobile agent paradigm, teamwork strategies among agents as well as ontology based policies is presented. Although, there is very limited work done in teamwork among mobile agents domain, the major strategies of generic teamwork approaches in literature including the hybrid techniques, human agent teamwork model and other unified framework techniques for addressing problems of heterogeneity in teamwork planning and execution were discussed.

The policy based approaches using condition-action rules were presented in various scenarios including pervasive environments, security domain and other network and distributed systems. The ontology based policies concept was highlighted by presenting major approaches in literature for achieving flexible and open architecture solutions. The usage of Web Ontology Language (OWL) was discussed for dynamic and semantic expressiveness. Additionally, the importance of agent based disaster management systems was highlighted in emergency environments keeping in view the teamwork aspects in order to use this domain in prototype application.

In next chapters, the teamwork approaches have been described where team leader and non team leader approaches are discussed. These approaches have been designed and analysed in context of goal definition, task creation and execution. Also, the distribution of operations strategy is highlighted for evaluating the performance of agents in specific teams. In addition, policy based architecture is mapped to relevant ontological solution and presented this architecture for mobile agents in multi-agent system frameworks. Also, Pi Calculus and Pi ADL are used to describe behaviour of teamwork among software agents in order to express its autonomous characteristics in earthquake disaster management. In this particular case, intelligent autonomous software agents are deployed to handle various features including monitoring of seismic activity, information sharing and collaboration as well as data management during earthquake as well as in post management relief activities. The agent based system will also support coordination and timely triggering of emergency services faster than its counterpart human personnel. Additionally, the system takes advantage from the autonomous, proactive and adaptive nature of software agents for efficient performance in dynamic circumstances.

Chapter 3

TEAMWORK IN MOBILE AGENTS

3.1 Introduction

With the increase in complexity of applications and its distributed nature, the need of efficient and effective teamwork among software agents is becoming more critical. Teamwork among software agents can be accomplished using various combinations of team member patterns. However, mobile agents require special teamwork operation strategy due to its mobility and specific task oriented nature.

A teamwork strategy has been proposed and analyzed in context of mobile agents, where a group of software agents is tasked to perform a joint operation. The proposed technique matches with the Honey Bee teamwork pattern in real life analogies, so it is named as Honey Bee teamwork architecture. Two possible scenarios for teamwork have been used to explain the proposed architecture for teamwork among software agents.

3.2 Honey Bee and Mobile Agents

There are a number of similarities exist between working strategy of honey bees and mobile agents when they are performing their tasks in a group. The resemblance has been highlighted between working strategy of honey bees and mobile agents. It forms the basis of design for proposed teamwork architecture.

In the scenario of honey bees network, the queen bee leads the hive and manages as well as coordinates the major operations. All other bees provide various kinds of services to the queen. In mobile agents' domain, team leader approach is proposed which manages, shares goal and plans as well as coordinates and monitors the operations of member agents. The team leader divides the top goal and plans the sub-goals for members and manages their operations and performance by information sharing, coordination and collaboration.

Honey bees roam around from flower to flower for fulfilling their major goal i.e. extracting nectar for making honey, which is later stored in the hive. Honey bees extract the nectar from many flowers as they fly. Bees retain it, return and release it at their hive at the designated cell. Similarly, mobile agents may visit various machines in order to fulfill their assigned tasks like searching for specific information. They extract the

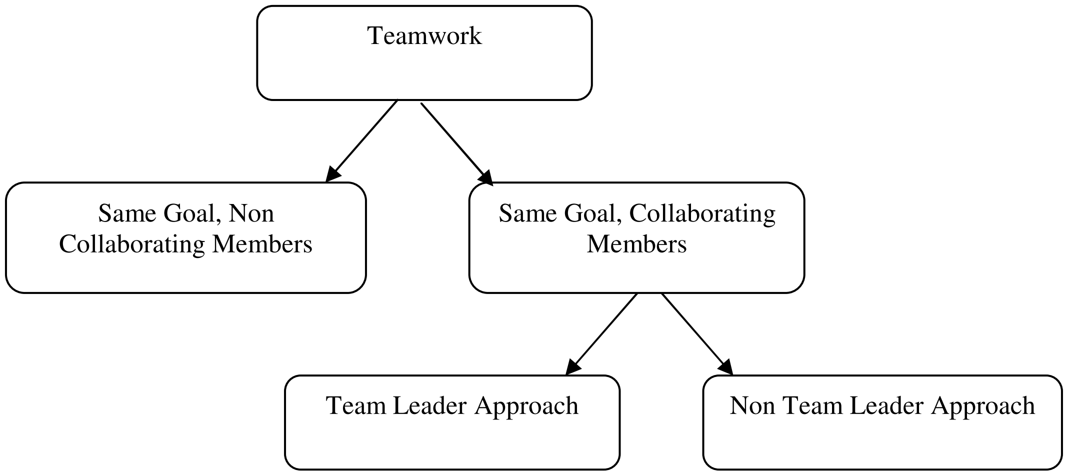

Teamwork
Same Goal, Non Collaborating Members
Same Goal, Collaborating Members
Team Leader Approach
Non Team Leader Approach

Figure 3.2 Classification of Teamwork Model

required data and process accordingly. Mobile agents may keep the extracted information or manipulate at destination machine and deliver result after returning to their parent machines or hosts.

In case, the queen bee is dead, there exists a well defined process for finding and making a new queen. A special food is fed to a particular bee; this process brings up the new queen. The queen adopts the role of new leader of the hive. In the context of software agents, fault tolerance mechanisms have been introduced for generating new team leader in case of malfunctioning. A promotion algorithm has been used in which the member agent is elevated as team leader depending on its seniority of time scale.

3.3 Teamwork Architecture

The general teamwork architecture is classified as shown in Figure 3.1. The teamwork architecture may be divided into two main categories. Firstly team of members, which are working for the same prime goal but there is no direct collaboration and communication between them. In such type of teamwork, members are not aware about their peer or neighboring members. The only thing identical among them is their prime goal which is being pursued by each member.

Other category in teamwork shows working of member agents to fulfill primary goal under identical plan. Such members coordinate and communicate with each other as per requirement of goals and operations strategy. This strategy is also sub-divided in two more categories. The first sub-category consists of team leader approach where a specific member is designated as leader and it is responsible for goal and plan division to its team members. In this approach, all information is shared through team leader among member agents of team. The second sub category is Non-team leader approach in which all members of team have direct communication with each other at peer to peer level without specifying a specific member as team leader.

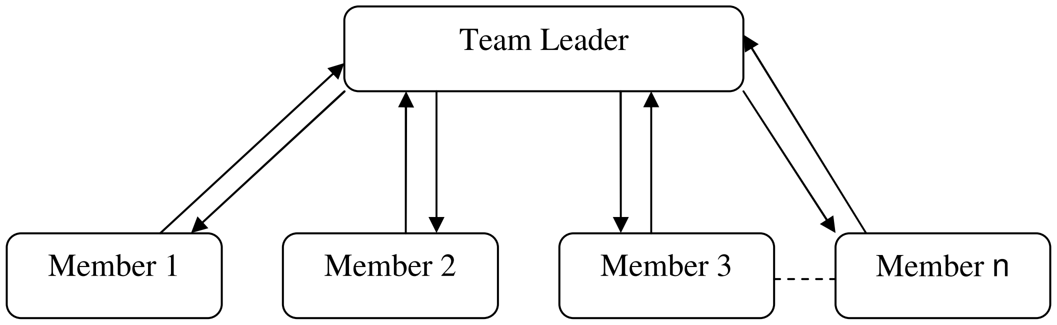


Figure 3.2 Interactions among Agents in Team Leader Approach

The specialized team leader approach has been proposed for mobile agents as inferred from honey bee working strategy. A comparison of both approaches is discussed in later part of the section along with rational for assigning priority to team leader approach over non team leader strategy.

The major terms are defined as following,

Goal for team members – g

Task to be performed – tk

Team leader agent - T_l

Team member agents - T_m

Communication factor in team leader approach - Cf_{TL}

Communication factor in non-team leader approach – Cf_{NTL}

3.3.1 Team Leader Strategy

In the Team Leader scenario, there is one dedicated agent which is leader of the team and its responsibility is to allocate the goal or plan as well as tasks operation strategy to each member in the team as per specification. In this approach, collaboration among member agents occurs in a hierarchical fashion. It transpires that collaboration among member agents of specific team is taking place by way of team leader. Such technique allows convenient sharing and integration of critical information among members of team in mobile agents' scenario. In this particular strategy, team leader agent shares the goal and tasks related information to its member agents in a tree like hierarchical structure as highlighted in Figure 3.2. The communication in this particular technique can be represented as 2n for n>1, where n is the number of member agents in a specified team.

The elaborated flow of activities in this algorithm of teamwork is described in Figure 3.3. At the start of application, user or owner of agent specifies the number of agents to be created in a team. After specification of team structure and primary goal, system creates the team leader and member agents. Team leader adds the members into its team and sets the communication pattern. The team leader divides the primary goal and its associated tasks into sub-goals as well as sub-tasks depending upon the specific number of member agents in the team. Team leader assigns these sub-goals and sub-tasks to members and execution is started. In this approach, all communication takes place through team leader agent which is coordinating and sharing all concerned information with members. As

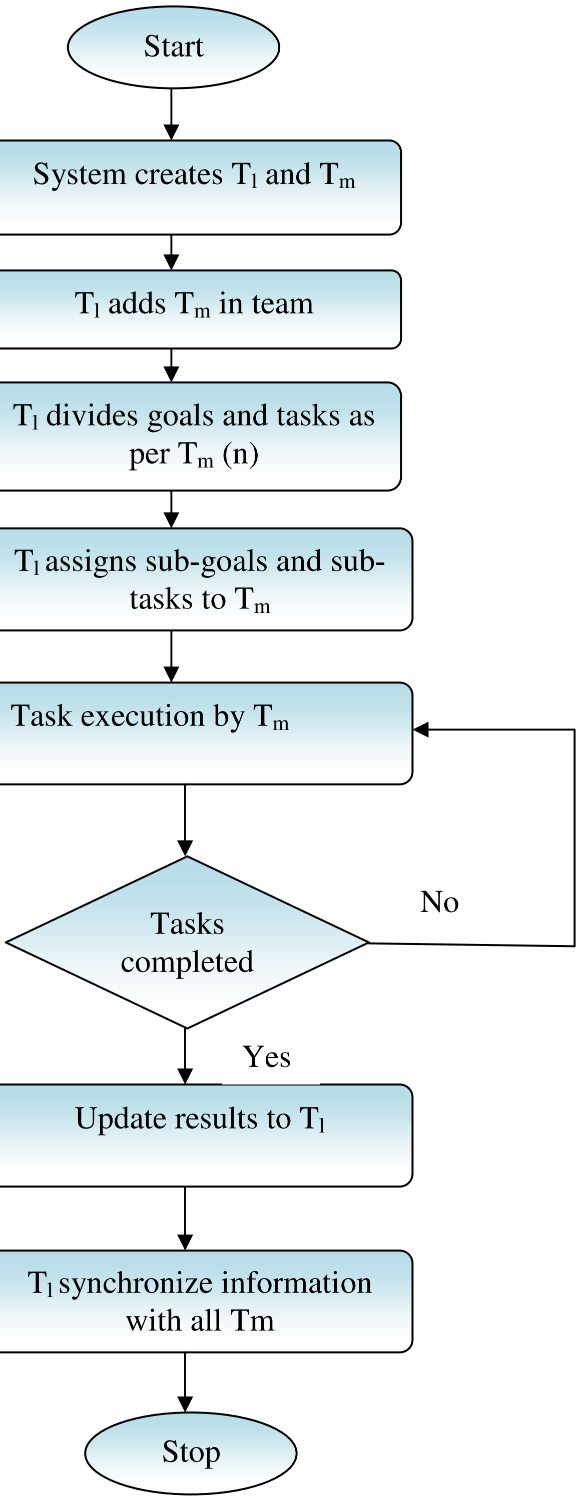

Start
System creates T_l and T_m
T_l adds T_m in team
T_l divides goals and tasks as per T_m (n)
T_l assigns sub-goals and sub-tasks to T_m
Task execution by T_m
Tasks completed
No
Yes
Update results to T_l
T_l synchronize information with all Tm
Stop

Figure 3.3 Team Leader Strategy

soon as the main goal is achieved and specified tasks are accomplished, team leader shares and synchronizes information with members and operation is concluded.

In domain of disaster management systems, the resource/target hunt scenario is used for proof of concept in this particular case as it is widely used example in the domain of teamwork. It is one of the major scenarios where robots teamwork strategies and their communication and coordination patterns are analyzed [81] by renowned research groups.

In our first example of target/resource hunt application scenario, the team leader agent and member agents are created by the system. Team leader agent assigns the prime goal data and area under consideration to each member agent for searching the required target location. In case of two agents in a team, the area under search is divided into two parts and allocated to team member agents. In case of four, eight or sixteen members in the specified team, the allotted area is further divided into respective parts and each member agent is assigned its area of search. The team leader agent adds the member agents in its team at the start of activity and later tracks member's progress as well as path of locations. When the target/resource is discovered, the team leader agent informs all member agents in its team and the search operation is concluded.

A particular job n execution in team leader scenario can be represented in tuple form as

$\{ T_l, T_m, g_n, tk_n, Cf_{TL} \}$.

The communication factor comparison is highlighted in evaluation section for further analysis and discussion about its effects on generic teamwork strategy.

3.3.2 Non Team-Leader Strategy

In Non Team Leader scenario, there is no team leader who divides the plan or set of actions for each member agent. The general plan or primary goal is shared by members at peer to peer level. The members work themselves according to main goal and plan in direct collaboration and coordination with each other.

In the Non Team Leader strategy, each agent in the team forms direct communication link with each other. In terms of implementation perspective, each member has two streams of communication; one is input and other output. The communication among agents in non team leader approach is much higher as compared with the team leader approach as shown in Figure 3.4, especially for team of two, three and four agents.

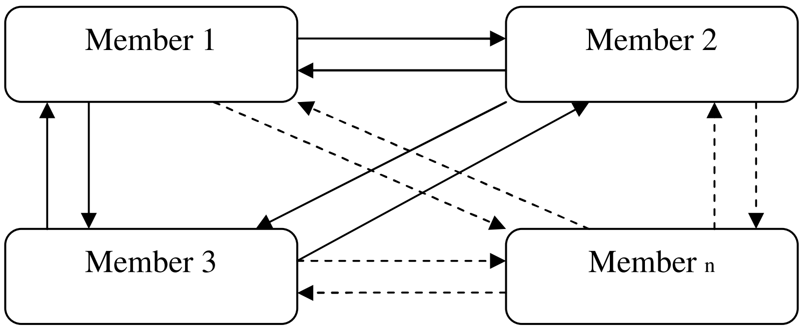


Figure 3.4 Interactions among Agents in Non Team Leader Approach

The flow of activities is also highlighted in Figure 3.5. In non-team leader approach, the system creates the team members at startup of application. Team members share the goals and tasks information with each other and synchronize their status. Members start execution on defined tasks while sharing its progress and current situation with peer members. As soon as the primary goal is achieved, information is synchronized and operation is stopped.

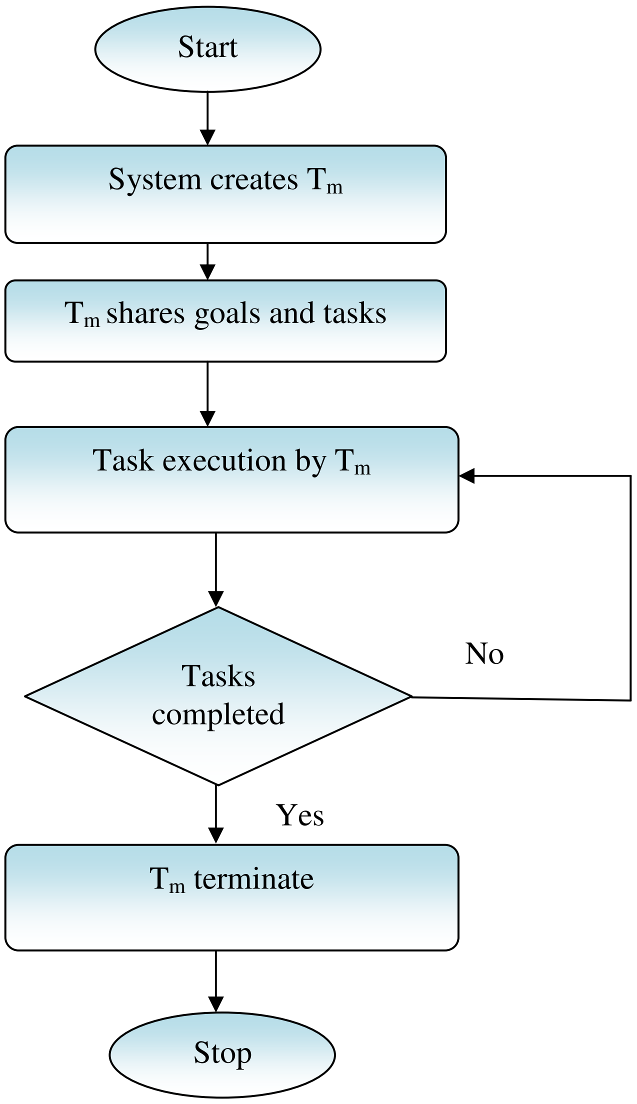


Figure 3.5 Non-Team Leader Strategy

Considering the proof of concept application of target hunt, the members are collaborating and sharing information with each other while searching the area for target/resource. They start searching the area on their paths and at the same time communicating with each other about their current location. If two members come face to face with each other or come across in same path which is already traversed by other agent, they change their direction and move towards the un-explored paths. When one of the agents finds the target, all member agents are informed and search operation is stopped.

As the number of agents increases in a team, the communication also increases respectively. The communication relationship in this case can be represented as n(n-1), where n is the number of member agents in the specified team.

A particular job n execution in non-team leader scenario can be expressed in tuple form as

$\{ T_m,\ g_n, tk_n, Cf_{NTL}\}$

3.4 Teamwork in Earthquake Management System (EMS)

Teamwork approaches have been expressed in multiple disaster management systems with focus on teams of rescuers in fire or explosion related disasters. We have used the example of earthquake management system to highlight the potential capabilities of software agents from start of emergency activities like sending alerts to monitor and manage the relief operations. This particular example is also discussed in upcoming chapters of thesis, highlighting the role of ontology based policies and formal modeling perspectives.

In EMS scenario as shown in fig 3.6, the main station and field stations are signifying the team leader strategy. As highlighted in the earlier example of team leader approach, the major information flow among member field stations is occurring through main station in order to minimize the communication overhead. In case of malfunctioning of the main station agent, one of the field stations is selected as leader station using the promotion policy. It is then used for information sharing, monitoring and management of tasks.

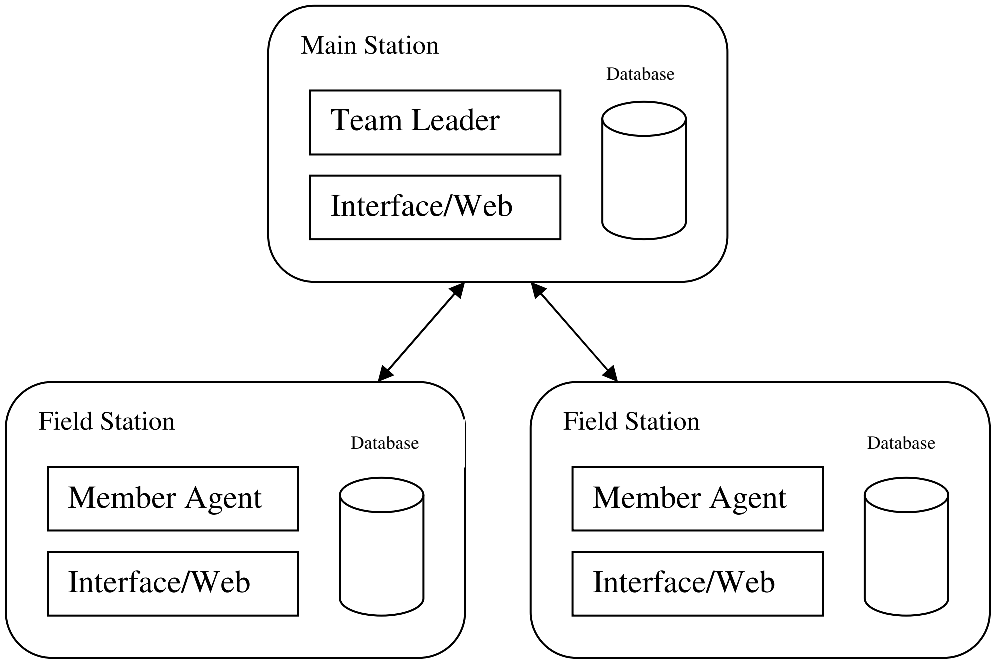


Figure 3.6 Interactions between Central and Field Stations

This EMS architecture design also forms one of the contributions and is discussed under major scenarios of proposed work. The major design features of Earthquake Management System are discussed in chapter 5 while the related implementation part is highlighted in chapter 6 under context of ontology based policies.

Chapter 4

SEMANTIC POLICIES

4.1 Introduction

In this chapter, a detailed description of system architecture regarding ontology based policies work is described. The research involves three major components. Firstly, the structure of policies is highlighted with its domain characteristics and classification. Afterwards, the role of ontologies and its vital association with policies is mentioned. The domain of disaster management systems with emphasis on earthquake management systems has been used as proof of concept application.

4.2 Role of Policies

Policy structure is classified as shown in figure 4.1 into three major types. Obligation policies cover all the security related and permission oriented policies in which majority of research is related to traditional policies concept. Management policies work is related

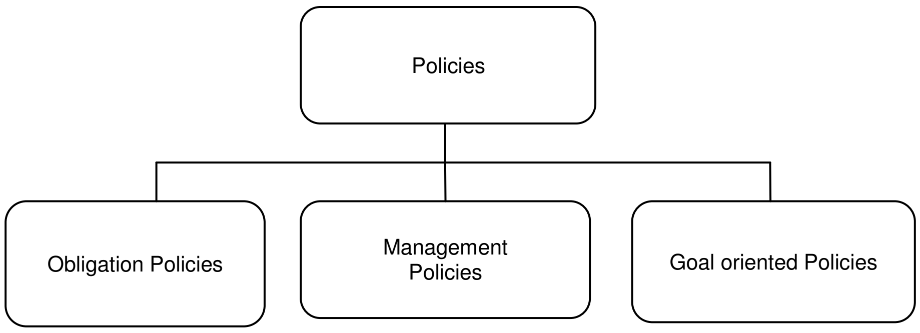


Figure 4.1 Policy Classification

to system management issues for example if there is missing code or program to open and execute a file, then it can be loaded from specific repository. However, the policies have much more potential and scope than they are currently being considered and used. In this thesis, the need for goal oriented policies has been highlighted. Such policies comprise of individual tasks and contribute towards a particular goal. These are later represented in ontologies form and its role has been highlighted, especially in mobile agents scenario.

4.3 Goal Oriented Policies

Goal oriented concept is gaining recognition in requirements engineering [42] where this approach is used for various phases including requirements elicitation, negotiation, specification and validation. The major issues are how goals can be represented and used under dynamic context. However, in this particular research work, the ontology based

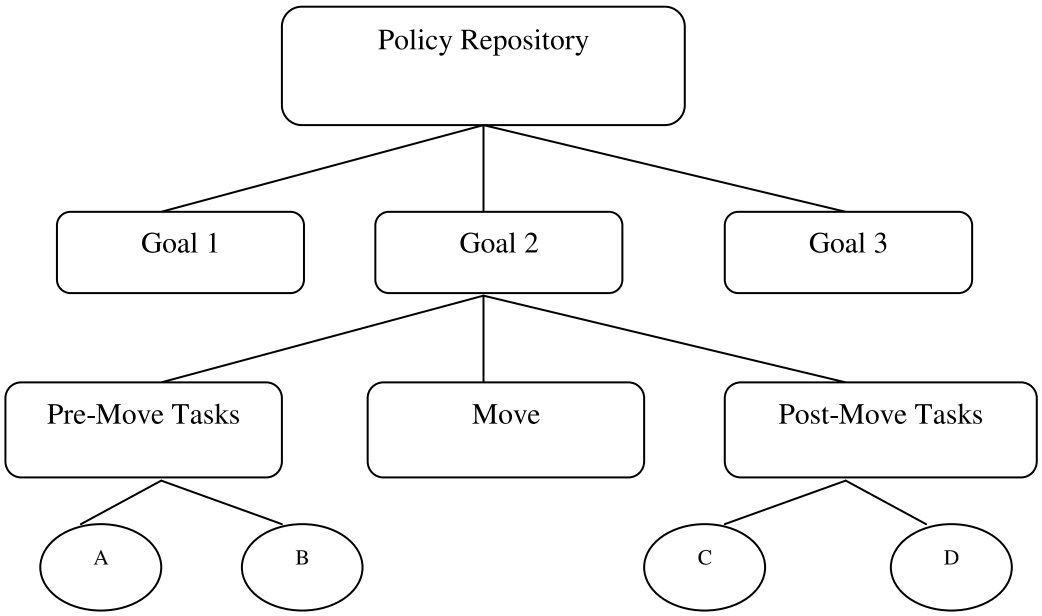


Figure 4.2 Policy Structure

goal oriented policies technique is used to address the issues of dynamic sharing of information and improvement of task execution mechanisms. Policies consist of tasks which need to be executed before movement of mobile agents, during movement and after movement at destination machine referred to as pre-move, move and post-move tasks. These actions or tasks are arranged in logical order and each set is attached with related sub-goal which is ultimately associated with primary goal and contributes towards its accomplishment. The users or developers of the system can define and associate these tasks with each goal while setting the policy structure of the desired domain application in multi agent system context. A general policy based structure is shown in Figure 4.2. A policy repository contains a number of goals and each goal is attached with sub-goals which are related to set of tasks, i.e. Pre move, Move, and Post move tasks of mobile agents. Pre-move and Post-move tasks consist of individual sub-tasks. In Figure 4.3, general policy template is shown where A, B, C, D are the individual tasks. Once defined and developed, the individual tasks may be later reused and customized according to required domain functionality.

On Goal: X
Pre-move tasks: A, B
Sub-tasks: A_1, A_2, B_1, B_2
Move
Post-move tasks: C, D
Sub-tasks: C_1, C_2, D_1, D_2
Where A, B, C, D are individual tasks

Figure 4.3 Generic Policy

The user on whose behalf a mobile agent is destined to execute, specifies its goal dynamically at run time. Firstly, the system matches the goal given by user with the goals which are mentioned explicitly in the policy repository. If a goal, which is mentioned by user matches with already defined goal in the repository, the associated sub-goals and tasks are loaded autonomously. Each pre-move task needs to be executed before movement of mobile agent from source to other destination machines. Each sub-task is picked one by one and its definition is loaded from list of tasks class. This particular class includes the implementation aspects of each task. After all the pre-move tasks are executed, the control is passed to move task, which executes the migration of mobile

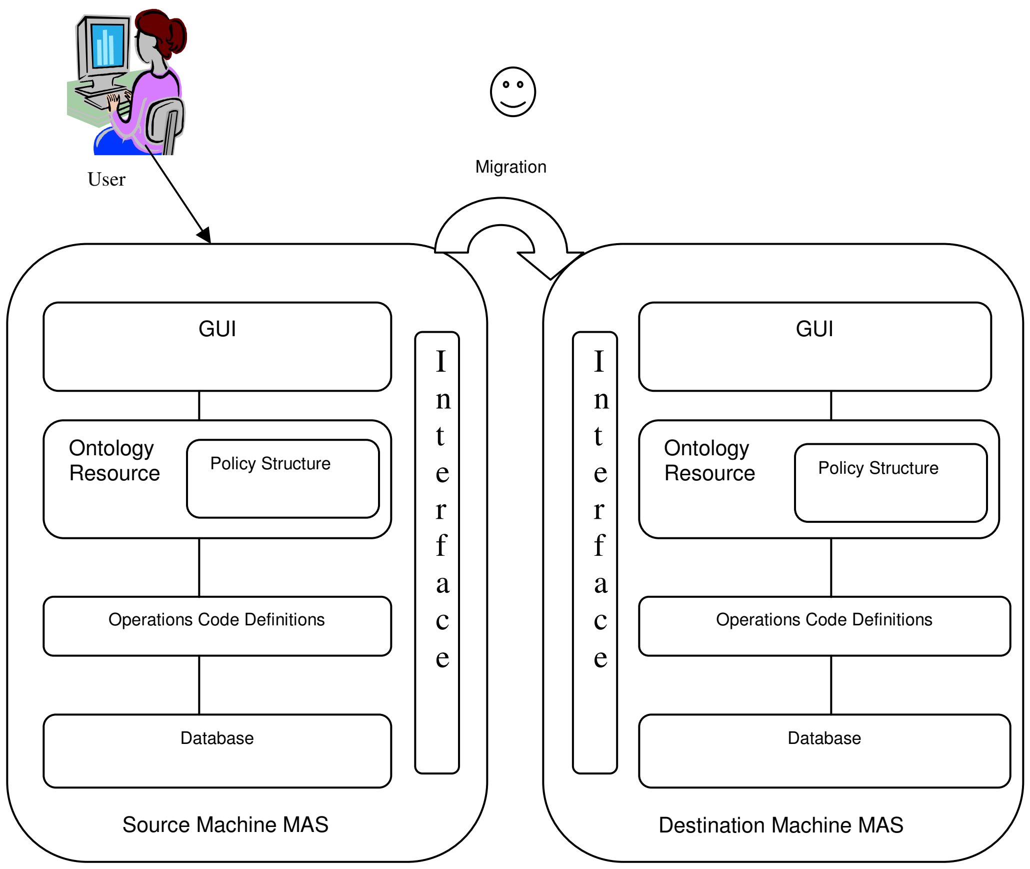


Figure 4.4 Generic Architecture

agent on selected destination machine while taking IP address of target as the key input. Then control goes to post-move list and checks any sub-tasks to be executed. Post-move tasks contain those tasks which need to be executed after the movement of mobile agent on the destination machine.

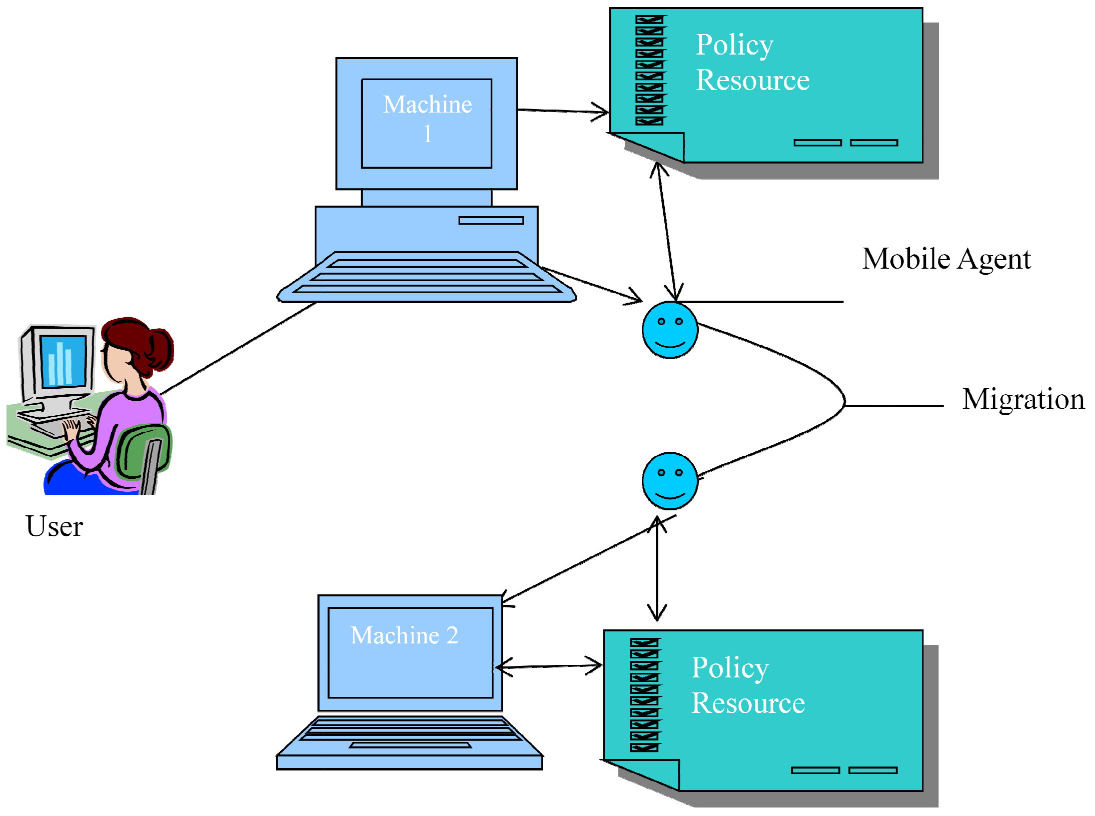


Figure 4.5 Policy sharing mechanism

In the second case where the goal entered by a user is a new goal, which is not defined in existing resource, the user is asked to elaborate its structure in terms of sub-goals and actions by defining its associated sequence of tasks for execution in various phases. The user can enter its new goal and customize its sets of sub-tasks particular to domain functionality and add it to the goals repository. This defined goal could be later reused to save user time and efforts in defining the same job repeatedly.

4.4 Ontology-Policy Architecture

The ontology based policies are distributed and shared on multiple machines. In case a mobile agent has to carry its entire task-related information and sets of conditions and actions with it, it would increase its size and affect its performance and execution. With the advantage of sharing and distributing such policies resource structure, the mobile agent can invoke its associated tasks definition at run time even after migrating to its target machines according to its specified goal. The proposed architecture which includes policy sharing and interaction mechanism for mobile agents is shown in Figure 4.4 and Figure 4.5. At start of itinerary on source machine, a mobile agent retrieves the required set of information relevant to its assigned goal from policy resource and executes the associated tasks. After moving to destination machine, it invokes the remaining set of tasks and the required information, which are part of that particular assigned goal activity but need to be run on destination machine after migration.

Web Ontology Language (OWL) is selected in this particular case because it is a standard approach adopted by W3C [8]. In comparison to FIPA ontology [83] approach, OWL structure provides more efficiency and flexibility. Also, it is more suitable in proposed work scenario of mobile agents.

In the ontology based policy architecture, the notion of goals is signified as class in OWL terminology. Similarly the pre-move, move, and post-move tasks are also expressed as OWL classes. The subtasks or specific values in goals and tasks category have been expressed as OWL individuals in the particular classes. The implementation procedure or related coding part of each individual task is mapped and expressed to an independent java file. The links or associations in a particular policy structure are expressed as properties in OWL terminology. The prime benefit of employing OWL in this case is publishing of proposed ontology structure online on the web. In scenario of distributed deployment of multi-agent system on various machines, mobile agents can interact and retrieve the required information more conveniently. Additionally, this architecture allows persistent storage of information as compared to orthodox techniques of saving dynamic run-time information in hard-coded form or databases. In this architecture, the system policies are expressed in ontology form and software agents extract the goal and task structure for execution of related activities in the designated style.

In ontology based working scenario, when a goal is entered by user at source machine, the system verifies the goal as already existing or newly set by user. It extracts the pre-move sub-tasks from ontology resource which are implemented in form of goal oriented policy structure. The system loads the code implementation details of individual tasks and executes it. After executing all the pre-move tasks, the mobile agent migrates to target host with assigned goal information. If the destination host possesses same prime goal specifications, mobile agent invokes the associated post-tasks from local ontology repository. The execution of such tasks is initiated after loading code from associated implementation class files. It executes the list of tasks which are associated with particular assigned goal. In case of new or unknown goal, the destination host upgrades the local ontology repository after synchronizing with source machine and executes the relevant code.

This framework is more focused for goal oriented task related policies; however it could also be used for obligation and management policies domain, like in order to control the movement and interaction pattern of mobile agents. The administrator of multi-agent system can use such policies structure to control and restrict the movement of mobile agents to specific locations. The administrator can either block certain locations for mobile agents to migrate, or create a mechanism to block the execution of specific activities at source or destination machines.

The OWL ontology is accessed by mobile agents by providing URI and creating the respective Owl model through Jena APIs as mentioned in following sample statements.

```
String uri = "http://localhost/policyontology.owl";
  OWLModel owlModel = ProtegeOWL.createJenaOWLModelFromURI(uri);
```

4.5 Summary

In this chapter, the classification of policies as well as proposed architecture of ontology based policy work is presented. The structural hierarchy has been shown along with concept of goal oriented policies approach to handle complexity of distributed applications. Additionally the detailed architecture of tasks division and hierarchy as per specific primary goal is discussed to highlight the potential of enriching mobile agents' capability through simplification of its goal and tasks structure. Also, ontologies have been used to represent the policy structure consisting of individual goals and tasks. This adds further flexibility during runtime operations of mobile agents in dynamic online and distributed environment.

Chapter 5

ROLE OF AGENTS IN EMS

5.1 Introduction

In this chapter, the design and analysis of Earthquake Management System application is highlighted using formal approach, which is proof of concept application in the proposed research work. This section particularly describes the use of novel way of applying specific formal techniques in designing software agent based systems and contribute towards main research work especially in Autonomous Decentralized Systems (ADS) [85, 86] context of autonomous coordination and collaboration.

5.2 EMS Application

In the proposed EMS application as shown in figure 5.1, the field workers are linked to respective field stations which in turn connect them to main station for information sharing and synchronizing. The utilization of software agents' approach is more appropriate in distributed and dynamic environment. In case of less or no coverage area, software agents which are executing on PDAs can assimilate and disseminate required data autonomously as soon as the connection is restored. The information can also be synchronized with other member stations through the main station.

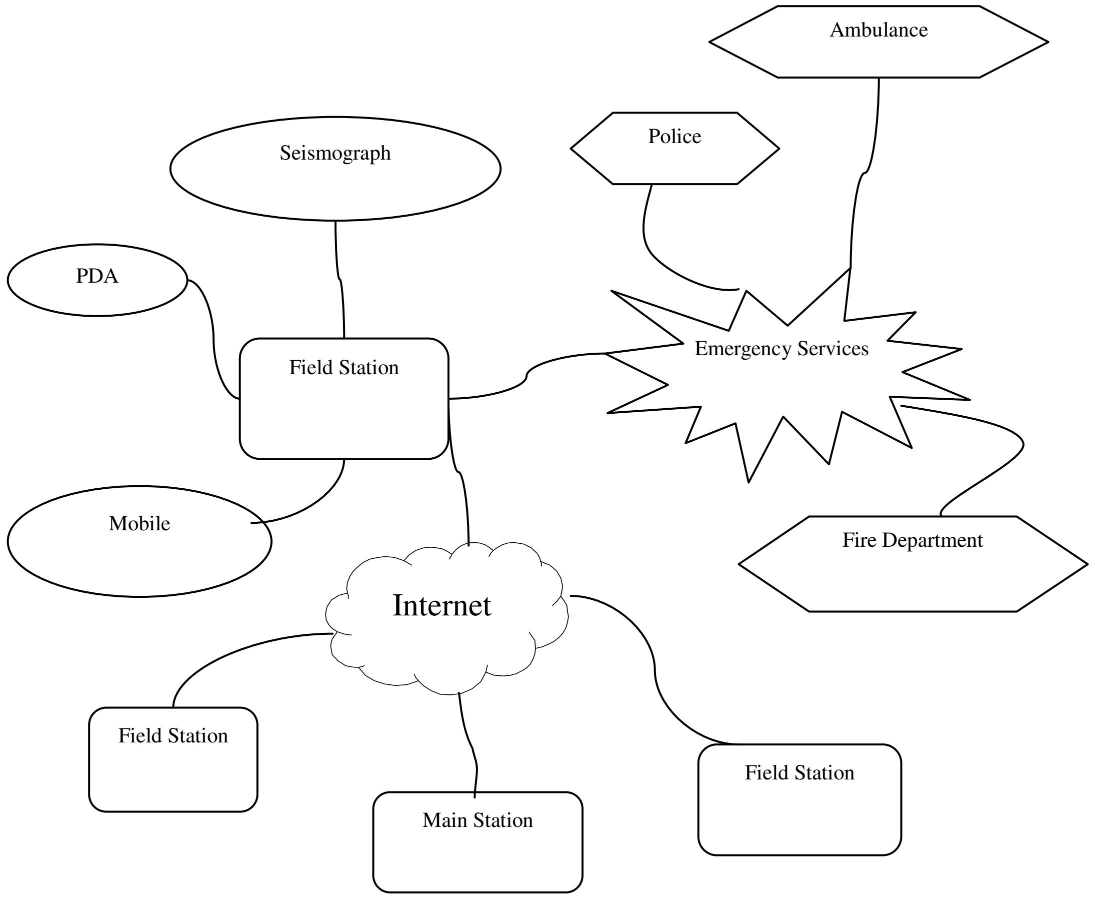


Figure 5.1 EMS Overall Design

The generated data is communicated among software agents residing on PDAs or base stations autonomously through direct communication channels. The information about disaster or record of causalities may be communicated for general use by publishing it online on web pages. The software agents autonomously retrieve and disperse the necessary data efficiently in coordination and collaboration with critical entities. The existing systems in disaster management domain are installed in widespread locations for monitoring and evaluating the scale of disastrous event. The incorporation of autonomous agents' paradigm is anticipated to be more suitable and appropriate technology for this particular domain.

5.3 Formalizing System Architecture

The EMS architecture is formally proposed and analyzed which autonomously support various features of typical disaster handling systems. It facilitates in generating alerts, information delivery to respective relief agencies and communication among various entities in the event of earthquake as discussed in Chapter 4.

In emergency situations, the slow response of human operators causes delay in communication and coordination. The use of agents is proposed for effective and efficient performance. Considering an example, if the reading from a seismograph goes above some predefined benchmark value, the agent associated with it triggers the emergency services by sending the alarm messages to other software agents, which are deployed in various services related departments. These software agents also share information and coordinate activities among various departments. It shares the essential information among integrated systems as well as makes it available for persistent storage and update in database.

In case of request of some particular information, software agents perform the tasks of human operators in delivering necessary data and service support relevant to earthquake related activities. The specialized agents retrieve the required information by querying the associated databases and update the outcome to desired recipients. This approach exemplifies to be efficient strategy of distributed data searching and retrieval.

5.4 Operation Activity

In the proposed software agents based EMS architecture, the following key agents have been considered as shown in figure 5.2 in order to carry out various operations. Information Service Agent, Field Service Agent, Personalized Service Agent, Emergency Service Agent, Personal Assistant (PA) and database representing agent entity.

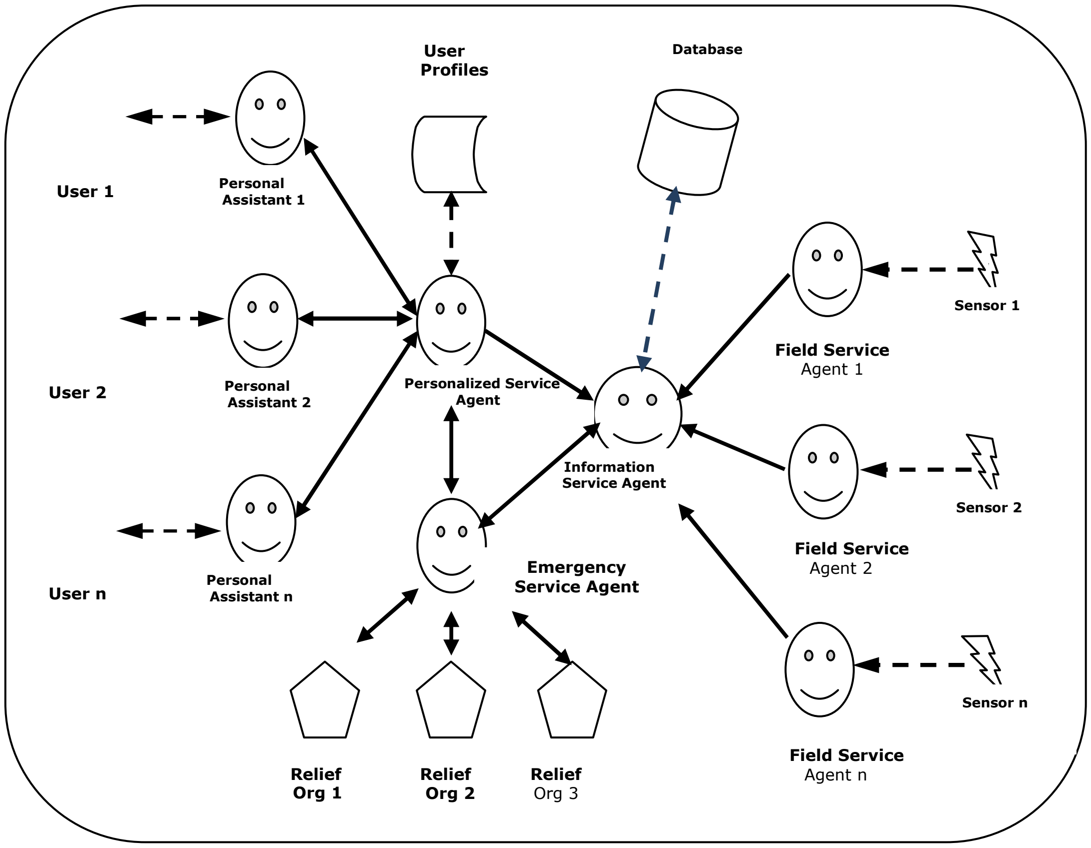


Figure 5.2 EMS Architecture

The Field Service Agent (FSA) plays a critical role in monitoring of seismic waves scale and delivering the updated data to the Information Service Agent (ISA). It also issues alerts in case current seismic value crosses the specified benchmark value as obtained from seismograph. The Information Service Agent analyses the received data and invokes the next strategy to handle the event. If severe earthquake activity is confirmed, the alerts are sent to Emergency Service Agent for further actions.

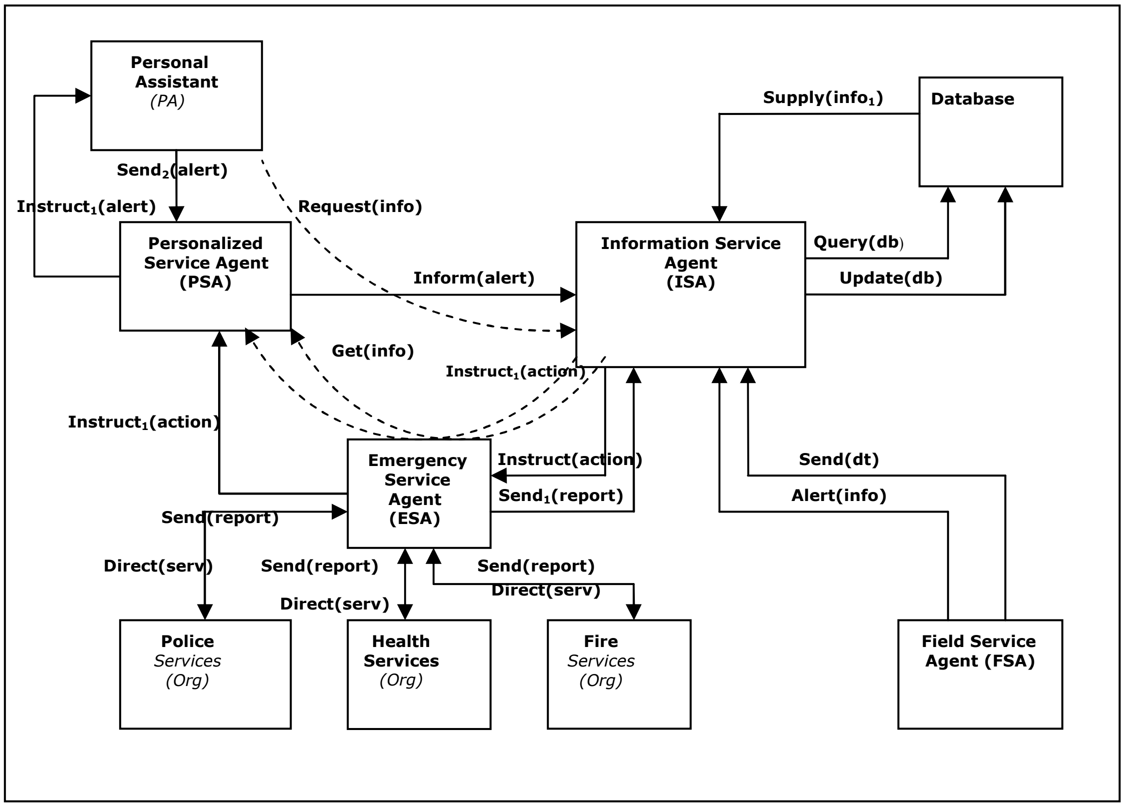


Figure 5.3 Agents in EMS and Communication Channels

The Emergency Service Agent coordinates and collaborates with emergency response and relief organizations like police, fire stations and ambulances. As per information and orders received from Information Service Agent, necessary instructions are passed to such relief services. There are special kinds of seismic waves which are generated at the start of earthquake activity as well as during its peak time. These waves are identified as “P and S waves” [87] by the geologists. The field service agents (FSA) can detect generation of such waves and issue alarm messages as early warnings for evacuation.

The bold links in figure 5.3 shows the direct communication channels among the various system agents. The dotted links in the above mentioned figure highlights the communication through the transitional agents. The emergency response agencies including fire, ambulance and police organizations are communicated by Emergency Service Agent (ESA). The operation tasks are assigned by Emergency Service Agent (ESA) according to predefined plan. The results and feedback are shared with Information Service Agent (ISA) for upgrading of data and strategy in the database. The detailed formal specification and verification of EMS application is presented in chapter 6.

5.5 Summary

In this chapter, the detailed description of earthquake management system (EMS) is discussed. The overall working scenario is shown highlighting the need and importance of agent based systems usage in such disaster management domain. The division of tasks according to structure of software agents has been made to balance the independent working of each agent in its respective domain. Lastly, the detailed channel and information flow among agents has been highlighted.

Chapter 6

IMPLEMENTATION

6.1 Proof of Concept Application – EMS

The design of Earthquake Management System (EMS) is based on software agents for monitoring and management of activities from start of earthquake to relief work. In chapter 5, the overall design of earthquake management system (EMS) was expressed in detail.

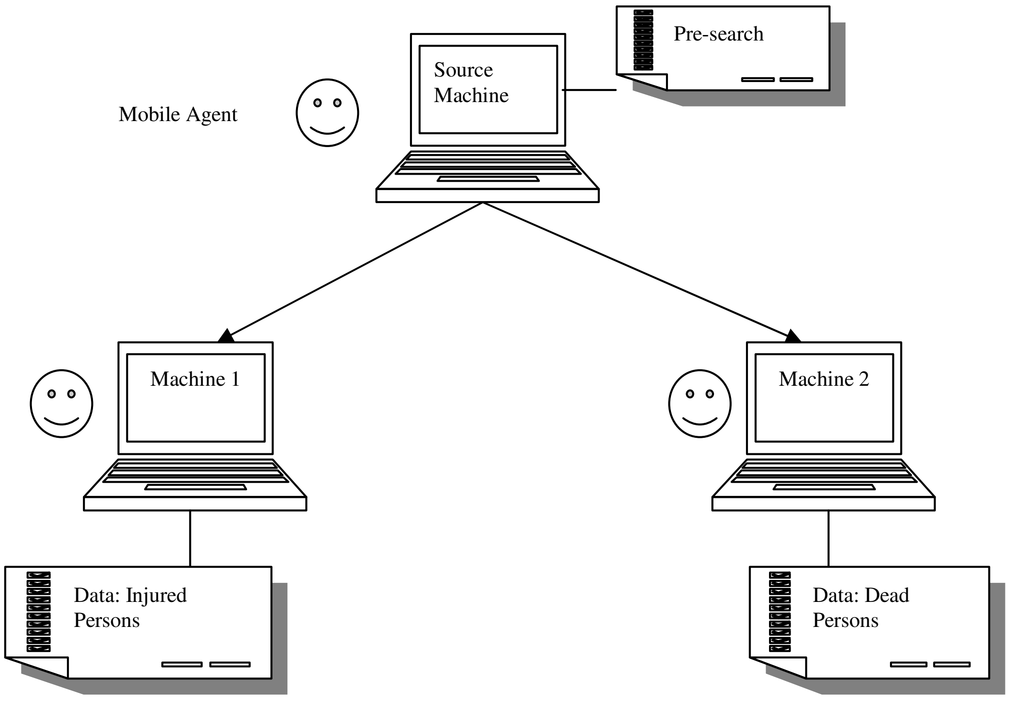


Figure 6.1 Information Searching and Retrieval in EMS

An example scenario is discussed in the following part as one of the component of the proposed Earthquake Management System (EMS). A proof of concept application has been designed and developed in which an agent searches, moves and extracts the required data about a specified person. The mobile agent moves to connected machines as per assigned goals and tasks. This application exemplifies the scenario of data searching and extraction in distributed architecture through ontology based goal oriented technique.

The mobile agent related operation is highlighted as proof of concept scenario. The data about injured and dead persons is placed separately on two machines as shown in Figure 6.1. In the proposed scenario, an agent executes a pre-search task as a starting activity. This task represents an operation under pre-move activity at the source host. The mobile agent checks the current information of the specified person and determines its status either injured or dead in the recent earthquake activity. The mobile agent migrates to the associated destination host for further query execution. The target host possesses the detailed information of causalities including the name of individuals, ID number and resident city data.

The multi-agent system generates the mobile agent. The users or owners of the agents enter the primary goal “Search” in the current example. The system will check the associated tasks with the specified goal and extracts the required sequence of tasks/operations to execute. In this particular case, the pre-move activity is “pre-search” operation in which agent firstly searches local source machine to extract data about specific person whether injured or dead. According to existing information, the concerned agent migrates to the target host. The mobile agent identifies the required table and executes the SQL query for information retrieval. Mobile agent explores the related database and retrieves the assigned goal related information.

```
On: Earthquake
Tasks: Alarm, Identify, Data_Update*, Query*
On Goal: Data_Update
Pre-Task: Get_Info, Identify_Machine
Move
Post-Task:Update_Data
On Goal: Query
Pre-Task: Pre-search
Move
Post-Task:Opsearch

*represents the tasks need to be executed by mobile agents
```

Figure 6.2 Policy Example in Prototype Application

6.2 Implementation

The proof of concept application is implemented using SAGE [82, 84] multi-agent system. The ontologies have been implemented in OWL using Protégé [9]. Afterwards, Protégé as well as Jena [10] APIs are used to interact and extract the specified ontology structure by mobile agents.

The policy-setup is invoked after booting the multi-agent system and before creation of mobile agents. In this mechanism, the tasks/operations are implemented individually using java language and saved in the application code specific file. These tasks are later picked one by one for placing in the policy structure. Similarly, these tasks/operations are picked in the defined order and executed by calling its defined code from the source code specific file.

Figure 6.2 highlights the layout of tasks and sub-tasks as well as its relationship and structure of execution flow in case of earthquake activity. The primary tasks or operations in the proposed scenario include the following major activities.

(1) Alarm – This process issues the alerts and alarm messages for the concerned emergency response services like fire department, ambulance and police agencies.

(2) Identify – This activity traces out the related seismograph in the specific area and field station from where seismic activity is originating.

(3) Data_Update* – In this process, the agents update the upcoming data from earthquake hit area among attached field stations and departments.

(4) Query* - In this activity, the agents make queries and extracts the data from related field stations or services.

The last two activities have been considered as major tasks as it involves the use of mobile agents for performing the desired actions. The "Data_Update" task includes the activities of receiving the input data, identifying and migrating to the associated target host for addition or update of data in the pre-defined structure. The specified agent moves to the destination host and executes the sub-task in the specified order.

In the "Query" task, the user assigns the prime goal as "Search" to mobile agent in GUI. This goal is matched with the pre-defined goal definitions in policy repository. If it matches the pre-specified goal, the user is notified of loading pre-defined tasks structure and implementation details. In the Pre-move task category, there exists one sub-task defined as “Pre-Search” in the current example. While executing the “Pre-Search” activity, mobile agent executes the search operation at source host for retrieving the necessary information before moving to next destination machine.

On completion of all the Pre-move tasks, the system picks the Move task. In, the “move” activity, a request is generated to multi-agent system for migration of mobile agent to next destination machine. The multi-agent system takes the destination host data primarily the destination IP address and the mobile agent moves to the target host. In the Post-move category, “OpSearch” operation is mentioned in the current example. The OpSearch operation includes the activities of executing the search operation at target host by making queries as per required parameters. The mobile agent retrieves the data and sends the results to the parent host.

In the example scenario of EMS, the mobile agent executes “Pre-Search” operation at source host and finds out the recent information status of particular person as “dead”. The agent picks the attached target host information on which the detailed record of dead persons is kept in form of database. The mobile agent migrates to the specific target host.

On reaching the destination machine, the mobile agent executes the “Opsearch” task. The agent executes the search operation by interacting with the database deployed locally. It retrieves the specified person’s data including his ID and resident city information.

Afterwards, the agent returns the results to the source machine. Such combination of individual sub-tasks which are present in tasks hierarchy fulfills its sub-goals and ultimately contribute towards the key goal of application.

6.3. Formal Model using Pi-Calculus

6.3.1 Pi-Calculus Notation

Pi-calculus is a mathematically defined formal method technique which is employed to model the processes and information flow. Pi-calculus supports the representation of parallel execution of processes as well as its communication channels [11-14]. The syntax or notation comprises of primarily prefixes for formal modeling of information channels and processes.

In the Pi-calculus terminology, a Process is an independent thread of control or execution entity. Additionally, a channel represents the connection between the two processes for information exchange [14]. The communication takes place by sending and receiving messages over the inter-connected channels. An overview of the pi-calculus syntax is mentioned below, where Table 6.1 highlights the basic syntax for two example processes expressed as P and Q. Further detailed information about Pi-Calculus is available in various resources [11-14].

Table 6.1. Syntax of Pi-Calculus

Process	**Representation**	**Explanation**
Empty	0	The process where no action takes place.
Parallel	P|Q	Two processes P and Q running in parallel with each other.
Output	$\overline{a}\langle x\rangle \cdot P'$	A process which sends message x over a channel "a" and behaves as process P` afterward.
Input	$a(x) \cdot P'$	A process which waits on channel "a" to receive a value bound to variable x and behaves as process P` afterward.
Non-deterministic choice	P + Q	The process where either process P or Q executes.
Repetition	!P	Infinite number of process P running in parallel.
Match	[x = y]P	A process which behaves as P provided x and y are the same. Otherwise nothing happens.
Restriction	(vx)P	A process which behaves as P where x is a local channel and used only for communication among processes

		within the scope of P.
Silent Action	τ	Nothing observable happens, i.e., action without interaction with environment.

6.3.2 Agents Description

The roles of above mentioned agents are described in this section by highlighting the major functions and responsibilities.

As referred in figure 5.2, the agents and their interactions are modeled in the system. The roles of various agents are described in this section by highlighting the major functions and responsibilities.

6.3.2.1 Field Service Agent

The field service agent (FSA) monitors the seismic waves which originate from a specific area in connection with locally attached seismograph. The field service agent issues the alerts and alarm messages to information service agent if the scale measure crosses a pre-defined benchmark. The Field Service Agent uses the communication channels of "Send" and "Alert" to send information from seismograph to Information Service Agent for necessary action. Information Service Agent receives the information and interacts with database using update and supply channels.

$$FSA \stackrel{\text{def}}{=\!=}$$
$$(!\overline{Send}\langle dt\rangle \cdot \overline{Alert}\langle info\rangle \cdot FSA' \mid Send(m) \cdot$$
$$Alert(n) \cdot \overline{Update}\langle m\rangle \cdot Supply(info_1) \cdot ISA')$$

6.3.2.2 Information Service Agent

The Information Service Agent (ISA) shares the information from the Field Service Agents (FSA) with the appropriate working agent as well as system database. The information service agent distinguishes the earthquake hit area and communicates the related data to Emergency Service Agent (ESA) accordingly.

The channels of communication between the Information Service Agent and the database processes are Update(db), Query(db) and Supply($info_1$). The Information Service Agent can send data from other agents like the Emergency Service Agent and Personalized Service Agent to the database through the "Update" communication channel using name db. Information Service Agent can also issue query to database for necessary information, for instance, post-earthquake reports along the "Query" communication channel. The feedbacks of queries and strategies to be adopted in response to current event are sent from database to the Information Service Agent through the "Supply" communication channel.

$$ISA \overset{\text{def}}{=\!=}$$
$$((\overline{Send}\langle dt\rangle \cdot \overline{Alert}\langle info\rangle \cdot FSA' \mid Send(m) \cdot Alert(n)$$
$$\cdot\overline{Update}\langle db\rangle) + (\overline{Inform}\langle alert\rangle \cdot PSA' \mid Inform(x)$$
$$\cdot\overline{Query}\langle db\rangle) \cdot Supply(info) \cdot \overline{Instruct}\langle action\rangle \cdot ISA' \mid$$
$$Instruct(y).ESA') + (\overline{Send}\langle report\rangle \cdot ESA' \mid Send(m)$$
$$\cdot\overline{Update}\langle db\rangle \cdot ISA') + (\overline{Instruct}\langle instruct_1\rangle \cdot ESA' \mid$$
$$\overline{Instruct}\langle x\rangle \cdot PSA')$$

6.3.3.3 Emergency Service Agent

The emergency service agent (ESA) shares the data received from information service agent with emergency response services. It collaborates among the major operations of services like ambulance, fire and police departments. The emergency service agent also coordinates with personalized service agents by issuing alerts as well as assigning task related instructions.

$$ESA \overset{\text{def}}{=\!=}$$
$$(Instruct_1(y) \cdot PSA' \mid \overline{Instruct1}\langle action\rangle \cdot \overline{Direct}\langle serv\rangle \cdot$$
$$ESA' \mid Direct(z) \cdot Org'$$
$$+(\overline{Send}\langle report\rangle \cdot Org' \mid Send(x) \cdot \overline{Send}\langle report\rangle \cdot ESA')$$

6.3.4.4 Personalized Service Agent

The personalized service agent (PSA) configures and transforms the tasks to respective personal assistants. Such configuration includes assignment of goals and tasks, mode of communication as well as individual preferences. The data from personal assistants is merged appropriately by PSA and communicated to Information Service Agent for further action. If personal assistants issue any alert or alarm message, PSA forwards the necessary data ISA on "inform" channel or ESA on "instruct" channel for necessary action.

$$PSA \overset{\text{def}}{=\!=}$$
$$\left(\overline{Send_1}\langle alert\rangle \cdot PA' \mid send_1(x) \cdot \overline{inform}\langle alert\rangle \cdot PSA'\right)$$
$$+\left(\overline{Instruct_1}\langle action\rangle \cdot ESA' \mid Instruct_1(x) \cdot \overline{Instruct_2}\langle action\rangle \cdot PSA'\right)$$

6.3.5.5 Personal Assistant

The personal assistant (PA) acts as representative agents for human task force. The information from PSA is communicated to personal assistants for human related operations. In case of any field observation, the data is shared with ISA through PSA for update to all attached system agents and update in database. The personal assistant agents can be deployed on PDAs, mobile phones and other handheld computational devices in pervasive environment.

$$PA \stackrel{\text{def}}{=\!=}$$
$$(\overline{Instruct_2}(action).PSA'|Instruct_2(x).PA') +$$
$$(\overline{Send_2}(alert).PA'|Send_2(y).PSA')$$

6.3.6.6 Database

The Database (DB) consists of data and information about the earthquake related events as well as operations strategy in the form of goals and tasks. The DB also stores the newly generated data and up to date information which is sent by the ISA. DB agent updates the existing record if the data is received on “Update” channel. If ISA sends a query for retrieval of data on “Query” channel, the DB agent executes the query and extracts the required data from underlying records. It sends back the results to ISA on the “Supply” channel back to Information Service Agent.

$$DB \stackrel{\text{def}}{=\!=}$$
$$\left(\overline{Update}\langle db\rangle \cdot ISA' \mid Update(x) \cdot 0\right) +$$
$$\left(\overline{Query}\langle db\rangle \cdot ISA' \mid Query(y) \cdot \overline{Supply}\langle info_1\rangle \cdot DB'\right)$$

6.4 Major Activities

Various scenarios are modeled in formal notations and expressed in following sub-sections.

6.4.1 Earthquake Detection

In the activity of earthquake detection, the FSA informs about the scale readings to ISA and generates alerts and alarm messages. Alternatively, PA can also communicate any field observation data through the PSA for further action.

$$EDetection \underset{=}{\text{def}}$$
$$\left((\overline{Send}\langle dt\rangle \cdot \overline{Alert}\langle info\rangle \cdot FSA')\right)$$
$$+\begin{pmatrix}(\overline{Inform}\langle alert\rangle \cdot PSA' \mid Inform(x)\cdot \overline{Query}\langle db\rangle \cdot \\ Supply(info_1)\cdot ISA')\end{pmatrix}$$

6.4.2 Emergency Service Initiation

The ISA shares the up-to-date data with attached system agents. It communicates the instructions to ESA for onward sharing with emergency response organizations.

$$EServiceInitiation \text{ def}$$
$$(\overline{Instruct}\langle action\rangle \cdot ISA') \mid (Instruct(x)\cdot$$
$$\overline{Instruct_1}\langle action\rangle \cdot ESA' \equiv (\overline{Instruct}\langle instruct_1\rangle \cdot$$
$$ESA' \mid Instruct \cdot (instruct_1)\overline{Instruct_1}\langle x\rangle \cdot$$
$$PSA' \xrightarrow{\tau} ISA \mid \overline{Instruct_1}\langle x\rangle \cdot PSA')$$

6.4.5 Service Agencies Directives

In this activity, the Emergency Service Agent issues the tasks or orders to emergency response services on “direct” channel for necessary operation.

$$ServAgencyDir \underset{=\!=}{\text{def}} \overline{Direct}\langle serv\rangle \cdot ESA' \mid Direct(z) \cdot Agency'$$

6.4.6 Database Update

In this activity, the data which is received from Emergency Service Agent or Field Service Agent is updated in database through “Update” communication channel.

$$DBUpdate \underset{=\!=}{\text{def}}$$
$$(\overline{Send_1}(report).ESA' \mid Send_1(x).\overline{Update}(db).ISA')$$
$$+(\overline{Send}(dt).FSA' \mid Send(y).\overline{Update}(y).ISA')$$

6.5 Pi-ADL Specification

Pi-ADL is defined as Architecture Description Language. [16, 88, 93] It is a formal method technique which has its foundation in Pi-calculus and supports the modeling of system behavior in parallel execution environment. The major focus of PI-ADL is expressing the system architecture in a formal way for reasoning and verification. Pi-ADL is more advanced and enriched in formal syntax and notations as compared to Pi-calculus. The models and specification expressed in Pi-ADL are closer towards execution. The formal system specification is considered as hyper code. It is converted to intermediate language in Pi-ADL.NET tool or “ProcessBase” language in the Archware framework [15]. Using the Pi-ADL enabled tools; these formal specifications can be executed for validation and verification of the system architecture as well as information flow.

Pi-ADL facilitates in formal modeling of system structure and behavior by expressing the processes, internal and external interactions through the connecting information channels. The executable processes are expressed as “behavior” and “abstraction”. The inter-connecting information channels are specified through “via” keyword. The data input and output from/to processes, is expressed through “send” and “receive” formal syntax respectively.

The term “abstract implementation” is used for Pi-ADL specifications because the formal specifications can be executed with the help of tools for checking inconsistency, redundant information or any loop holes in system modeling. In the current proof of concept application, Pi-ADL.NET tool has been used for executing the formal specifications for validation and verification. The syntax errors during compile time as well as in the execution phase facilitates in identifying the system analysis and design discrepancies.

6.5.1 EMS Specification in Pi-ADL

The major activities in agent based EMS system have been specified using Pi-ADL formal notation and mentioned in following sub-sections.

6.5.1.1 Flow of Information Analysis

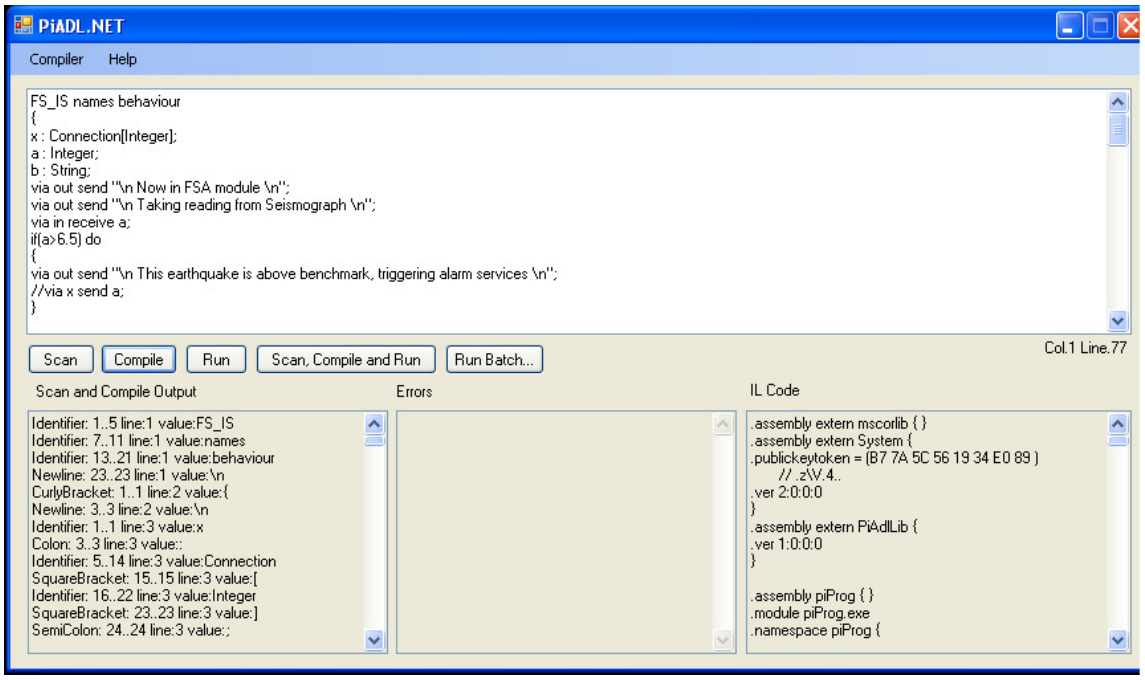
The combined system specification which shows flow of information to various modules which is represented in Pi-ADL specifications and executable in Pi-ADL.NET tool is mentioned as follows,

```
FS_IS names behaviour
{
x : Connection[Integer];
a : Integer;
b : String;
via out send "\n Now in FSA module \n";
via out send "\n Taking reading from Seismograph \n";
via in receive a;
if(a>6.5) do
{
via out send "\n This earthquake is above benchmark, triggering alarm services \n";
//via x send a;
}
else do
{
via out send "\nThis earthquake is within normal range\n";
```

```
}

via isa send a where {x renames yy};

}

value isa is abstraction (avalue : Integer)

{

//x : Connection [Integer];

yy: Connection [Integer];

direct : Connection [String];

service_requested : String;

via out send "\n Now in ISA module \n";

via out send "\n Received following value from FSA \n";

via out send avalue;

service_requested = "Activate Ambulance_Fire_Police";

if (avalue>6.5) do

{

via out send "\n Activating Emergency services...\n";

via out send "\n Sending service request from ISA to ESA module \n";

//via direct send "Police, Ambulance, Fire";

via esa send service_requested where {direct renames esadir};

 }

else do

{
```

```
via out send "\n Earthquake within normal range, Emergency services are not activated \n";
}
 }
value esa is abstraction (service : String)
{
esadir : Connection[String];
y : Connection [any];
serviceag: Connection [any];
tup : tuple[String, String];
via out send "\n Now in ESA module \n";
via out send "\n Received following request from ISA module \n\n";
via out send service;
tup=tuple ("Ambulance_Fire_Police", "Urgent response needed");
//via serviceag send tup; //Sending signals to hardware equipment
via out send "\n\n Emergency services activated \n\n";
}
```

Figure 6.3 shows the verification process of pi-adl specifications, which are error free and converted to intermediate language for architecture description as well as execution. It can also be used with other third generation languages for further detailed implementation.



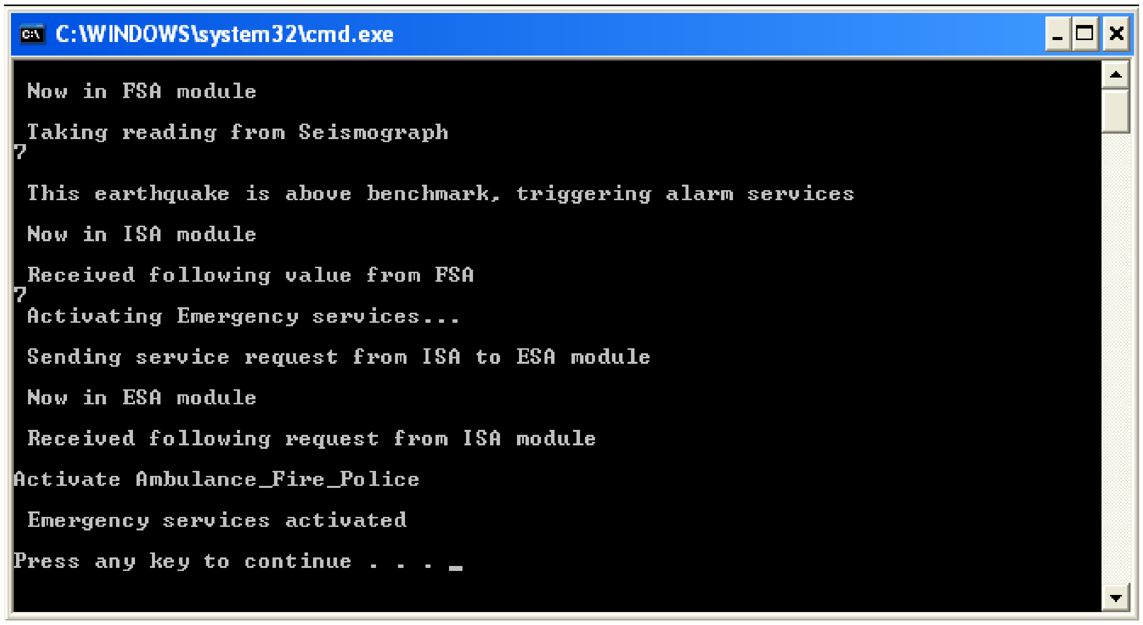
```
FS_IS names behaviour
{
x : Connection[Integer];
a : Integer;
b : String;
via out send "\n Now in FSA module \n";
via out send "\n Taking reading from Seismograph \n";
via in receive a;
if(a>6.5) do
{
via out send "\n This earthquake is above benchmark, triggering alarm services \n";
//via x send a;
}
```

```
Identifier: 1..5 line:1 value:FS_IS
Identifier: 7..11 line:1 value:names
Identifier: 13..21 line:1 value:behaviour
Newline: 23..23 line:1 value:\n
CurlyBracket: 1..1 line:2 value:{
Newline: 3..3 line:2 value:\n
Identifier: 1..1 line:3 value:x
Colon: 3..3 line:3 value::
Identifier: 5..14 line:3 value:Connection
SquareBracket: 15..15 line:3 value:[
Identifier: 16..22 line:3 value:Integer
SquareBracket: 23..23 line:3 value:]
SemiColon: 24..24 line:3 value:;
```

```
.assembly extern mscorlib { }
.assembly extern System {
.publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
     // .z\V.4..
.ver 2:0:0:0
}
.assembly extern PiAdlLib {
.ver 1:0:0:0
}

.assembly piProg { }
.module piProg.exe
.namespace piProg {
```

Figure 6.3 Verifying Pi-ADL Specifications

Fig 6.4 shows the execution result of the above mentioned specification. It highlights the flow of information through various modules of proposed application architecture.



```
 Now in FSA module

 Taking reading from Seismograph
7

 This earthquake is above benchmark, triggering alarm services

 Now in ISA module

 Received following value from FSA
7
 Activating Emergency services...

 Sending service request from ISA to ESA module

 Now in ESA module

 Received following request from ISA module

Activate Ambulance_Fire_Police

 Emergency services activated

Press any key to continue . . . _
```

Figure 6.4 Execution of Pi-ADL Specifications

6.5.1.2 Accessing Database Information

The storing and retrieving of database information as represented according to Pi-ADL specification and verified to be syntax error free through Pi-ADL.NET tool is as follows,

```
ISA_DB names behavior.
{
update : Connection [view[Id:Integer, Person_Name:String, Status:String, City:String, Hospital:String]];
querydb : Connection [any];

Query : Connection [view [Id: Integer, Person_Name: String, Status: String, City:String, Hospital:String]];
vi : view [Id: Integer, Person_Name: String, Status: String, City:String, Hospital:String];
que: any;
//loc : location [any];
choose
{
compose
{
via Query receive vi;
and
via update send vi;
//via loc send vi;
}
or
via querydb receive que;
} }
```

6.6 Summary

This chapter discusses the prototype implementation of goal oriented task based policies concept for mobile agents. The role of task based policies is highlighted which are made of various conditions and actions and associated with particular goals. The main goals contain hierarchy of sub-goals which are linked with sub-tasks. These policies structure is represented in ontologies using OWL. Mobile agents moving on various machines can access URI and extract as well as execute various commands and actions through protégé and Jena APIs in context of assigned goals.

Additionally, the formal representation of proposed EMS application is discussed using Pi-Calculus and Pi-ADL techniques. This application domain has been proposed as major candidate for highlighting the effectiveness and efficiency of agent based systems which can be expressed in detail using formal notations. In this case, software agents have been used for observing, tracing and administering the operations during the earthquake event from initial detection of seismic waves to emergency relief operations. The formal notations help to produce and analyze un-ambiguous and non-redundant flow of activities in context of overall architecture based on software agents.

Chapter 7

EVALUATION

7.1 Teamwork Evaluation

In the first prototype application of target/resource hunt example, teamwork approaches have been analyzed by taking measurements of their goal or task accomplishment time. The job of software agents is to find a specific resource in a grid area, mentioned in a two dimensional map. Agents are coordinating with each other using two schemes of teamwork as discussed in chapter 3.

The results obtained are shown in Figure 7.1. In Team Leader strategy, all agent members of team are executing in parallel threads. As the number of agents increases in a team and area of operation is divided more and more, there is some overhead of parallel execution of agents which are running on single machine.

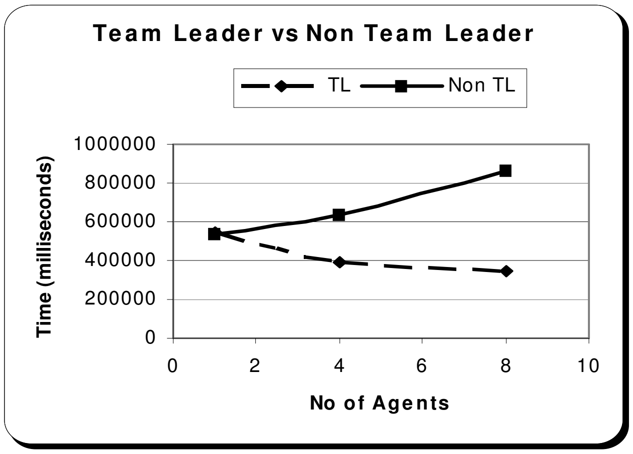


Figure 7.1 TL vs Non TL Approaches

Non-Team Leader strategy highlights two overheads while execution. First overhead is the parallel execution of member agents in which there is no division of plan or area under search. The second overhead is the large amount of interactions during the coordination and collaboration among member agents during execution in order to fulfill the primary goal and allocated tasks.

Team Leader approach facilitates in implementation perspective due to its efficient communication capability. Also, this technique supports effective sharing and dissemination of assigned goals as well as associated task related information. This strategy exhibits better results especially in the scenarios of higher number of member agents.

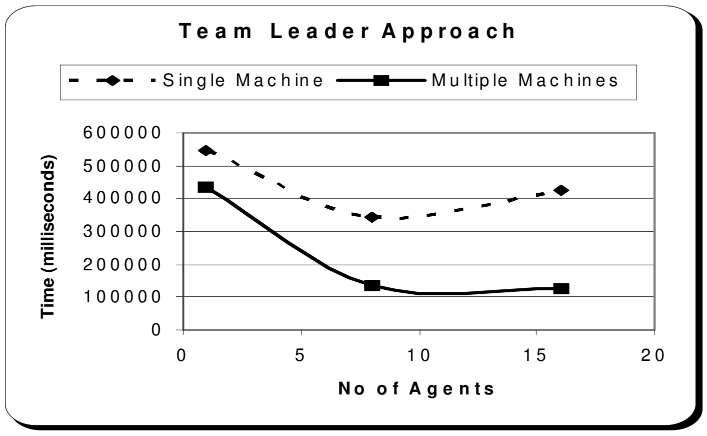


Figure 7.2 Team Leader Approach – Single vs Multiple Machines

The Team Leader and Non Team Leader approaches have also been analyzed on multiple machines as highlighted in Figures 7.2 and 7.3 respectively. Three machines were used for this experiment on which member agents were executing while in team leader approach the leader agent was running on fourth machine. Software agents were

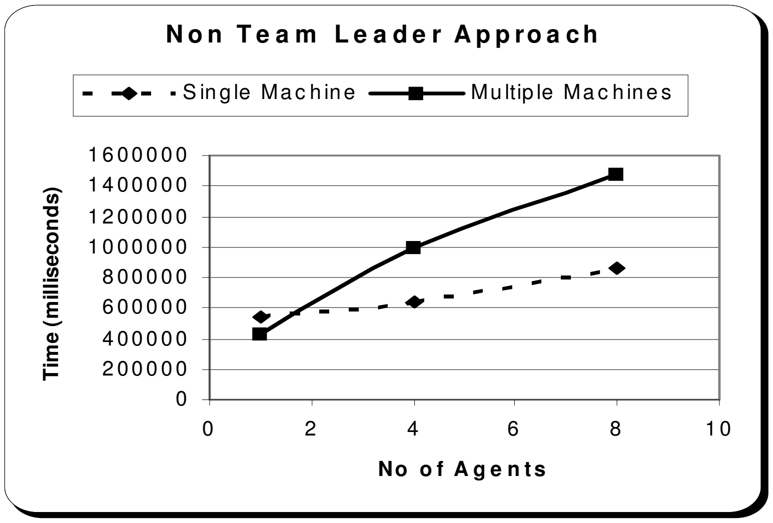


Figure 7.3 Non Team Leader Approach – Single vs Multiple Machines

deployed on these machines. The agents were executing on the Sage [82] multi agent system. The member agents were communicating and coordinating with each other by inter-machine messages through HTTP protocol. Figure 6 highlights the comparison of Team Leader strategy where agent interactions were evaluated on single as well as multiple machines. The evaluation performed on single machine highlights less time taken in task accomplishment in the beginning of execution. However, as the number of member agents is raised, the overhead of parallel execution starts appearing. This places more load on the system and overall performance of system begins to decline.

In case of member agents' execution on multiple machines, the task achievement time keeps low in the start of application. As the number of agents begins to increase on the specific machine, there is moderate increase in the time taken to achieve specified goal. However, the overall performance still shows better results as compared to single machine results.

The results of Non Team Leader strategy is shown in Figure 7.3. The key difference between single machine and multiple machines results is more vivid in the scenario of Non Team Leader approach. It is due to the overheads of communication and coordination among number of agents. The communication is intra-machine when agents are executing on one machine. However, when the agents are distributed and deployed on multiple machines, the coordination and collaboration among them becomes as inter-machine communication. This form of communication enhances the overhead to large extent. With the increase in number of agents in a specific team, and they perform their operations collaborating and coordinating with each other on distributed machines, the more overhead appears and makes significant performance loss. The following equation may be inferred from the above mentioned results.

$$Cf_{TL} < Cf_{NTL}$$

It is concluded that the Team Leader strategy proves to be more efficient and effective than using Non Team Leader strategy in teamwork among higher number of agents. The results are more evident when the team member agents are deployed in distributed fashion and there is major overhead of communication and coordination due to limited infrastructure and/or available bandwidth in system networks.

A tree like hierarchal structure exhibiting team leader strategy among software agents may be designed for more complex and distributed environments. The team leader at each level may be assigned sub-goals and sub-tasks depending on its position in the hierarchy. The sub-goals achieved by member agents and respective team leaders will lead to accomplishment of prime goal at the top of hierarchy.

7.2 Ontology based Policies Evaluation

The proposed approach has been evaluated in context of proof of concept application acting as a module in the earthquake management system. The outcome is analyzed against conventional approaches which are deployed in analogous circumstances. The existing techniques include web based communication, ACL messages in multi-agent system as well as basic mobile agent strategy for querying, retrieving and updating data on remote hosts.

One of the techniques for information retrieval and data update at remote hosts is usage of HTTP protocol in the communication messages. Websites are accessed through web browsers, which support html format. Although it is most widely used technique but it has specific limitations. For example, the users need to browse through number of web pages in order to access certain category of information. Additionally, in case of addition of repetitive data, one may require reloading or refreshing the web page for data entry repeatedly. The prime issue is making data query for information searching and retrieval. The user might be receiving only the static data from target host due to fixed parameters usage in the queries. It limits the user capability to interact with the database in a flexible and efficient way. Users need to interact with the database by generating customized queries in a limited way. In the proposed example scenario, the web pages were created related to earthquake management system data where users query and receive injured or dead persons data for evaluation.

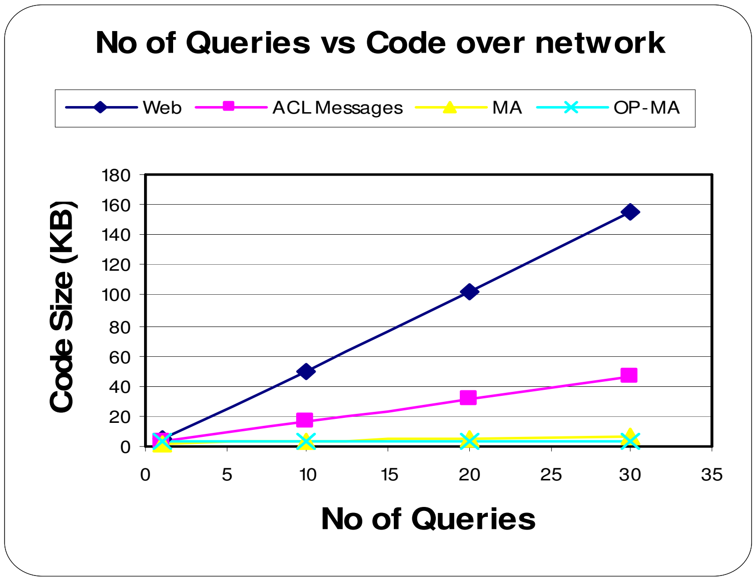


Figure 7.4 Comparison of Existing Technologies for Code Movement over Network

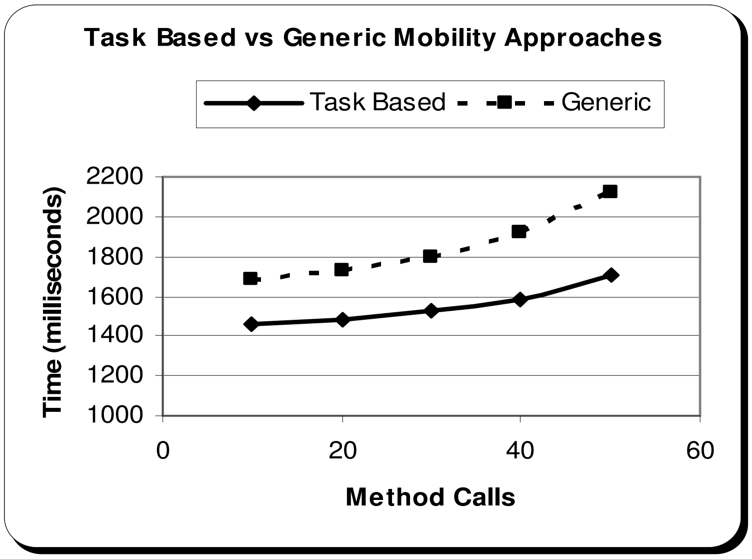


Figure 7.5 Comparison of Policy based Operations vs Generic Mobility Strategy

In multi-agent system domain, agents also coordinate with each other through Agent Communication Language (ACL) messages. In current application scenario, when software agents try to make communication with remote database for information exchange using ACL messages, the performance declines due to overhead of heavy network data exchange. Additionally, the ACL message structure is complex and requires user's intervention on both sender and receiver sides in order to represent queries in dynamic environment.

The basic mobile agent strategy is more efficient than html based web pages and ACL messages in multi-agent systems. Mobile agents can carry and process the code itself on the target host for accessing and retrieving required information. The SQL based queries may be run repetitively with combination of various user-defined parameters. Additionally, the data may be controlled more effectively like accessing or updating the records of a specific table or other sources. However, the creation and definition of various mobile agent related tasks requires more knowledge and capability for novice users as compared to exploring other alternative techniques. It is due to the tightly constrained structure of application and complexity of integrating mobile agent code with the primary application. This drawback can be addressed by employing the ontology based policy approach.

The policy based mobility technique usage by employing task oriented policies, provides a solution to express tightly constrained statements in form of policies. Policies comprising pre-defined goals and associated tasks may be altered without affecting other application code statements in order to avoid a ripple effect. However, the creation of the policies and its associated structure of goals and tasks in specified order appear to be complicated activity. In order to address such limitations, ontology based technique is used to represent policy structure of the mobile agent. The ontologies which are designed and developed in OWL offer more flexibility and less execution overhead. The mobile agent can interact with the specified ontology on multiple hosts by using URI of the published ontology. The agent retrieves the specific goal oriented task based policy structure and executes the related operation for accomplishment of primary goal. The expression of tasks structure in OWL ontology provides a more flexible solution as compared to other conventional techniques where policy statements are mentioned in hard coded or tabular form.

Figure 7.4 shows the evaluation of above mentioned traditional and newly proposed techniques. In this case, the number of data queries to be executed, is compared with amount of information/data exchanged over the network between the two hosts. SAGE multi agent system was deployed on both machines for execution and communication support of individual agents. The primary goal is to query about person's name at remote host and retrieve the related information including its current status, ID number and resident city data. In case of html based communication, web pages were created and connected through web browser. In context of Agent Communication Language (ACL) approach, ACL messages were sent and received through multi-agent system support. Similarly, mobile agent was created on the agent system. It migrated to the target host carrying the code for query execution with itself. The mobile agent processed the code on destination machine and returned the results to source machine. Additionally, mobile agent with ontology based policy structure was also executed with specified goal and results were retrieved.

The results highlights that the ontology based policy technique which is used in context of mobile agent, generates less amount of data to be sent over the network for accomplishment of primary goal. In web pages case, the change in requirements at client side requires similar update on server side. In the ACL scenario, if the message is altered for addressing new requirements, the receiver also needs to be upgraded. However, mobile agent carries the code with itself and requires only the execution support at the target host.

The evaluation of proposed architecture is also highlighted using two mobility techniques defined as Generic mobility approach and policy based approach as shown in Figure 7.5. The generic mobility approach shows the strategy of moving all code and state of mobile agent in one process/step to destination host. The policy based migration technique highlights the approach of classifying the code as per tasks hierarchy. The approach allows transferring of only particular code and state which is required for execution at destination machine. If the associated tasks structure is already available at target host with implementation specifications, the reusability benefits are gained. In such case, the related task data along with necessary parameters are transferred and local resources are exploited as much as possible. Additionally, the user can customize the target host activities by creating necessary processes at destination host and provides its interfaces to guest mobile agent. In this way, mobile agent executes the local tasks at arrival with its own source host defined parameters and interacts with the target host locally in a flexible way. In prototype application, the mobile agent migrates only with necessary parameter required for searching the database instead of moving the whole code to target host.

Table 7.1 - Analysis of Generic and Policy based Operations Mobility Mechanisms

Attributes	Generic Mobility	Ontology-Policy Mobility
Code Reusability	No	Yes
Limited Bandwidth usage	Yes	Yes
System resources usage	Moderate	Moderate
User Flexibility	Moderate	High
Adaptability	Low	Moderate

The two mobility approaches can be analyzed qualitatively as presented in Table 7.1. It shows the analysis of mobility mechanisms in case of generic as well as ontology based policy mobility techniques. Considering the code reusability parameter, generic mobility exhibits a limited approach. The mobile agent program defined for a particular functionality needs to be re-programmed completely for other task or job. In case of ontology based policy mobility, once the major goal and structure is defined in ontology form, the tasks structure and goal hierarchy can be reused. In case of new circumstances, the specific task can be altered or upgraded without changing the whole structure of application. Both generic and ontology-policy mobility uses limited bandwidth as compared to conventional client server distributed paradigms because these mobility techniques uses mobile agent technique for performing various distributed operations. Also, both techniques use system resources to a moderate level because the software agents are running on an underlying multi-agent system. However, the advantages of using agents based approach covers this mild limitation.

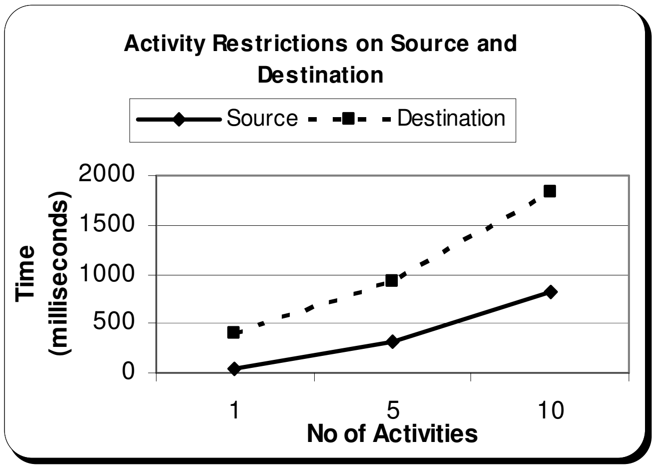


Figure 7.6 Comparison of Applying Restrictions

In generic mobility case, a user needs extensive knowledge and expertise to program a mobile agent application and develop various conditions in hard coded form like where to move which components under specific events and conditions. However, more user flexibility is achieved in case of ontology-policy mobility technique because the major basic building blocks are already defined for particular scenarios and user needs only to invoke the primary goal with related parameters. In case a new scenario is needed, user can pick and assemble individual tasks and operations under a primary goal in specific order and application can be executed in a flexible way. There is low adaptability in case of generic mobility technique because when mobile agent moves to destination machine, it cannot change its behavior or execution strategy as per new changes or format at destination host. In case of ontology-policy mobility technique, various operations can be defined at target host and its interfaces are provided to mobile agent on it arrival for tasks execution in dynamic environment.

The proposed policy based strategy could also be used for regulating the task execution and movement pattern of mobile agents. Users or developers can employ various conditions in order to block certain activities which are restricted to be run at destination or preventing the migration of mobile agent to certain IP addresses. A comparison of two scenarios is highlighted in figure 7.6 which shows time taken to block certain activities while applying restrictions on task related information of mobile agents at source machine as compared to blocking of activities at destination machine.

7.3 Evaluation of Formal Specifications

The proposed earthquake management system has been modeled and specified using Pi-calculus and Pi-ADL formal techniques. These approaches facilitates in elaborated modeling, specification, analysis, design and abstract implementation for verifying the system reliability and elimination of inconsistencies. Although, formal logic based approaches like propositional and predicate logic, description logic, Z Notation support the expression of system structure however, there are major deficiencies while modeling the behavior of systems. These limitations are further visible when there is a need to formally express parallel and concurrent processes. Additionally, these logic based notations possess limited capability to specify communications channels for information flow as well as new process state after execution of specific command. Pi-ADL is more advanced and enriched in syntax and formal notations then conventionally used logic based approaches. Pi-ADL also provides support to represent messages in form of tuple, view, union, variant and quote. The database is represented by location while collection of data can be expressed by way of set, bag and sequence statements.

Pi-ADL.NET tool was used for executing and validating the formal specifications as mentioned in chapter 6. The formal modeling approach facilitates in eliminating the redundant information as well as specification errors and inconsistencies. This process enhances the overall reliability of the proposed modeled system. Similarly, the structure and behavior as well as information flow can be traced and analyzed for each specific agent.

7.4 Summary

In this chapter, the evaluation work is presented in three stages. Firstly, the evaluation of two major teamwork architectures including team leader and non-team leader approaches has been highlighted. The comparison highlights the rising difference in execution performance when there is higher number of agents having significant communication among them while residing on distributed machines. Team leader strategy proves to be more efficient and effective teamwork technique than traditionally used non team leader approach in group of mobile agents.

Also, the evaluation of ontology based policies approach is highlighted comparing it with current existing techniques. As per conclusion from results, the proposed approach simplifies the goal based task execution as well as generates less amount of code which is required to be sent over network for specific task and consequently provides more efficient solution. Lastly, the execution and validation approach for formal specifications is discussed as compared to existing logic techniques as well as validation tool usage.

Chapter 8

DISCUSSION

In this chapter, a review and analysis of evaluation and results is presented as well as its significance is highlighted in context of future applications.

8.1 Analysis of Teamwork Results

In chapter 3, the major classification of teamwork architecture was presented in addition to highlighting the two main techniques. As the complexity and diversity of applications will increase, teamwork among mobile agents will be a major characteristic in future applications. It is difficult to incorporate all desired functionality in single agent; therefore a group of agents coordinating with each other and sharing the main goal proves to be much efficient and effective solution.

In this particular work, the teams of software agents which were executing on agent platform on distributed machines were assigned a primary goal. The architecture of two teams was designed in a way that teams firstly share goals with each other and then contribute towards a joint primary goal. However the sharing of information and particular task division was achieved through team leader and non-team leader approaches.

According to results achieved, it was determined that team leader approach is much efficient than the non-team leader approach especially when agents are assigned the tasks, which are required to be executed on distributed and multiple machines. This difference is wider as the number of agents in a team increase with high number of communication and interaction among them. In non-team leader approach, the tasks of primary goal division and allocation of sub-goals and sub tasks to team members as well as their coordination and information sharing during tasks execution makes significant overhead on the overall performance of particular team and task accomplishment time. However, team leader approach where goal and task information is shared in a hierarchy fashion proves to be a much efficient and effective technique.

8.2 Review of Semantic policies approach

The ontology based policies approach reinforces the teamwork architecture goal and task division characteristics. As chapter 3 discusses the efficiency of team leader technique, chapter 4 describes the ontology based policies technique for goal and task division as well as execution for generic applications. The goals were described in the form of policies consisting of conditions and actions, which were represented in ontology form for online accessibility and usage. Similarly the tasks were divided in a hierarchy form and linked with the goals.

This ontology based policy technique helped in convenient assignment of goals and associated tasks setup information for mobile agents which were earlier difficult to setup and execute as desired especially in dynamic environments. Additionally, as per results obtained, the proposed technique helps in reducing size of the mobile agents leading to less amount of code and data movement over network. The combination of simple task execution as well as reduced size leads to less amount of time consumption in performing desired functionality leading towards an efficient solution. The expression of tasks structure in ontological form and publishing it online provides more flexibility and dynamic accessibility to mobile agents, which move from machines to machines over internet and interact locally through online available tasks and goal structure. For future applications, this paradigm will lead to development of simple setup of e-commerce or related online websites which can host and interact with mobile agents effectively in an efficient way.

8.3 Revisiting the EMS Application

The role of agents is highlighted in earthquake management system in chapter 5. The objective is to propose the use of software agents in disaster management domain and express their capabilities to handle complex application scenarios. An example of information retrieval by mobile agents in EMS domain is highlighted in combination with description of complete application structure and behavior along with the role of agents. This leads to opening of major application domain where software agents execute in a teamwork fashion highlighting capability of executing goal related operations in dynamic environments. In order to describe the major structure and behavior of proposed application, Pi-Calculus and Pi-ADL formal techniques have been used.

8.4 Summary

This chapter discusses the research outcomes of the proposed research work as presented in previous chapters. It reflects the need for using teamwork architecture solution in mobile agents and critical importance of various parameters like communication in distributed infrastructure as well as ontology based policies solution for flexibility and efficiency. Additionally, the behavior of proposed architecture including its goal and tasks setup as well as execution strategies has been discussed. Lastly, it highlights the role of agents in proposed application for further usage and experimentation.

Chapter 9

CONCLUSIONS

In this thesis, an efficient teamwork strategy has been presented for group of mobile agents in addition to ontology based goal oriented policy techniques. This teamwork approach or joint operations strategy by group of mobile agents is designed to tackle the growing complexity of application domain where individual and isolated stationary or mobile agents are not able to accomplish their desired tasks effectively. Although, there is significant work going on in the domain of teamwork for agents but very limited work is focused especially on team of mobile agents. The proposed teamwork strategy has been conceptualized from observation of working pattern of Honey-Bees and so it was named after it as Honey-Bee teamwork architecture. The classification was highlighted based on goal sharing and interaction pattern among software agents. The two major approaches including team leader and non-team leader strategies were analyzed and evaluated using prototype application.

Also, goal oriented ontology based policies architecture is proposed in context of FIPA compliant multi-agent systems especially for mobile agents. This strategy was proposed in order to address the issues of efficiently creating and executing various tasks which are associated with goals in a hierarchical structure ultimately fulfilling the primary goal of application. The ontologies were developed using OWL as it is standardized approach and it is convenient to create and use through supporting tools like Protégé and Jena APIs.

The mobile agents could access the policy structure represented in ontologies by providing the URI of published ontology on distributed machines. This provides more flexible approach than traditional hard-coded policy statements. The proposed approach was analyzed in the context of earthquake management system (EMS) as proof of concept application.

Lastly, the earthquake management system (EMS) has been designed and analyzed using formal approaches including Pi-Calculus and Pi-ADL. The formal technique helped in modeling and specification of the agent based earthquake management system (EMS) by enhancing the reliability and flexibility of the proposed system. The formal approaches supported in explicitly defining the information flow and reducing the redundant information for effectively coordinating various processes and exhibiting reliable application behavior. This strategy also contributes towards the proposed research work.

The research questions which were mentioned in chapter 1 related to teamwork are answered in subsequent chapters where the limitations of mobile agents teamwork architecture are highlighted along with two distinct types of teamwork strategies. Also, the quantitative analysis shows the efficient technique in scenarios of single and multiple machines transpiring inherent concept of communication overheads. Additionally, other questions were addressed in recent chapters by discussing the proposed work of ontology based policies in mobile agents, where the limited capability of mobile agents' task execution was discussed. The policy based approach was highlighted in this domain along with its limitations. Furthermore, in order to address the deficiencies, the proposed solution of ontology based policy architecture is argued. Lastly, the role of agents is presented to mention the structure and behavior of proposed EMS application in context of disaster management systems which is expected to be the upcoming application domain for analyzing future teamwork concepts.

9.1 Future Work

As the applications are becoming more complex, complicated and distributed in nature day by day, there is a need to focus on the newly emerging aspects of coordination and collaboration among software agents including highly flexible and reliable mechanisms for information and assimilation and dissemination strategies. Also, there is a need to improve the mobile agent capability in order to overcome the interoperability constraints and access widely distributed resources effectively over the web.

Additionally, there is a necessity to focus on team building issues of software agents in general and mobile agents in particular where a balance of varying rationalities, expertise or specializations could improve the working capability of team to a great extent for the desired mission. The immediate future plan is to integrate the proposed proof of concept application with web-based system in order to facilitate mobile agent operations on web related services and provide a platform for convenient accessibility and usage.

PUBLICATIONS

Journal Papers/Book Chapters

- Sarmad Sadik, Mukaila Alade Rahman, Arshad Ali, H Farooq Ahmad, Hiroki Suguri, "Modeling High Assurance Agent Based Earthquake Management System using Formal Techniques", Journal of Supercomputing, Springer. Volume 52, Number 2 / May, 2010, pp 97-118.
- Sarmad Sadik, Arshad Ali, H. Farooq Ahmad, Hiroki Suguri, "Honey Bee Teamwork Architecture in Multi-Agent Systems" Extended Paper in CSCW in Design, published by Springer in LNCS 4402, 2006.

Conference Papers

- Sarmad Sadik, Mukaila Rahman, Arshad Ali, H. Farooq Ahmad, Hiroki Suguri, "A Formal Approach for Design of Agent Based Earthquake Management System (EMS)" The Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2008), Thailand.
- Sarmad Sadik, Arshad Ali, H Farooq Ahmad, Hiroki Suguri, "Policy based Ontology Framework for Mobile Agents", 6th IEEE International Conference on Computer and Information Science (ICIS 2007), July 11-13, Australia.

- Sarmad Sadik, Maruf Pasha, Arshad Ali, H Farooq Ahmad, Hiroki Suguri, “Policy Based Migration of Mobile Agents in Disaster Management Systems”, IEEE International Conference on Emerging Technologies, Nov 13-14, Pakistan.
- Sarmad Sadik, Arshad Ali, H Farooq Ahmad, Hiroki Suguri, “Using Honey Bee Teamwork Strategy in Software Agents”, 10th International Conference on CSCW in Design May 3-5, 2006 Nanjing, China
- Sarmad Sadik, Arshad Ali, H Farooq Ahmad, Hiroki Suguri, “Policy Based Approach to Enhance Task Execution Performance of Mobile Agents”, The 2006 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'06), June 26-29, 2006, USA.

REFERENCES

1. A. Fuggetta, G. Picco, and G. Vigna, “Understanding Code Mobility”, *IEEE Trans. Software Engineering*, May 1998.
2. Paolo Bellavista, Antonio Corrad, Cesare Stefanelli, “Mobile Agent Middleware for Mobile Computing”, *IEEE Computer*, March 2001
3. Matthew Johnson, Jeffery Bradshaw, Paul Feltovich, Renia Jeffers, Hyuckchul Jung, Andrzej Uszok, “A Semantically Rich Policy Based Approach to Robot Control”, *International Conference on Informatics in Control, Automation and Robotics*, France, 2006.
4. Rebecca Montanari, Emil Lupu and Cesare Stefanelli, “Policy-Based Dynamic Reconfiguration of Mobile-Code Applications”, *IEEE Computer*, July 2004.
5. D. Fensel, “Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce”, 2nd edition. Springer-Verlag, Berlin, 2003.
6. David Lewis, John Keeney, Declan O’ Sullivan, “Policy-based Management for Resource-Specific Semantic Services”, *Proceedings of the 1st Annual Workshop on Distributed Autonomous Network Management Systems*, Ireland,2006.
7. Michael Wooldridge, An Introduction to Multi-agent Systems, John Wiley & Sons Press, 2002.
8. W3C,Online: http://www.w3.org/ 2007
9. Protégé, Online: http://protege.stanford.edu/ 2007
10. Jena, Online: http://jena.sourceforge.net/ 2007

11. R. Milner: A Calculus of Communicating Systems. LNCS 92, Springer Verlag, 1980.

12. R Milner, Communicating and Mobile Systems: the Pi-Calculus, Cambridge University Press, 1999.

13. Parrow J. An Introduction to the Pi-Calculus, Handbook of Process Algebra. - Elsevier, 2001.

14. J.M. Wing,: FAQ on Pi-Calculus, December 2002 Online: http://www.cs.cmu.edu/~wing/publications/Wing02a.pdf

15. ArchWare: Architecting Evolvable Software. European RTDProject Online: www.architecture-ware.org. 2007

16. Balasubramaniam, D, Morrison, R, Kirby, GNC, Mickan, K, Norcross, S. ArchWare ADL Release 1 User Reference Manual. ArchWare Project IST-2001-32360 Report D4.3. 2004

17. 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.

18. Paolo Bellavista, Antonio Corrad, Cesare Stefanelli, "Mobile Agent Middleware for Mobile Computing", *IEEE Computer*, March 2001.

19. Peter Braun, Steffen Kern, Ingo Mueller, Ryszard Kowalczyk "Attacking the Migration Bottleneck of Mobile Agents" *Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*. Utrecht, Netherlands, 2005

20. Zara Field, P. W. Trinder, Andr´e Rauber Du Bois, “A Comparative Evaluation of Three Mobile Languages” *Proceedings of the 3rd international conference on Mobile technology, applications & systems*, Thailand, 2006.
21. Tudor Marian1, Bogdan Dumitriu, Mihaela Dinsoreanu, Ioan Salomie, "A Framework of Reusable Structures for Mobile Agent Development" *Proceedings of IEEE International Conference on Intelligent Engineering Systems (INES2004)*, Cluj-Napoca, Romania, 2004.
22. Pauli Misikangas, Kimmo Raatikainen, "Agent Migration between Incompatible Agent Platforms", *20th IEEE International Conference on Distributed Computing Systems*, Taiwan, 2000.
23. FIPA Agent Management Support for Mobility Specification, http://www.fipa.org/specs/fipa00087/index.html 2001
24. Giacomo Cabri, Letizia Leonardi, Franco Zambonelli, “Weak and Strong Mobility in Mobile Agent Applications”, *Proceedings of the 2nd International Conference and Exhibition on The Practical Application of Java*, PA JAVA 2000, Manchester (UK), April 2000
25. Arne Grimstrup, Robert Gray, David Kotz, Maggie Breedy, Marco Carvalho, Thomas Cowin, Daria Chac´on, Joyce Barton, Chris Garrett, and Martin Hofmann, “Toward Interoperability of Mobile-Agent Systems”, *6th IEEE International Conference on Mobile Agents*, Spain 2002.
26. Holger Peine and Torsten Stolpmann, “The Architecture of the Ara Platform for Mobile Agents”, Lecture Notes In Computer Science; Vol. 1219, Springer 1997.

27. Michael Bursell, Richard Hayton, Douglas Donaldson, Andrew Herbert, "A Mobile Object Workbench", *Proceedings of the Second International Workshop on Mobile Agents*, Germany 1998.
28. Frazer Bennett, Tristan Richardson, Andy Harter, “Teleporting - Making Applications Mobile”, *Proceedings of 1994 Workshop on Mobile Computing Systems and Applications*, USA
29. Li Tang, Bernard Pagurek, “A Comparative Evaluation of Mobile Agent Performance for Network Management” *Proceedings of the 9th IEEE International Conference on Engineering of Computer-Based Systems*, Sweden 2002.
30. L. M. Camarinha-Matos, João Rosas, Ana-Inês Oliveira, "A Mobile Agents Platform for Telecare and Teleassistance" *Proceedings of the 1st International Workshop on Tele-Care and Collaborative Virtual Communities in Elderly Care, TELECARE 2004*, In conjunction with ICEIS 200, Portugal.
31. Takuya Iizuka, Angel Lau, Tatsuya Suda, "A Design of local resource access control for mobile agent in PDA", *Proceedings of the Asian-Pacific Conference on Communications*, Japan 2001.
32. Milind Tambe, Wei-Min Shen, Maja Mataric, David V. Pynadath, Dani Goldberg, Pragnesh Jay Modi, Zhun Qiu, Behnam Salemi. “Teamwork in Cyberspace: Using TEAMCORE to Make Agents Team-Ready” *Proceedings of the AAAI* , USA 1999.
33. M. Tambe, E.Bowring, H.Jung, G.Kaminka, R. Maheswaran, J. Marecki, P.J.Modi, R.Nair, S.Okamoto, J.P.Pearce, P.Paruchuri, D.Pynadath, P.Scerri,

N.Scerri, N.Schurr, P.Varakantham. “Conflicts in teamwork: Hybrids to the rescue.” *In Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Netherlands, 2005.

34. Paul Scerri, Alessandro Farinelli, Steven Okamoto, Milind Tambe, "Allocating Tasks in Extreme Teams", *In Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS)* Netherlands, 2005.
35. Maarten Sierhuis, Jeffrey M. Bradshaw, Alessandro Acquisti, Ron van Hoof, Renia Jeffers, Andrzej Uszok. “Human-Agent Teamwork and Adjustable Autonomy in Practice.” In *Proceeding of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Japan, 2003
36. Paul Scerri, Katia Sycara, Milind Tambe, "Adjustable Autonomy in the Context of Coordination", *In AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, USA 2004.
37. Seng W. Loke. “The A-Team Design Pattern: Useful Properties for Using Teams of Mobile Agents.” *In Proceedings of International Conference on Intelligent Agents, Web Technology and Internet Commerce (IAWTIC)*, Vienna, Austria, February 2003.
38. Milind Tambe. “Agent Architectures for Flexible, Practical Teamwork.” *National Conference on Artificial Intelligence*, Rhode Island, 1997.
39. David V. Pynadath and Milind Tambe. “Team Coordination among Distributed Agents: Analyzing Key Teamwork Theories and Models”. I*n Proceedings of the*

AAAI Spring Symposium on Intelligent Distributed and Embedded Systems, USA, 2002.

40. Hyuckchul Jung, Milind Tambe. "Performance Models for Large Scale Multiagent Systems: Using Distributed POMDP Building Blocks." *In Proceedings of the second International Joint conference on Agents and Multiagent Systems (AAMAS)*, Australia, 2003.
41. Wikipedia Online: http://en.wikipedia.org/wiki/Policy 2007
42. Evangelia Kavakli, Pericles Loucopoulos, "Goal Driven Requirements Engineering: Evaluation of Current Methods", *8th CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Austria, 2003.
43. Rebecca Montanari, Gianluca Tonti, Cesare Stefanelli, "A Policy-based Mobile Agent Infrastructure", *Proceedings of the Symposium on Applications and the Internet (SAINT'03)* USA 2003.
44. Kun Yang and Alex Galis, "Policy-driven Mobile Agents for Context-aware Service in Next Generation Networks" Lecture Notes in Computer Science, Volume 2881, Springer 2003.
45. Radu Litiu, Amgad Zeitoun, "Infrastructure Support for Mobile Collaboration", *Proceedings of the 37th Hawaii International Conference on System Sciences*, USA, 2004.
46. Ken'ichi Takahashi, Satoshi Amamiya, Tadashige Iwao, "An Agentbased Framework for Ubiquitous Systems", *Challenges in Open Agent Systems '03 Workshop*, Melbourne, Australia, 2003.

47. 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.

48. Giacomo Cabri "Role based Infrastructure for Agents" , *The 8th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2001)*, Bologna, Italy, 2001.

49. 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.

50. Asnat Dadon-Elichai, "RDS: Remote Distributed Scheme for Protecting Mobile Agents", *The third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, USA, 2004.

51. J Ametller, S. Robles, J.A.Ortega-Ruiz, "Self-Protected Mobile Agents", *The third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, USA, 2004.

52. Policy research forum, http://www.policy-workshop.org/ 2007

53. Jade, http://jade.tilab.com/ 2007

54. Aglets, http://www.trl.ibm.com/aglets/ 2007

55. David Wong, Noemi Paciorek, Tom Walsh, Joe DiCelie, Mike Young, Bill Peet, "Concordia: An Infrastructure for Collaborating Mobile Agents", *Proceedings of the First International Workshop on Mobile Agents*, Germany 1997.

56. Kresimir Jurasovic, Gordan Jezic, Mario Kusek, "A Performance Analysis of Multi-Agent Systems", *International Transactions on Systems Science and Applications*, volume 1, number 4, 2006.

57. L.Ismail and D.Hagimond, "A Performance Evaluation of the Mobile Agent Paradigm. In OOPSLA'99," *Proceedings of the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications*, USA 1999.

58. Larry Korba, Ronggong Song, "A Reputation Evaluation Framework for Mobile Agents", *Proceedings of the 5th International Workshop on Mobile Agents for Telecommunications Applications*, Morocco 2003.

59. Sven van der Meer, Declan O Sullivan, David Lewis, Nazim Agoulmine, "Ontology Based Policy Mobility for Pervasive Computing", *9th IFIP/IEEE International Symposium on Integrated Network Management*, France, 2005.

60. Maroun Chamoun, Rima Kilany, Ahmed Serhrouchni, "A Semantic Active Policy-based Management Architecture", *Proceedings IEEE Workshop on IP Operations and Management*, 2004.

61. Matthew Johnson, Jeffery Bradshaw, Paul Feltovich, Renia Jeffers, Hyuckchul Jung, Andrzej Uszok, "A Semantically Rich Policy Based Approach to Robot Control", *International Conference on Informatics in Control, Automation and Robotics*, France, 2006.

62. Niranjan Suri, Jeffrey Bradshaw, Mark Burstein, Andrzej Uszok, Brett Benyo, Maggie Breedy, Marco Carvalho, David Diller, Paul Groth, Renia Jeffers, Matt Johnson, Shri Kulkarni, James Lott, "Toward DAML-based Policy Enforcement

for Semantic Data Transformation and Filtering in Multi-agent Systems”, *Autonomous Agents and Multi Agent Systems (AAMAS)*, Australia, 2003.

63. Stephan Grimm, Steffen Lamparter, Andreas Abecker, Sudhir Agarwal, Andreas Eberhart, “Ontology based Specification of Web Service Policies”, *Proceedings of Semantic Web Services and Dynamic Networks*, Germany 2004.
64. David Lewis, John Keeney, Declan O’ Sullivan, “Policy-based Management for Resource-Specific Semantic Services”, *Proceedings of the 1st Annual Workshop on Distributed Autonomous Network Management Systems*, Ireland,2006.
65. Nejla Amara-Hachmi, “An Ontology-based Model for Mobile Agents Adaptation in Pervasive Environments”, *Proceedings of the 4th ACS/IEEE International Conference on Computer Systems and Applications*, UAE, 2006.
66. YuhJong Hu, “Combining Ontology and Rules as Service Constraint Policy for P2P Systems” *WWW 2005*, May 10-14, Japan.
67. John Keeney, Kevin Carey, David Lewis, Declan O’Sullivan, Vincent Wade, “Ontology-based Semantics for Composable Autonomic Elements”, *Proceedings of the Workshop of AI in Autonomic Communications at the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland July 2005.
68. Saket Kaushik, Duminda Wijesekera, Paul Ammann, “Policy-Based Dissemination of Partial Web-Ontologies” *Proceedings of the 2005 workshop on Secure web services*, USA, 2005.

69. David Lewis, Kevin Feeney, Kevin Carey, Thanassis Tiropanis, Simon Courtenage, “Semantic-based Policy Engineering for Autonomic Systems”, *WAC 2004*, Germany.

70. Advanced National Seismic System. Online: www.anss.org 2006

71. Sahana – IBM Sahana. Online: http://www.sahana.lk/ 2006

72. Peña-Mora, F and Mathias, C. “AVSAR: A collaboration system for disaster search and rescue operations using autonomous vehicles.” *itaec:* 2004.

73. Ajay K. Rathi and Rajendra S. Solanki. “Simulation of traffic Flow during Emergency Evacuations: A Microcomputer –based Modeling System.” *Proceedings of the 1993 Winter Simulation Conference*, USA.

74. Bartel Van de Walle and Murray Turoff. “Emergency Response Information Systems: Emerging ZTrends and Technologies.” *Communications of the ACM*. March 2007/Vol. 50, No. 3.

75. Lucian Vlad Lita, Jamieson Schulte and Sebastian Thrun. “A ultiAgent System for Agent Coordination in Uncertain Environments” *Proceedings of the fifth international conference on Autonomous agents*. Montreal, Quebec, Canada. Pages: 21 – 22, 2001.

76. Wei Chen and Keith S. Decker “Managing Multi-Agent Coordination, Planning, and Scheduling.” *AAMAS'04*, July 19-23, 2004, New York, New York, USA.

77. KAshcroft, J., D.J. Daniels, and S.V. Hart. 2002. “Crisis Information Management Software (CIMS) Feature Comparison Report.” NIJ Special Report 197065, U.S. Department of Justice, National Institute of Justice, Washington, DC. Online. Available: http://www.ncjrs.gov/pdffiles1/nij/197065.pdf.

78. Frank Fiedrich. "An HLA-Based Multi-Agent System for Optimized Resource Allocation after Strong Earthquakes." *Proceedings of the 2006 Winter Simulation Conference*, USA.

79. Cédric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget, Didier Rémy, "A Calculus of Mobile Agents", Lecture Notes In Computer Science; Vol. 1119, published by Springer.

80. Haiping Xu, "A Model-Based Approach for Development of Multi-Agent Software Systems" PhD Thesis, University of Illinois at Chicago, 2003.

81. Treasure Hunt Project for Human-Robot Teamwork by Carnegie Mellon University – Online: http://www.cs.cmu.edu/~treasurehunt/index.html 2008

82. Scalable fault tolerant Agent Grooming Environment (SAGE) Demo, *In the Fourth International Joint Conference on Autonomous Agents and Multi agent Systems (AAMAS)* Utrecht, Netherlands, 2005.

83. FIPA Ontology Service Specification, http://www.fipa.org/specs/fipa00086/XC00086C.html 2005

84. 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*, USA 2004.

85. Kinji Mori, "Autonomous Decentralized Systems for Service Assurance and Its Application", LNCS 4526, 2007.

86. Kinji Mori, “Trend of Autonomous Decentralized Systems”, *Proceedings of 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS)*, China 2004.

87. Geology Labs Online – http://www.sciencecourseware.org/VirtualEarthquake/VQuakeExecute.html 2005

88. Dharini Balasubramaniam, Ron Morrison, Kath Mickan, Graham Kirby, Brian Warboys, Ian Robertson, Bob Snowdon, R Mark Greenwood, Wykeen Seet, “Support for Feedback and Change in Self-adaptive Systems”, *Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems*, USA 2004.

89. Another Bisimilarity Checker (ABC) Online: http://lamp.epfl.ch/~sbriais/abc/abc_ug.pdf 2007

90. Mobility Workbench Online: http://www.it.uu.se/research/group/mobility/mwb 2007

91. Zawar Qayyum, Flavio Oquendo, “The Pi-ADL.NET project: An Inclusive Approach to ADL Compiler Design”, WSEAS Transactions on Computers, May 2008.

92. Danny Weyns, “A Pattern Language for Multi-Agent Systems”, IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA 2009), UK September 2009.

93. Flavio Oquendo, “π-ADL: An Architecture Description Language based on the Higher-Order Typed π-Calculus for Specifying Dynamic and Mobile Software Architectures”, ACM Software Engineering Notes, May 2004.

APPENDIX – A

A.1: The generic policy structure containing the goals, pre-conditions and post conditions infrastructure is described as follows,

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.myontologypolicyontology.owl#"
  xml:base="http://www.myontologypolicyontology.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="DoActions">
    <owl:disjointWith>
      <owl:Class rdf:ID="Preconditions"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="PostConditions"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Goals"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#PostConditions">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Goals"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#Preconditions"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#DoActions"/>
  </owl:Class>
  <owl:Class rdf:about="#Goals">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasPostcond"/>
        </owl:onProperty>
        <owl:someValuesFrom rdf:resource="#PostConditions"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
```

```
        <owl:someValuesFrom rdf:resource="#DoActions"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasAction"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:about="#Preconditions"/>
        </owl:someValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasPrecond"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf
rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="#Preconditions">
    <rdfs:subClassOf rdf:resource="#Goals"/>
    <owl:disjointWith rdf:resource="#DoActions"/>
    <owl:disjointWith rdf:resource="#PostConditions"/>
  </owl:Class>
  <owl:ObjectProperty rdf:about="#hasPrecond">
    <rdfs:domain rdf:resource="#Goals"/>
    <rdfs:range rdf:resource="#Preconditions"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasPostcond">
    <rdfs:domain rdf:resource="#Goals"/>
    <rdfs:range rdf:resource="#PostConditions"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasAction">
    <rdfs:range rdf:resource="#DoActions"/>
    <rdfs:domain rdf:resource="#Goals"/>
  </owl:ObjectProperty>
  <owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
      <PostConditions rdf:ID="Opsearch"/>
    </owl:distinctMembers>
  </owl:AllDifferent>
  <Goals rdf:ID="Search"/>
  <owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
      <Preconditions rdf:ID="Presearch"/>
    </owl:distinctMembers>
  </owl:AllDifferent>
  <Preconditions rdf:ID="Macconnected"/>
  <owl:AllDifferent>
```

```
    <owl:distinctMembers rdf:parseType="Collection">
      <Goals rdf:about="#Search"/>
    </owl:distinctMembers>
  </owl:AllDifferent>
  <owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
      <Preconditions rdf:about="#Presearch"/>
      <Preconditions rdf:about="#Macconnected"/>
    </owl:distinctMembers>
  </owl:AllDifferent>
  <owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
      <DoActions rdf:ID="Move"/>
    </owl:distinctMembers>
  </owl:AllDifferent>
</rdf:RDF>
```

A.2: The prototype policy example which as expressed in figure 6.2, is shown in ontology form as follows,

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY p1 "http://www.owl-ontologies.com/assert.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1257926648.owl#"
     xml:base="http://www.owl-ontologies.com/Ontology1257926648.owl"
     xmlns:p1="http://www.owl-ontologies.com/assert.owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about=""/>
    <owl:AllDifferent>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#Alarm"/>
            <rdf:Description rdf:about="#Identify"/>
            <rdf:Description rdf:about="#Data_Update"/>
            <rdf:Description rdf:about="#Query"/>
        </owl:distinctMembers>
    </owl:AllDifferent>
    <owl:AllDifferent>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#Earthquake"/>
        </owl:distinctMembers>
    </owl:AllDifferent>
    <owl:AllDifferent>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#Get_Info"/>
            <rdf:Description rdf:about="#Identify_Machine"/>
            <rdf:Description rdf:about="#Pre-search"/>
        </owl:distinctMembers>
    </owl:AllDifferent>
    <owl:AllDifferent>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#Update_Data"/>
            <rdf:Description rdf:about="#Opsearch"/>
        </owl:distinctMembers>
    </owl:AllDifferent>
    <Tasks rdf:ID="Alarm"/>
```

```
<Tasks rdf:ID="Data_Update">
    <hasPremove rdf:resource="#Identify_Machine"/>
    <hasPremove rdf:resource="#Get_Info"/>
    <hasPostmove rdf:resource="#Update_Data"/>
</Tasks>
<Goals rdf:ID="Earthquake"/>
<PreTasks rdf:ID="Get_Info"/>
<owl:Class rdf:ID="Goals"/>
<owl:ObjectProperty rdf:ID="hasPostmove">
    <rdfs:domain rdf:resource="#Tasks"/>
    <rdfs:range rdf:resource="#PostTasks"/>
    <owl:inverseOf rdf:resource="#isPostmoveOf"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPremove">
    <rdfs:domain rdf:resource="#Tasks"/>
    <rdfs:range rdf:resource="#PreTasks"/>
    <owl:inverseOf rdf:resource="#isPremoveOf"/>
</owl:ObjectProperty>
<Tasks rdf:ID="Identify"/>
<PreTasks rdf:ID="Identify_Machine"/>
<owl:ObjectProperty rdf:ID="isPostmoveOf">
    <rdfs:domain rdf:resource="#PostTasks"/>
    <rdfs:range rdf:resource="#Tasks"/>
    <owl:inverseOf rdf:resource="#hasPostmove"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isPremoveOf">
    <rdfs:domain rdf:resource="#PreTasks"/>
    <rdfs:range rdf:resource="#Tasks"/>
    <owl:inverseOf rdf:resource="#hasPremove"/>
</owl:ObjectProperty>
<PostTasks rdf:ID="Opsearch"/>
<owl:Class rdf:ID="PostTasks">
    <rdfs:subClassOf rdf:resource="#Tasks"/>
    <owl:disjointWith rdf:resource="#PreTasks"/>
</owl:Class>
<PreTasks rdf:ID="Pre-search"/>
<owl:Class rdf:ID="PreTasks">
    <rdfs:subClassOf rdf:resource="#Tasks"/>
    <owl:disjointWith rdf:resource="#PostTasks"/>
</owl:Class>
<Tasks rdf:ID="Query">
    <hasPremove rdf:resource="#Pre-search"/>
    <hasPostmove rdf:resource="#Update_Data"/>
</Tasks>
<owl:Class rdf:ID="Tasks">
    <rdfs:subClassOf rdf:resource="#Goals"/>
</owl:Class>
<PostTasks rdf:ID="Update_Data">
    <isPostmoveOf rdf:resource="#Query"/>
```

```
        <isPostmoveOf rdf:resource="#Data_Update"/>
    </PostTasks>
</rdf:RDF>
```