J Supercomput (2010) 52: 97–118
DOI 10.1007/s11227-009-0266-9

Modeling high assurance agent-based Earthquake Management System using formal techniques

Sarmad Sadik · Alade Rahman · Arshad Ali · H. Farooq Ahmad · Hiroki Suguri

Published online: 21 February 2009

Abstract Disaster management systems are complex applications due to their distributed and decentralized nature. Various components execute in parallel with high need of coordination with each other. In such applications, interaction and communication issues are difficult to model and implement. In this paper, we have proposed agent-based Earthquake Management System (EMS) which is modeled and analyzed using formal approach. Traditionally, such systems undergo through various transformations starting from requirement models and specification to analysis, design and implementation. A variety of formal approaches are available to specify systems for analyzing their structure and behavior; however, there are certain limitations in using these techniques due to their expressiveness and behavior requirements. We have adopted combination of Pi-calculus and Pi-ADL formal languages to model EMS from analysis to design. The formal approach helps to enhance reliability and flexibility of the system by reducing the redundant information. It reduces chances of

S. Sadik (✉) · A. Rahman · A. Ali
NUST Institute of Information Technology, Rawalpindi, Pakistan
e-mail: sarmad@niit.edu.pk

S. Sadik
e-mail: ssadik@acm.org

A. Rahman
e-mail: rahmalade@niit.edu.pk

A. Ali
e-mail: arshad.ali@niit.edu.pk

H.F. Ahmad · H. Suguri
Communication Technologies, Sendai, Japan

H.F. Ahmad
e-mail: farooq@comtec.co.jp

H. Suguri
e-mail: suguri@comtec.co.jp

errors by explicitly mentioning working flow of information. Additionally, a prototype application is presented as proof of concept in EMS context. We have also evaluated our formal specification by using ArchWare and ABC tools; also, comparison of prototype application with major existing techniques is highlighted.

Keywords Modeling · Agent systems · Formal methods · High assurance

1 Introduction

A number of natural calamities have triggered alarm for developing more efficient and effective approaches towards disaster management systems. Floods, mudslides, storms, and earthquake are some of the major calamities. Earthquake, which is one of the most destructive of them, accounted for a major share. For instance, earthquake tragedies in the last four years include Bam, Asian Tsunami and Kashmir with a total of 267,000 lives lost among other after-effects [36].

Efforts have been directed to disasters management in general, and earthquakes in particular, so that the effects of natural disasters after their occurrence could be reduced [12, 35]. The existing approaches have proved to some extent inadequate in combating this natural debacle. The problems of such disasters include unpredictable incidences, remoteness and terrain inaccessibility of affected areas and, most importantly, untimely relief operations. Usually, there is a delay in providing these urgent services largely due to improper means and channel of information flow.

In a typical earthquake management, there is a need for earthquake monitoring, alerting the appropriate relief agencies and potential victims [33]. Figure 1 is a view of an encompassing level of activities in combating earthquake disaster, starting from sensing of hazards to the provision of relief supply, where agents are deployed to support such activities. In this paper, it is referred to as an Earthquake Management System (EMS). The proposed system has a number of processes which are executing in parallel with each other, like monitoring of earthquake, management of information, addressing external user queries, as well as coordinating the emergency service departments.

Autonomous decentralized systems' (ADS) [23, 24] paradigm is evolving since 1977 for high assurance and online reliable systems. Autonomous decentralized systems' (ADS) approach is well suited to address such complex applications where autonomous controllability and coordinability are major issues. The agent-based EMS is designed as an ADS application and its development process is highlighted from specification phase to implementation. Software agents have proved to be valuable resources as deployed in many critical mission projects [43]. Specifically, agents have been used in hazard detection [37], crisis management [2], risk management [25], space control [10], control of power grid [38] and prevention of bio-attacks [9]. In order to specify and model parallel and concurrent execution of processes, a combination of Pi-calculus and Pi-ADL is used, as this technique provides major support for such kind of concurrent and distributed applications [32]. This formal approach helps in modeling key aspects of proposed system in high detail and removes major chances of errors and redundant information loopholes. Additionally, European

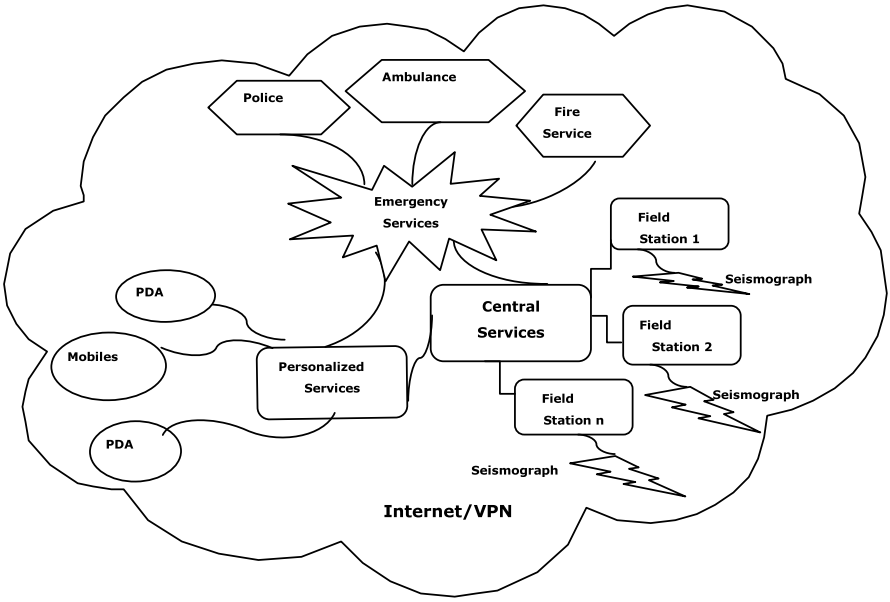

Fig. 1 Activities in earthquake relief operations

funded project ArchWare [6] toolkit provides support in order to specify the core architecture details of proposed system in Pi-ADL, and also helps to execute the specifications for validation.

As a proof of concept, we have made a prototype application in domain of earthquake management system (EMS). This new era of applications is differentiated by use of advance technologies to provide efficient solutions for this complex problem domain. These systems need intelligent coordinated behavior to facilitate human or working personnel. The proposed architecture is highlighted as an example scenario of distributed information search and retrieval.

The related work is described in Sect. 2. The proposed EMS architecture is presented in Sect. 3, while formal model using Pi-calculus and Pi-ADL notation are mentioned in Sects. 4 and 5, respectively. Proof of concept application is discussed in Sect. 6. Section 7 presents the evaluation of our approach. Conclusion and future work are presented in Sects. 8 and 9, respectively.

2 Related work

Researchers have used the concept of agents in disaster management systems emphasizing on the concept of teamwork [39]. However, the work was oriented towards generic emergency events like fire and explosion. The giant stride towards the development of the US earthquake monitoring [1] is limited in autonomy and requires human intervention. Also IBM has made an open source management system [34] for post-disaster events. It however requires manual usage and the autonomous features of the system are limited.

Other projects in the area of agent-based disaster management include SR emergency team for relief and evacuation with agents [27, 30, 41], and MAS planning and coordination in relief operation [11, 19]. In terms of emergency relief response, work in this regard includes Crisis Information Management Software [18] and an HLA-Based Multi-Agent System for Optimized Resource Allocation [13]. Yet, all these systems have no provision for active decision making based on prevailing situations.

Our work differs from these approaches as we have used the agent technology to produce autonomous and more efficient system in earthquake crisis management. Here, intelligent agents are deployed in monitoring, information sharing and data management in the event of an earthquake. This agent-based system will also support coordination and timely triggering of emergency services more rapidly than human operators. More importantly, the system benefits from the autonomous and adaptive nature of agents for efficient performances.

3 EMS architecture

An efficient architecture has been proposed that automatically and efficiently provides services like alerting, coordinating and collaborating in earthquake crisis. This system is based on software agents [32, 43]. Five major software agents to be involved in this particular system are illustrated in Fig. 2. These are Information Service Agent (ISA), Field Service Agent (FSA), Personalized Service Agent (PSA), Emergency Service Agent (ESA) and Personal Assistant (PA).

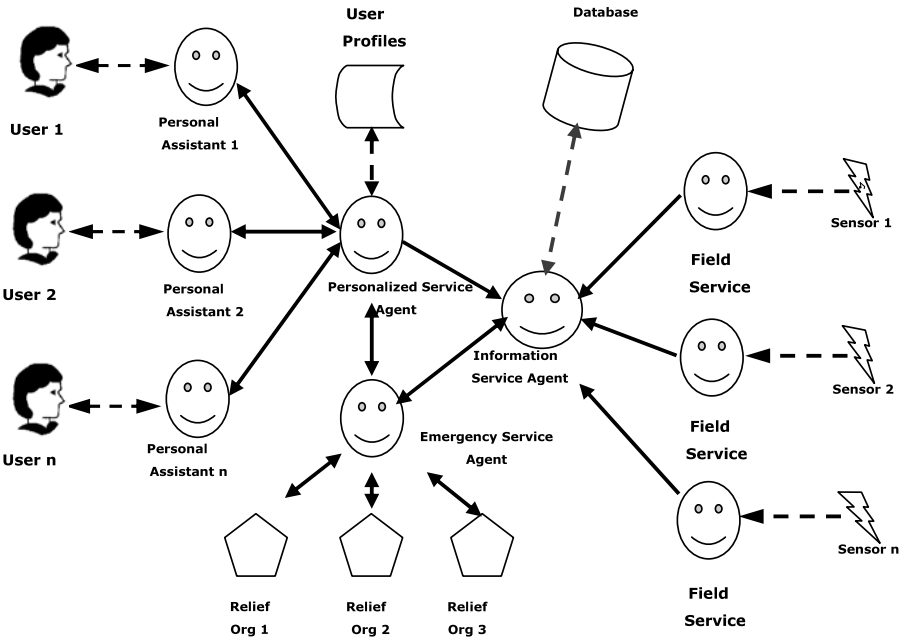

Fig. 2 EMS architecture

3.1 Roles of agents in the EMS

The roles of these agents are as described as follows.

3.1.1 Field Service Agent (FSA)

This agent is responsible for detecting wave range with the help of seismograph and continuously updates the Information Service Agent (ISA). It alerts the ISA of earthquake occurrence if the reading reaches a benchmark.

3.1.2 Information Service Agent (ISA)

It coordinates the readings from the Field Service Agents based on the received data. It identifies the disastrous area and informs the Emergency Service Agent (ESA) accordingly.

3.1.3 Emergency Service Agent (ESA)

It coordinates all the activities of the emergency services (police, health, fire) and other relief organizations. It alerts and gives appropriate instructions to the Personalized Service Agent based on information and directives received from the Information Service Agent.

3.1.4 Personalized Service Agent (PSA)

This agent personalizes the required services to the Personal Assistants (PAs). These services include language, communication and preferences among others. Information from the Personal Assistants for necessary update and verification is also sent to the Information Service Agent through the Personalized Service Agent.

3.1.5 Personal Assistant (PA)

This acts as a personal assistant to human operators. The messages from the Personalized Service Agent are transmitted through the PA for necessary emergency actions. Likewise, any observed and/or perceived report can also be sent to the Information Service Agent through the Personalized Service Agent. This PA can be deployed on PDAs, mobile phones and other ubiquitous devices.

The Database (DB) contains records of past occurrences and strategies employed, as well as the latest updates as sent by the Information Service Agent.

3.2 Operation scenario

The Field Service Agent detects wave range, sends the readings to the Information Service Agent and alerts it depending upon the benchmark reading of the seismograph. The Information Service Agent analyzes the data, consulting the database for optimum strategy to be adopted. If earthquake is confirmed, the Emergency Service Agent is alerted for prompt actions. The Emergency Service Agent coordinates the

relief organizations like police, fire stations and ambulances. Appropriate instructions are given to these organizations based on the information and directives received from the Information Service Agent. Likewise, the field service agent may issue alert during generation of the initial starting waves and big-scale waves which are defined as "P and S waves" [16] by the geologists. The Personalized Service Agent sends personalized information to the particular individuals through the Personal Assistant. The Personalized Service Agent also passes observed and/or perceived report from the Personal Assistant to the Information Service Agent. The Personal Assistant passes directives from the Personalized Service Agent to the individuals. After the operation, the Emergency Service Agent produces report of events that transpired to the Information Service Agent for necessary update of the database. With this and earlier information, the database is queried in analyzing the event for necessary public information release. These queries include death toll, hospitalization, property loss and missing people information.

The bold links in Fig. 3 is a direct communication among the respective agents while the dotted links are via the intermediate agents. The reports as requested by Personal Assistant can be sent back via the Emergency Service Agent. The relief organizations such as the police, fire and health services interact with the Emergency Service Agent. The directives may be issued for action plans by the Emergency Service Agent, and reports are sent back after each operation for necessary update.

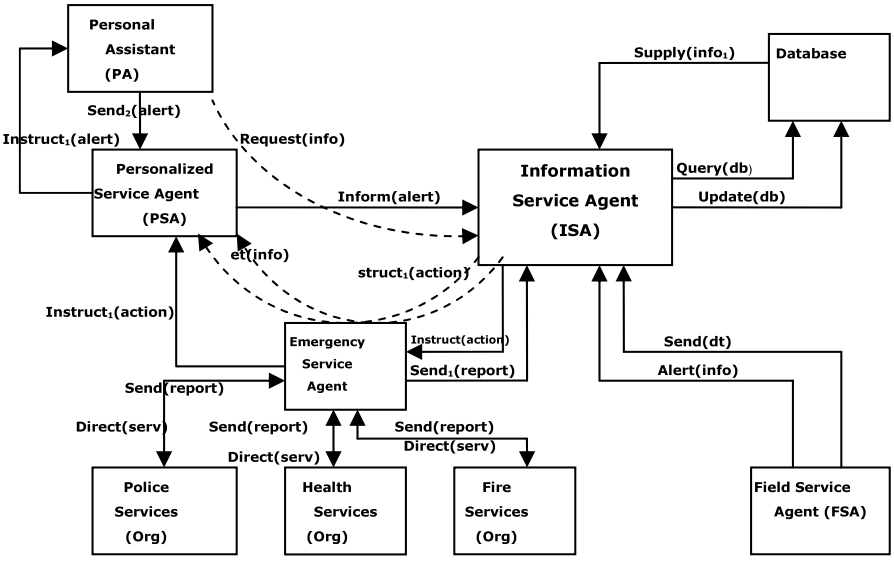

Fig. 3 Agents in EMS and their communication channels

4 Formal model using Pi-calculus

4.1 Pi-calculus notation

The Pi-calculus is a mathematical notation that can be used to describe process interactions not only sequentially, but also for parallel communication [20, 21, 26]. Its syntax consists of a set of prefixes and process expressions which can be used to represent communication between processes through channels.

Process is an independent thread of control while a channel is an abstraction of the communication link between two processes [42]. These interactions occur by sending and receiving of messages over channels. The summary of this syntax is described below. For any two processes P and Q, we have the following definitions as in Table 1, the details could be found at online resources [20, 21].

4.2 Agents description

As highlighted in Fig. 3, we modeled the agents and their interaction in the system as below.

4.2.1 Field Service Agent (FSA)

The Field Service Agent uses the channels of "Send" and "Alert" to send data and information from seismograph to Information Service Agent for necessary action. Information Service Agent receives the information and interacts with database using

Table 1 Syntax of Pi-calculus

Process	Representation	Explanation
Empty	0	The process where no action takes place.
Parallel	P|Q	The process comprising processes P and Q running in parallel.
Output	$\bar{a}\langle x\rangle \cdot P'$	The process that sends message x over a channel a and behaves as process P' afterwards.
Input	$a(x) \cdot P'$	The process that waits on channel a to receive a value bound to variable x and behaves as process P' afterwards.
Non-deterministic choice	P + Q	The process where either process P or Q runs.
Repetition	!P	Infinite number of processes P running in parallel.
Match	$[x = y]$P	The process that behaves as P provided x and y are the same. Otherwise nothing happens.
Restriction	(υx)P	The process that behaves as P. However, x is a local channel and can only be used for communication only between processes within the scope of P.
Silent Action	τ	Nothing observable happens, i.e., action without interaction with environment.

"Update" and "Supply" channels.

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

4.2.2 Information Service Agent (ISA)

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 received from other agents like the Emergency Service Agent and Personalized Service Agent, to the database through the "Update" channel using name db. Information Service Agent can also query the database for necessary information, for instance, post-earthquake reports along the "Query" channel. Feedbacks of queries and

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

strategies to be adopted in order to combat occurrences can be sent from the database to the Information Service Agent through the "Supply" channel.

4.2.3 Emergency Service Agent (ESA)

The Emergency Service Agent receives information from Personalized Service Agent. It also directs the emergency services for initiating required action.

$$\begin{aligned} ESA \stackrel{\text{def}}{=} & (Instruct_1(y) \cdot PSA' | \overline{Instruct}_1\langle action\rangle \cdot \overline{Direct}\langle serv\rangle \cdot \\ & ESA' | Send(x) \cdot \overline{Send}\langle report\rangle \cdot ESA') \end{aligned}$$

4.2.4 Personalized Service Agent (PSA)

In case of alert issued by Personal Assistant, Personalized Service Agent receives the information and sends alert to Information Service Agent on "inform" channel, as well as to Emergency Service Agent the required action which needs to be done.

$$\begin{aligned} PSA \stackrel{\text{def}}{=} & (\overline{Send}_1\langle alert\rangle \cdot PA' | send_1(x) \cdot \overline{inform}\langle alert\rangle \cdot PSA') \\ & + (\overline{Instruct_1}\langle action\rangle \cdot ESA' | Instruct_1(x) \cdot \overline{Instruct_2}\langle action\rangle \cdot PSA') \end{aligned}$$

4.2.5 Personal Assistant (PA)

PA receives instructions from Personalized Service Agent or it can send alerts to PSA for onward communication if field workers physically detect any catastrophe.

$$\begin{aligned} &PA \stackrel{\text{def}}{=} \\ &(\overline{Instruct_2}\langle action\rangle \cdot PSA' | Instruct_2(x) \cdot PA') \\ &+(\overline{Send_2}\langle alert\rangle \cdot PA' | Send_2(y) \cdot PSA') \end{aligned}$$

4.2.6 Database (DB)

Database receives information from Information Service Agent and updates the record if information is received on "Update" channel. In case of "Query" channel it searches the record and returns information on supply channel back to Information Service Agent.

$$\begin{aligned} &DB \stackrel{\text{def}}{=} \\ &(\overline{Update}\langle db\rangle \cdot ISA' | Update(x) \cdot 0) \\ &+(\overline{Query}\langle db\rangle \cdot ISA' | Query(y) \cdot \overline{Supply}\langle info_1\rangle \cdot DB') \end{aligned}$$

4.3 Sample scenario

The following scenarios are modeled.

4.3.1 Earthquake detection

This involves either the Field Service Agent sends readings with alert information, or the Personal Assistant sends a signal through the Personalized Service Agent and subsequent confirmation from the database. Earthquake is asserted if the Field Service Agent reading is equivalent to or greater than a benchmark figure.

$$\begin{aligned} &EDetection \stackrel{\text{def}}{=} \\ &\left((\overline{Send}\langle dt\rangle \cdot \overline{Alert}\langle info\rangle \cdot FSA')\right) \\ &+\left(\begin{array}{l}(\overline{Inform}\langle alert\rangle \cdot PSA' | Inform(x) \cdot \overline{Query}\langle db\rangle \cdot \\ Supply(info_1) \cdot ISA') \leftrightarrow Equal(x, info_1, Supply)\end{array}\right) \end{aligned}$$

4.3.2 Emergency service initiation

The Information Service Agent sends instruction as next line of action and the emergency services are alerted for actions. Personal Assistant through the Personalized Service Agent is instructed for necessary preparations.

$$\begin{aligned} &EServiceInitiation \text{ def} \\ &(\overline{Instruct}\langle action\rangle \cdot ISA') | (Instruct(x) \cdot \\ &\overline{Instruct_1}\langle action\rangle \cdot ESA' \equiv (\overline{Instruct}\langle instruct_1\rangle \cdot \\ &ESA' | Instruct \cdot (instruct_1)\overline{Instruct_1}\langle x\rangle \cdot ISA | \overline{Instruct_1}\langle x\rangle \cdot PSA') \end{aligned}$$

4.3.3 Service agencies' directives

Here, the Emergency Service Agent directs the emergency services organizations on actions to take.

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

4.3.4 Database

Information from both the Emergency Service Agent and the Field Service Agent is sent to the Database for update.

$$\begin{aligned} &DBUpdate \stackrel{\text{def}}{=} \\ &(\overline{Send}\langle report\rangle \cdot ESA' | Send_1(x) \cdot \overline{Update}\langle db\rangle \cdot ISA') \\ &+(\overline{Send_1}\langle dt\rangle \cdot FSA' | Send(y) \cdot \overline{Update}\langle y\rangle \cdot ISA') \end{aligned}$$

4.3.5 Report generation

The database can be queried for necessary information as may be required from time to time.

$$\begin{aligned} &CrisRptGen \stackrel{\text{def}}{=} \\ &\overline{Query}\langle db\rangle \cdot ISA' | Supply(info_1) \cdot 0 \end{aligned}$$

5 Pi-ADL specification

Pi-ADL (Architecture Description Language) [7, 8] is based on Pi-calculus and focuses primarily on the architecture description. It is richer in formal notation than Pi-calculus and provides support for specification of data close to implementation perspective. Pi-ADL specification comes under class of hyper-code which is later converted to "ProcessBase" language in ArchWare tool [6] for execution and verification of modeled system. Pi-ADL is architecture description language which supports to model architecture in more detail after its specification in Pi-calculus.

Pi-calculus helps to model the structure and behavior of system by explicitly defining the running processes, communication channels and information flow. We have taken major processes (critical services) and specified them in Pi-ADL using its standard syntax and semantics. The communication channels are represented as "via" keyword while input and output is identified by "send" and "receive" notations. Pi-ADL is used because it provides support for executable specifications (hyper-code) which are considered as specifications as well as prototype code for ArchWare tool in order to validate. However, these specifications and ArchWare tool have some limitations as compared with professional programming languages like Java, C++, etc., and are primarily used during the system development process for core architecture modeling, analysis and validation. We have also executed this Pi-ADL specification using ArchWare tool for proposed architecture validation. The errors during compilation time or in execution help to verify removal of loopholes or any other inconsistencies. Also a trace file is generated with ".tr" extension which records all the information flow of input and output.

5.1 EMS specification in Pi-ADL

Limited scenarios have been discussed in this paper due to space restraints. The first scenario presented is sending alert from Field Service Agent to Information Service Agent.

```
! Sending alert from FSA to ISA
{Value FS=abstraction ( )
{ Value fsend=connection (view[svalue:Integer] );
via fsend send view(svalue);
}
Value CS=abstraction( )
{ value calert = connection(Any);
Value csrecfs=connection(view [svalue:Integer] )
Via csrecfs=connection(view [svalue:Integer] )
If sval::svalue ≥ 6.5
Do CS_ES( ); }
Value AlertService = abstraction ( );
{ compose FS and CS;
fsend unifies csrecfs; } }
```

A scenario of calling emergency service is represented from Information Service Agent to Emergency Service Agent.

```
! Calling Emergency Service
value CS = abstraction()
    {value e = connection(view[service:String,location:String,description:String])
    via e send view(service,location,description)
    via e receive view(status)    }
value ES = abstraction()
    {    value ee = connection(Any),rr=connection(Any),hh=connection(Any),
pp=connection(Any),bb=connection(Any)
    via e receive y:view[serv:String,loc:String,desc:String]
if y::serv==‘Rescue’ do via rr send view(y::loc,y::desc)
if y::serv==‘Police’ do via hh send view(y::loc,y::desc)
if y::serv==‘Hospital’ do via pp send view(y::loc,y::desc)
if y::serv==‘FireB’ do via bb send view(y::loc,y::desc) }
value R = abstraction()
 {value r = connection(Any)
 via r receive y:view[location:String,description:String] }
value P = abstraction()
 {value p = connection(Any)
 via p receive y:view[location:String,description:String] }
value H = abstraction()
 {value h = connection(Any)
 via h receive y:view[location:String,description:String] }
value FB = abstraction()
 {value b = connection(Any)
```

```
  via b receive y:view[location:String,description:String] }
value callServiceComposite = compose { c1 as FS()
     and   c2 as ES() and   c3 as R()   and   c4 as P()
     and   c5 as H() and   c6 as FB()
   where { c1::e unifies c2::ee and       c2::rr unifies c3::r
            and c2::pp unifies c4::p  and c2::hh unifies c5::h
              and c2::bb unifies c6::b } }
```

Storing and retrieving database information is also highlighted as in the following.

```
{ value CS = abstraction( );
  { value update = connection(view[Id:Integer, Person_Name:String, Status:String,
City:String, Hospital:String]);
  value Query = connection(Any);
  value q=Any( );
  via Update send view(Id, Person_Name, Status, City, Hospital); or
  compose { via Query send q;
  via Query receive view (Id, Person_Name, Status, City, Hospital); }
  { value DB=abstraction ( );
    { via Update receive view (Id,Person_Name, Status, City, Hospital);
  Value db=location(0);
Via db send view(Id, Person_Name, Status, City, Hospital); }
or via Query receive q;
via db receive val:Any;
via Query send val; }
value AccessDB = abstraction( );
{ compose CS and DB; }
```

6 Proof of concept—example scenario

Agents take the job of human operators in providing services related to earthquake data. As agents receive a request to transfer or share some data, they search through database and send the results exhibiting a rapid mechanism for information sharing and distribution. In this paper, we have discussed an example scenario as a particular module of EMS (Earthquake Management System) by developing a prototype application where mobile agent searches information record of particular person on distributed machines highlighting an example of distributed information searching and retrieval.

The data about injured and dead persons is deployed separately on two machines as shown in Fig. 4. This represents an example under distributed information searching and retrieval scenario. In current case, mobile agent first performs a pre-search task which comes under category of pre-move task at source machine checking about status of a person and then on basis of its status injured or dead, it moves to related destination machine. The destination machine contains database of persons including their names, id and related city information. Users need to enter goal "Search" in this example after creating the mobile agent. The only pre-move task in this goal (pre-search) is to first search local machine to extract data about particular person whether

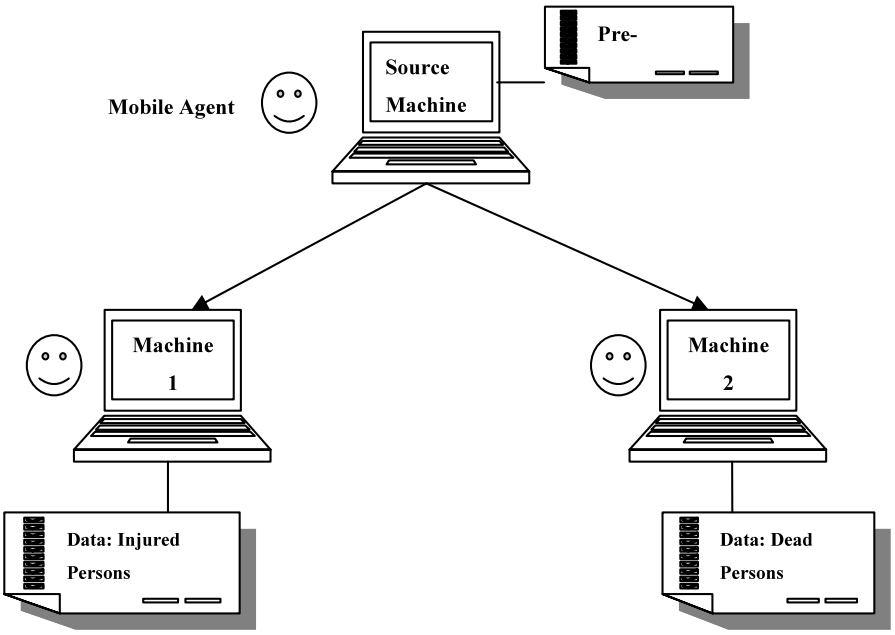

Fig. 4 Information searching and retrieval in EMS

injured or dead and move to particular destination machine. On destination, see the particular table and extract required values, analyze whether information is related to injured or dead person. Depending upon status, mobile agent moves to specified machine in order to find particular table and extract required values by executing SQL queries.

OWL was chosen for this system because it is standardized by W3C [40] and it is more flexible approach as compared to FIPA ontology [14] structure, especially in this particular case. We have represented the goals, pre-move tasks, move, and post-move tasks as classes in OWL. The particular sub-tasks are represented as individuals in these classes and their implementation details in a separate Java file. The relationships in policy structure are represented using OWL properties. The major advantage of using OWL is that the ontologies created in it can be easily published on web and easily accessible, especially by mobile agents executing on multi-agent systems on distributed machines. Also, it is convenient to add values for persistent storage which otherwise needs to be stored in database or hard-coded. Ontology is used primarily for representation of policies in which mobile agent can get the task structure of a particular goal and execute its mentioned tasks in specified order.

6.1 Implementation

We have implemented prototype application using FIPA [15] compliant SAGE multi-agent system [3, 4, 31]. Protégé [28] is used to implement these ontologies in OWL and later used the Protégé and Jena [17] APIs to access and manipulate these ontologies.

Fig. 5 Policy example in prototype application

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

A mechanism is created to separately define the tasks, which are actually executed and mentioned in sets of Pre-move and Post-move tasks. Each particular task has its implementation coded in a Java file.

Figure 5 shows the example of tasks and their relationships which are performed in case of earthquake. There are four major tasks: (1) sending alarm messages to related departments ("Alarm"), (2) identifying the seismograph attached with particular field station ("Identify"), (3) updating data as soon as it starts coming from one field station and department to others ("Data Update*"), and (4) querying data deployed in distributed way on various department machines ("Query*"). We have identified the last two tasks as major operations which need to be done using mobile agents.

In "Data_Update", the sub-tasks for mobile agent are to get the data (person name) which needs to be added or updated and identifying the particular destination machine. It migrates to target machine and executes the task of updating the data. In the "Query" task, the user while creating the mobile agent program enters his goal as "Search" in GUI after creating the agent. This goal is matched with the goal definitions in policy repository and as it matches the goal, the user is notified of loading already defined sets of tasks. The sub-task which is defined in Pre-move is "PreSearch". The "PreSearch" sub-task performs search at source machine. After successful execution, the control goes to Move. This "move" task serves as the mobility request. It takes the target machine information and mobile agent migrates to the destination machine using the multi-agent system support. In Post-move tasks, the OpSearch sub-task is added. OpSearch sub-task consists of executable code which performs search operation at destination machine according to given parameters and returns the results.

For instance, a mobile agent after performing "PreSearch" task at source machine, determines that the current status of specified person is "dead." It selects the destination machine where the database of dead persons is deployed among other categories and moves to destination machine. It performs "Opsearch" sub-task at a target machine, searches the local database and extracts out the related person's information including his ID number and related city. It converts the related information to ACL format and sends it back to parent host. All of these tasks contribute toward their goal and fulfilling the prime goal of the application.

7 Evaluation

The proposed system is modeled with Pi-calculus and Pi-ADL. This formal technique helps in elaborated specification, analysis, design and implementation for enhancing reliability and flexibility of proposed system. Other logic based languages, like propositional and predicate logic, description logic, Z notation, although support in defining the structural aspects of systems, however there are major limitations while expressing the behavior of systems, especially parallel and concurrent processes. Additionally, there are issues in explicit identification of communications channels and information flow, state of process after receiving new information, etc. Linda is also a modeling language which has five major operations—out, in, rd, inp and rdp—which are applied to tuple and tuplespace, however Pi-ADL is much more rich in notations and could support messages based on tuple, view, union, variant and quote. Also the value may be stored and retrieved using location while collection of values could be represented using set, bag, sequence.

We have also used Pi-calculus tools including ABC (Another Bisimiliarity Checker) [5] and MWB (Mobility Work Bench) [22] which are frequently employed for analyzing the syntax and semantics as well as verification of specified system [29]. ABC implements the equivalence checker of MWB and it used to analyze the Strong Bisimulation which checks the behavioral equivalence of two processes, as well as stepwise information flow analysis. This technique helps in identifying the redundant information and specification errors in order to increase the overall reliability of the proposed modeled system.

Firstly we defined the agents using the ABC tool [5] and then applied various commands for analysis and verification, as mentioned in Fig. 6.

```
abc > agent FSA(send,alert,update,supply) =
('send.'alert.FSA|send.alert.'update.supply.ISA)
Agent FSA is defined.

abc > agent ISA(send,alert,update,inform,query,supply,instruct) =
('send.'alert.FSA|send.alert.'update +
'inform.PSA|inform.'query.supply.'instruct.ISA|instruct.ESA +
'send.ESA|send.'update.ISA + 'instruct.ESA|'instruct.PSA)
Agent ISA is defined.

abc > agent ESA(instruct,direct,send) =
(instruct.PSA|'instruct.'direct.ESA + send.'send.ESA)
Agent ESA is defined.

abc > agent PSA(send,inform,instruct) = ('send.PA|send.'inform.PSA +
'instruct.ESA|instruct.'instruct.PSA)
Agent PSA is defined.

abc > agent PA(instruct,send) = ('instruct.PSA|instruct.PA +
'send.PA|send.PSA)
Agent PA is defined.

abc > agent DB(update,query,supply) = ('update.ISA|update.0 +
'query.ISA|query.'supply.DB)
Agent DB is defined.
```

Fig. 6

After defining the agents in ABC tool, as specified previously using pi-calculus, we have now analyzed their bisimilarity property by using command "eq". The proposed agents are not redundant and have their distinct well-defined roles and behavior.

We checked for all agents and the result showed no strong relation among the agents; however, due to space limitations, limited agent scenarios are discussed in Fig. 7 and Fig. 8.

It shows that all agents are not bisimilarly equivalent to each other. The system can also show representation of channels in x terms which is recorded and analyzed in all agent cases. For example, in case of PSA,

$$\text{abc} > \text{show PSA}(\backslash x0, x1, x2)\ (\ (\prime x0.\text{PA} \mid x0.\prime x1.\text{PSA}) + (\prime x2.\text{ESA} | x2.\prime x2.\text{PSA}\ )\ )$$

The "step" command shows the interactive behavior of system stepwise and information flow is analyzed. Firstly it provides the possible outcomes of given expression and then it follows the particular entered step number for further details.

In this way, behavior and interactive information flow could be analyzed for each particular agent, which helps in distinguishing the information channels and path traceability as well as eliminating redundancy and errors in specification and design.

```
abc > eq ESA PSA
The two agents are not strongly related (1).
Do you want to see some traces (yes/no) ? yes
traces of
ESA
PSA
-\%0->
-\%0->
(\x0,x1,x2)((('x0.PA | x0.'x1.PSA) + ('x2.ESA | x2.'x2.PSA)) |
'\%0.'in.ESA)
('in.PSA | '\%0.PA)
```

Fig. 7

```
abc > step PSA
1: { } => PSA --\%0-> ('in.PSA | '\%0.PA)
2: { } => PSA --out-> ('out.PSA | 'out.ESA)
3: { } => PSA --'\%0->
   (\x0,x1)((('x0.PSA | x0.PA) + ('x1.PA | x1.PSA)) | \%0.'in.PSA)
4: { } => PSA --'out->
   (\x0,x1,x2)((('x0.'x1.ESA | x0.PSA) + x2.'x2.ESA) |
out.'out.PSA)
Please choose a commitment (between 1 and 4) or 0 to exit: 1
1: { } => ('in.PSA | '\%0.PA) --'\%0->
   (\x0,x1)((('x0.PSA | x0.PA) + ('x1.PA | x1.PSA)) | 'in.PSA)
2: { } => ('in.PSA | '\%0.PA) --'in->
   (\x0,x1,x2)((('x0.PA | x0.'x1.PSA) + ('x2.ESA | x2.'x2.PSA)) |
'\%0.PA)
```

Fig. 8

Pi-calculus and Pi-ADL help to great extent in modeling such features in distributed and decentralized environments. ArchWare tool is also used to execute and validate the formal specifications. It also generates a trace file with ",tr" extension which records all the information flow including inputs and outputs. This helps to verify and validate the formal specifications of the proposed system.

7.1 Evaluation of prototype application

We have evaluated our approach using prototype of EMS application by comparing to traditional techniques used for performing similar kinds of operation. The traditional techniques are Web Pages, ACL Messages, and simple Mobile Agent strategy.

7.1.1 Web pages

One of the ways to query or add information at remote machine is using standard HTTP messages. HTML web pages need to be created manually and accessed using web browsers. However, it is a very static approach which has certain limitations, like users may need to browse a number of web pages to reach a specific category of information. Also, if one has to enter multiple data, he may need to reload the page repeatedly. The major issue is querying the information. The user may get only the static information from destination machine through fixed parameters on target web page. In this case, one needs to manipulate or search for related items. This is not a flexible solution. In our evaluation process, we created a website of EMS where the user needs to query and enter injured or dead persons' information.

7.1.2 ACL messages

Software agents may communicate with each other using ACL messages. In current case, when we need to make a large amount of transactions with remote database using ACL messages, the performance degrades rapidly. Also, it is difficult to express queries at runtime by users.

7.1.3 Mobile Agent (MA)

Mobile agent is a more efficient approach than ACL client/server messages as it can take the code with itself and execute it at destination machine. The SQL queries can be executed any number of times and information may be manipulated efficiently like searching or updating particular table, etc. However, creating and setting up tasks for a mobile agent may require more expertise for novice users as compared to using other alternative techniques because application code is more tightly constrained and difficult to program. This limitation may be addressed using ontology-based policy techniques.

7.1.4 Ontology Policy Mobility (OP-MA)

Policies containing goals and tasks arranged in specified order may be changed without affecting other code statements to avoid a ripple effect. But, the definition of the

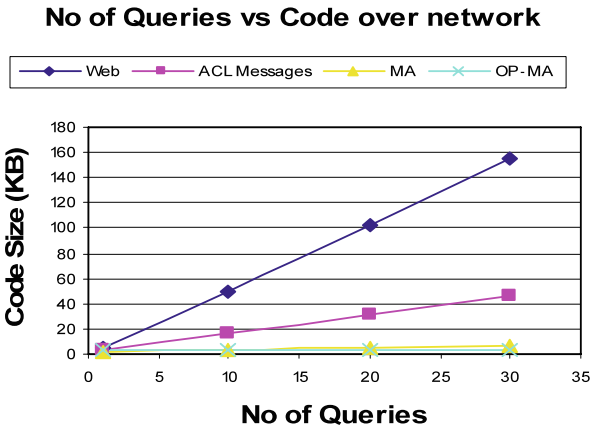

Fig. 9 Comparison of existing and proposed technologies for code movement over network

policy by user and interaction with policies by mobile agent may seem difficult task, especially if it needs to move onto different machines. Ontology-based technique to represent policy structure for a mobile agent is more suitable in these circumstances. The ontologies developed in OWL provide benefits of convenient creation as well as flexibility. The mobile agent can access particular ontology on various machines by giving URI of published ontology and extract the particular policy structure. Its representation in OWL ontology makes a more flexible solution as compared to other existing techniques where policy statements are represented in hard-coded form.

Figure 9 highlights the comparison of these technologies where number of queries to be executed is compared with amount of data moved over network. Two machines were used for the experiment with Sage multi-agent system deployed on them. The goal is to query about person's name at remote machine and extracting out details about its particulars like status, id and city. For web pages, a site was created using HTML and accessed through web browser. For ACL method, ACL message was sent and reply received accordingly in the same format. Mobile Agent was created and sent to destination machine along with code; it processed the code there and returned the results.

Mobile agent based on policy structure having reference of particular ontology was executed and results were received.

This shows that the ontology technique produces fewer amounts of data which is sent over network for completion of particular job. Also, in web pages, the change in requirements at client side needs a change in server side. In ACL case, if the message is changed upon new requirements, the receiver also needs to be updated. However, for mobile agent, the agent takes the code with itself and only execution support is required at the destination end.

8 Conclusions

In this paper, we proposed a formal approach for agent-based earthquake management system (EMS) as a proof of concept for developing ADS application. The formal technique justifies to be used for Autonomous Decentralized Systems in order to

improve the reliability and flexibility of the system by enhanced specification, analysis and design. EMS framework is modeled using the Pi-calculus and Pi-ADL formal methods in order to reduce redundant information by explicitly highlighting the information flow as well as process coordination for reliability and lower chances of error. The formal specifications have been evaluated by using ArchWare and ABC tools. This agent-based system would autonomously support human in detecting, coordinating and providing prompt and efficient relief services in the advent of earthquake. A prototype application is also discussed as a proof of concept in EMS domain. The policy structure representation of desired tasks in OWL ontology makes a very flexible solution as compared to traditional hard-coded statements. The evaluation of various existing strategies is made using prototype application of distributed information searching and retrieval in context of earthquake management system.

9 Future work

We intend to integrate it with a web-based application for convenient accessibility and usage. Also, there is a need to define the mechanism for collaborative behavior among groups of mobile agents. In the areas where an agent solely cannot achieve its goal, agents require teamwork with varying rationalities and working jointly to achieve a desired task in association and support of each other. Therefore, we intend to explore the teamwork architecture in context of policy-based systems.

References

1. Advanced National Seismic System (2008) www.anss.org
2. AGENTLINK (2005) Multi-agent systems in crisis management: the combined systems – case study. Dec. 2005
3. Ahmad HF, Suguri H, Ali A, Malik S, Mugal M, Shafiq O, Tariq A, Basharat A (2005) Scalable fault tolerant agent grooming environment—SAGE agent platform. In: 4th International joint conference on autonomous agents and multi agent systems (AAMAS). Demo, Utrecht, Netherlands, 2005
4. Ali A, Ahmad HF, Abbas Z, Ghafoor A, Mujahid R, Suguri H (2004) SAGE: next generation multi-agent system. In: Proceedings of the international conference on parallel and distributed processing techniques and applications, USA, 2004
5. Another Bisimilarity Checker (ABC) (2008) Online: http://lamp.epfl.ch/~sbriais/abc/abc_ug.pdf
6. ArchWare (2008) Architecting evolvable software. European RTD Project www.architecture-ware.org
7. Balasubramaniam D, Morrison R, Kirby GNC, Mickan K, Norcross S (2004) ArchWare ADL release 1 user reference manual. ArchWare Project IST-2001-32360 Report D4.3
8. Balasubramaniam D, Morrison R, Mickan K, Kirby G, Warboys B, Robertson I, Snowdon B, Greenwood RM, Seet W (2004) Support for feedback and change in self-adaptive systems. In: Proceedings of the 1st ACM SIGSOFT workshop on self-managed systems, USA, 2004
9. Carley M, Fridsma KM, Casman DB, Yahja E, Altman A, Chen NL-C, Kaminsky B, Nave D (2006) BioWar scalable agent-based model of bioattacks. IEEE Trans Syst Man Cybern (March)
10. Castillo L, Fdez-Olivares J, Gonz'alez A (2002) Shifting AI planning technology from automated manufacturing to autonomous operation and control in space missions. In: Workshop on AI planning and scheduling for autonomy in space applications, 2002
11. Chen W, Decker KS (2004) Managing multi-agent coordination, planning, and scheduling. In: AAMAS'04, 2004
12. Earthquake Management (2008) www.earthquakemanagement.net/

13. Fiedrich F (2006) An HLA-based multi-agent system for optimized resource allocation after strong earthquakes. In: Proceedings of the winter simulation conference, USA, 2006
14. FIPA Ontology Service Specification (2008) http://www.fipa.org/specs/fipa00086/XC00086C.html
15. FIPA (2009) Foundation for Intelligent & Physical Agents. http://www.fipa.org
16. Geology Labs Online (2008) http://www.sciencecourseware.org/VirtualEarthquake/VQuakeExecute.html
17. Jena (2007) http://jena.sourceforge.net/ 2007
18. Kashcroft J, Daniels D, Hart S (1970) Crisis Information Management Software (CIMS) Feature Comparison Report. NIJ Special Report. http://www.ncjrs.gov/pdffiles1/nij/197065.pdf
19. Lita L, Schulte J, Thrun S (2001) A multi-agent system for agent coordination in uncertain environments. In: Proceedings of the fifth international conference on autonomous agents, Canada, 2001
20. Milner R (1980) A calculus of communicating systems. In: LNCS, vol 92. Springer, Berlin
21. Milner R (1999) Communicating and mobile systems: the Pi-calculus. Cambridge University Press, Cambridge
22. Mobility Workbench (2008) Online: http://www.it.uu.se/research/group/mobility/mwb
23. Mori K (2004) Trend of autonomous decentralized systems. In: Proceedings of 10th IEEE international workshop on future trends of distributed computing systems (FTDCS), China, 2004
24. Mori K (2007) Autonomous decentralized systems for service assurance and its application. In: LNCS, vol 4526. Springer, Berlin.
25. Pan JI, Huang KJ, Lee YH, Yang CK, Shih S-Y (2007) An agent-based self-risk assessment and monitoring system for cardiovascular disease patients. In: Proceeding of Telehealth, 2007
26. Parrow J (2001) An introduction to the Pi-calculus, Handbook of process algebra. Elsevier, Amsterdam
27. Peña-Mora F, Mathias C (2004) AVSAR: a collaboration system for disaster search and rescue operations using autonomous vehicles. Int J IT Archit Eng Constr
28. Protégé (2007) http://protege.stanford.edu/ 2007
29. Puhlmann F (2007) Soundness verification of business processes specified in the Pi-calculus. In: LNCS, vol 4803. Springer, Berlin.
30. Rathi AK, Solanki RS (1993) Simulation of traffic flow during emergency evacuations: a microcomputer based modeling system. In: Proceedings of the winter simulation conference, 1993
31. Sadik S, Ahmad HF, Ali A, Suguri H (2007) Policy-based ontology framework for mobile agents. In: 6th IEEE international conference on computer and information science (ICIS07), Australia, July 2007
32. Sadik S, Rahman A, Ali A, Ahmad HF, Suguri H (2008) A formal approach for design of agent-based earthquake management system. In: Proceedings of the ninth ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing (SNPD2008), Thailand
33. Sadik S, Pasha M, Ali A, Ahmad HF, Suguri H (2006) Policy based migration of mobile agents in disaster management systems. In: Proceedings of IEEE international conference on emerging technologies, Pakistan, 2006
34. Sahana (2008) IBM Sahana is open source software available on Sourceforge.net. http://www.sahana.lk/
35. Scalem M, Bandyopadhyay S, Sinha S, Sircar A (2005) Decentralised Disaster Management Information Network (DDMIN): research discussions on the inherent technological and strategic implementation issues and proposed solutions. In: CISTM, India, 2005
36. Scaruffi P (2008) The worst natural disasters ever. Online www.scaruffi.com/politics/disaster.html
37. Schroeder BA, Schwan K, Aggarwal S (1997) Software approach to hazard detection using on-line analysis of safety constraints. In: Proc. IEEE symposium on reliable distributed systems, Oct. 1997, pp 80–87
38. Smathers DC, Goldsmith SY (2001) Agent concept for intelligent distributed coordination in the electric power grid, 2001. Online: http://certs.lbl.gov/pdf/sand00-1005.pdf. Accessed 2008
39. Tambe M, Bowring E, Jung H, Kaminka G, Maheswaran R, Marecki J, Modi PJ, Nair R, Okamoto S, Pearce JP, Paruchuri P, Pynadath D, Scerri P, Scerri N, Schurr N, Varakantham P (2005) Conflicts in teamwork: hybrids to the rescue. In: Proceedings of the fourth international joint conference on autonomous agents and multiagent systems (AAMAS), Netherlands, 2005
40. W3C (2007) http://www.w3.org/ 2007

41. Walle B, Turoff M (2007) Emergency response information systems: emerging Z-trends and technologies. Commun ACM (March)
42. Wing JM (2002) FAQ on Pi-calculus, December 2002. http://www.cs.cmu.edu/~wing/publications/Wing02a.pdf
43. Wooldridge M (2002) An introduction to multi-agent systems. Wiley, New York

Sarmad Sadik is a Ph.D. student in NUST School of Electrical Engineering and Computer Science. His research interests include teamwork architecture among software agents, ontology-based policies, as well as applying formal techniques for analysis and design of agent systems. He has presented his research work in various conferences in Netherlands, Japan, China, Australia and UAE. He is member of IEEE, ACM and ACIS.

Alade Rahman is a Ph.D. student in University of Lagos, Nigeria. He is currently under one year research fellowship programme at NUST School of Electrical Engineering and Computer Science, Pakistan under the supervision of Dr. H. Farooq Ahmad. He has research interests in web services and application of formal techniques in agent-based supply chain methods.

Arshad Ali is Professor and Director General at NUST School of Electrical Engineering and Computer Science (SEECS), Pakistan. He received Ph.D. from University of Pittsburg, USA. He has research interests in grid computing, network monitoring and management, as well as semantic grid. He is playing a key role in collaboration of SEECS with Stanford Linear Accelerator Center (SLAC) USA, Center of European Nuclear Research (CERN) Switzerland, University of West England (UWE) UK and California Institute of Technology (CALTECH) USA. He is a member of IEEE and ACM.

H. Farooq Ahmad is Specialist Engineer in Communication Technologies (Comtec), Japan. He received Ph.D. from Tokyo Institute of Technology in 2002. Additionally, he has joint appointment as Associate Professor at NUST School of Electrical Engineering and Computer Science (SEECS), Pakistan. His research interests include autonomous decentralized systems, multi-agent systems, autonomous semantic grid, health level 7 and semantic web. He is a member of IEEE and IEICE.

Hiroki Suguri is Vice President and Chief Technology Officer with Communication Technologies, which he co-founded in 2000. He received Ph.D. from Iwate Prefectural University in 2004. He has research interests in multi-agent systems, grid services and semantic grid. He received Certificate of Appreciation from Foundation for Intelligent Physical Agents in 2002 and Award for Key Role in Developing Research Linkages between Japan and Pakistan from Pakistan Software Export Board in 2004. He is a member of IPSJ, IEEE, ACM and IEICE.