Design and Formal Specifications of an Intelligent and Adaptive Multi-agent Interface Using Z-Specification Language

Abstract: *The continuous growth in processing and communication capabilities led to the massive distributed computational environments e.g. the Internet. These environments are often called as open environemnts being characterized by massive geographical distribution, high dynamics, no global control, lack of security and high heterogeneity. Building a distributed application for such an environment is a complex task. Research efforts have been directed to manage such complexities throught he development of new paradigms, theories and technologies for distributed applications. Within this context, software agents have received special attention due to its flexibility and potential use in various fields such as network management, e-commerce, active networks etc. However, there is a crucial lack concerning specification and development methodologies for the Multi Agent Frameworks (MAF). The main intent of this work is to present a generic and formal approach to represent MAF that fits in with prototyping and simulation of homogenous as well as heterogeneous domains.*

Dimple Juneja
M.M. Institute of Computer Technology
and Business Management (MCA)
M.M. University, Mullana (Ambala)

S.S. Iyengar
Dept. of Computer Science and Engg.
Louisiana State University,
Baton Rouge, Louisiana, USA

Keywords: *Software agents, homogeneous and heterogeneous domain, multi-agent frameworks*

1. INTRODUCTION

Agent community [1] is emerging as a new paradigm for modeling various complex but intelligent distributed systems. The richness of the agent metaphor is due to its ability to analyze, design and implement systems based upon the central notions of agents, their interactions and the environment which they perceive and in which they act. Although many Multi-Agent Systems (MAS) [2] have been designed, there is a crucial lack concerning specification and development methodologies.

To facilitate agent-based applications, a middleware infrastructure is needed to support mobility, security, fault tolerance, distributed resource and service management, and interaction with services. Such middleware makes it possible for agents to perform their tasks, to communicate, to migrate, etc.; but also implements security mechanisms to, for example, sandbox agents to prevent malicious code harm the local machine, or vice versa, protect an agent from tampering by a malicious host.

The current work proposes such a middleware that facilitates communication between heterogeneous domains and it adopts the Z specification language [3] for two major reasons. First, it provides modularity and abstraction and is sufficiently expressive to allow a consistent, unified and structured account of a computer system and its associated operations. Such structured specifications enable the description of systems at different levels of abstraction, with system complexity being added at successively lower levels. Second, Z schemas are particularly suitable in squaring the demands of formal modeling with the need for implementation by allowing transition between specification and program. Thus the approach used for formal specification is pragmatic. Moreover, Z is gaining increasing acceptance as a tool within the artificial intelligence community [4,5] and is therefore appropriate in terms of standards and dissemination capabilities.

The paper is structured as follows: Section 2 provides an overview of Z describing the basic notations to representing agents and their autonomy. Section 3 targets towards the work of distinguished researchers highlighting the need for formal specification methods. Section 4 presents the main intent of this paper i.e. proposes an agent-based middleware and its formal specification that can be used both as the basis of an implementation, and also as a precise but general framework not only for exiting homogenous and heterogeneous domains but also for further research.

2. Z-SPECIFICATION LANGUAGE

The Z formal specification language is grounded in mathematics and based on typed set theory and first-order logic [6]. It can model problems or systems as sets, relations and functions [7]. A Z specification can contain global variables, components of schemas, and local variables. Schemas group variables, restricting their values, and describing their static and dynamic aspects. The static aspects include two sides, namely the states occupied, and the in-variant relationship that are maintained as the system moves from one state to another state [8]. The dynamic aspects include three facets, namely the possible operations, the relationship between their in-puts and outputs, and the changes of the states. Morever, each schema has a declaration (or signature) part and a predicate part. A declaration declares or imports a set of variables. A predicate part specifies the relationships between values of variables. A framework to be specified is usually structured into four parts namely, environment, objects, agents and autonomous agents [9]. A formal model initially describes the environment and then, through increasingly detailed description, defines objects, agents and autonomous agents to provide an account of a general multi-agent system.

Table 1. Components of Formal Specification in Z

Component	Definition
Environment	Abstract description of the world, simply a collection of attributes
Object	Cluster of the attributes in the environment with defined capabilities i.e. action primitives which can be performed by the object leading to change in the state of that environment.
Agent	An object with goals i.e. greater functionality over objects
Autonomous Agent	Self-motivated agents, follow their own agendas

Considering the basic notion of Z-specification language, it is evident that the language is suitable for formally specifying a multiagent interface in a distributed environment. The significant property that Z specification language allows a structured specification to be written by describing a system at its highest level of abstraction with further complexity being added at each successive lower level of abstraction has formed the basis of this work. The upcoming section presents a survey of literature review in the related field and the formal model of the proposed work appears in the subsequent sections.

3. RELATED WORK

A middleware is an interface or software that consists of a set of enabling services that allow multiple processes running on one or more machines to interact across network. It is essential for client/server applications and also for communication across heterogeneous platforms. This section highlights the work of eminent researchers in the related field of agent technologies and formal specification methods. In fact, this section justifies the need of formal specification of agent based middleware by analyzing existing middleware designs and their formal specifications which have made an attempt to resolve various issues pertaining to open environments.

Authors [10] emphasizes that the specification process must provide the underlying rationale for the system under development and also shall guide subsequent design, implementation and verification phases. However, their work further identifies the issue that although there exists a variety of specification formalisms in the multi-agent environments however, such formalisms put the emphasis on the former role and do not provide a basis to fulfill the later.

Work in [11] states that the exiting methods are abstract and unrelated to concrete computational models. Authors in [12] made a good attempt to bridge the gap between the abstract and the concrete level by providing the system specification using a prototyping process. Carvalho et.al [13] proposed a method for open system infrastructure development considering risk analysis concepts to provide a structured way to specify, implement, monitor and maintain systems requirements however the authors failed to provide a conceptual framework to aid the assessment of existing alternatives. In order to narrow the gap between multi-agent formal modeling and multi-agent practical systems, Yu and Ca [14] proposed a novel architecture description language for MAS (ADLMAS) rooted in BDI model is proposed, which adopts Object-Oriented Petri nets but failed to incorporate learning ability of MAS. The authors[15] have reported formalization and prototype of a system of agents for generating reports consisting of references to the literature on several technical areas. Kacem and Kacem [16] made a good attempt by formally specifying MAS to provide a more concrete specification using CSP-Z. Miller and McBurney [17] proved the suitability of several TCOZ constructs in specifying multi-agent systems however their work shows no concern to undesirable behaviors.

A critical look at the available literature highlights the fact that there have been attempts to design agent-based middleware for homogeneous as well as heterogeneous domains very few researchers have targeted towards the formal but generic specification of multiagent frameworks that fits into prototyping and simulation oriented processes.

4. THE PROPOSED AGENT-BASED MIDDLEWARE

The key to all application in open and heterogeneous environments is the middleware architecture which uses a smart intermediary between traditional servers and heterogeneous clients. The fundamental driver for the proposed architecture is the inability of server to handle the incredible variation in software, hardware and networking capabilities of agents where agents are codes that are autonomous in the sense that they may migrate when it is necessary and to where they need. The upcoming section presents the high level view of proposed middleware and z-specification of all participating agents.

A. *High Level View and Ecology of Agents in Proposed Middleware*

Primarily, the proposed agent-based middleware comprises of six agents namely user agent, interface agents, Server Agent, Registry Agent, Sociological Agent and a finite number of task agents. Each agent is possessed with a set of capabilities that limits its contribution in achieving a goal but making it better domain oriented and thus increasing its efficiency. The domain set of capabilities of each agent is listed in table 2.

It may be noted that each individual agent may achieve a goal first through the appropriate use of its own capabilities and second through the successful exploitation of the capabilities of others. In both the cases, however an agent must be able to recognize existing relationships between other agents in order that it can successfully perform the relevant task either within the social constraints of the current situation or by altering those relationships without inadvertently and deleteriously affecting its own relationship with others. The high level view of the proposed middleware is shown in Figure 1 and is explained as follows:

1. The *user agent* gathers the information such as input message or goals to be achieved, host id and destination id from the user and forwards the complete set to interface agent.

Table 2. Agents and Their Responsibilities

Agent	Responsibilities
User Agent (UA)	UA provides an interface between user and Interface Agent, semantically enhances the user-input adds ontological terms, and requests a search to be performed.
Interface Agent (IA)	An Agent that receives all authenticates UA and incoming messages, acts as mediator between UA and server agent, convert requests to standard formal, analyses requests, Generate messageid, Output messages to destination.
Server Agent (OA)	Takes commands from IA and time stamp each goal in the order of their arrival and contacts registry agent for a particular task agent to adopt the goal and perform the task.
Registry Agent (RA)	Keep track of all agents that advertise their capabilities. It provides a unique id to all registered agents. Return the id of task agent that will actually serve the purpose but in case, no ideal task agent is being registered with RA, RA returns id of sociological agent that has the special ability to perform composition of services provided by task agents.
Sociological Agent (SA)	Combines services provided by various task agents to provide the service close to the request. It increases the probability of success of providing service if not accurate but relevant and close to user's request. The new service composed gets itself registered in the name of new TA with RA and is given a unique id.
Task Agents (TA)	Lowest Level Agents that actually provide the service requested. These can coordinate and cooperate with each other via CA.

2. *Interface agent* gets activated automatically and immediately calls the authentication module to authenticate user agent. In case, there are no goals pending, all agents remain in *sleep state*. This would lead to efficient resource utilization. If authenticated, converts the message to standard RDG format and generates a goal id. From this point on, the input message is treated as a new goal to be achieved. The goal is then forwarded to server agent for further processing. The specification for interface agent is depicted in Figure 2.

3. Now, the state of Server agent changes from sleep to *active* and it adopts the goal and in turn maps the same to the store. The store contains *the set of beliefs* i.e. the direct achievable goals in the form of knowledge base. This mapping is successful, it releases the goal to interface agent else forwards the goal to registry agent. The server agent is now free to process any other pending new goals, However, if no *allgoalsachieved* and server agent state changes to *sleep*. Figure 3 presents the formal specification of server agent.

4. Registry *agent* registers the task agent along with their defined capabilities, therefore it looks for the *task agent* with a similar capability. The state of the goal changes from *new goal* to *existing goal*. If the existing goal matches either exactly or can be plugged into the capabilities of and\ of the already existing task agents, the task is performed and task agent returns the result to destination id directly. However, if neither a matching capability is found nor an agent is free to take up the task, registry agent invokes *sociological agent* as it is possessed with composition abilities. Formal specification of registry agent and task agent is given in Figure 4 and Figure 5 respectively.

5. *Sociological agent,* checks the temporary log for the happening frequency and the priority of existing

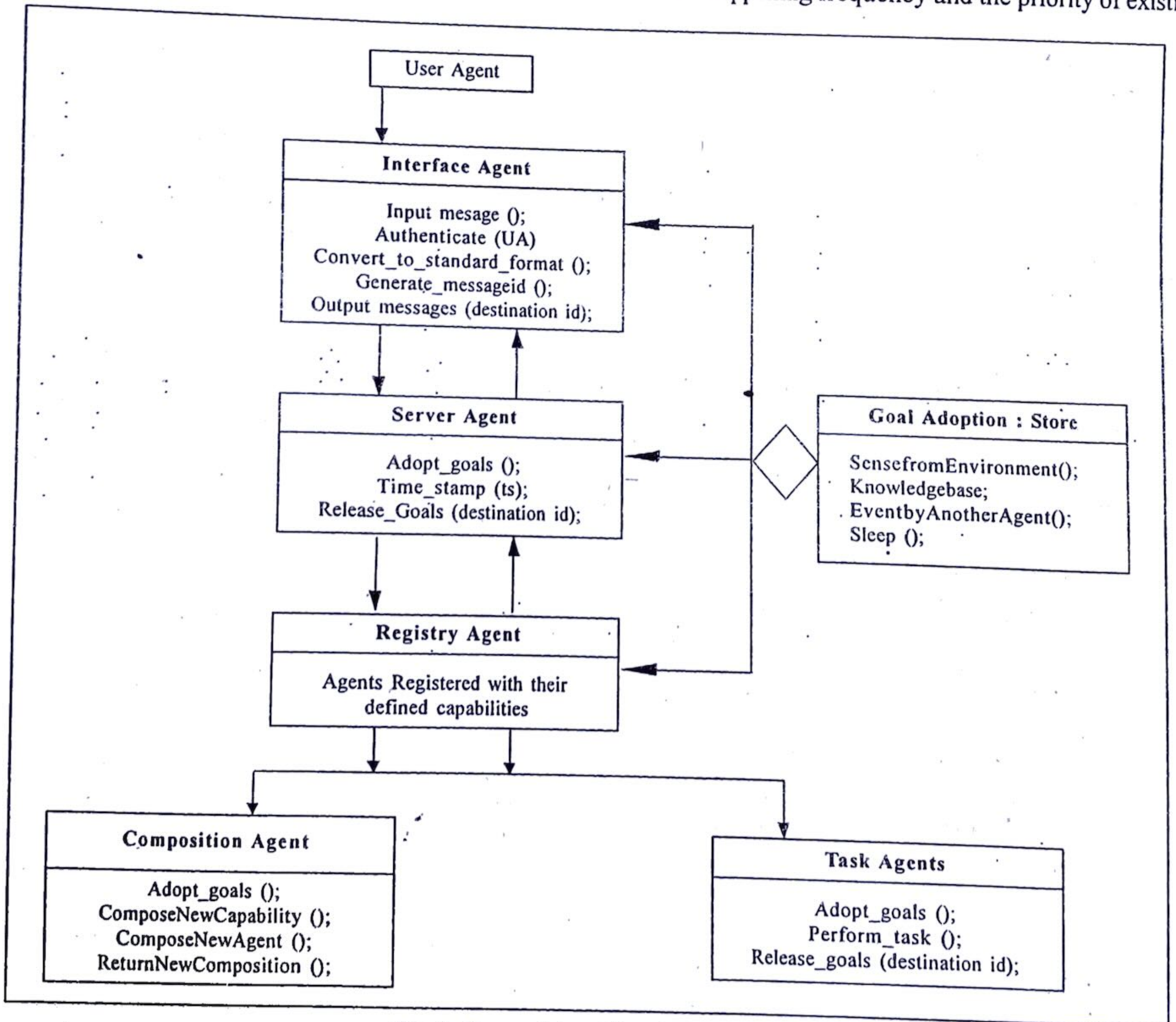

Fig. 1. Proposed Agent-based Middleware

```
InterfaceAgentPerception
Inputmessage, outputmessage: message
Useragent, Serveragent: Agent
Interfaceagentadoptsgoals, Interfaceagentfowardsgoal, Severagentadoptsgoal
Allgoalsachieved, Eventbyuseragent, Serveragentreleasegoals: Environment
goal: Goals
Activate, sleep: Action
-----------------------------------------
∀ im: Inputmessage; sa: severagent; ua: useragent; as action
    Eventbyuseragent ≠ { }
    ⇒ ∀ env: Environment; as: Action |
            as = Activate;
    ∃ iaag: interfaceagentadoptsgoals |
      if iaag.authenticate(ua)=True
            iaag.goal ∈ im.goal
            iaag.generategoalid(iaag.goal)
∀ env. Environment; as Action |
    env ∈ dom Interfaceagentforwardsgoal • as ∈ dom(Severagentadoptsgoal)
∀ env. Environment; as: Action |
    env ∈ dom Allgoalsachived ⇒ as=sleep;

∀ om: outputmessage, sa: severagent; ua: useragent, as: action
    Eventbyseveragent ≠ { }
    ⇒ ∀ env: Environment; as: Action |
            as = Activate;
    ∃ sarg: severagentreleasegoals ^ iaag: interfaceagentadoptsgoals |
    if iaag.authenticate (sa)=True
            iaag.goal ∈ sarg.goal
            iaag.releasegoalid (iaag.goal)
∀ env: Environment; as: Action |
            env ∈ dom Allgoalsachieved ⇒ as=sleep;
```

Fig. 2. An Interface Agent Perception

```
Sever Agent Perception
sa: serveragent
existinggoal, newgoal: Goals
Severagentadoptsgoal, severagentreleasegoals, Allgoalsachieved : Environment
Activate,sleep → Action
-----------------------------------------
∀ eg: existinggoal, env: Environment; as: Action |
∃ sa: severagent | eg.goals ≠ { }
∀ ng newgoal, env: Environment; as Action |
⇒  as = Activate
    env ∈ dom severagentadoptsgoals • as ∈ dom (timestamp (sa.ng))
    ⇒       sa.eg ∪ sa.ng ∈ sa.eg
∀ env. Environment, as: Action |
    env ∈ dom Allgoalsachieved • as ∈ dom (serveragentreleasegoals env)
    ∃ sarg: serveragentrelasagoals ^ iaag: interfaceagentadoptsgoals |
            sarg.goal ∈ iaag.goal
            ⇒ as = sleep.
```

Fig. 3. A Server Agent Perception

```
Registry Agent Perception
Interfaceagent, Severagent, SocialogicalAgent, taskagent: RegisteredAgents
InternalExecutableCapability, ExternalExecutableCapability Capabilities
RegisterNewCapability,    RegisterNewAgent,    ReturnDesiredTaskAgentId,
ReturnCompositionagentId : Environment
Activate, sleep →Action
-----------------------------------------
RegisteredAgents ≠ { } ^ InternalExecutableCapability ≠ {}
∀ nc:NewCapability, ra:RegisteredAgents; env. Environment; as: Action |
⇒ as = Activate
    env ∈ dom RegisterNewCapability |
    ⇒ ra:Capabilities ∪ ra.nc ∈ ra.capabilities v ra.Capabilities ∩ ra.nc={}
∀ na:NewAgent; env: Environment; as: Action |
    env ∈ dom RegisterNewAgent |
    ⇒ na ∪ ra ∈ ra
⇒ as = sleep
```

Fig. 4. Registry Agent Perception

```
Task Agent Perception
ActivationbyRegistry Agent,ActivationbySocialogicalAgent, Sleep: Action
Taskagentadoptsgoals, Taskagentreleasegoals, Allgoalsachieved: Environment
-----------------------------------------
∀ as: Action |
⇒ as=ActivationbyRegistryAgent ≠ {} v ActivationbySociologicalAgent ≠ {}
∀
Eventbyanotheragentactions ∪ sensefromenvironmentactions=perceivingactions;
Eventbyanotheragentactions ∩ sensefromenvironmentactions={};
dom taskadoptsgoals={Store};
∀ env.Environment; as: Action |
    env ∈ domtaskadoptgoals • as ∈ dom(taskadoptsgoalsofagents env)
    ⇒ as=Enventbyanotheragentactions;
∀ evn.Environment; as: Action |
    env ∈ dom taskadoptgoals • as ∈ dom(tasksensefromenvironment env)
    ⇒ as=sensefromenvironmentactions;

∀ env.Environment; as: Action |
    env ∈  dom taskadoptgoals = {} • as ∈ dom(Allgoalsachived env)
⇒  as=sleep;
```

Fig. 5. Task Agent Perception

goal. If the priority of the goal is high, it composes a new agent and for all other priority levels it simply generates a new capability and returns the same to registry leading to an increase in probability of achieving all goals and hence making the system more and more intelligent. The specification of agent incorporated is given in figure 6.

V. CONCLUSIONS AND FUTURE WORK

In this work a formal specification for an agent-based middleware that work in homogenous as well as heterogeneous domains had been proposed. The presented

```
Sociological Agent Perception
Serveragentgoal, taskagentgoal: Goal
DirectAdoption, ChainAdoption, Composition : Goal Adoption
ActivationbyRegistryAgent, Sleep→Action
─────────────────────────────
⇒ as: Action |
as=ActivationbyRegistryAgent
RegisteredAgents≠{}   ^InternalExecutableCapability≠{}
                      ^ExternalExecutableCapability={}
∀ ga: GoalAdotption • (ga.serveragentgoal ∈ serveragent)
                      ^ (ga.taskagentgoal ∈ taskagent)
if (ga.serveragentgoal ≠ ga.taskagentgoal)
    goal ∉ dom (GoalAdoption) • (ga.serveragentgoal ^ ga.taskagentgoal)
    ∀ sa: SociologicalAgent • goal ∈ sa.goal
∀ ag Activegoal • (dom activegoal ⊆ ra.capabilities) ^ (dom existinggoal ⊆ Goal)
^ (dom executablegoal ⊆ Composition)
∀ g: Goal, p : plan, env: Environment | (sociologicalagent g p env) ⊇ capabilities
∀ as Action |
as=sleep
```

Fig. 6. Sociological Agent

specification identifies and characterizes agents only. Initially, the proposed schemas are being specified at the highest level of abstraction and then, by incrementally increasing the detail in the specification, the system complexity has been added at appropriate levels. Most importantly, the use of Z in such a unique manner has provided a general mathematical framework within which different models, and even particular systems, can be defined and contrasted. In this work, formal definition for all participating agents allows a variety of different architectural and design views to be accommodated within a single unifying structure. All that is required in this specification is a minimal adherence to features of, and relationships between, the agents described therein. This work does not specify how the control structures should function, but instead how the control is being directed.

References

1. M.J. Wooldridge and N.R. Jennings. Intelligentagents: Theory and practice. *The Knolwedge Engineering Review,* 10(2): 115-152, 1995.

2. N.J.E. Wijngaards, B.J. Overeinder. M. van Steen, and F.M.T. Brazier, Supporting: Internetscale multi-agent system. *Data Knowledge Engineering,* 41(2-3): 229-245, 2002.

3. Michael Luck and Mark d'Inverno, "Structuring a Z Specification to Provide a Formal Framework for Autonomous Agent System". In Zum '95: The Z Formal Specification Notation, J. Bowen and M. Hinchey, (ed.), Lecture Notes in Computer Science, 967, 47-62, Springer-Verlag, Heidelberg, 1995.

4. I.D. Craig. The formal specification of ELEKTRA. Research Report RR 261. Department of Computer Science, University of Warwick, 1994.

5. R. Goodwin, Formalizing properties of agents. Technical Report CMU-CS-93-159. Carnegie-Mellon University, 1993.

6. Bowen, J.P. (2001): Experience teaching Z with tool and web support. ACM SIGSOFT Software Engineering Notes, 26(2), 2001, pp. 69-75.

7. Fangjun Wu,Tong Yi, "Measuring Z Specifications", ACM SIGSOFT Software Engineering Notes, September 2004 Volume 29 Number 5.

8. Spivey, J.M. (1992): The Z notation: a reference manual (second edition).London: Prentice Hall, 1992.

9. M. Luck and M. d'Inverno. A formal framework for agency and autonomy. In *Proceedings of the First International Conference on Multi-Agent Systems,* 1995.

10. Vincent Hilairel, AbderKoukaml, Pablo Gruerl,and Jean-Piere Miuller, "Formal Speci_cation and Prototyping of Multi-Agent Systems". In ESAW '000: Proceedingsof the First International Workshop on Engineering Societies in the Agent World, 114-127, Springer-Verlag, 2000.

11. Mark d' Inverno, Michael Fisher, Alessio Lomuscio, Michael Wooldridge. Formalisms for multi-agent systems. In FirstUK WOrkshop on Foundations of Multi-Agent Systems, 1996.

12. T. Lissajoux, V. Hilaire, A. Koukam, and A. Caminada. Genetic algorithms as prototyping tools for multi-agent systems: Application to the antenna parametersetting problem. In Springer Verlag. editor, Lecture Note in Articial Intelligence, number 1437 in LNAI, 1998.

13. G. Carvalho,R.Paes, R. Choren, and C. Lucena. Towards aRisk Driven Method for Developing Law Enforcement Middleware. 3rd Int. Workshop on Agent-Oriented Methodologies, OOPSAL '2004, Vancouver,BC, Canada, October 2004.

14. *Zhenhua Yu* and *Yuanli Cai,"* Object-OrientedPetrinets Based Architecture Description Languagefor Multi-agent Systems", IJCSNS International Journal of Computer science and Network Security, Vol. 6 No. 1B, January2006.

15. Torrii Murphy and Albert Esterline, "Formal Specification and Implementation of a Multi-agent Information System Using Schema Based Reasoning". Lecture notes in Computer Science.

16. Ahmed Hadj Kaceml and Najla Hadj Kacem, "From Formal Specification to Model Checking of MAS Using CSP-Z and SPIN", *International Journal of Computing & Information Sciences Vol. 5, No. 1, April 2007, On-Line.*

17. Miler, T. McBurney, P.: Multi-agent system specification using TCOZ. In MATES (2005) 216-221.