Indian Journal of Science and Technology Vol. 5 No. 7 (July 2012) ISSN: 0974- 6846

Formal modeling towards a dynamic organization of multi-agent systems using communicating X-machine and Z-notation

Ghulam Ali[1], SherAfzal Khan[2]*, Nazir Ahmad Zafar[3] and Farooq Ahmad[1]

[1]*Faculty of Information Technology, University of Central Punjab Lahore, Pakistan*
[2]*Department of Computer Sciences, COMSATS Institute of Information Technology, Attock, Pakistan*
[3]Department of Computer Science, College of Computer Science and Information Technology, King Faisal University, Al Hassa, Saudi Arabia
sherafzal@ciit-attock.edu.pk*, nazafar@kfu.edu.sa, {ghulamali,dr.farooq}@ucp.edu.pk

Abstract
The real world is a dynamic place where things change in an unexpected way. Software must be able to adapt these changes to work efficiently in the real world. Modeling of the multi-agent system implies modeling of the agent's dynamic structure and behavior, including their ability to communicate with other agent of the systems and dynamically organize their formation over time. In this research we used two different formal methods, communicating stream X-machine and Z notation, for writing the formal specification of multi-agent systems with a dynamic structure and behavior. Both the modeling techniques possess different characteristics which are discussed through the modeling process of multi-agent system. A case study of biologically inspired multi-agent system is taken to illustrate the proposed modeling approach.

Keywords*:* Communicating Stream X-machine; Z notation; multi-agent system; formal modeling.

Introduction

In recent years the multi-agent systems technology has generated lots of incitements, because of its promise as a new paradigm for conceptualizing, designing, and developing complex software systems. Developing stand-alone agent models is one of the basic techniques for modeling multi-agent systems. In multi-agent system modeling, a complex system is viewed as a large number of autonomous communicating entities. While modeling such systems the main focus is to identify the components of a system, to discover interactions among them and their local behaviors and global behaviors (Zhao*,* 2009). The global system behavior emerges from the local behaviors of the individual components, and their interactions as addressed in (Khalesian & Delavar*,* 2005). Multi-agent computing is being used to model the large number of multi-behavior distributed applications in different areas of real life such as supply chain management, workforce management, distributed computing, business process management, e-health care, knowledge discovery and knowledge sharing (Mitkas, 2005).

A number of methodologies for the development and modeling of multi-agent systems has been developed such as AUML, AML, MaSE, GAIA, X-Machine, and Stream X-Machine. Moreover, many formal software engineering methodologies and techniques are devised to develop multi-agent systems such as Finite State Machines and Petri Nets which capture the essential features, but fail to describe the system completely (Eleftherakis *et al.,* 2005). We briefly review the current state of play in the area of multi-agent software engineering and the proposed methodology of our research. The existing formal and informal agent-modeling techniques lack the ability to express efficiently the dynamics of the system. So far, there has been limited progress in developing formal modeling techniquesthat facilitate all crucial phases of correct system design, modeling and verification. Thus, the integration of communicating stream X-machine and Z notation is useful for describing the state space of a system, and facilitates flexible and intuitive modeling and verification of multi-agent systems. Communicating stream X-machine system provides a modular approach for system development and to establish communication between components of the system. The Z notation(Spivey, 1989) is a formal language used for specifying and modeling software and hardware systems(Khan *et al*., 2011; Khan *et al*., 2011b; Zafar*et al.,* 2011). Z comes with a standardized mathematical toolkit based on set theory and predicate logic based functions. It is one of the most widely used formal specification languages in formal methods.The existing communicating X-machine components can be used to build system model. In this research we investigate the way that how the existing communicating X-machine models can be used to model the multi-agent system that could be dynamically organized.

To develop multi-agent systems which can be dynamically organized, one needs a modeling technique that facilitates the development process as common in other disciplines. Recently, a number of formal modeling techniques and methodologies have been proposed for multi-agent systems modeling such as Multi-agent System Engineering (Deloach,1999), Geometry, Algebra, Informatics and Applications, X-Machine, and Stream X-Machine. Odell *et al.,* (2000) extended Unified Modeling Language to accommodate the distinctive requirements of agents.

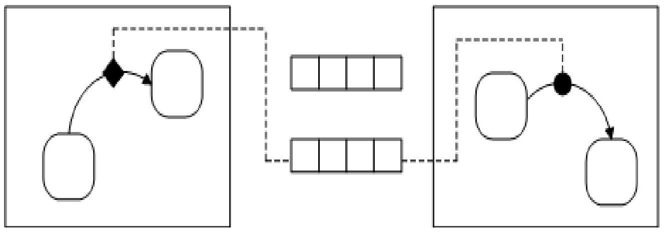
To handle the communication mechanism between multi-agents the Agent Communication Language was introduced (Chaibdraa *et al.,* 2002). Communicating X-machines are also introduced to model the systems composed of communicating agents (Stamatopoulou *et al*., 2005). Wooldridgeand Ciancarini (2000) discussed the tools used to develop multi-agent applications, the current issues in agent-oriented software engineering are described by Bordini*et al.* (2008), and future directions for multi-agent software engineering are addressed by Winikoff (2009). The issues in industrial acceptance of multi-agent technology are discussed by Akbari (2010).

Formal methods are also being used to model and specify the multi-agent systems. They are one of the most promising technologies for precisely specifying, modeling, and analyzing complex systems (Xudong *et al.,* 2006). Many formal models are developed for modeling and analysis of multi-agent systems such as Petri Nets (Celaya*et al.,* 2009). Marzougui *et al*. (2010) introduced an agent Petri nets to model communication and dynamic behavior of multi-agent system. A formal modeling framework for developing multi-agent systems with dynamic structure and behavior is presented by Kefalas (2005). Several attempts are made to use formal methods to specify different aspects of multi-agent systems. One of them was to formalize the Belief Desire Interest architecture using Z (Rao & Georgeff, 1995). Benerecetti *et al*. (1998) used model checking of multi-agent systems. Goodwin (1993) developed a framework using Z specification language to describe the agent's task and environments. Luck *et al*. (1997) described that how the formal agent framework can be refined and used to support the dynamic organization of multi-agent systems.

The main challenge in modeling dynamically organized multi-agent systems is managing their dynamic structure which may change dynamically. An explicit answer to this challenge is the use of some form of formal modeling techniques to automatically and systematically verify the completeness and correctness, as well as can model the dynamic structure and behavior of the system. In this research we use communicating stream X-machine system for modeling the stand alone communicating agent models. These models are organized as communicating X-machine systems, which are dynamically organized by invoking the rules to control the dynamic structure and behavior of the system. A state function is used to control the dynamic behavior of the systems it stores the current state, current memory status and last applied function of the agent.

Dynamic communicating X-machine system (CXMS)*:* It provides a mechanism in which a number of communicating components can communicate and dynamically organized. In dynamic CXMS the communication between the communicating stream X-machines is confirmed by means of multiple streams. The dynamic CXMS provides a common platform on which a number of communicating stream X-machines can communicate. Fig. 1 shows how two stream X-machines can communicate with each other. A set of rules is defined to control the dynamic configuration of the system. These rules are used to define the way that how the dynamic operators will be affecting the structure of the communicating system. For the dynamic configuration of the system, the system must know the current state, memory status and last function that have been applied by each agent.

Fig. 1. Communicating X-machines



Dynamic operators: A number of operators have been defined to ascertain the dynamic configuration of the communicating X-machine systems which are illustrated as follows:

Attachment operator: This operator is used to establish the communication between a set of CSXM components and existing communicating X-machine system (set of communicating X-machine components), i. e., ATT: $C \times Z \rightarrow Z'$.C is the set of standalone communicating X-machines and Z is the set of systems which can communicate with each other. Z' is the new communicating X-machine system which established different communication channels between communicating components C and communicating systems Z. Only the function φof the newly attached communicating X-machinehas been changed which can read from other component's input streams and can write on other component's output streams.Through this way the whole system Z' becomes collection of cooperating components, which can send and receive messages from other components.

Detachment Operator: This operator is used to remove the communication channels between a set of communicating X-machine components from a communicating X-machine system with which it currently communicates, i. e., DET: $C \times Z \rightarrow Z'$. C denotes the set of communicatingcomponents; Z is a communicating system,where C is a subset of Z. Z' is the new state of system Z without C. All the communication channels and relationships between C and its input, output streams with the remaining components of the system are removed.

Generation Operator: This operator is used to create and introduce a new communicating component into the system. If other components request to communicate with newly created component then communication channels are established, i. e., GEN: $C \times Z \rightarrow Z'$. C denotes the newly created component and Z denotes the existing system. Whereas Z' is the new state of the system.

Destruction Operator: To remove a particular communicating component from the system the destruction operator is used. it removes the communicating components from the system along with the communication channels, i. e., DES: C × Z → Z'. C denotes the recently removed components and Z is the old state of the system. Whereas Z' is the new state of system Z without.

X-Machine model of an ant agent: Here we take the biological inspired intelligent agent as a case study to model a multi-agent system. The model of ant agent is derived from (Stamatopoulou*et al*., 2007). The stream X-machine model of ant agent is illustrated in Fig. 2.

The stream X-machine model of ant agent consists of five states, which the ant can be in: *Inactive, Dead, Taking, Giving,* and *Hungry*. The state *Inactive* shows that the agent holding enough food quantity. The state *hungry*shows that the ant is hungry and it holding the food quantity below its hunger threshold. When a non-hungry ant meets a hungry ant it shares the food with it, and the hungry ant that meet the non-hungry ant receives food from it. The ant is in *Dead* state when food quantity dropped to zero. The memory of the agent holds its current location, the amount of food it holds, threshold level to indicate below that level the agent becomes hungry, food consumption rate that an ant consumed in a time unit, and the amount of food quantity to share with other agent that is hungry.

These stream X-machine models can communicate by directing the output of one X-machine function as input to another X-machine. In this scenario the X-machine models of an ant can communicate while sharing food. The communicating stream X-machine model of an ant is same except these functions, *give food, take enoughfood* and *take lessfood*. The function *givefood* shows that it gives a specified amount of food to a hungry ant while writing the output to communicating stream. Similarly the functions *take enoughfood* and *take less food* indicates that it receives a certain amount of food quantity from another ant by reading the input from a communicating stream. In Fig.3 the symbols ♦ and ● indicates an output and input message respectively.

Fig. 2. Stream X-machine model of an ant

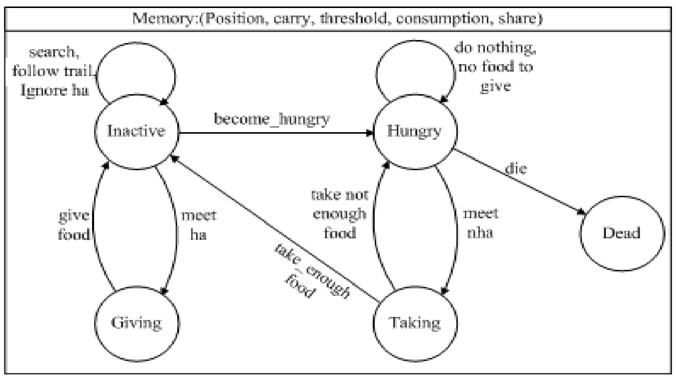
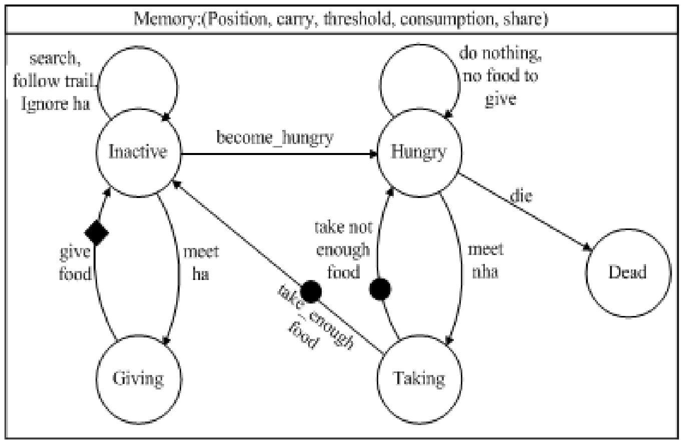

Memory:(Position, carry, threshold, consumption, share)
search, follow trail, Ignore ha
Inactive
become_hungry
Hungry
do nothing, no food to give
die
Dead
give food
meet ha
take not enough food
meet nha
take_enough food
Giving
Taking

Fig. 3. Communicating X-machine model of an ant


Memory:(Position, carry, threshold, consumption, share)
search, follow trail, Ignore ha
Inactive
become_hungry
Hungry
do nothing, no food to give
die
Dead
give food
meet ha
take not enough food
meet nha
take_enough food
Giving
Taking

*Formal modeling using Z:*In this section formal specification of X-machine models of an ant in Z notation is given.The agent models are: (a) dynamic communicating X-machine system, and (b) and dynamic structure and behavior of multi-agent system. The complete formal specification of abstract X-machine models SXM and CSXM which are used to specify these agent models are presented in our previous work (Zafar*et al*., 2010).

Design of communicating ant agent: We reused the formal specification of stand-alone ant agent *Ant* and communicating ant agent *CAnt* is presented in our previous work (Ali & Zafar, 2010). There is only need to define the dynamic communicating X-machine system.

Dynamic communicating X-machine system: A dynamic communicating X-machine system (DCXM) contains a number of CXMs that can exchange messages with other CXMs of the system. A DCXM is defined as $Z=((C_i)_{i=1,\dots,n}$, CR, R, GC) where: i) $C_i$denotes the current communicating component,ii) CR is a functionwhich shows the communication mechanism among a number of communicating X-machine components, i. e., $CR \subseteq C \times C$, where CR is a subset of relation C × C and $C = \{C_1,\dots, C_n\}$. A relation $(C_i, C_k) \in CR$ shows that the communicating X-machine component C_i can send a message to a reciprocal input stream of the communicating X-machine component C_k for any $i, k \in \{1, . . . , n\}$, iii) *R*denotes the set of rules that pertain to the configuration of the system, i.e. these rules defines the way that how dynamic operators will be affecting the organization of the system, iv) whereas*GC* contains the definitions of the existing communicating X-machine components as well as the components which may be added to the system. These definitions are called the genetic codes of dynamic communicating X-machine components.

The formal specification of abstract dynamic communicating X-machine system (CXMS) is described in schema Z where we introduced a variable C of type power set of CSXM to define a set of communicating stream X-machines. CR is a relation of type power set of *(CSXM × CSXM)*, where *c* and *c1* are communicating stream X-machines which can communicate with each other.

```
__CSXM ________________________________
ΔSXM
SISO: SigmaIn × Memory → SigmaOut × Memory
SIOS: SigmaIn × Memory → SigmaOut × Memory
ISSO: SigmaIn × Memory → SigmaOut × Memory
ISOS: SigmaIn × Memory → SigmaOut × Memory
is: seq (seq SigmaIn); os: seq (seq SigmaOut)
________________________________________
```

```
__Z____________________________________
C: ℙ CSXM
CR: ℙ (CSXM × CSXM)
________________________
∀c, c1: CSXM | c ∈ C ∧ c1 ∈ C ∧ c ↦ c1 ∈ CR ∧ c ≠ c1
  • (∃m, m1: c . memory; s: SigmaIn; g: SigmaOut
      | s ∈ ran c . ist ∧ g ∈ ran c . ost
      • (s, m) ∈ dom c . SISO ∧ (g, m1) ∈ ran c . SISO)
   ∧ (∃m, m1: c . memory; s: SigmaIn; g: SigmaOut
      | s ∈ ran c . ist ∧ g ∈ ran c1 . ost
      • ((s, m) ∈ dom c . SIOS ∧ (g, m1) ∈ ran c . SIOS))
   ∧ (∃m, m1: c . memory; s: SigmaIn; g: SigmaOut
      | s ∈ ran c1 . ist ∧ g ∈ ran c . ost
      • ((s, m) ∈ dom c . ISSO ∧ (g, m1) ∈ ran c . ISSO))
   ∧ (∃m, m1: c . memory; s: SigmaIn; g: SigmaOut
      | s ∈ ran c1 . ist ∧ g ∈ ran c1 . ost
      • ((s, m) ∈ dom c . SIOS ∧ (g, m1) ∈ ran c . ISOS))
________________________________________I
```

Invariants: For each *c* and *c1* of type CSXM where *(c, c1)* is in relation CR, there exists *m*, *m1* of type memory of machine c. Input and output alphabets *s* and *g* are of type SigmaIn and SigmaOut respectively. If *s* belongs to the range of standard input stream *ist*of machine *c* and *g* belongs to the range of standard output streams *ost* of machine *c* then the partial function *(s, m), (g, m1)* belongs to function *SISO*. Further if *g* belongs to the output stream of machine *c1* then the partial function *(s, m), (g, m1)* belongs to *SIOS*. Similarly, if *s* belongs to the input stream of machine *c1* then the partial function *(s, m),(g, m1)* belongs to *ISSO*, otherwise the partial function *(s, m), (g, m1)* belongs to *ISOS*.

Design of multi-agent system: A communicating multi-agent system consists of a number of communicating agents, which can exchange messages with agents of the system. A communicating multi-agent system is defined as MultiAgentSystem= ((Ci)i=1,..., n, CR) where:

```
__Ant__________________________________
ΔSXM
search, follow_trail,become_hungry, ignore_ha,
do_nothing,die, meet_nha, meet_ha: FUNCTION
take_enoughfood, give_food, take_less_food,
no_foodto_give:FUNCTION
________________________________________
```

```
__CAnt_________________________________
ΞAnt
ΔCSXM
________________________
SISO′ = {search, follow_trail, become_hungry, ignore_ha,
do_nothing, die, meet_nha, meet_ha, no_foodto_give}
SIOS′ = {give_food}
ISSO′ = {take_enoughfood, take_less_food}
ISOS′ = {} ∧ is′ = is ∧ os′ = os
________________________________________
```

```
__MultiAgentSystem_____________________
C: ℙ CAnt
CR: ℙ (CAnt × CAnt)
________________________
∀ant: CAnt; x, y, carry, thr, con, sh: ℤ | ant ∈ C ∧ x ⩾ 0 ∧ y ⩾ 0
∧ carry ⩾ 0 ∧ con ⩾ 1 ∧ thr ⩾ 1 ∧ sh ⩾ 1 • ant . q0 =Inactive ∧
ant . m0 = ((x, y), carry, thr, con, sh)
∀ant, ant1: CAnt | ant ∈ C ∧ ant1 ∈ C ∧ ant ↦ ant1 ∈ CR
∧ ant ≠ ant1 • (∃m, m1: ant . memory; iq: seq SigmaIn; oq:
seq SigmaOut; s: ant . alphaIn; g: ant . alphaOut |iq ∈
ran ant . is ∧ oq ∈ ran ant . os • s ∈ ran iq ∧ g ∈ ran oq
∧ (s, m) ∈ dom ant . SISO ∧ (g, m1) ∈ ran ant . SISO)
∧ (∃m, m1: ant . memory; iq: seq SigmaIn; oq: seq SigmaOut;
s: ant1 . alphaIn; g: ant . alphaOut | iq ∈ ran ant1 . is
∧ oq ∈ ran ant . os • (s ∈ ran iq ∧ g ∈ ran oq ∧ (s, m)
∈ dom ant . ISSO ∧ (g, m1) ∈ ran ant .ISSO))
∧ (∃m, m1: ant . memory; iq: seq SigmaIn; oq: seq SigmaOut;
s: ant . alphaIn; g: ant1 . alphaOut | iq ∈ ran ant .is ∧ oq ∈ ran
ant1 . os • (s ∈ ran iq ∧ g ∈ ran oq ∧ (s, m) ∈ dom ant . SIOS
∧ (g, m1) ∈ ran ant . SIOS))
________________________________________
```

(a) $C_i$is a i-th communicating agent, (b) *CR* is a function which defines the communication mechnanism between the ant agents, i. e., *CR ⊆ C × C* andC = {C_1,... , C_n}.A tuple *(C_i, C_k) ∈ CR* denotes that the *CAnt* component *C_i* can output a message to a corresponding input stream of the *CAnt* component *C_k* for any i, k ∈ {1, . . . , n}, i▪ k.

Invariants: (a) This specification shows that we are not defining all the agents from scratch, we just specify a single agent of one type and then we can create instances of these agents with different initial memory values and start state according to the requirement, (b) For each ant and ant1 of type CAnt where (ant, ant1) is in relation CR, there exists m, m1 of type memory of agent ant. Input and output alphabets s and g are of type SigmaIn and SigmaOut respectively. If s belongs to the range of standard input stream is of agent ant and g belongs to the range of standard output streams os of agent ant1 then the partial function (s, m), (g, m1) belongs to function SISO. Further if g belongs to the output stream of agent ant1 then the partial function (s, m), (g, m1) belongs to SIOS. Similarly, if s belongs to the input stream of agent ant1 then the partial function (s, m),(g, m1) belongs to ISSO, otherwise the partial function (s, m), (g, m1) belongs to ISOS.

The formal specification of Multi-agent system is described in schema *MultiAgentSystem* where we introduced a variable C of type power set of *CAnt* to define a set of communicating ant agents. *CR* is a relation of type power set of *(CAnt× CAnt)*, where *ant* and *ant1* are communicating ant agents which can communicate with each other. In the specification of communicating multi-agent system we reused the predefined communicating ant agent models to design the complete system, therefore, there is no need to model the communicating agent model from scratch.

To control the dynamic behavior of the systems it is required to remember the current state, current memory status and last applied function of the agent. To fulfill this requirement we define the schema S which is a set of 3-tuples S={qc, mc, fc}, where qc is the state in which C_i is in, mc is the memory value of C_i, and fc is the last function that has been applied in C_i.

S

$\Xi\ CAnt$
$qc: Q;$
$mc: Memory$
$fc: FUNCTION$

$qc \in states$
$mc \in memory$
$fc \in function$

SZ is a state of a system of communicating agents which is defined as a set of S components which contains the three elements the current state, current memory and last applied function.

SZ

$sz: \mathbb{P}\ S$

$\forall s: S \bullet\ s \in sz$

We specify the attachment operator in schema *Attachment* which establishes communication between two existing agents. It takes two communicating agents of type *Cant* as an input and establishes the communicating between these two agents.
This schema takes two agent *ci!*and*cj!* as input which belongs to C set of communicating agents and add to relation CR to establish the communication channels between them.

Attachment

$\Delta CSXMS$
$ci!: CAnt;\ cj!: CAnt$

$ci! \in C$
$cj! \in C$
$CR' = \{(ci!, cj!)\} \cup CR$

The scheme *Detachment* is defined to remove the communication channels of an existing agent from the system. It takes an agent of type *CAnt*as and input and verifies it that it must be an existing agent and removes its communicating channels from the system.

Detachment

$\Delta CSXMS$
$c!: CAnt$

$c! \in C$
$CR' = \{c!\} \lhd CR$
$CR' = CR \rhd \{c!\}$

The *Generation* schema creates a new agent of type *CAnt*and added this agent into the system. The variable c! is defined to show that it is an input and it is not a part of the system. It does not belong to the set of communicating agents of the systems, which means that it is a newly created agent.

Generation

$\Delta CSXMS$
$c!: CAnt$

$c! \notin C$
$C = \{c!\} \cup C$

The destruction schema is defined to remove an existing*CAnt* agent from the system along with all the communication channels which allow it to communicating with the other agents of the system.

Destruction

$\Delta CSXMS$
$c!: CAnt$

---

if $c! \in C$
then $CR' = \{c!\} \lhd CR \wedge CR' = CR \rhd \{c!\} \wedge C' = C \setminus \{c!\}$
else $C' = C \setminus \{c!\}$

Prior to the removal of an agent from the systems, we have to remove its communicating channels from the system and then we remove the agent from the system.

Conclusion

This dual modeling approach supported the behavioral modeling, data modeling and property analysis of the multi-agent systems. We have used the Z notation in a modular fashion in such a way that to specify the communicating X-machine we have reused the predefined specification of X-machine. In communicating X-machine, communication and data are considered as two separate distinct activities which provide the benefit of reusability of stream X-machine models. The dynamic communicating X-machine system facilitates the designer to reuse the predefined communicating stream X-machine models to design the complete system, therefore, there is no need to model the communicating X-machine components from scratch. These X-machine models are alsoreused to specify the stand-alone agent, communicating agent and a complete multi-agent system consisting of communicating agents. The rebuilding of any multi-agent system requires only modeling the communication part of the system. The use of rules invokes the operators to control the dynamic structure and behavior of the system. These X-machine models and operators are specified and verified using Z notation. The developed formal agent models based on communicating X-machines are also verified and analyzed using computer tool Z/EVES.

This relationship has reduced the implementation issues of X-machine models. These models will be used for formal testing of the multi-agent systems to prove the correctness of the model. Further, it will increase the confidence in correctness and completeness for the construction of multi-agent systems.

Reference

1. Akbari OZ (2010) A Survey of agent-oriented software engineering paradigm: Towards its industrial acceptance. *J. Comput. Engg. Res.* 1(2), 14-28.
2. Ali G and Zafar NA (2010) Formal modeling of multi-agent systems using communicating stream X-Machine and Z Notation. *Proce. Intl. Conf. Intelligence & Info. Technol.*pp: 523-527.
3. Benerecetti M, Giunchiglia F and Serafini L (1998) A model checking algorithm for multi-agent systems. *Proce. 5thIntl. Workshop Intelligent Agents V, Agent Theories, Architec. & Languages.*pp: 163-176.
4. Bordini RH, Dastani M and Winikoff M (2008) Current issues in multi-agent systems development. *Proce. 7thInt. Joint Conf.Autonomous Agents & Multiagent Sys.,* ACM.pp: 233-240.
5. Celaya JR, Desrochers AA and Graves RJ (2009) Modeling and analysis of multi-agent systems using petri nets. *J. Comput.* 4(10), 981-996.
6. Chaibdraa B and Dignum F (2002) Trends in agent communication language. *Comput. Intelligence*. 18(2), 89-101.
7. Deloach SA(1999) Multiagent systems engineering. *Methodol. & Language For Desig. Agent Sys., Agent-Oriented Info. Sys.,*Seattle WA, USA.
8. Eleftherakis G, Kefalas P, Sotiriadou A and Kehris E (2005) Modeling biology inspired reactive agents using x-machine. *Proc. World Acad. Sci., Engg.& Technol*.pp: 63-66.
9. Goodwin R (1993) Formalizing properties of agents. *CMU-CS*.pp: 93-159.
10. Kefalas P, Stamatopoulou L and Gheorghe M (2005) A formal modelling framework for developing multi-agent systems with dynamic structure and behaviour. *Proc.4thInt. Central & Eastern Europ. Conf. Multi-Agent Sys. & Appl.* pp:122-131.
11. Khalesian M and Delavar MR (2008) A multi-agent based traffic network micro-simulation using spatio-temporal GIS. *Intl. Archives Photogrammetry, Remote Sensing & Spatial Info. Sci.* 12(7), 31-35.
12. Khan SAand Zafar NA (2011) Improving moving block railway system using fuzzy multi-agent specification language.*Int. J. Innovative Computing, Info.& Control*. 7(7(B)), 4517-4534**.**
13. Khan SA, Zafar NA and Ahmad F (2011b) Petri net modeling of railway crossing system using fuzzy brakes.*Intl. J. Physical Sci.* 6(14), 3389-3397.
14. Luck M, Griffiths N and d'Inverno M (1997) From agent theory to agent construction: a case study. *Proc. 3rd Int. Workshop Agent Theories*, Architec. & Languages.pp: 49-63.
15. Marzougui B, Hassine K and Barkaoui K (2010) A New formalism for modeling a multi agent systems: agent petri nets. *J. Software Eng.& Appl.*3, 1118-1124.
16. Mitkas P (2005) Knowledge discovery for training intelligent agents: methodology. tools and applications.*Autonomous Intelligent Sys., Agents &Data Mining, Lecture Notes in Comput. Sci.*3505, 2-18.
17. Odell J, Parunak HVD and Bauer B (2000) Extending UML for agents. *Proce. Agent Oriented Info. Sys. Workshop (AOIS) 17th National Conf. Artificial Intelligence*. pp: 3-17.
18. Rao AS and Georgeff M (1995) BDI-agents: from theory to practice. *Proce. 1stInt. Conf. Multiagent Sys.*pp:312-319.
19. Spivey JM (1989)The znotation.*Reference Manual. Englewood Cliffs, NJ, Prentice-Hall.*
20. StamatopoulouI, Gheorghe Mand Kefalas P (2005) Modelling of dynamic organization of biology-inspired multi-agent systems with communicating x-machines and p systems. 5th Workshop in Membrane Computing, Springer-Verlag, Berlin, Milan Italy. 3365,389-403.
21. Winikoff M (2009) Future Directions for agent-based software engineering. *Int. J. Agent-Oriented Software Eng.* 3(4), 402-410.
22. Wooldridge M and Ciancarini P(2000) Agent-oriented software engineering: the state of the art.*1stInt. Workshop on Agent-Oriented Software Eng.*pp:1-28.
23. Xudong H, Huiqun Y and Ding Y (2006) Formal methods for specifying and analyzing complex software systems. *Modern Formal Methods & Appl., Springer.* pp: 123-150.
24. StamatopoulouI,Sakellariou I,Kefalas P and Eleftherakis G (2007) Formal modeling for in-silico experiments with social insect colonies.*Proc.11thPanhellenic Conf. Info.*pp: 79-89.
25. Zafar NA and Ali G (2010) Transformation of X-machine to Z notation enhancing modeling power for distributed systems. *Proc. Int. Conf. Software Engg. Theory & Practice.* pp: 54-61.
26. Zafar NA, Khan SA and Araki K (2012) Towards the safety properties of moving block railway interlocking system. *Int. J. Innovative Comput., Info & Control.* (Accepted) 8(7).
27. Zhao Cand Wang L (2009) Computational modeling of complex systems: case study in real world. *Proc. 9th Int. Conf. Hybrid Intelligent Sys.*pp: 33-38.