Constraint Specification in Multi Agent System

[1]Mangilal Sharma, [2]Mauajama Firdaus, [3]Rajib Kumar Chatterjee, [4]Anirban Sarkar
[1, 2, 3] Computer Centre, National Institute of Technology Durgapur 713209, India
[4] Department of Computer Applications, National Institute of Technology Durgapur 713209, India
{mangi.sharma25@gmail.com, mauzama.03@gmail.com, chatterjee.rajib@gmail.com, sarkar.anirban@gmail.com}

Abstract—Multi-Agent System (MAS) consists of innumerable independent and active units, called agents, which collaborate with each other to attain pre-determined goals or delegated objectives. Agents being uncoupled, scattered, event-driven and self-motivating in nature any MAS is distributed and large scale. Hence the analysis of MAS are intricate and crucial. This work proposes formal analysis method for defining and specifying properties, resources, knowledge and related constraints of an agent in a MAS. Agents need to reason about the interactions through constraints. These constraints needs to be satisfied for invoking an event. Definite templates to exemplify these aspects of MAS are generated for proper analysis. To justify the expressivity of the proposed methodology an illustration have been considered. The proposed approach has been mapped graphically to an extended version of Multi-Agent System Architecture Graph (Ex-MASAG). Lastly, a software tool named drawFAM has been proposed to design the proposed Ex-MASAG.

Keywords—Agent; MAS; Multi-Agent System; Constraints; Resources; Properties; Knowledge; Design Tool

I. INTRODUCTION

In recent years, the characteristics of large and complex software has been derived by an easy and better approach called Agent Oriented Paradigm. Designing and developing application by using autonomous, dynamic and event-driven software entities referred to as agents is the main idea behind Agent Oriented Systems. In the absence of continuous user interaction autonomous entities called agents are able to perform their task [1]. In recent literatures like [2, 3, 11, 13] the concepts of MAS has been properly discussed for the better perception of the agents and its role in a MAS.

Analysis of certain aspects of agents namely constraints, properties, resources and knowledge in a MAS is a significant issue towards efficient design and development of such systems. In relation to MAS, the term constraints means the condition that needs to be satisfied by an agent to get its pre-specified objectives. The state dependent properties that must hold at any state is expressed by a static constraint. *Whereas* a dynamic constraint refers to the conditions that vary from one or more states. Using these constraints, agents in MAS are able to reason during collaboration to interact successfully to attain the goal. Properties of an agent usually refers to the basic trait of an agent in MAS. The term Knowledge refers to the set of information necessary and crucial for the proper understanding and development of a MAS. The theoretical understanding of a MAS constitutes explicit knowledge and the practical perception is referred to as implicit. Resources are present in the environment in which an agent resides. These environmental resources are necessary to an agent to deliver the defined service. A agent can carry on systematic reasoning only if the insight of the resources, property and knowledge is proper. For making the local decisions constraints related to property, resources and knowledge is to checked and satisfied. In domains like distributed scheduling, distributed resource allocation agents must reason about the interactions between their local decisions in order to obtain good global performance. In order to symbolise these types of automated reasoning problems, researchers in MAS have suggested the concept of constraints as a key to the paradigm. Agents that collaborate successfully needs reasoning about interactions in a collaborative but decentralized manner. between individual agent decisions.

Some recent literatures [4, 5, 6, 7, 8, 9, 10] have dealt with the specification of knowledge, resources, property and constraints in MAS. In [6], Knowledge Management lifecycle model was proposed consisting of six phases but no formal analysis of knowledge was done. Agent Knowledge Interchange Mechanism was proposed in [7] which provided a robust method using diverse tools for collaboration. Proposal [10] was inspired by the perception of different methods in knowledge sharing where an agent could design a global model from dispersed local models held by individual agents in a MAS. In [8], emphasis was on resource coordination in multi-agent systems with no focus on analysis of resources of an agent in a MAS. Grid Resource Management method for a resource-cost based MAS was proposed in [9]. Literatures like [4, 5] help in better understanding of constraints as a whole. In [4], a method to handle dynamic constraints using XML updation was proposed. But this work did not considered all types of dynamic constraints. Proposal [5] presented a graphical modelling notation but limited to Schematron for expressing dynamic constraints. However a very few literatures have focussed on the specification of constraints as some conditions to be checked before providing some service. For effective collaborations agents in MAS need to reason about the interactions between them. The local decisions made by agents and the interactions between them can have significant implications on the performance of the MAS as a whole. Formal analysis of MAS and its active components was done partially in [14] where these issues of constraints, resources were overlooked.

Hence this paper proposes a logic based formal approach for the identification and analysis of constraints, properties, resources and knowledge of an agent in a MAS which is needed for the complete and proper analysis of MAS. An agent may need to satisfy some constraints about the interactions to invoke an event. Some inherent properties of an agent, environmental resources and some knowledge is required to fulfil and reason the constraints and deliver the defined service. The templates generated has been mapped graphically to an extended version of Multi-Agent System Architecture Graph (MASAG) [16] called as Ex-MASAG. A suitable case study has been taken to illustrate the proposed concepts. A design tool *drawFAM* has been developed to efficiently map the various templates obtained after analysis of MAS to Ex-MASAG.

II. MAS Conceptual Architecture and MASAG

A. *MAS Conceptual Architecture*

MAS Conceptual Architecture [14] in general defines the agents and their inter relationship. This is required to hypothesise the environmental entities, agents, their events, collaborations and interactions in the MAS.

An agent Ag in such an environment *Env* can be defined [12] as, *Ag = [Role, E, C, Q, PR, K, S, I]* where, each agent plays a distinct set of roles to provide the set of services *S;* the set of events which may occur from the user, from some other collaborative agent, or on achieving some condition or changes in states of *Res* is denoted by *E*; a set of constraints *C* to be satisfied in order to respond to some event; *Q* is set of resources those are available for agents to reach its objective; *PR* is the set of agent properties which holds the agent's state and also maintains the set of resources *Q* acting on the agent. *K* is the knowledge base; *I* denotes a set of interactions between candidate agents and their events and vice versa. Constraints of an agent is related to the agent's property, resource and knowledge. The availability of these needs to be checked and satisfied as per the requirements before an event occurs. The path <A1, C1, E1, S1, A2> in figure 1 is referred to as a constraint full interaction path where the constraint C1 will be checked before E1. Whereas the path <A1, E2, A2> in the same figure comprises an interaction path which is constraint less.

TABLE I. Role Collaboration Template

ROLES	EVENTS	SERVICES
Agent Name :- Student Agent (SA)		
R1: College seeker	**E1:** Higher education degree **E2:** Suitable college found	**S1:** Searching for notification **S2**: Search college **S3**: Get college info
R2: Admission seeker	**E3:** Registration	**S4:** Admission info **S5:** Application form fill up
Agent Name :- College Agent		
R3: Collegeinfo provider	**E4:** Application procedure	**S6:** Print out notification **S7:** form submission
R4: Service Provider	**E5:** Successful enrollment **E6**: Successful admission	**S8:** Taking test and interview **S9:** Publication of result **S10:** Update Student Data

The conceptual architecture of MAS can be illustrated using the Student Enrolment System (SES). This SES will help a student, who is searching for a suitable college after completion of his/her basic school education. The main objective of SES is to search a good college according to the requirements of the student, to provide hostel accommodation and scholarship and finally to get the student registered in the college.

This SES can be realized using MAS. In this scenario, the MAS comprises of two agents the *Student Agent(SA)* and the *College Agent(CA)*.The CA will provide the services like sending college as well as courses information, providing hostel accommodation, send scholarship information and finally creating student record. Whereas, the SA provide the services like sending application, initiating the search of college and courses. Acquiring information about hostel and scholarship. Hence the SA will play the role of *information seeker (R1)* and *service seeker (R2)* while the CA will play the role of *information provider (R3)* and *service provider (R4).*

B. *MASAG(Multi Agent System Architecture Graph)*

The dynamic facets of the MAS architecture was defined as a combination of collaborative agents and their corresponding interactions confined in an environment and which can be expressed as Multi-Agent System Architecture Graph (MASAG) [14]. The two graph semantic based components of MASAG are Agenut Interaction Graph (AIG) and Agent Collaboration Graph (ACG). Established on the roles played by the agents and collaboration between them ACG studied the cooperativeness of the agents in MAS. On the other hand, through the related events and interaction paths in MAS the agent level interactions was studied in AIG. The MASAG consisting of the AIG and ACG is shown in Figure 1 and 2.

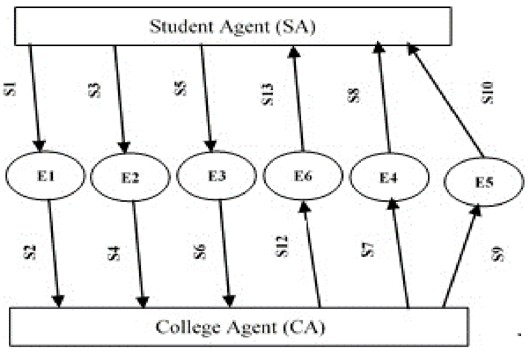


Fig. 2. AIG of Student Enrolment System

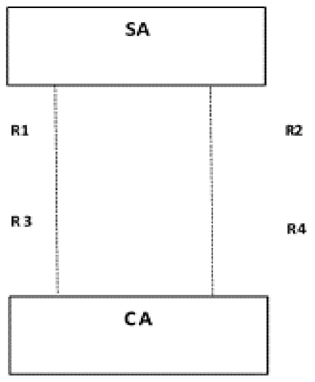


Fig. 1. ACG of Student Enrolment System

III. Proposed Extended Methodology

A. *Extended Agent Interaction Graph: Ex-AIG*

Precisely, AIG is a labeled bipartite directed graph and was expressed as G = (U, V, D, L) in [16]. But while defining Ex-AIG, an additional element is added to the above definition of AIG. Hence formally Ex-AIG can be expressed as $EX - AIG = \{(U, V, C), (D1, D2, D3), L1, L2\}$ where,

- *U* and *V* are disjoint set of vertices where, *U* is the set of all distinct Agents, *V* is the set of all distinct Events on which those agents will response.
- *C* is the set of constraint vertices of an agent that needs to be satisfied by an agent for an event to be performed.
- *D1* is the set of edges between an event and agent vertex and vice versa to exemplify the Interactions and expressed as $((U \times V) \cup (V \times U))$
- *D2* is the set of edges between agent and its constraints vertex depicting relationship between them. It can be expressed as *U X C*.
- *D3* shows the relationship of constraints with their event vertex and can be expressed as *C X V*.
- *L1* and *L2* are set of distinct labels expressed on the edges *D1*, *D2* and can be defined as $L1(D1) = s1$ and $L2(D2) = s2$ where *s1* and *s2* are the service.

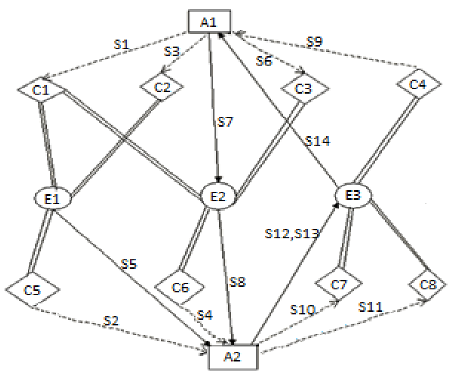


Fig. 3. Formal diagram of Ex-AIG

TABLE II. Description of Components used in Ex-AIG

Components	Description
▭	Agent Vertices (U)
○	Event Vertices (V)
◇	Constraints of an Agent (C)
→	Relation between an Agent & Event Vertices (D1)
⇢	Relationship of Constraints with their Agents Vertices (D2)
═	Relationship of Constraints with their Event Vertices (D3)
Labels	Denotes the Services (L1 + L2)

B. *Properties of Ex-AIG*

The extended AIG derived from MASAG defined above holds the following properties

- It is a labelled tripartite directed graph with three distinct sets of vertices comprising of agent, event and constraints.
- Edges in the graph depicts the various constraint full and constraint less distinct alternative interaction paths between two agent vertices. $\{a_1 s_1 c_1 e c_2 s_2 a_2\}$ is a constraint full interaction path and $\{a_1 s_1 e\, s_2 a_2\}$ is a constraint less interaction path. The edges have labels to represent the services.
- The interaction paths may have multiple constraints followed by multiple events. After event, they may have multiple constraints succeeded by multiple services. The possibilities can be formalised as:

  $\{a_1 s_1^+ c_1^* e^+ c_2^* s_2^+ a_2 \mid a_1 \in A,\ s_1 \in S,\ c_1 \in C,\ c_2 \in C - c_2,\ s_2 \in S - s_1,\ a_2 \in A - a_1\} \ldots\ldots\ldots\ldots\ldots\ldots..(1)$
- Multiple events may be invoked simultaneously on satisfaction of a single constraint.
- Depending on the direction of the interaction paths agent vertices can be *requester* and *responder*.
- The services may be a part of *pre-condition* required before the constraint is checked for an event to occur. This process $\{C(s, p, q, k) \to e\}$ is referred to as state change of an agent.
- Services may also be a part of *post-condition* where the service will be obtained after an event has occurred and constraint checked. This process $\{C'(p', q', k') \to s'\}$ is called as state check on service.

C. *Formal Analysis*

STEP 1: Identification of Resources

Resources denoted by Q $(Q \subset Res)$ is the set of resources in the environment those are accessible and essential for the achievement of the goals of an agent. Resources needs to be available to an agent for delivering the corresponding services after event is invoked. Hence identification of resource can be formalised as: $Q = \exists a \exists r \exists q \exists s \{(a \in A) \wedge (r \in R) \wedge ((q \in Q)) \wedge (s \in S) \wedge (a \to r) \wedge (a \to q)\} \to (List(s)) \ldots\ldots\ldots..(2)$

Where, Res is the environmental resources and *Q* is the set of resources available to an agent. The resource *q* is available to an agent with role *r* to give the list of services *List(s)*.

STEP 2: Generation of Resource Template

From the expression of resource given above a resource template which is allocative in nature is obtained as shown in Table III. Analysis of the resource template will help to get the information of the resources available to an agent to perform its events. As shown, agent *A1* while playing the roles *R1* and *R2* in the system can have more than one resource namely *Q1* and *Q2*. Using resource *Q1* agent will give service *S1*, *S2* and *Q2* will be required for providing the services *S3* and *S4*.

TABLE III. RESOURCE TEMPLATE FOR AN AGENT

Agent (A)	**Role (R)**	**Resource (Q)**	**Service (S)**
A1	R1	Q1	S1
			S2
	R2	Q2	S3
			S4

STEP 3: Identification of Properties

Properties, denoted by *PR,* is the set of agent properties which holds the agent's state to complete execution of the defined services. Formally, properties of an agent can thus be defined as: $PR = \exists a \exists r \exists p \exists s\{(a \in A) \wedge (r \in R) \wedge (p \in PR) \wedge (s \in S) \wedge (a \rightarrow r) \wedge (a \rightarrow p)\} \rightarrow \big(List(s)\big) \dots \dots \dots (3)$

Where, an agent playing the role *r* will provide the listed services *List(s)* and then update the property *p*. If any agent is having properties *p* then it implies that the state of the agent is maintained by *p* with service *s*.

STEP 4: Generation of Properties Template

From the formal definition a property template can be acquired as in Table IV. Analysis of the properties template will help in gaining the detailed knowledge of the properties required to maintain the state of an agent to get the services. As given, agent *A1* with role *R1* and *R2* will provide the services *S1, S2, S3* and *S4* and update the property *PR1* and *PR2*.

TABLE IV. PROPERTIES TEMPLATE FOR A SINGLE AGENT

Agent (A)	**Role (R)**	**Service (S)**	**Property (PR)**
A1	R1	S1	PR1
		S2	
	R2	S3	PR2
		S4	

STEP 5: Identification of Knowledge

Knowledge, symbolised by *K*, is a set of information that makes an agent capable of achieving its predefined goals. It is essential to an agent that invokes the events to attain the desired services. Formally, this set of information can be defined as: $K = \exists a \exists r \exists k \exists s\{(a \in A) \wedge (r \in R) \wedge (k \in K) \wedge (s \in S) \wedge (a \rightarrow r) \wedge (a \rightarrow k)\} \rightarrow \big(List(s)\big) \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots (4)$

Where, *K* is the knowledge that gets updated periodically by the environment and *k* is the knowledge possessed by an agent to provide service *s* after the event *e* is invoked. The knowledge *K* keeps getting updated with the help of *k*.

STEP 6: Generation of Knowledge Template

From the definition of knowledge the corresponding knowledge template is shown in the table V. Analysis of the knowledge template will help in gaining a detailed description of the information necessary for an agent to perform its events. An agent *A1* in the roles *R1* and *R2* requires a set of information namely *K1* and *K2* for services *S1, S2, S3* and *S4*.

TABLE V. KNOWLEDGE TEMPLATE FOR AN AGENT

Agent (A)	**Role (R)**	**Knowledge (K)**	**Service (S)**
A1	R1	K1	S1
			S2
	R2	K2	S3
			S4

STEP 7: Identification of Constraints

Constraints expressed by *C* is a set of all dynamic and static constraints which needs to be satisfied for an event to be invoked. If source of constraint is environment then constraint is static and if source is agent then constraint is dynamic. Constraint is dependent on properties, resources, and knowledge.

Hence constraint can be formalised as $C = \exists a \exists r \exists p \exists k \exists q \exists s \exists c \exists e \Big\{(a \in A) \wedge (r \in R) \wedge (p \in PR) \wedge (k \in K) \wedge (q \in Q) \wedge (s \in S) \wedge (c(p,k,q) \in C) \wedge (e \in E) \wedge (a \rightarrow r) \wedge (a \rightarrow p,k,q,s) \wedge \Big(\Big(\big(s \wedge c(p,k,q)\big) \vee (s) \rightarrow e\Big) \rightarrow \big((c(p,k,q) \rightarrow s) \vee s\big)\Big)\Big\} \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots .. (5)$

Where agent *a* with role *r* have a constraint *c* related to property *p*, resource *q* and knowledge *k*. In case of a constraint full interaction for an event *e* to be invoked there will be a service with a constraint. In case of a constraint less interaction a service *s* will invoke the event *e*.

STEP 8: Generation of Constraint Template

From the dynamic constraint defined above, a constraint template for every agent is displayed in Table VI. Analysis of the constraint template will help in gaining the comprehensive knowledge of the conditions a distinct agent must fulfil in order to invoke an event. This is referred to as *pre-condition* of an event. After the event a constraint may be required to be satisfied to get a distinct service from the collaborating agent. This is referred to as the *post-condition* of an event. As shown an agent *A1* with roles *R1* and *R2* can have more than one constraints namely *C1* and *C2*. In role *R1* if *C1* related to property *PR1*, knowledge *K1* and resource *Q1* is satisfied then event *E1* will be invoked.

TABLE VI. CONSTRAINT TEMPLATE FOR AN AGENT

Agent (A)	**Role (R)**	**Pre-Condition**	**Event (E)**	**Post-Condition**
A1	R1	<S1, C1 (PR1, K1, Q1)>	E1	<C3 (PR3, K3, Q3), S3>
		<S2, C2 (PR2, K2, Q2)>		<C4 (PR4, K4, Q4), S4>
		<S5, C5 (PR5, K5, Q5)>	E2	<C6 (PR6, K6, Q6), S6>
	R2	<S7>	E3	<C8 (PR8, K8, Q8), S8>
		<S9, C9 (PR9, K9, Q9)>	E4	<S10>

IV. ILLUSTRATION USING CASE STUDY

A case study of Student Enrolment System (SES) has been considered to illustrate the above proposal. This SES will help a student searching for a suitable college and get himself admitted after completion of his basic school education.

This SES can be realized using MAS. Here the MAS comprises of two agents the Student Agent (SA) and the College Agent (CA). The CA provides the services like *SendCollegeInfo*, *SendCourseInfo*, *SendScholarshipinfo*, etc. SA will provide services like *SendApplication, GetCollegeInfo, GetCoursesInfo* and so on. Hence the SA plays the role of *information seeker (R1)* and *service seeker (R2)* while the CA plays the role of *information provider (R3)* and *service provider (R4)*. The constraint templates corresponding to the SA and CA is shown respectively in Table VII and VIII.

TABLE VII. CONSTRAINT TEMPLATE FOR STUDENT AGENT

Agent (A)	Role (R)	Pre-Condition	Event (E)	Post-Condition
Student Agent (SA)	Information Seeker (R1)	<S1:GetCollegeInfo,C1:CollegeReputation(PR1:Decision Taking,K1:College Info,Q1:Qualification)>	E1:Suitable CollegeFound	<S2:SendCollegeInfo, C2:InfoAvailability(K2:CompleteInfo,Q2:Infrastructure)>
		<S3:GetCourseInfo>	E2:NeedCourseInfo	<S4:SendCourseInfo,C3:CurriculumWellDefined(PR3:Sender,K3:Curriculum,{Q3:Faculty, Q4:CoursePanel})>
	Service Seeker (R2)	<S5:SeekAdmission,C4:ReadyForAdmit(PR4:Eligible,K4:Marks,Q5:AdmissionForm)>	E3:ApplyForAdmission	<S6:ReceiveApplication>

TABLE VIII. CONSTRAINT TEMPLATE FOR COLLEGE AGENT

Agent (A)	Role (R)	Pre-Condition	Event (E)	Post-Condition
College Agent (SA)	Information Provider (R3)	<S7:GiveScholarshipInfo,C5:ScholarshipAvailability(PR5:ScholarshipProvider,K5:TypesOf Scholarships,Q6:ScholarshipSection)>,<S8:ScholarshipNotice,C5:NewScholarship(K5:ScholarshipSource)>	E4:Scholarship InfoAvailable	<S9:ApplyFor Scholarship>
		<S10:GiveHostelInfo>	E5:HostelInfoAvailable	<S11:ApplyForHostel>
	Service Provider (R4)	<S12:AdmissionStart, C6:LastDate(PR6:UniversityApproval,K6:ApplicantShortlist,)>	E6:ReadyToEnrol	<S13:TakeAdmission>

The graph in the Fig. 4 below has been designed using the details shown in tables VI and VII. The two agents in the graph are the SA and the CA. The SA has nine constraints as stated in table VI whereas the CA has five constraints as stated in table VII. Events are invoked after all the constraints for each event is satisfied. The event *E1* occurs only after the constraint *C1* is satisfied by *SA* to get the service *S1*. The event *E3* needs *C10* and *C11* to be satisfied to give the services *S8* and *S9*. While the event *E4* requires *C12* and *C13* to be fulfilled to get the service *S10* and *S11*. Similarly the other events are performed after fulfilling its respective constraints.

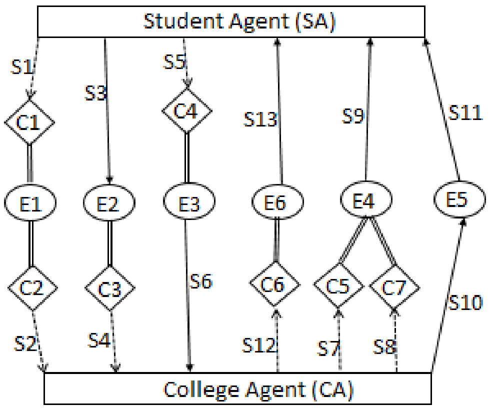
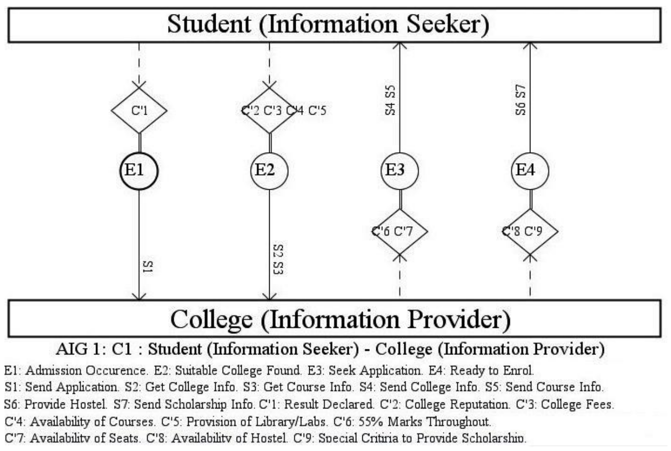


Fig. 4 The Ex-MASAG of the Student Enrolment

V. TOOL TO DESIGN EX-MASAG

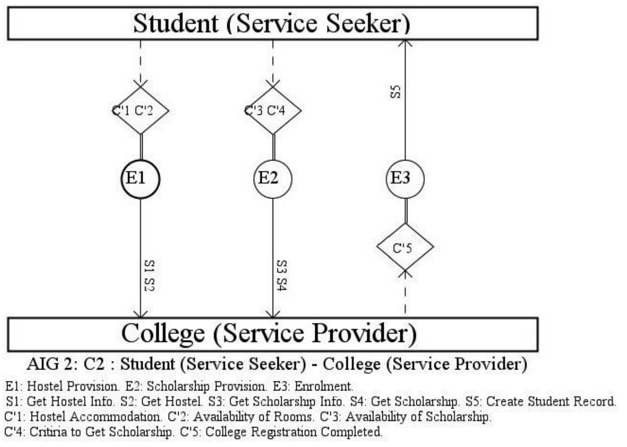
Ex-MASAG is a graphical representation to effectively demonstrate the collaborations and interactions between agents in a MAS based on some constraints to invoke some events for getting some services. It has extended the MASAG's AIG diagram [14] by taking care of few more important characteristics Constraints, Properties, Resources and Knowledge.

To get the corresponding Ex-AIG an efficient tool is required to automate the process of drawing. Many existing standard tool like UML can be used for drawing ACG & Ex-AIG which is very complex in large MAS scenario. This is because each element needs to be drawn manually. To solve this problem an efficient and simple tool called *drawFAM* has been developed using Java, to map the concepts of Ex-MASAG. The end



Student (Information Seeker)
C'1
C'2 C'3 C'4 C'5
S4 S5
S6 S7
E1
E2
E3
E4
C'6 C'7
C'8 C'9
S1
S2 S3
College (Information Provider)
AIG 1: C1 : Student (Information Seeker) - College (Information Provider)
E1: Admission Occurence. E2: Suitable College Found. E3: Seek Application. E4: Ready to Enrol.
S1: Send Application. S2: Get College Info. S3: Get Course Info. S4: Send College Info. S5: Send Course Info.
S6: Provide Hostel. S7: Send Scholarship Info. C'1: Result Declared. C'2: College Reputation. C'3: College Fees.
C'4: Availability of Courses. C'5: Provision of Library/Labs. C'6: 55% Marks Throughout.
C'7: Availability of Seats. C'8: Availability of Hostel. C'9: Special Critiria to Provide Scholarship.

Fig. 5 Snapshot of first Ex-AIG generated by tool



Student (Service Seeker)
C'1 C'2
C'3 C'4
S5
E1
E2
E3
C'5
S1 S2
S3 S4
College (Service Provider)
AIG 2: C2 : Student (Service Seeker) - College (Service Provider)
E1: Hostel Provision. E2: Scholarship Provision. E3: Enrolment.
S1: Get Hostel Info. S2: Get Hostel. S3: Get Scholarship Info. S4: Get Scholarship. S5: Create Student Record.
C'1: Hostel Accommodation. C'2: Availability of Rooms. C'3: Availability of Scholarship.
C'4: Critiria to Get Scholarship. C'5: College Registration Completed.

Fig. 6 Snapshot of second Ex-AIG generated by tool

user/MAS architect/MAS designer needs to complete the agent profiling through a user friendly wizard to create/draw the ex-AIG. This will be possible after an extended analysis of the MAS.

The case study of Student Enrolment System (SES) used in section IV has been used to demonstrate *drawFAM* tool. Figure 5 and 6 shows the Ex-AIGs of the aforesaid case study obtained with the help of *drawFAM* tool. The agent profiles acquired from the analysis is used to get the concerned EX-AIG.

VI. EVALUATION OF Ex-AIG

In the context of MAS, the dynamic facets of MAS architecture is a combination of collaborative agents and their interactions present in an environment and expressed as MASAG. AIG is a component of MASAG that gives the agent level interaction through event and interaction paths. Ex-AIG is a development from the AIG as it provides the following benefits

- Agents need to reason about the interactions through constraints. These constraints needs to be satisfied for invoking an event. Constraints specification and satisfaction has been expressed using Ex-AIG.
- Multiple event invocation is handled due to fulfilment of some constraints.
- Ex-AIG also expresses the pre and post condition of service invocation from any agent.

The comparison among AIG and Ex-AIG can be summarized in Table IX

TABLE IX. COMPARISON BETWEEN AIG AND EX-AIG (PROPOSED)

Graph Type / **Properties**	**AIG**	**EX-AIG**
Partite	Bipartite	Tri partite
Vertices	Two	Three
Constraint Specification	No	Yes
Invocation of Agents	Single Agent at a time	Multiple Agent at a time
Semantic of Edges	Represent Interactions	Represent 1) Interaction between agent and event vertices 2) Relationship between agent and constraints and vice versa
Resource, knowledge Specification	No	Yes
Tool Support	No	Yes

VII. CONCLUSION AND FUTURE WORK

A formal analysis methodology has been proposed for a MAS. This methodology formally helps in identifying and representing the different aspects of an Agent, namely the Constraints, Properties, Resources and the Knowledge. The proposed methodology follows an ordered approach towards logic based formal analysis of the agents in MAS. The different aspects namely the structural and dynamic aspects has been well represented by the help of several templates. A case study has been taken for the better understanding of the proposed methodology.

Finally, the Ex-MASAG has been expressed and designed for mapping the proposed approach into architectural design of MAS. The Ex-AIG is capable of representing the constraints specified that needs to be satisfied to invoke an event. Also the pre and the post condition for supplication of service from event is expressed through Ex-AIG.

In this work, the Ex-MASAG has been designed for mapping the proposed approach in a system consisting of two agents. The future objective would be to design the Ex-MASAG for a much larger system consisting of several agents. Also, to further incorporate the properties, resources and knowledge of an agent into the MASAG.

REFERENCES

[1] Luck, M., McBurney, P. & Preist, C. 2003. Agent Technology: Enabling Next Generation Computing — A Roadmap for Agent Based Computing, AgentLink.http://www.agentlink.org/roadmap

[2] P. K. Biswas, “Towards an agent-oriented approach to conceptualization”, Journal of Applied Soft Computing, Vol. 8(1), PP 127-139, 2008.

[3] G. Cabri, L. Leonardi, L. Ferrari, F. Zambonelli, “Role-based software agent interaction models: a survey”, Knowledge Eng. Review, Vol. 25(4), pp. 397 – 419, 2010.

[4] Norfaradilla Wahid, Eric Pardede, "Single Transition Constraint for XML Update Validation", NBIS, 2012, 2013 16th International Conference on Network-Based Information Systems, 2013 16th International Conference on Network-Based Information Systems 2012, pp. 24-31.

[5] Wahid, N. & Pardede, E. (2015), 'XSDyM: An XML graphical conceptual model for static and dynamic constraints.', Computer Standards & Interfaces 37 , 60-72 , June 2014.

[6] M. E. Nissen, M. N. Kamel and K. C. Sengupta, “Toward Integrating Knowledge Management, Processes and Sys-tems: A Position Paper,” Proceedings of the AAAI Sym-posium on Bringing Knowledge to Business Processes, Stanford, 2008.

[7] Marc J. Raphael and Scott A. Deloach," A Knowledge Base For Knowledge-Based Multi-Agent System Construction", Proceedings of the National Aerospace and Electronics Conference (NAECON),(2000).

[8] Gifty Edwin and Michael T. Cox, "Resource Coordination in a Single Agent and Multi-Agent Systems", in ICTAI, PP 18-24, 2001.

[9] Hao Xiaoqing and Chen Jia, " Resource-cost-based Multi-agent Systems Scheduling for Grid Resource Management", In: Computer Application and System Modeling (ICCASM), 2010 International Conference on (Volume:12).

[10] Kao-Shing Hwang, Senior Member, IEEE, Wei-Cheng Jiang, and Yu-Jen Chen," Model Learning and Knowledge Sharing for a Multi-agent System With Dyna-Q Learning", In: IEEE Transactions on Cybernetics (Volume:45 , Issue: 5), 2014.

[11] Rajib Kumar Chatterjee, Anirban Sarkar, " Analysis of Multi Agents Systems: A Formal Approach", 10th International Conference(IEEE), pp 1-6, TENCON 2014.

[12] Neha, Rajib kumar Chatterjee, Anirban Sarkar, “High Level Petri Net Based Behavioral Model For Multi Agent System”, 3rd International Conference on Advanced Computing and Communications (ACC 2013), pp 25 – 30, September 2013.

[13] Rajib kumar Chatterjee, Neha, Anirban Sarkar. (2015). Behavioral Modeling of Multi Agent System: High Level Petri Net Based Approach. International Journal of Agent Technologies and Systems (IJATS), 7(1), 55-78. doi:10.4018/ijats.2015010104

[14] Anirban Sarkar, “Modeling Multi Agent System Dynamics: Graph Semantic Based Approach”, 10th International Conference on Service Systems and Service Management, pp. 664-669, 2013.