Vol 8. No. 2 June, 2015

African Journal of Computing & ICT

© 2015 Afr J Comp & ICT – All Rights Reserved - ISSN 2006-1781
www.ajocict.net

Multi-agent Based Course Allocator Using GAIA Methodology and JADE Framework

K.T. Igulu
Department of Computer Science
Rivers State Polytechnic,Bori
Rivers State, Nigeria.
E-mail: igulukt@gmail.com,
Phone: +2347018827522

Z.P. Piah
Department of Computer Science
Rivers State Polytechnic, Bori
Rivers State, Nigeria.

P.O. Asagba PhD
Department of Computer Science
University of Port Harcourt
Rivers State, Nigeria.

ABSTRACT

This paper discusses our work on the development of an interactive multi-agent system that automates university course allocation. The system receives as input the courses to be offered in a particular semester, their respective credit units and appropriately allocate it to competent lecturers based on their teaching profile. The system is composed of two agent types- The administrator agent and the lecturer agents which were the final concrete artifacts of the system. Four roles (allocation handler, environment monitor, administrator assistant and lecturer assistant) were discovered in the analysis stage. However, the allocation handler and environment monitor were subsumed by the administrator and lecturer agents at the design stage. The elicitation, analysis, design to implementation were quite natural. This proves Agent-Based Software Engineering (ABSE) as a viable paradigm. We used GAIA methodology for analysis and design and Java Agent DEvelopment (JADE) framework for implementation.

Keywords- Mmulti-agent; course allocation; Gaia, JADE; software engineering; intelligent, agent.

African Journal of Computing & ICT Reference Format:
K.T. Igulu, Z.P. Piah & P.O. Asagba (2015). Multi-agent Based Course Allocator Using GAIA Methodology and JADE Framework. Afr J. of Comp & ICTs. Vol 8, No. 2. Pp 41-52.

I. INTRODUCTION

It is a known fact that scheduling is an NP problem. A problem where there is no universally accepted optimal solution. Most Universities do this manually and of course becomes so tedious with the growing number of students and courses offered in a University. Along with the innate limitations or problems of traditional manual systems, manual course allocation and scheduling has the following key problems: 1) Keep and manage record of the previous data. 2) Meeting person. 3) Manage multiple queries for the same subject. 4) Make availability of the interested subjects for a faculty member. 5) Manage rooms for delivering lectures. Every university has a number of schools with respective heads and each school has a number of departments with respective heads. A department is composed of lecturers, students and other teaching enhancement facilities.

In a well-established university, a department should offer degrees ranging from undergraduate to post graduate. Each degree depending on the duration, has several levels (e.g., Bachelor of Science in Computer Science should have first, second, third and fourth year level students) administered concurrently in every academic session. The task of allocation is usually the job of the head of the department or can also be delegated to another fellow who proves competent. Allocation of this kind is never void of irregularities and anomalies. The problem faced by academic departments is the inefficient/ inappropriate allocation of courses to competent lecturers. In our interview with the faculty in charge of course allocation in the department of Computer Science, University of PortHarcourt, the following were discussed; 1.) Allocation is dedicated to a particular lecturer 2.) Number of courses to be offered is known a priori, 3.) Number of classes is also known a priori, 4.) The faculty needs information regarding the areas of expertise of the lecturers. 5.)

The subjects are communicated to lecturers and they are required to give their choices according to their levels and a minimum number of credit units to offer. 6.) When reply is received, courses-lecturers table is made. 7.) The course allocation administrator willfully allocates the courses. 8.) The number of courses to be allocated is determined by the designation of the lecturer. Professors are given less but highly technical load. 9.) Designation also determines the level course that can be assigned to a lecturer. E.g: an MSc holder can't be allowed to teach doctorate students. 10.) Conflict is bound to occur. 11.) These conflicts are usually resolved on a one to one basis. 12.) The administrator lecturer prepares an allocation chart. 13.) Conflict is bound to occur.

This work discusses a typical university course allocation problem using ABSE paradigm. The system receives as input the courses to be offered in a particular semester, their respective credit units and appropriately allocate it to competent lecturers based on their teaching profile. The system is composed of two agent types- The administrator agent and the lecturer agents which were the final concrete artifacts of the system. Four roles (allocation handler, environment monitor, administrator assistant and lecturer assistant) were discovered in the analysis stage. However, the allocation handler and environment monitor were subsumed by the administrator and lecturer agents at the design stage. Figure 1 shows the interactions among the agents in the system. There are m number of Lecturer agent instances and one instance of Admin agent. The Admin Agent performs the allocation, monitors the environment for any request and does resolve conflicts among lecturer agents

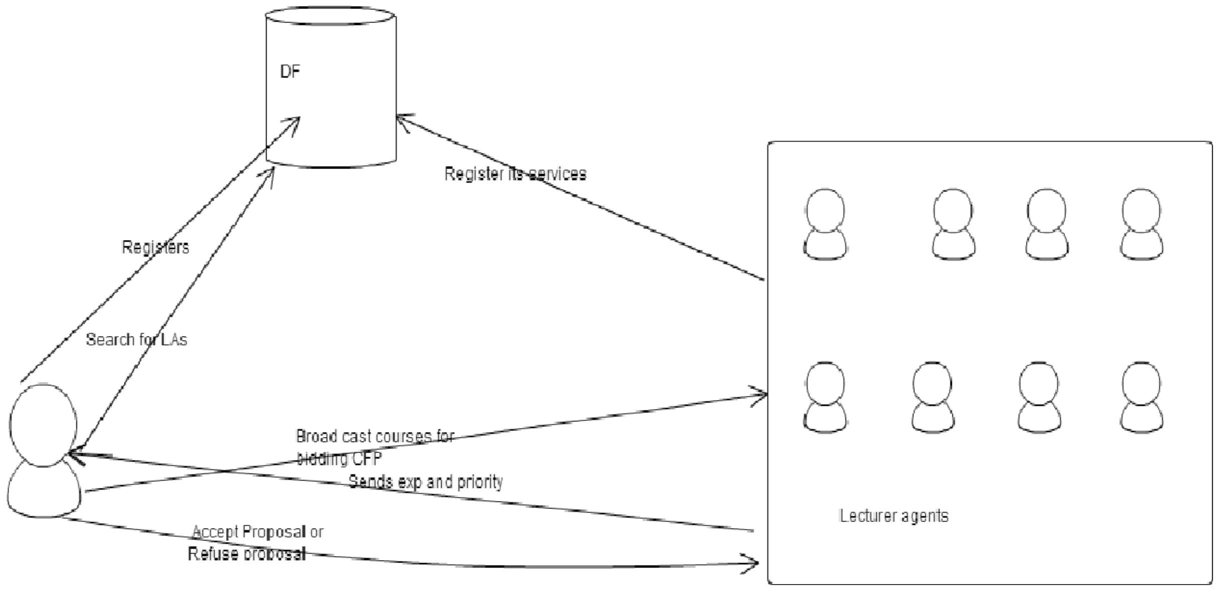


Figure 1. Interactions in the system

In this work, we assume that 1.)Each level(e.g., first year) has a single sections. 2.) No external lecturer is needed 3.) All courses are offered by the assumed department.

The paper is organized as follows: In section 2, we discuss works that are related to our work. In section 3, we discuss our motivations for agency and the general benefits of using ABSE paradigm. Section 4 introduces the Gaia methodology. Section 5 discusses the analysis phase of the system with respect to Gaia standards. Section 6 provides information about the design phase of the system. Section 7 gives the implementation details of the system. Finally, section 8 wraps up our work.

2. RELATED WORKS

In [17], the authors argued that for certain classes of problem, adopting a multi-agent approach to system development affords software engineers a number of significant advantages over contemporary methods. If a problem domain is particularly complex, large or unpredictable, it might be only way it can be reasonably addressed is to develop a set of modular components that are specialized at solving a particular aspect of it.

In [22] the system is composed of intelligent agents but does not follow any Agent-Based software Engineering (ABSE) conventions. Other notable works can be found in [5] and [6] that uses stimulated annealing and genetic algorithm respectively. [1] uses Heuristic approach in generating the schedule. Literatures on time table scheduling can also be found in [14,17,31,40,41,43] whereas works on Gaia and Jade can be found in [8,9,16,18,20,22,34]. It is worthwhile to note that most of the works focused on time tabling not course allocation.

3. MOTIVATION/BENEFITS OF AGENCY

The human-like characteristics of agents provide a high abstraction level which may simplify the modeling and implementation of systems for complex domains. Agents can be trusted to pursue their goals and take initiative to interact only when needed; this independence reduces the need for external communications. Their autonomy leads to encapsulation of functionality, and coupling is reduced because agents do not provide any control point to external entities. The following list highlights some of the main dimensions along which agent systems are believed to enhance performance, these aspects are further elaborated on in [19]:

Computational efficiency because concurrency of computation is exploited. This requires that the communication is kept minimal, e.g. by transmitting high-level information and results rather than low-level data.

Reliability Components that fail can be gracefully recovered. Agents with redundant capabilities or appropriate inter-agent coordination are found dynamically and can take up responsibilities of agents that fail.

Maintainability A system composed of multiple components is easier to maintain because of its modularity.
Responsiveness The modularity of a multi-agent system leads to the possibility of handling anomalies locally without propagating them to the whole system.

Flexibility Agents with different abilities can adaptively organize to solve a given problem. An agent can also have a number of plans for reaching its goal and adapt its strategy to changes in the environment.

In order to deal with the complexities, the timeliness response and avoiding subjective impositions of course allocation and yet maintaining robustness and flexibility, great level of autonomy must be maintained. We identified some conflicts that may arise in course of allocation. These conflicts require negotiation without much influence on the negotiating parties.

In [24,27] intelligent software agents(or intelligent agents or simply agents) are characterized as being autonomous, proactive, reactive, social, flexible and robust, as well as situated in an environment which they can sense and act upon. We found AOSE ideal to deal with the problems identified above. In fact, Multi agent systems provide the modularity that we want and the agents' social ability makes them capable of meeting different restrictions and goals through negotiation and collaboration.

4. GAIA METHODOLOGY

The Gaia methodology deals with both the macro-level (societal) and the micro-level (agent) aspects of systems [26]. The Gaia methodology is based on organizational metaphor. It is assumed that the software entity is a collection of various self-functioning roles of the organization with their dedicated responsibilities towards meeting the global organizational objective. In fact in most complex systems, the system could be modeled as composing of sets of organizations with their respective organizational responsibilities (functionalities).

A software system is conceived as the computational instantiation of a (possibly open) group of interacting and autonomous individuals (agents). Each agent can be seen as playing one or more specific roles: it has a well-defined set of responsibilities or sub goals in the context of the overall system and is responsible for pursuing these autonomously. Such sub goals may be both altruistic (to contribute to a global application goal) or opportunistic (for an agent to pursue its own interests). Interactions are no longer merely an expression of interdependencies, and are rather seen as a means for an agent to accomplish its role in the system. Therefore, interactions are clearly identified and localized in the definition of the role itself, and they help characterize the overall structure of the organization and the position of the agent in it [26].

The evolution of the activities in the organization, deriving from the autonomous execution of agents and from their interactions, determines the achievement of the application goal, whether an a priori identified global goal (as, e.g., in a workflow management systems where altruistic agents contribute to the achievement of a specific cooperative project), or a goal related to the satisfaction of individual goals (as, for example, in agent-mediated auctions, whose purpose is to satisfy the needs of buyer and seller agents), or both (as, for example, in network enterprises exploiting market mechanisms to improve efficiency)[26].

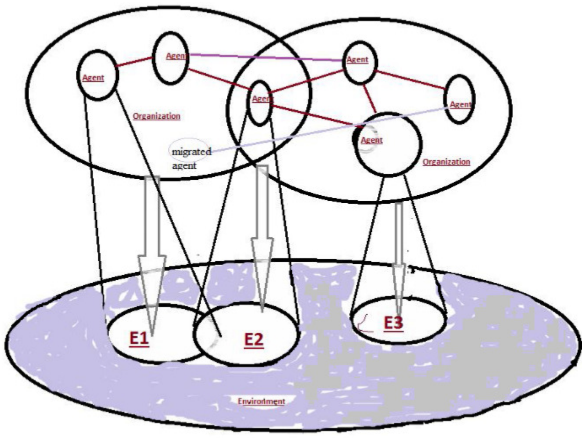


Figure 2. Canonical view of a multi-agent System [9]

The organizational perspective leads to a general architectural characterization of Multi-agent System (MAS) as depicted in Figure 2. A simpler system will eventually be modeled as a single organization. However, as complexity increases, programming practices like modularity and encapsulation suggest decomposing the system into different sub organizations consisting of a considerable number of agents to pursue its (sub)-organizational goal(s). In an organization, agents may interact to share computational tasks or for exchange of knowledge. In such system as depicted in Figure 2, there could be intra-organization (within) or/and inter-organizational (outside) interactions. In some other complex computation, agents may need to migrate from one organization to another. An agent may perform one or more roles in an organization [9].

Moreover, the MAS is completely immersed in an environment which is basically an ensemble of resources that the agents may need to interact with to accomplish their role. Of course, interaction with the environment occurs via some sort of sensors and actuators-mechanisms enabling agents to perceive and act upon some part of the environment. Such portion of visibility is determined by agent’s specific role, as well as by its current status.

The first proposed Gaia methodology consists of two iterative phases, analysis and design. Gaia does not address requirement elicitation but does not necessarily ignore it. Gaia is usually open to any platform of implementation but experience shows that it is better implemented with a FIPA compliant platform like JADE [16,34,35]. Thus our choice of implementation platform.

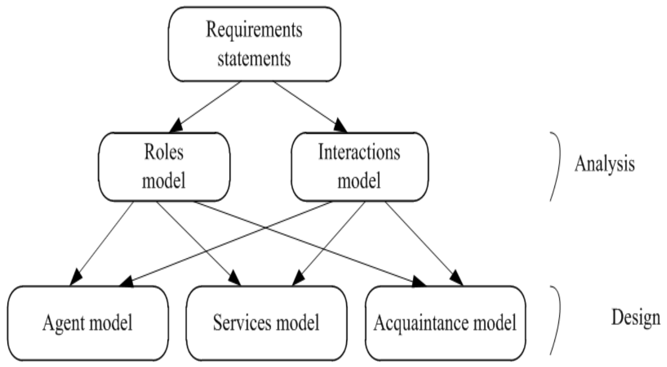


Figure 3. Gaia methodology Phases and Relationships[26]

5. ANALYSIS PHASE

The analysis phase involves building conceptual or abstract models that may not directly impact the system. It assumes that the analyst has conceptualized the problem and is clear of what the system should and should not do. Here, *roles* are identified and their interactions are modeled. The two artifacts produced at the end of the analysis phase of Gaia are *role* model and *interaction* models. The roles may not be detailed at the analysis stage [9].

5.1 Role Models

Roles consist of four attributes- *responsibilities, permissions, activities and protocols.*

Responsibilities (Rs) are said to be a key attribute of a role since they determine the functionality. Responsibilities are of two kinds- **liveness propertie**s-the role has to add something good to the system and **safety properties**- the role must prevent that something bad happens to the system.

Permissions(Ps) are the rights associated with a role. They identify the resources that are available to that role in order to realize its responsibilities.

Activities of a role are computations associated with the role that may be carried out by the agent without interacting with other agents. Activities are thus “private” actions.

Protocols are computations that require interaction with other agents. (P,A) will be used for Protocols and Activities and (D) for description in the role model because of space.

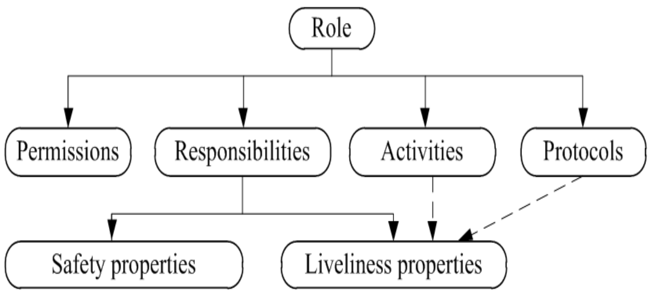


Figure 4. **Role and its attributes [26]**

With thorough analysis, we identified four roles in the system. Environment monitor, Lecturer Assistant Administrator Assistant and Allocation Handler.

Having identified the roles, we now move on to documentation of the roles. We will only show documentation of three roles-Administrator Assistant, Lecturer assistant and allocation Handler roles.

Role Schema: AdminAssistant[ADA]	
D	*It publishes requests for bidding of courses; it subscribes to the yellow pages knowledgebase and receives notification if a LA role registers; It receives commitment messages from LAs and relays same to AH; it publicizes the result of allocation to respective LA.*
P, A	*RegisterDF, QueryDF, SubscribeToDF, SaveLAs, RegisterCourse UpdateCourse, BroadCastMessageForbidding, ReceiveCommittmentToOffer, SendAllocationInstruction, AllocationChartPublishing*
Ps	*reads, updates LADataS, AllocationChart, CourseDataS Creates LADataS, CourseDataS*
Rs	**Liveness:** *RegisterDF . InitializeLADataS|| InitializeCourseDataS .SubscribeToDF. ((ReceiveMessageFromDF.SaveLAServices) +||RegisterCourses. (SendRequestForAllocation .ReceiveCommitmentToOffer)+. PublishAllocationResult+)||[UpdateCourses||UpdateLA]*.[AlterAllocation]**
	Safety: *CourseList not nil, LAlist not Nil, No Course is Allocated to more than one LA,*

Figure 5. Admin Assistant Role model

Role Schema: LecturerAssistant[LA]	
Ds	*It acts on behalf of a lecturer; it does most intelligent decisions a lecturer is supposed to take; it registers its service(lecturer)to DF; it replies the ADA when sent message to decide course to offer by choosing from experience profile; the lecturer can also set the priority of course to choose.*
P,A	*RegisterDF DecideCoursePriority, ChooseCourse , QueryAvailableCourses, SaveAllocation , SendsSatisfactoryMessage*
Ps	*Creates, reads, and updates own experience profile.*
Rs	**Liveness:** *RegisterDF . InitializeCourseTaughtDataS. [QueryAvailableCourses]* ReceiveMessageToOfferCourse +.SendCommitmentToOffer+.ReceiveCourseAllocated+.SendsSatisfactoryMessage.SaveAllocation* **Safety**: *true*

Figure 6. Lecturer Assistant Role model

Role Schema: AllocationHandler[AH]	
D	*Does the allocation computation; it resolve allocation conflicts when they arise, it forwards allocation chart to ADA*
P, A	*AllocationOfCoursesToLecturers, CheckForConflict, SendAllocationStatusMessage*
Ps	*Creates, reads and updates AllocatonChart*
Rs	**Liveness:** *InitializeAllocationChart .ReceiveAllocationMessage + .Allocate +.[ConflictResolutionModule]*. SendAllocationStatus* **Safety:** *one course to one lecturer*

Figure 7. Allocation Handler Role model

Vol 8. No. 2 June, 2015

African Journal of Computing & ICT

© 2015 Afr J Comp & ICT – All Rights Reserved - ISSN 2006-1781
www.ajocict.net

7.2 Interaction Model

The model consists of set of protocol definitions, one for each type of inter-role interaction. Here, a protocol can be viewed as an institutionalized pattern of interaction [9,26].

A protocol definition consists of the following:
Purpose: Brief textual description of the nature of the interaction (e.g., "AllocateCourses");
Initiator: the role(s) responsible for starting the interaction (e.g. admin);
Responder: the role (s) with which the initiator interacts;
Inputs: Information used by the role initiator while enacting the protocol (list of courses to be allocated);
Output: Information supplied by/to the protocol responder during the course of the interaction (e.g., allocation chart);
Processing: brief textual description of any processing the protocol initiator performs during the course of computation.

We present some of the protocols of our system.

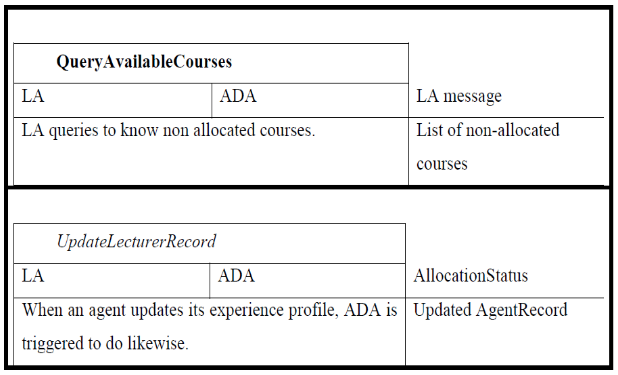
PublishRequestForBidding		CourseList, LecturerList
ADA	LA	
Packages courses and sends request to all discovered LAs for bidding.		CommitmentToOfferMessage

UpdateAgentRecord		Yellow page notification
DFA	ADA	
When an agent registers in DF, DF notifies ADA of its presence and ADA updates its AgentRecord		Updated AgentRecord

SendAllocationMessage		CommitmentToOfferMessage
ADA	AH	
After Receipt of commitment to offer message from LA, ADA relays same to AH for allocation		Partial Allocation Chart

AllocationStatusMessage		PartialAllocationChart
ADA	LA	
After Receipt of allocation status from AH, ADA sends the status to various LAs.		Allocated Courses

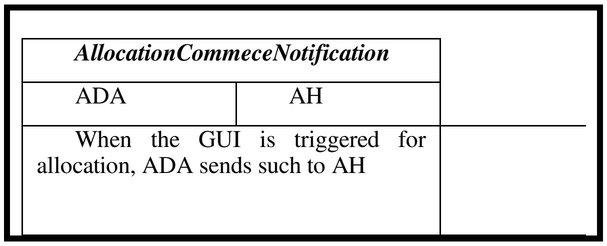
Figure 8. Protocols related to Admin Assistant Role



QueryAvailableCourses		
LA	ADA	LA message
LA queries to know non allocated courses.		List of non-allocated courses

UpdateLecturerRecord		
LA	ADA	AllocationStatus
When an agent updates its experience profile, ADA is triggered to do likewise.		Updated AgentRecord

Figure 9. .Protocols related to Lecturer Assistant Role



AllocationCommeceNotification		
ADA	AH	
When the GUI is triggered for allocation, ADA sends such to AH		

Figure 10. Protocol related to Allocation Handler Role

6. SYSTEM DESIGN PHASE

The analysis phase is basically the conceptualization of the system. It produces the input for the design phase. The activities of design phase involve the transformation of the abstract entities (represented in role and interaction models) of the analysis phase to concrete entities that may have direct impact on the realization of the system[9,26].
Agent Type, Services and Acquaintances models are identified and documented in this phase. The succeeding sections present the artifacts of Gaia design phase with respect to our case study. We exclude discussion of the Service model for privacy reasons.

6.1 *Agent Type Model*

Agent types are the counterparts of objects in object-oriented approaches. They are basic design units of an agent-based system and their realization at runtime is agent instances. Agent types in the system under development are defined on the basis of the roles that they play. In most cases, there is one-one mapping between roles to agent types. Gaia represents Agent Type with a rectangle and a role with an oval [8]. It uses annotation to represent the number instances of such agent mapped and a directed edge from role to agent.

In our system, the Lecturer Assistant Role is mapped to Lecturer Agent whereas the Environment monitor fades out during analysis and allocation handler is subsumed by the Admin Agent. The agent type model is depicted in Figure 11 below.

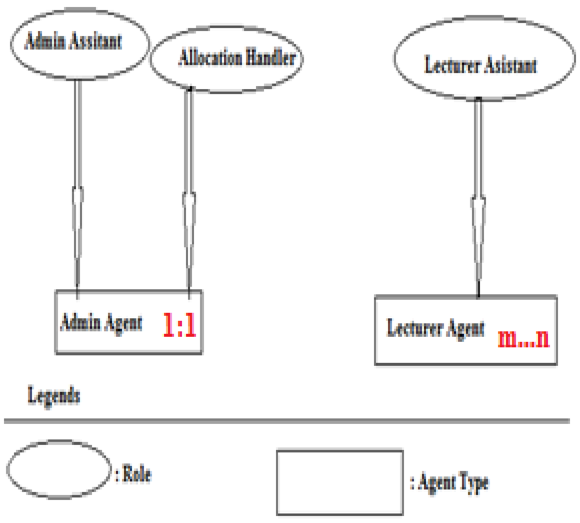


Figure 11. Agent Type Model

6.2 Acquaintance Model

The Acquaintance model depicts the communication links existing between agent types. It is in fact a directed graph in which nodes represent agent types and arcs show communication pathways.

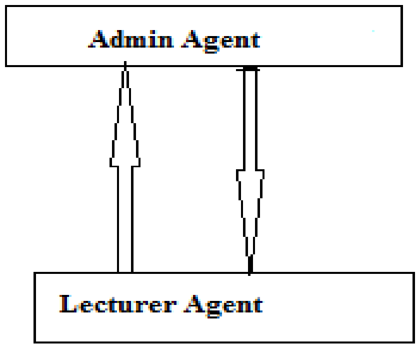


Figure 12. Acquaintances model

7. IMPLEMENTATION

As we have stated, we use JADE for implementation. We now discuss the implementation of the system based on JADE. JADE is FIPA compliant [16, 34, 35].

7.1 Admin Agent Class

The adminAgent class is the core class of this MAS. It embeds the core functionalities such as

- searching and discovery of lecturer agents,
- initiating conversation by sending appropriate messages to Lecturer Agents,

- Checking the appropriating of accepting a proposal from a lecturer agent
- Appropriately allocating courses to lecturer and
- Resolution of conflicts.

In order to implement Jade agents in GUI, Jade distribution comes with middleware support based on java Window tool Kit and Swing APIs. Our AdminAgent class extends the AgentGui instead of jade.core.Agent. In Jade, the tasks an agent performs are embedded in its behaviours. Usually, an agent has one or more behaviours depending. Jade comes with a number of behavioral classes which can be used to achieve diverse tasks of an agent[8,16, 44].

The following are the behaviours of Admin Agent;

- **OneShotBehaviour** is extended to achieve DF() registration.
- **TickerBehaviour** is used when the start allocation is triggered. This behavior searches the DF (Directory Facilitator[16]) every 60 seconds for 5 minutes after it has received the trigger. A TickerBehaviour repeats periodically using the millisecond parameter passed to its constructor. It does not stop except stopped explicitly. This behavior embeds the task of searching the DF for LECTURER_TYPE service of lecturer agents. It then adds the sendCallForProposalToAllLecturerAgent behavior that sends them CALL FOR PROPOSAL (CFP) (FIPA performative [10,12,13]) message with appropriate content. Here the content is a list of AgentAction objects called Offer (a class with Course attribute), a part of CourseAllocationOntology. The behavior also handles the incoming proposal and the rest of the conversation between lecturer agents and carries the allocation process.
- **WakerBehaviour** is used to terminate the Ticker behaviour by calling the stop() method of TickerBehaviour. This behavior waits for 6 minutes and then triggers the transfer of the content of partialAllocationTable to mainAllocationTable and also alerts the AgentGui.
- **SequentialBehaviour** embeds the behaviours described above.

7.2 Lecturer Agent Class

The Lecturer Agent (LA hereafter) handles all communication with Admin Agent using appropriate message attributes. It has two behaviours-OneShot and cyclic behviours to register and communicate with Admin respectively.

7.3 Ontology

As human, we communicate with a using the symbol, syntaxes and semantic of a particular language. The language's syntax and semantic must be known and understood by the communicating parties[7,32]. One problem that may occur in agent approach is the choice of format of encoding the content of a message (here, ACLmessages).

Sending information as primitives (Strings, numbers or characters) will only turn to be efficient if we are interested in primitive values. Serializing objects would have been the next option but serialized object can't be decoded on transit. Conveniently, Jade provides us with content language and ontology for converting and checking the semantics information encoded by another agent. One of the importance of ontology is that agent architecture may differ in the sense language and platform[7, 10,11,12,13,16].

It is clear however that, if on the one hand information handling inside an agent is eased, on the other hand each time agent A sends a piece of information *I* to agent B

- "A" needs to convert its internal representation of *I* into the corresponding ACL content expression representation and B needs to perform the opposite conversion.
- Moreover B should also perform a number of semantic checks to verify that *I* is a meaningful piece of information, i.e. that it complies with the rules of the ontology by means of which both A and B ascribe a proper meaning to *I*.

The support for content languages and ontologies provided by JADE is designed to automatically perform all the above conversion and check operations.

By Jade design recommendations, we found the structure suitable;

- CourseAllocationOntology
- CourseAllocationVocabulary
- Course as Concept
- Lecturer as Concept
- Problem as Concept
- Conflict as Concept
- Offer as Action
- Allocated as Predicate
- Taught as predicate

We adopted the SL0 as our content language.

7.4 Allocation Core and Conflict Resolution

Here we present how the allocation core is done and how the agent resolves several conflicts that could arise. For simplicity, we will depict the conflict resolution with a flowchart.

African Journal of Computing & ICT

© 2015 Afr J Comp & ICT – All Rights Reserved - ISSN 2006-1781
www.ajocict.net

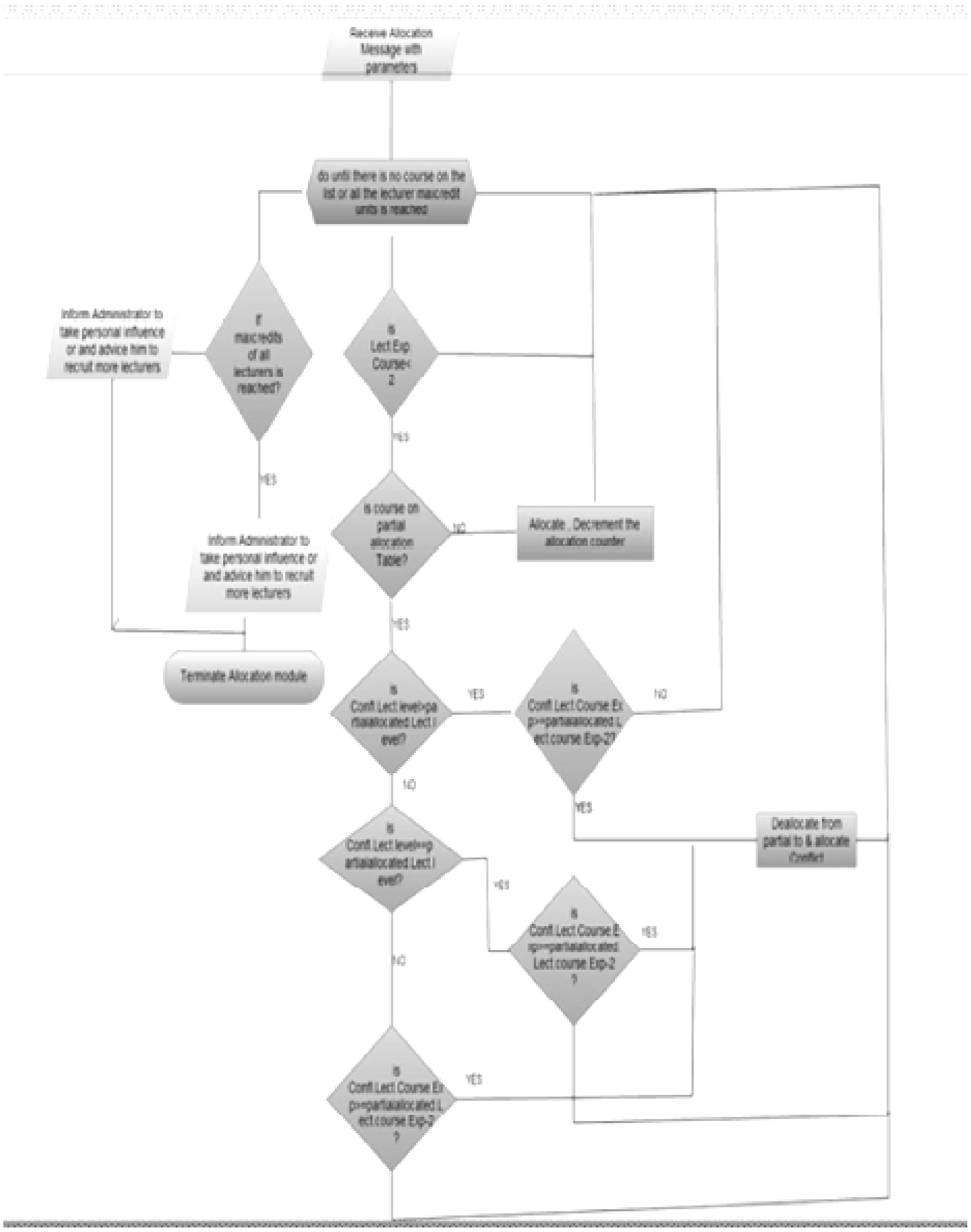

Receive Allocation Message with parameters
do until there is no course on the list or all the lecturer max credit units is reached
If max credits of all lecturers is reached?
Inform Administrator to take personal influence or and advice him to recruit more lecturers
YES
Inform Administrator to take personal influence or and advice him to recruit more lecturers
Terminate Allocation module
is Lect Exp Course< 2
YES
is course on partial allocation Table?
NO
Allocate , Decrement the allocation counter
YES
is Confl Lect level>partialallocated Lect level?
YES
is Confl Lect Course Exp>=partialallocated Lect course Exp-2?
NO
YES
Dealocate from partial to & allocate Conflict
NO
is Confl Lect level==partialallocated Lect level?
YES
is Confl Lect Course Exp==partialallocated Lect course Exp-2 ?
YES
NO
is Confl Lect Course Exp>=partialallocated Lect course Exp-2 ?
YES

Figure 13. Conflict Resolution

8. CONCLUSION/ FUTURE WORK

We acknowledge the fact that considerable efforts have been channeled towards school timetabling research but it is worthy to note that few of the works uses the concept of agency and very few follows ABSE paradigm.

Our work- Multi-Agent Based Course Allocator is fully ABSE compliant. We follow the Gaia methodology convention in the analysis and design of the system. The final artifacts of the design phase (Agent model, service model and acquaintance model) are directly mapped to the various classes provided by JADE framework.

The Admin Agent performs the allocation, monitors the environment for any request and does resolve conflicts among lecturer agents. This system has proven to increase the efficiency and a high level of appropriateness of course allocation with respect to a lecturer's choice and experience. The administrator can alter allocation when necessary and be able to update lecturer and course records when necessary. This is an extreme situation. As a matter of fact, the human administrator has less or no work except for this extreme case.

With respect to applications, ABSE is still at its infancy. In fact the software industry is yet to embrace it but it is envisaged that ABSE will be what OO (object-oriented) paradigm is today in less than two decades.
As future work,

- Adding learning capability. Learning in agent systems is a very interesting topic which we would have loved to have the time to investigate further. This capability could be introduced at various places in our design and would, at least in theory, which could eventuate to a more dynamic agent system.
- Making it a web-based so that system communicates with human via email.

ACKNOWLEDGEMENT
We acknowledge the researchers whose works were cited in our work especially the Gaia and JADE teams.

REFERENCES

[1] A. Nanda, M. P. Pai, and A. Gole, "An Algorithm to Automatically Generate Schedule for School Lectures Using a Heuristic Approach,"International Journal of Machine Learning and Computing vol. 2(4) , 2012.

[2] A. S. Rao, M. P. Georgeff, "Formal Models and Decision Procedures for Multi-agent Systems," Technical note 61, Australian AI Institute, level 6, 171 La Trobe Street, Melbourne, Australia. 1995.

[3] A.S. Rao, M.P. Georgeff, E.A. Sonenberg, "Social Plans: A Preliminary Report," Proc. of the 3rd European Workshop on ModellingAutonomous Agents in a Multi-Agent World (MAAMAW 91), pp. 57-76. Elsevier, Amsterdam. 1992.

[4] B. Brewington, "Mobile agents for distributed information retrieval," In Intelligent Information Agents, pp. 355-395. Springer, Berlin, 1999.

[5] D. Abramson "Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms," Management Science, vol. 37(1), pp. 98-113, 1991.

[6] D. Abramson, J. Abela, "A Parallel Genetic Algorithm for the Solving the School Timetabling Problem," Proc of the Fifteenth Australian Conference: Division of Information Technology, C.S.I.R.O, pp. 1-11, 1991.

[7] F. Tim, F. Richard, M. Don and M. Robin, "KQML as an Agent Communication Language," Proc. of the Third International Conference on Information and KnowledgeManagement, pp.456–463, 1994.

[8] F. Bellifemine, G. Caire, T. Trucco, G. Rimassa,: Jade Programmer's Guide. JADE 4.3 http://jade.tilab.com/doc/programmersguide.pdf, 2013.

[9] F. Zambonelli, N.R. Jennings, A. Omicini, M.J Wooldridge, Coordination of Internet Agents: Models, Technologies and Applications, chapter 13. Springer-verlag, Agent-Oriented Software Engineering for Internet Applications, 2000.

[10] FIPA: The foundation for intelligent physical agents. See http://www.fipa.org/

[11] FIPA-OS: A component-based toolkit enabling rapid development of FIPA compliant agents: http://fipa-os.sourceforge.net/

[12] FIPA Specification part 2 - agent communication language. The text refers to the specification dated 16 April 1999.

[13] FIPA specification XC00061E: FIPA ACL Message Structure Specification (2000) http://www.fipa.org

[14] G.S. Bello, M.C. Rangel, M.C.S. Boeres, "An Approach for the Class /Teacher Timetabling Problem," Proc. of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT2008), http://w1.cirrelt.ca/~patat2008/ PATAT_7_ PROCEEDINGS/Papers/Boeres-WA2b.pdf, 2008.

[15] I. Rezek, D. S. Leslie, S. Reece, S. J. Roberts, A. Rogers, R. K. Dash, N. R. Jennings,"On Similarities between Inference in Game Theory and Machine Learning," Journal of AI Research, vol. 33 pp.259-283, 2008.

[16] Jade site [online] http://jade.tilab.com/

[17] L. Tran-Thanh, A. Rogers, N. R. Jennings, "Long-term Information Collection with Energy Harvesting Wireless Sensors: A Multi-armed Bandit based Approach," Journal of Autonomous Agents and Multi-Agent Systems,vol. 25 (2). 2012.

[18] L. Padgham, M. Winikoff, Developing Intelligent Agent Systems- A Practical Guide. John Wiley & Sons, Ltd, 2004.

[19] L. Engmo, L. Hallen, Software Agent Applied in Oil Production. A master degree thesis, Department of Computer and Information Sciences, Norwegian University of Science and Technology, Norway, 2007.

[20] M. Niazi, A. Hussain (2010), "Agent-based Computing from Multi-agent Systems to Agent-based Models: A Visual Survey," 2010.

[21] M. P. Singh, N.M. N.M. Asher, "Towards a Formal Theory of Intentions," Proceedings of the European Workshop (JELIA-90), vol. 478,pp. 472-486. Springer, Berlin, 1991.

[22] M. P. Tariq, M. Waqar Mirza, R. Akbar, "Multi-agent Based University Timetable Scheduling System (MUTSS)," IJMSE, vol. 1(1), 2010.

[23] M. Wooldridge, An Introduction to Multi-agent Systems, John Wiley and Sons ltd., 2002.

[24] M. Wooldridge, "The Logical Modelling of Computational Multi-Agent Systems,"PhD thesis, Department of Computation, UMIST, Manchester, UK. 1992.

[25] M. Wooldridge,"Verifying that Agents implement a Communication Language," Proc. of the 16th National Conference on Artificial Intelligence (AAAI-99), Orlando, FL, pp. 52-57.

[26] M. Wooldridge, N.R. Jennings, D. Kinny,"The Gaia Methodology for Agent-oriented Analysis and Design,"Journal of Autonomous Agents and Multi-Agent System, vol. 3(3), pp. 285–312, 2000.

[27] N. R. Jennings, M. Wooldridge, "Agent-Oriented Software Engineering,", Handbook of Agent Technology. AAAI/MIT Press, 2001.

[28] N. R. Jennings, M. Wooldridge, Agent Technology: Foundations, Applications and Markets. Springer, Berlin. 1998.

[29] N. R. Jennings, K. Sycara, M. Wooldridge, "A Roadmap of Agent Research and Development,"Journal of Autonomous Agents and Multi-Agent Systems, vol. 1(1), pp. 7-38, 1998.

N. Spanoudakis, P. Moraitis, "Using ASEME Methodology for Model-Driven Systems Development," Springer Workshop, AOSE Toronto, Cannada, 2010.

[30] O. Mihaela, O.MAS,_ "UP-UCT: A Multi-Agent System for University Course Timetable Scheduling," International Journal of Computer, Communications & Control, vol. 2, pp. 94-102, 2007.

[31] P. R. Cohen, C. R. Perrault, "Elements of a Plan Based Theory of Speech Acts,"Cognitive Science, vol. 3, pp. 177-21, 1979.

[32] P. Vytelingum, T. D. Voice, S. D. Ramchurn, A. Rogers and N. R. Jennings, "Theoretical and practical foundations of Large-scale Agent-based Micro-storage in the Smart Grid," Journal of AI Research, vol. 42. 2011.

[33] P. Moraïtis, E. Petraki, N. I. Spanoudakis , "Engineering JADE Agents with the Gaia Methodology," Proc. Agent Technologies, Infrastructures, Tools, and Applications for E-Services, 2006, pp. 77–91, Berlin: Springer-Verlag.

[34] P. Moraïtis, N. Spanoudakis, "The Gaia2jade Process for Multi-Agent Systems Development," Applied Artificial Intelligence, vol. 20, 2006, pp. 251–273, Taylor & Francis Group, LLC, doi: 10.1080/08839510500484249

[35] Prometheus Methodology Tools, [online] Available: http://www.cs.rmit.edu.au/agents/pdt/index.shtml [21/01/2013].

[36] R. Kota, N. Gibbins and N. R. Jennings, "Decentralised approaches for Self-organising Agent Systems,"ACM Trans on Autonomous and Adaptive Systems, vol.**7** (1) , 2012.

[37] R.V.M. Lorena, (2011), "Requirement Modelling for Multi-agent Systems, " A master degree thesis, 2011.

[38] S. J. Russell, P. Norvig; Artificial Intelligence: Mordern Approach., Englewood Cliffs:, New Jersey 07632: Prentice Hall,1995.

[39] T. Birbas, S. Daskalaki, E. Housos "School Timetabling for Quality Student and Teacher Schedules," Journal of Scheduling, vol. 12(2), pp. 177-197, April 2009.

[40] T. Birbas, S. Daskalaki , E. Housos,"Timetabling for Greek High Schools," Journal of the Operational Research Society, vol. 48(2), pp. 1191-1200, December 1997

[41] T. Rahwan, T. Michalak, M. Wooldridge and N. R. Jennings, "Anytime Coalition Structure Generation in Multi-agent Systems with positive or negativeExternalities," Artificial Intelligence, vol.186**,** pp**.** 95-122, 2012.

[42] T. Takeshi, T. Megumi, "Production of the Time Table Management System of Commercial Course University," Proc. World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education, vol. 30, pp.48-52, 2006.

[43] V. Jean, N. Ambroise, *Jade Tutorial and Primer* [online] available: http://www.iro.umontreal.ca/~vaucher/Agents/Jade/JadePrimer.html

[44] X. Luo, C. Miao, N. R. Jennings, M. He, Z. Shen, "KEMNAD: A knowledge Engineering Methodology for Negotiating Agent Development," Journal of Computational Intelligence, vol.28(1), pp. 51-105, 2012.