AGENT ROLE LOCKING (ARL): THEORY FOR MULTI AGENT SYSTEM WITH E-LEARNING CASE STUDY

Salaheddin J. Juneidi, George A. Vouros

Department of Information and Communication Systems Engineering
School of Sciences
University of the Aegean

ABSTRACT
Advances in methods and techniques for software engineering are crucial for industrial and commercial applications, as these systems are required to operate in increasingly complex, distributed, open, dynamic, unpredictable, and inherently highly interactive environments. This article presents Agent Role Locking (ARL) theory supported by a case study as an example of engineering complex systems with autonomous entities, and managing their inherent complexity during analysis, design and implementation. Agent Role Locking (ARL) theory provides a new conceptualization of the relation between agents and roles in MAS. ARL calls on modification of UML interaction diagrams by introducing AIP diagram to preserves the distinguishing characteristics of agent software entities.

KEYWORDS
Agent Oriented Software Engineering (AOSE), Multi Agent System (MAS), AUML, super role, atomic role, Agent Interaction Protocol (AIP).

1. INTRODUCTION
Agent oriented software engineering (AOSE) has been introduced as a new paradigm for engineering complex software systems. Indeed, advances in software engineering are crucial for industrial and commercial applications, as software systems are required to operate in increasingly complex, distributed, open, dynamic, unpredictable, and inherently highly interactive application environments.

Agent based computing appears to be very promising for building such systems, integrating entities with agency characteristics into software applications [Bradshaw 1997, Wooldridge 1995]. Although many tools and applications are considered as agent software integrated applications [Odell 2002], but they mostly, do not follow any formal software engineering process. Recent surveys and evaluation of the most prominent approaches to AOSE [Teveit 2001, Juneidi & Vouros 2004] in terms of software engineering criteria as well as agent based computing criteria [Juneidi & Vouros 2004, Ardis 1995, Shehory 2000], reveal that none of the given methodologies are fully satisfactory to AOSE.

Most AOSE approaches agree on the importance of roles during analysis [Jennings 2000, Bradshaw 1997, Odell 2003, Odell 2004]. However, roles are not reflected in the final system design and implementation, which creates an engineering gap. This gap affects the development of complex systems with autonomous entities, restricts the level of complexity that can be handled during analysis, design and implementation of systems, resulting into systems with rigid structure, and with no adaptation abilities. As stated in [Juneidi 2004] a new engineering paradigm and a new way of thinking must emerge to adopt agents smoothly into the software development process.

This paper introduces Agent Role locking (ARL) theory which is an approach to AOSE that integrates role-oriented and goal-oriented system analysis, emphasizing on agents, roles and on their interplay, clearly distinguishing between agent and role entities. ARL has its own view of Multi-Agent System (MAS), gives agents entities the freedom to perform any “appropriate” role. ARL has its own methodology for analyzing MAS by defining environment; organizations belong to the environment, super roles and atomic roles of these organizations. It defines a new dynamic structure of agents’ interactions, Agent Interaction Protocol
(AIP), this dynamic structure of agency will then be integrated to Unified Modeling Language (UML), to be extended into Agent-UML.

This article is structured as follows: In section 2 we present a case study on e-learning system. This case study goes hand by hand with ARL definitions throughout the article. Section 3 explains ARL theory and its approaches towards MAS structure. Section 4 defines new developments on integrating agent diagrams into UML specifically through dynamic structure based on ARL definitions and assumptions. And finally, section 5 concludes this paper.

2. CASE STUDY (DISTANCE LEARNING)

We developed this case study as an example of managing MAS software environment using ARL theory. Normally, this software environment is characterized as a complex, distributed, open, dynamic, and unpredictable software. Now on we are going to explain how to apply ARL theory in a real software application such as distance learning. Distance learning is a part of e-learning system. E-learning refers to a wide range of applications and processes designed to deliver instruction through electronic means. Usually this means over the Web. Functional specification for e-learning systems captured according universities, firms, and e-learning digests interested in e-learning technology. The entire e-learning environment Figure 1 has the following functional organizations:

- **E-learning administration**: users (author, learner, instructor, reviewer, administrator) identification and certificates (keys) assignment, users profile (enrolment, interests, payment...etc), events, recourse, and course scheduling, user–user co-ordination and collaboration
- **Learning content management**: co-ordinate between users for adding and updating the learning materials into database repository, aggregate Learning Objects (LO) according to courses and various personalized user interests
- **Learning Management**: primary tasks in distance learning, login, course registration, view schedules and general portals ...etc. Teaching / learning through synchronous / asynchronous user’s collaboration, online course and class management, virtual classroom, student attendance and course completion reports (certificate), student assessment and testing and test scoring.

Added to these organizations, we get database repository, which includes learning contents of e-learning system; these contents are arranged into Reusable Learning Objects (RLO). Learning Object LO is a small piece of instruction or information that targets a specific performance goal; it might be a single graph or content.
video file or just a text, every agent type has different attitude to these RLO. The issue of aggregating and retrieving RLO is not in our context but it must be mentioned as a part of e-learning system essential element of e-learning. Distance learning is mostly represented in learning management organization. As shown in Figure 1 e-learning system have different agent types:

- **Learner**: requests to enroll in some tailor lessons or courses, he/she must follow up registration rules and payments. Attend / collaborate into the enrolled courses and give out the required assessments of registered course.
- **Instructor**: responsible for conducting the courses (synchronous / asynchronous), preparing tests and assignments for learner, collaborate with author, admin for updating courses contents in database repository.
- **Author**: preparing learning content material, receives requests from learner, instructor, administrator to add/change the learning contents. Submit this material to the reviewer for final approval.
- **Reviewer**: reviewing the submitted material from the author for approval, engages into course formulation and performance checking, through co-ordination with learner, instructor, and administrator. The reviewer has responsibilities and privileges of the author as well as instructor, the reviewer only review for material by other authors.
- **Administrator**: mostly responsible for user management organization, facilitating other agents collaboration, updating users (learner, author, instructor, and reviewer) profiles, scheduling events, courses formulation, add / update database repository according to reviewer approval by forming and personalizing learning objects for courses.
- **Dbase agent**: provides access to RLO in database repository according to course or profile personalization.

3. AGENT-ROLE LOCKING (ARL).

ARL introduces a new view of multi-agent systems structure. The main concept on which the whole approach is based on is role decomposition. The first step of analyzing MAS is to define the “boundaries” of that system, by these boundaries we set the environment we working with, MAS environment is set by major function and requirement specifications of a given system. For example, if we are talking about distance learning, so we have to take in consideration entire e-learning system as natural environment of distance learning. The next step is defining the organizations that belong to this environment. The organizational view identifies the most general objectives of the system; each organization has its associated objectives and tasks, which carried out by organizational positions with assigned responsibilities like the secretary, VP, manager …etc. E-learning environment has three main organizations. The organizational objectives are satisfied by positions and responsibilities, or roles, roles performers are agent entities.

3.1. Super & Atomic roles

ARL assumes that an individual agent can perform a single role in a given time. Therefore, it is important to determine the granularity of each role. For instance, if an agent is specified to play the role of studying at university, since agent is autonomous, it can change roles any time according internal state, so this agent could be as a reader, writer, chat-participant …etc. these eligible subsidiary roles of the student agent, which have different objectives responsibilities and can be performed any time by the student agent. Therefore, to reduce complexity, we have to decompose a very general role (super role) into more specific roles (sub-roles) until we reach to specific objectives to accomplished (atomic role). The motivation behind decomposition is to reduce the complexity of engineering MAS. Accordingly, we need fine-grained roles that can be played by discrete agents in the system.

To specify roles, the first step is to specify the super roles of each organization that reflect the coherent abstract services of the system. The set of super roles constitute the organizational view of the system being modeled. Analysis and specification of roles proceed in parallel to the analysis and specification of system’s objectives. Super roles correspond to the most general objectives of the organization. Each super role can be considered as a position within an organization being modeled. Having the super roles, one must specify the sub-roles of each role. The method for specifying sub-roles is to decompose each general objective of a given
role into more specific objectives. This process is called **objective normalization**. Objective normalization assures that no more than two agents communicate to accomplish one objective. The outcomes are atomic objectives that are carried out by **atomic roles** (e.g., check e-mails, send e-mail, perform pack-ups, buy, bid ... etc). Each atomic objective is accomplished by means of activities. We distinguish between two kinds of activities:

- **Independent activities**: Activities that belong to a role. Independent activities are performed individually by an agent, by using the objects in its disposal.
- **Dependent activities**: Activities that require two agents that play two distinguished atomic roles to communicate.

The objectives are **normalized** if and only if the activities of each objective (role) are either independent or dependent on any other same agent. Decomposition of roles proceeds until we reach atomic roles. These are the roles that have atomic objectives fulfilled either by independent or by dependent activities. In other words, the criterion to stop role decomposition, is that for each objective (role), there must be an activity that can be carried by a single agent (eventually locked in the role), either individually or by communicating with at most one other agent. In case the agent needs to communicate with more than one role player to fill full an objective, then for each of these role players’ agents, a new sub-role must be introduced.

![Organizational view (distance learning): Super roles and their sub-roles and atomic roles of Learning Management organization](image)

**Figure 2**: Organizational view (distance learning): Super roles and their sub-roles and atomic roles of Learning Management organization

Figure 2, depicts the Learning Management (LM) organization supported by four super roles essential for conducting distance learning processes. Each super role is decomposed into sub-roles and atomic roles. Agent may (**lock/unlock**) atomic roles. The atomic roles together specify super role functionality, super roles specify organizational functionality. The set of all organizations represent the (environment) functionality. Each agent must be able to determine its eligibility to play an atomic role. This eligibility is specified by super role assigned **key**. An agent which granted super role’s key is eligible to lock into any atomic role belongs to that super role. Agents perform atomic roles by two methods:

- **Role launching**: agent launches atomic role according to its internal state, functional requirement, or time bases.
- **Role satisfying**: agent performs atomic role that interdependent with an atomic role that been launched by another agent.

---

1 Due to one-to-one relation between atomic objectives and atomic roles, roles named according to objectives intended to accomplish.
As shown in Figure 2, role decomposition results into a role hierarchy from super-roles to atomic roles (top-down direction). LMS organization has super roles, each of which has been assigned with a key. This key specifies the constraints that agents must satisfy in order to play atomic roles. When an agent satisfies these constraints (has the key), then it can lock into and play these atomic roles. ARL is assigning keys for super roles, and give each agent the needed key(s) according to its responsibilities as in Figure 3. LMS organization has four super roles “course teaching” with key (θ), “course enrolment” with key (λ), “general tasks” with key (ω), and “assessment and performance” with key (π). All agents have the key (ω) to perform general tasks, but key (λ) is given to learner agent type, and keys (θ,π) are provided to the instructor. We can see key (π) also given to reviewer to conduct some course and material assessments as well as other keys from other organizational super roles.

![Figure 3. Agents types, instances and there keys](image)

3.2. Definitions In ARL:

The following definitions clarify expressions used MAS by ARL theory. It must be pointed that agent- role allocation is fully dynamic, the number of the agent instances vary as well as roles they perform. Each agent instance may lock, unlock, or instantiated into a different atomic role in a given time. According to the theory, and as shown in figure 3 agent types must own some super roles’ keys, which entitle them to perform ( launch / satisfy) atomic roles belong to that super role. The agent instances of any agent type will have the same key(s) given to that type:

**Agent type:** An agent type provides a blueprint of a specific agent category that can be instantiated. Instances of the same agent type share the same characteristics and each has its own id. For instance, Learning Agent type is denoted by L while its instances are denoted by L1, L2…

**Agent set:** An agent set is the set of all agents that have the same role key. Notice that while all instances of an agent type have the same role key(s), agent members in an agent set are not needed to be of the same agent type. For example agent set with the key (ω) can be {L1...Ln, R1…Rm}, were n and m are the number of agent instances of two types in a given time. (Note: reviewer may like to attend a class for performance check)

**Agent group:** Agent groups comprise agents locked into atomic roles, which interact in order to accomplish their dependent activities (i.e. these agents play interdependent atomic roles). Groups are denoted by Gi, i=1,2,3,… and are sets of tuples of agents. For instance in case of a virtual classroom the agents co-operate ( learner (L) and one instructor (I), Admin (A), Database agent (O)). Then the agent group at a given time of three learners in the classroom can be the following set of agent pairs G1={ (I1, A1),(I2,L1),(I3, L6) ,( I4, L12), (I5,O33) } (note: the thirty third (33) Database agent is communicating with this instant of Instructor at this time, the others are busy with other agents)

**Atomic Role Set (ARS):** Is the set of all atomic roles that belong to a super role. The ARS of “course teaching” super role is {arrange virtual class, run virtual class, learner check, get LO’s, replay to query}

**Super Role Set (SRS):** Is the set of all super roles in an organization. The SRS of LCMS organization is {aggregate RLO, content composing, add new content Dbase}

**Social system:** This comprises a set of interdependent atomic roles performed by agents in a period of time that have the potential for interaction towards achieving their objectives social system spans over organizations. (Note: social system is dynamic and not fixed)
Organization: It comprises a set of super-roles, agents’ permissions and keys, objects, as well as the interaction scenarios for performing the dependent activities of atomic roles. (Note: organizations in MAS environment are fixed)

Environment: consists of organizations and the social system with its all probabilities of agent collaboration and instantiation.

4. ARL integration with UML

UML has wide acceptance for modelling object-oriented software [Bauer 2000]. Since UML is the mostly widely known and use as current modelling language, so it’s naturally to think about integrating agent entities and behaviour into this modelling techniques. ARL has captured the complexity by role decomposition and controls dynamicity by manipulating role / agent entities as independent entities. In terms of modelling ARL support the idea of UML extension toward Agent UML.

Table 1.ARC table, some selected interdependent atomic roles

<table>
<thead>
<tr>
<th>Atomic role (launching)</th>
<th>Key</th>
<th>Atomic role (satisfying)</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrange virtual class</td>
<td>θ</td>
<td>open virtual class</td>
<td>β</td>
</tr>
<tr>
<td>Run virtual class</td>
<td>θ</td>
<td>Attend virtual class</td>
<td>α</td>
</tr>
<tr>
<td>Check learner</td>
<td>θ</td>
<td>Provide profile</td>
<td>α</td>
</tr>
<tr>
<td>get answer LO’s</td>
<td>θ</td>
<td>Content provide LO’s</td>
<td>χ</td>
</tr>
<tr>
<td>Instant massage</td>
<td>α</td>
<td>Replay to query</td>
<td>θ</td>
</tr>
<tr>
<td>Test online</td>
<td>π</td>
<td>Sit for exam</td>
<td>α</td>
</tr>
<tr>
<td>Open special course discussion</td>
<td>β</td>
<td>Special discussion</td>
<td>α</td>
</tr>
<tr>
<td>Registering to a course</td>
<td>λ</td>
<td>Add learner to a course</td>
<td>α</td>
</tr>
</tbody>
</table>

Having specified atomic roles in organizations, and having defined agents’ types, and having keys are spread on to agents types and to super roles. The next step is to set Atomic Roles Couples table (ARC). Table 1 defines the couples of interdependent roles and their candidate performing agents. Interdependent role contains dependant activities in one role that couldn’t be accomplish without the other role interaction. To specify the details of interaction between roles couples Agent Interaction Protocols (AIP) is used [Odell 2004]. Roles couples in ARC are represented by pairs; represent MAS interaction, communication and functionality. From the agent point of view, the system start functioning when an gent instance locks in a launching role (i.e. a role launched by the system). This role calls for another agent to lock into its interdependent satisfying role. The number of agents instances are variable start activating and deactivated roles according functional constrains, agent mental state, and agent-role switching constrains. Generally we have three forms of AIP:

• Simple atomic role coupling: atomic roles that are fully dependant on each other that means none of the atomic roles need special arrangements with other roles to commit its activities. For example figure 4, agent A launches X role, that interdependent on Y that been satisfied by agent B, agent role B/Y commit an independent activity, and dependent activities as communication with agent / role, A/X

![Figure 4. AIP Simple roles coupling](image-url)
• **Atomic role coupling with agent instantiation:** in this case atomic roles are cascade on couples, which carried out simultaneously, the acting agents of some role must instantiates itself to keep playing current role and make its instances to carry other atomic roles in role coupling scenario, an example Figure 5 shows two interdependent atomic roles X and Y, more than one agent instant launch X, the same number of agent instances of Agent B must satisfy role Y, to form roles interdependent couples of X and Y.

![Figure 5. AIP role coupling with agent instantiation](image)

• **Atomic role couples spanning:** atomic roles that communicate and cooperate, but one or both need third party role to commit some activities, in this case, atomic roles perform a spanning couple of roles using unlock \ lock mechanism. Take for example, figure 6, X and Y interdependent roles, but agent B must to leave (unlock) Y to launch (lock) into a third party interdependent roles couples (Z,W) before it gets back to its original role Y.

![Figure 6. AIP with agent role spanning.](image)

AIPs are integrated with other UML diagrams as a complementary diagram for specifying agents’ interaction. Returning to our case study, let’s consider the specification of the virtual classroom functionality. Starting from the UML Use Case diagram in figure 7, this can be integrated with AIPs. So we can specify the AIP given in figure 8 that gives representation to interaction processes among agents – roles in virtual classroom, which is a combination of simple, agent instantiation, and spanning role couples of AIP forms. We can notice that the Use Case diagram is in correspondence with the AIP diagram.
Figure 7. Use Case diagram for virtual class in e-learning

Figure 8. An entire view of roles couples needed to open and arrange virtual classroom with simple role coupling, role spanning and agent instantiation.
5. CONCLUDING REMARKS

Agent - Role Locking theory provides a new approach toward Multi Agent System analysis and design, the basic idea of ARL is, an agent should have always a role to perform, but it cannot perform more than one role at a given time. To apply this main idea we must determine the most fine grained role in which agent can perform in a given time. Atomic role are the most fine grained roles that represents normalized objectives. Atomic roles are a part of supers roles, super roles together represents organizations, organizations together (organizational view) is MAS environment. By this decomposition we manage to reduce complexity roles and dynamics of an agent

ARL has resolved many agent – role disputed issues in agent oriented engineering research field. Firstly, Pro-activeness and adaptability: By engineering agents and roles entities independently from each other, agent is free to launch or satisfy roles without specific preplanned scenario. Secondly system complexity management, naturally MAS is considered as a highly complicated system, that includes many interdependent roles, its elements (agents) interact and communicate to accomplish objectives. Role decomposition is the main factor for managing complexity by represents interaction into role coupling. Thirdly, agent autonomous behavior, by agent goal, a number of candidate roles the agent may lock / unlock into desired roles, instantiate to perform simultaneously other role according system constrains and functionality.

REFERENCES:


Bradshaw 1997 Jeffrey M. - Editor: Software Agents, American association for Artificial Intelligence ISBN 0- 262-52234-9,

Cognitive Design Solutions, Inc. with affiliation of CISCO systems Phoenix, Arizona, USA

E-learning post: is an intelligent digest of daily links to articles and news stories about Corporate Learning Community www.elearningpost.com


Juneidi S. J. 2004 Toward Programming Paradigms for Agent Oriented Software Engineering. IASTED International Conference on, Software Engineering (SE 2004), Innsbruck, Austria, 2004


Virtual University (VU): Nipomo, California, USA. Virtual University (VU) is the oldest online learning community on the World Wide Web.