Engineering an E-learning application using the ARL Theory for Agent Oriented Software Engineering

Salaheddin J. Juneidi, George A. Vouros
Department of Information and Communication Systems Engineering
School of Sciences
University of the Aegean
Karlovassi 83200, Samos
Greece
{jneidi,georgev}@aegean.gr

Abstract
Software engineering development is crucial for industrial and commercial applications as systems are required to operate in increasingly complex, distributed, open, dynamic, unpredictable, and inherently highly interactive environments. This work is being motivated by the need to engineer complex systems with autonomous entities, to manage systems' inherent complexity during analysis, design and implementation. This article presents the Agent Role Locking (ARL) theory that provides a new conceptualization of the relation between agents and roles in Multi Agent Systems. ARL concepts are being explained and illustrated using an e-learning system case study. ARL extends UML with both static and dynamic structures by means of role class, agent class diagrams and Agent Interaction Protocol (AIP) diagrams.

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 settings.

Agent based computing appears to be a very promising and natural development for building such systems, by integrating entities with agency characteristics into software applications. Therefore, AOSE, which appeals to the agent based computing paradigm, becomes a necessity rather than luxury for developing the required systems. Although many tools and applications concern with integrating agent entities into software applications [1], most of them do not follow any formal software engineering process.

Most AOSE approaches agree on the importance of roles during analysis [2,3,4,5]. However, roles are not reflected in the final system design and implementation. This 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, resulting into rigid systems, with no adaptation abilities. As stated elsewhere (e.g. in [6]) 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 Systems (MAS), emphasizing on agents' autonomous and flexible behavior: Agents may perform any "appropriate" role with respect to system objectives, permissions, own goals and other constraints imposed. ARL has its own methodology for analyzing MAS by defining the MAS environment, organizations that belong to the environment, super roles and atomic roles of these organizations. It defines the dynamic structure of agents' interactions by means of Agent Interaction Protocols (AIP), and the static structure of MAS entities by means of agent and role classes. An agent and a role class become an active class only when the agent entity plays (*locks into*) the role. The static and dynamic structures are integrated with the Unified Modeling Language (UML), to be extended into Agent-UML.

This article is structured as follows: Section 2 presents the ARL Multi-Agent Systems view. Section 3 presents the e-learning case study system. Section 4 explains the ARL theory and its approach towards the development of MAS. Section 5 presents the static and dynamic models' diagrams and their integration within UML. Finally, section 6 concludes the paper.

2. ARL: MAS view

The main concepts in ARL view of MAS structure are *Environment*, *Organization, Role,* and *Agent*.

Environment: According to ARL, identifying and modeling the environment involves determining all the entities, interactions among them and resources that entities in MAS can exploit, control or consume when they are working towards the achievement of system's objectives [7, 8]. This implies that everything (tangible or not) that affects system objectives must be

represented in the computational MAS environment. Therefore, ARL distinguishes between the physical environment in which MAS is situated and the MAS computational environment in terms of organizations, agents and roles.

Organizations: The organizational view in ARL comprises roles that correspond to systems' objectives. Roles are classified into *super roles*- that correspond to the most abstract system objectives- and *atomic roles* that correspond to fine-grained objectives that are subsidiary to the objectives of super roles. It must be pointed that:

- The organizational structure is fixed in terms of roles. Each organization has a fixed number of defined roles.
- Agents do not belong to any organization. However, at a specific time instance, some of these agents may (according to permissions granted, system needs and agents' internal state) lock to roles.

Roles: In a human organizational structure a role can be considered as a position. In ARL a role has a clear correspondence to system's *objectives*. Objectives are accomplished through activities, which as it will be explained, are distinguished to *dependent* and *independent* activities. These activities constitute the *responsibilities* of an agent that locks to the corresponding role.

Agents: Agents may perform legitimate (according to the permissions granted) roles in various organizations. Unlike other MAS methodologies, agents in ARL are the "activators" of the roles to be performed (and vice versa). Agent entities in ARL have different types[1], each of which has its own rights to perform some roles in MAS organizations.

3. The E-learning Case Study

The e-learning case study provides an illustrative example of developing a MAS application using the ARL theory.

E-learning refers to a wide range of applications and processes designed to deliver instruction through computational means. Usually this means over the World-Wide-Web [9,10,11].

Figure 1 shows the entire e-learning environment with the following organizations:

- **E-learning administration**. This organization is responsible for users' (author, learner, instructor, reviewer, administrator) identification and certificates (keys) assignment, users' profile (enrolment, interests, payment…etc) management, events, recourse, and course scheduling, user–user co-ordination and collaboration
- **Learning content management**. This organization has the objective to manage and co-ordinate the use of the learning material according to courses' objectives, personalized user interests, needs and preferences.
- **Learning Management**: This organization is responsible for users' login, course registration, provision of general information concerning courses and schedules, teaching / learning through synchronous / asynchronous user's collaboration, online course and class management, virtual classroom management, student attendance and course completion reports, testing and test scoring.

4. Agent-Role Locking (ARL) theory

The first step of analyzing MAS according to ARL is to define the MAS *environment*. This is set by the system's functional requirements and, as already pointed, it concerns all these aspects that affect system functionality. Specifying the MAS environment, one has to define the *organizations* that belong to this environment. To specify the organizational view one has to identify the most general objectives of the system. These objectives correspond to roles with responsibilities. Figure 1 specifies an ARL environment that comprises *1...a* organizations. Each organization comprises a number of *super roles.* These roles represent positions with responsibilities for achieving system objectives and identify permissions for playing roles, indicated by a *key*. Super roles correspond to abstract objectives and are decomposed to roles with subsidiary objectives and responsibilities. The decomposition process proceeds until we reach *atomic roles*. The distinction between super and atomic roles is further elaborated in section 3.1. Atomic role performers are agent entities that can lock into legitimate atomic roles in the organization. When locked into a role, agents take "responsibilities" to perform these roles. This ARL view of MAS, is consistent with the role definition in [12] and comprises:

- **Objectives**: Specify the functions that need to be performed.
- **Responsibilities:** Represent the activities that must be carried out for achieving the objectives.
- **Permissions:** Eligibility constrains for agents to lock into roles. These are represented by role/ agent keys.

[1] 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.

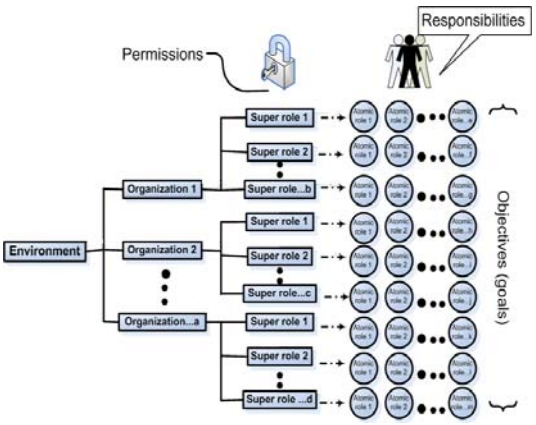

Figure 1: An overview of ARL on MAS environment, role decompositions and role specification.

ARL assumes that an individual agent can perform a single role at a given time. Therefore, it is important to identify fine-grained roles that can be played by agents to achieve specific objectives. For instance, if an agent is specified to play the role of studying at a university, this agent can play the role of a reader, writer, class-participant, etc. These are eligible subsidiary roles of the student agent, which have different objectives and responsibilities and can be performed by the student agent. Therefore, to reduce complexity, we have to decompose roles into more specific *sub-roles*.

To specify roles, the first step is to specify the *super roles* of each organization. As already pointed, super roles correspond to the most general objectives/functions of the organization. The set of super roles constitute the *organizational* view of the system being modelled. Analysis and specification of roles proceed in parallel to the analysis and specification of system's objectives. Having the super roles, one must specify their subsidiary roles. 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 (role players) communicate to accomplish one objective. The outcome includes atomic objectives that are carried out by *atomic roles* (e.g. check e-mails, send e-mail, buy, bid …etc).[2] Each atomic objective is accomplished by means of an activity. Such an activity can be of two kinds:

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

[2] Due to the close relation between atomic objectives and atomic roles, roles are named according to the objectives intended to accomplish.

Decomposition of roles proceeds until we reach atomic roles. These are the roles that have atomic objectives that can be fulfilled either by independent or by dependent activities. In other words, the criterion to stop role decomposition is that for each role, there must be an activity that can be carried by a single agent eventually locked in the role, either individually or in interaction with at most one other agent. In case an agent needs to communicate with more than one role player to fulfil an objective, then for each of these roles, a new sub-role must be introduced.

The motivation behind decomposition is to reduce the complexity of engineering MAS by identifying the fine-grained roles (*atomic roles)* that can be played by discrete agents in the system for achieving specific objectives.

Figure 2 depicts the Learning Management (LM) organization that comprises four super-roles. Each super role is decomposed into atomic roles. The atomic roles specify the functionality of the organization. The set of all organizations represent the functionality of the system as a whole.

Agents may perform (*lock/unlock into*) atomic roles. The eligibility of an agent to play an atomic role is determined by means of the *key* assigned to the super-role of this atomic role. An agent that has the super role key is eligible to lock into any atomic role which is subsidiary to that super role. Agents perform atomic roles by two methods:

- **Role launching:** An agent may launch an atomic role according to its internal state.
- **Role satisfying:** An agent may play an atomic role that is interdependent to an atomic role that has been launched by another agent.

5. Dynamic and static models

ARL provides the *dynamic model* for the representation of communication, interaction, and behaviour in MAS, and the *static model* for the representation of agent and role classes. ARL supports the idea of UML extension toward Agent UML: Proposed models integrate smoothly with UML.

The following sub-sections clarify these two upgraded types of models and provide examples from the e-learning case study.

launching role	key	satisfying role	key
Arrange virtual class	Θ	open virtual class	β
Run virtual class	θ	Attend virtual class	ω
Check learner	θ	Provide profile	α
get answer LO's	θ	Content provide LO's	χ
Instant massage	ω	Replay to query	θ
Test online	π	Sit for exam	ω

Table 1: ARC table, some selected interdependent atomic roles

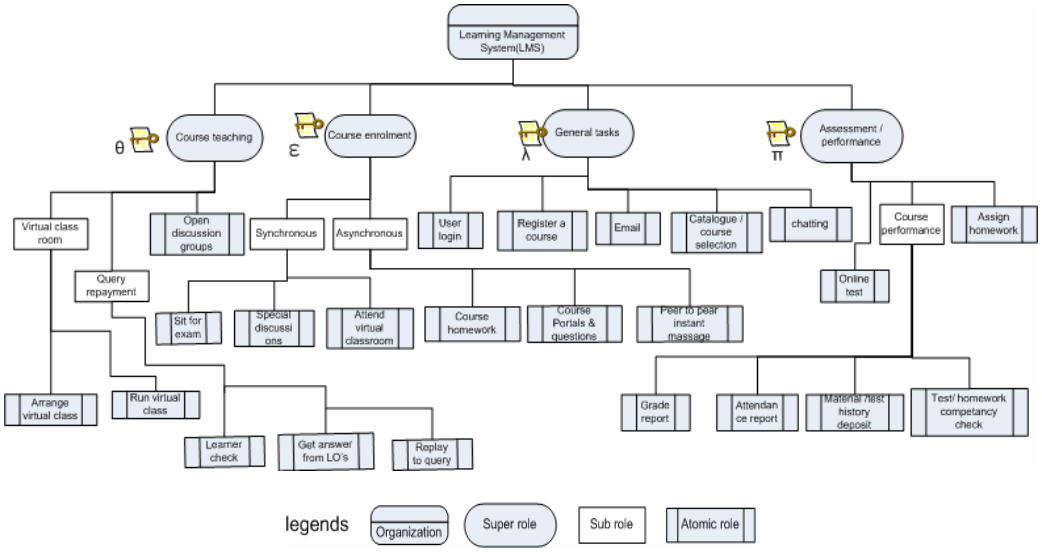

Figure 2: Organizational view (distance learning): Super roles and their sub-roles and atomic roles of Learning Management organization

5.1 Dynamic model

Having specified the atomic roles in each organization and the types of agents, keys are spread on agent types and super roles. Then we can collect interdependent atomic roles into the *Atomic Roles Couples table (ARC).* This table defines the couples of interdependent roles and the keys required by the candidate agents. An example of an ARC is provided in Table 1. *Interdependent roles* are roles with dependent activities. To specify the details of interaction between role couples ARL uses the Agent Interaction Protocol (AIP) that has been proposed in [13,14,15].

From the agent point of view, the system starts functioning when an agent locks into and launches a role that interacts with an interdependent role. The latter role calls for a satisfying agent, which when it locks into the role interacts with the former agent towards performing their dependent activity.

ARL distinguishes between three types of AIP:

- **Simple atomic role coupling**: In this case none of the atomic roles needs special arrangements with other roles to perform its activities.
- **Atomic role coupling with agent instantiation:** In this case, couples of atomic roles must be performed simultaneously. Acting agents (i.e. agents playing a role) must instantiate themselves to keep playing their roles and make their instances lock into other atomic interdependent roles.
- **Atomic role couples spanning**: In this case agents playing atomic roles interact, but at least one of them needs to lock to a third party role to perform some of its (dependent) activities. In this case, agents perform a spanning couple of roles using the unlock \ lock mechanism.

AIPs are integrated with other UML diagrams 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 3, interactions among interdependent roles shown in Figure 3 are specified by means of AIPs. So we can specify the AIP given in Figure 4 that represents the interaction among agents locked into the Instructor, Learner and Admin roles in a virtual classroom. This AIP shows a combination of simple, agent instantiation, and spanning role couples.

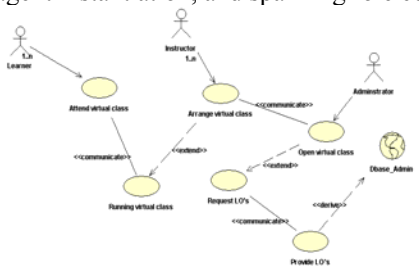

Figure 3: Use Case diagram for a virtual class in the e-learning system

5.2 Static Model

ARL proposes the specification of agent and role classes as two separate entities. When instances of agent classes lock to specific atomic roles, then the agent and the role become active. This new static view can be integrated with UML reaching to Agent UML that supports object as well as agent and role classes, to

represent static models of MAS. Separating these two types of entities (agents and roles) ARL can represent phenomena where an agent entity can "move" from one role to another without any pre-assigned agent-role mapping: Agent entities can be instantiated to perform atomic roles, agents can move freely and instantiate themselves according to system functionality constrains (agent–role switching constrains[14,16]) or according to their internal mental state.

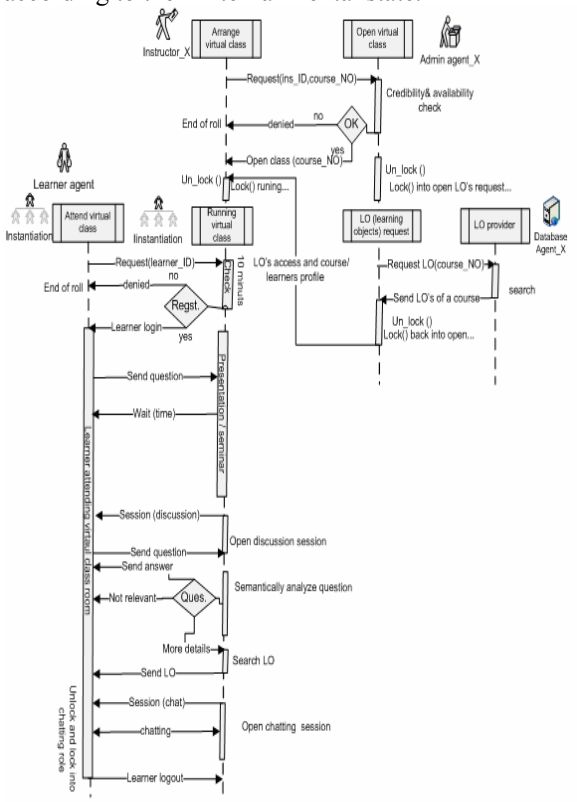

Figure 4: An entire view of role couples needed to open and arrange a virtual classroom

Agent Class. As shown in Figure 5 this class has the following attributes:

Agent Type –ID: The name of the agent type with an identification number determined by agent instantiation.

Internal State: Description of agents' internal state, including the role to which the agent has locked for a time interval and the candidate roles for the agent to perform next. For more advanced agent architectures, the internal state may contain agent's beliefs, desires, goals and intentions as well as other knowledge elements.

Key-role: This is the certificate that entitles an agent to play some roles. An agent may have more than one key. However, it can lock only to one role at each time.

Agent Goals: Each agent type may have some goals that are set by the agent designer. Goals are normally consistent to the role objectives.

Role sensors and preceptors: Methods for the agent to sense the roles that need to be satisfied, and to perceive the physical environment in which it is situated. For more advanced agents these methods may enable agents to learn new keys.

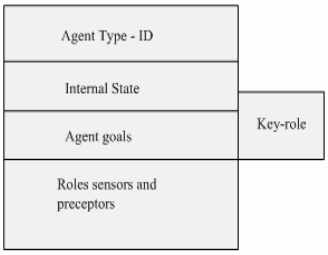

Figure 5: Agent – Class

Role Class. The role class represents atomic roles and, as Figure 6 shows, it has the following attributes:

Flag-vacancy: This is binary value attribute, which shows that the corresponding role needs to be served by an agent.

Role-name: A description of role objectives. In case of atomic roles this is the atomic objective of this role.

Role constrains and rules: It represents special conditions and constrains that need to be satisfied for performing the role.

Service description: Represents the dependent and independent activities of the role. More detailed descriptions of independent activities can be specified by means of UML activity diagrams or state charts.

Role-key slot: A method that checks whether an agent is eligible to lock into this role.

Communication Acts & Protocols: These specify the protocols for dependent activities. More detailed descriptions of protocols can be provided by means of Agent Interaction Protocols (AIP).

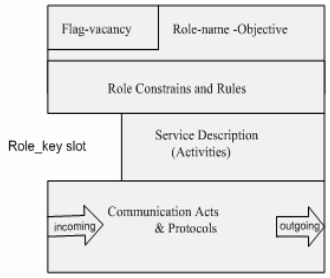

Figure 6: Role – Class

6. Concluding Remarks

The Agent - Role Locking theory provides a new approach towards engineering Multi Agent Systems. The basic idea of ARL is that an agent, to be active, must have a role to perform, but it cannot perform more than one role at a given time.

ARL proposes the following additions to UML towards Agent UML (AUML):

- Modeling dynamic aspects of MAS by means of Agent Interaction Protocol (AIP), so as to capture the system dynamics and interaction though agent-role to agent- role communication.
- Modeling static structure of MAS by means of distinguishing between agent and role classes.

ARL has resolved many agent – role disputed issues in the agent oriented engineering research field. Firstly, pro-activeness and adaptability: By engineering agents and roles entities independently from each other, agents are free to launch or satisfy roles without a specific preplanned scenario. Secondly, it supports managing system complexity. MAS are considered to be highly complicated systems, including many interdependent roles and entities interacting to accomplish objectives. Role decomposition is the main method for managing complexity. Thirdly, ARL supports agents' autonomous behavior, since an agent may lock to candidate roles, as well as instantiate itself to perform simultaneously other roles according to system's constrains and functionality.

References:

1. Bernhard Bauer, Jorg P. Muller, James Odell. Agent UML: A Formalism for Specifying Multiagent Software Systems. *Proceedings, ICSE 2000 Workshop on Agent-Oriented Software Engineering AOSE* 2000, Limerick. Springer Verlag, pp. 121-140
2. Marc-Philippe Huget, Extending Agent UML Sequence Diagram, F. Giunchiglia ct al. (Eds.) : *AOSE 2002 ,Lncs 2585*, pp 150-161, Springer-Verlag Berlin Heidelberg 2003.
3. Jemes Odell, H. Van Dyke Parunak, Sven Brueckner, John Stuart : Temporal Aspects of Dynamic Role Assignment, *AOSE 2003, LNCS 2935,* pp. 201-213, Springer – Verlag Berlin Heidelberg 2004
4. Ioannis Partsakoulakis and George Vouros. Roles in MAS: Managing the Complexity of Tasks and Environments. Multi-Agent Systems: An application Science, T. Wagner (eds.), Kluwer Academic, 2004
5. James Odell, H. Van Dyke Parunak, Mitchell Fleischer: The Role of Roles, in *Journal of Object Technology,* vol. 2, no. 1, January-February 2003, pages 39-51.
6. S. J. Juneidi, Toward Programming Paradigms for Agent Oriented Software Engineering. *IASTED International Conference on Software Engineering (SE 2004*), Innsbruck, Austria, 2004 pp 431-436.
7. M. Wood and S.A. DeLoach. An overview of the multiagent systems engineering methodology. In Agent-oriented software engineering (P. Ciancarini and M.J. Wooldridge Eds), *Proceedings of the First International Workshop (AOSE-2000),* Lecture Notes in Artificial Intelligence, Vol. 1957, Springer-Verlag, 2001, pages 207-222.
8. Michael Wooldridge , Nicholas Jennings, David Kinny. The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agent and Multi- Agent Systems*, Kluwer Academic Publishers pp. 285-312, Netherlands, 2000.
9. E-learning post: *http://www.elearningpost.com*.
10. Cognitive Design Solutions, Inc. with affiliation of *CISCO systems* Phoenix, Arizona, USA.
11. Virtual University (VU): Nipomo, California, USA.
12. M.Wooldridge and N.R. Jennins Intelegent Agents: Theory and Practice ,*The Knwledge Eng. Rev*. Vol l.,10, Nov.2,1995
13. Marc-Philippe Huget, Extending Agent UML Sequence Diagram, F. Giunchiglia ct al. (Eds.) :proceedings *AOSE 2002*, pp 150-161, Springer-Verlag Berlin Heidelberg 2003.
14. Salaheddin J. Juneidi, George A. Vouros. Agent Role Locking (ARL): Theory for Multi Agent System with E-Learning Case Study, *IADIS International Conference Applied Computing* 2005 Algarve, Portugal February 2005.
15. Jorg P. Muller, Barhard Bauer, Jemes Odell, Agent UML : A *formalism for Specifying Multiagent Interaction*, Springer – verlag , Berlin , pp. 91- 103, 2001
16. Salaheddin J. Juneidi, George A. Vouros, Agent Role Locking (ARL): Theory for Agent Oriented Software Engineering, *IASTED International Conference SE November* 2004, MIT, Cambridge, MA. USA.