A Role-Based Approach in Dynamic Task Delegation in Agent Teamwork

Yu Zhang*

Department of Computer Science, Trinity University, San Antonio, TX 78212

yzhang@cs.trinity.edu

Abstract—**This paper presents a role-based agent teamwork language called RoB-MALLET (Role-Based Multi-Agent Logic Language for Encoding Teamwork). Roles have been used to form multi-agent theories and guide the construction of multi-agent architectures and systems. Different from the most existing agent teamwork models that emphasize practical reasoning based on beliefs, desires and intentions (BDI), a role-based teamwork language stresses the concept of organization and thereby cooperation. RoB-MALLET is able to handle unexpected uncertainties and allow for the conceptual specification of teamwork knowledge for reuse. Finally, we present several experiments that explore plan reusability of RoB-MALLET, show that it is flexible in supporting simultaneity of invocation and evaluate its efficiency in supporting teamwork.**

Index Terms—**Role-based Collaboration, Task Delegation, Multi-Agent Systems, Agent Teamwork, Agent Plan**

I. INTRODUCTION

Teamwork has recently become increasingly important in many disciplines, such as virtual training [1], internet-based information integration [16], Robocup soccer [17] and interactive entertainment [19]. A great amount of research has been done to facilitate multiple agents to work closely as a team in complex and dynamic domains, in which agents are autonomous, heterogeneous distributed over various platforms [29, 27, 7, 34]. However, ensuring that agents work as an effective team remains a very difficult challenge [28].

Teamwork theories, such as Cohen and Levesque's joint intention theory [10], Grosz and Kraus' shared plan theory [14] and Jennings' joint responsibility [15], have explored the critical mental states underlying teamwork driving agents to act together as a team and the interactions leading their individual actions to team efforts. The critical mental states include operators, mutual beliefs, shared goals, team plans, and joint intentions. Through interactions, agents transform their team mental states (such as joint goals, joint intentions and team plans) to individual mental states (such as individual goals, individual intentions and individual

operator/plan); and the evolution of mental states eventually leads to the individual actions.

To apply these teamwork theories to simulate teamwork, a teamwork language is demanded to explicitly express the mental states underlying teamwork. Particularly in complex and dynamic domains, many unexpected uncertainties could occur, such as dynamic changes in team's goals, team members' unexpected failures to fulfill their responsibilities, dynamic decision-making in dynamic environment, and dynamically backing up other team members. The decisions on handling these uncertainties are tightly related to the mental states underlying teamwork. To specify the knowledge about handling these uncertainties, it is more important for the teamwork language to explicitly express the mental states underlying teamwork. Considering the perspective of software engineering, the teamwork language would better allow to specifying teamwork knowledge conceptually for being reused, particularly, team plans are better specified in terms of abstract entities, instead of specific agents, so as to be reused by different teams of agents.

Roles are increasing recognized as a valuable abstraction for meeting the above requirements. STEAM (Shell for TEAMwork) uses roles to deal with the case that an individual defects from the group [28]. Stone and Veloso introduced roles as a mechanism for specifying an agent's internal and external behaviors and decomposing team tasks [26]. JACK Teams introduced the team reasoning entity, which encapsulates team behaviors through roles and team plans [6]. In ROPE (Role Oriented Programming Environment), invocation consists of passing an agent a role and an execution environment, and it is up to the agent to carry out the role in that context [4]. TOP (Team-Oriented Programming) focuses on defining the notions of joint team mental states [30, 25], and OMNI (Organizational Model for Normative Institutions) define the social structures and interactions within a group of agents [11, 32].

In this paper, we propose a teamwork language called RoB-MALLET (Role-Based Multi-Agent Logic Language for Encoding Teamwork), an extension of MALLET [12], which has rich expressivity to specify the critical mental states underlying teamwork explicitly and allows dynamic team structures. Our approach is to first develop a team organization that maintains a team process. The team organization is a hierarchy of shared goals, the role-based plans to achieve the goals, and the

team structures for agents to execute the plans (i.e., the team assignments and role variable selections for executing the plans). A team process is distributed among the team of agents by each agent maintaining a team organization and their own individual process; the individual processes are complementary to each other; and the overlapping of shared mental states is only in the agents' team organization. Using team organization can dramatically reduce the communication for maintaining shared mental models, increase the concurrency of executing team processes, and further improve team performance.

Second, we define three formal concepts, position, role and role variable, so that team plans, called role-based plans, are specified in terms of these concepts instead of specific agents. Unlike the informal concept of role used in other teamwork architectures, such as the roles used in STEAM [28] and TOP [30] and the variant of role (agent variable) used in MALLET [12], the concepts of position, role and role variable are formalized based on the classificatory concepts in role theory literature that capture the behavioral aspects of role [33]. A particularly important construct in RoB-MALLET is the role-based plan. This is specified in terms of a virtual team of roles and a set of role variables. The actions associated with roles and role variables are specified in the process of the role-based plan. The temporal ordering on the actions is specified by the constructs of flow control. Moreover, a role-based plan contains a set of constraints on delegating the roles in the plans to agents. Before a team of agents actually executes a role-based plan, each role in the plan needs to be delegated to an agent; and the delegations of all roles to the agents must satisfy the constraints on delegating the roles in the plans to agents. During the plan execution, a role is dynamically selected to fill a role variable; and the selected role must satisfy the constraints on the selection. We use a CSP (Constraint Satisfaction Problem) algorithm [18] to search an admissible team assignment for a team of agents to execute a role based plan and to select a role for a role variable selection. In addition, we show how role-based plans could be used as actions together with individual operators in planning algorithms (either forward or backward state-space search), and give a simple version of planning algorithm that just searches a role-based plan for a team of agents, without decomposing a goal to a sequence of states and using operators.

Third, we develop the mechanisms for task decomposition and task delegation, which are compatible with the evolution of mental states and practically lead to individual actions, based on role-based plans. Without loss of generality, a task of a team of agents is that the agents invoke a role-based plan. To enable our mechanisms of task decomposition and task delegation, we define a notion of responsibility in terms of what a responsibility contains and how a responsibility impacts the mental states of the agent(s) taking the responsibility. The team responsibility of all the roles in a role-based plan contains all temporally ordered actions associated with the roles and their impact on the mental states of the

agents taking the team responsibility. The individual responsibility of a role (or role variable) contains the temporally ordered actions associated with the role (or role variable) and their impact on the mental states of the agent taking the responsibility.

II. TEAM ORGANIZATION

A team organization is a tree structure, showed in Fig 1, representing the hierarchy of shared goals, the plans to achieve the goals, and the team assignments for executing the plans. Each node in the tree structure is a tuple (A, G, P, S), where A is a team of agents, G is a shared goal of A, P is a role-based plan (described in Sec. III) to achieve G, and S is the team structure of A invoking P. The agents may be working on multiple goals at the same time. To accommodate this, a special node is used as the Root of the tree structure which points to simultaneous goals at the top level. In Fig. 1, the agents are involved in a set of simultaneous goals $G1$, $G2$, …, Gi.
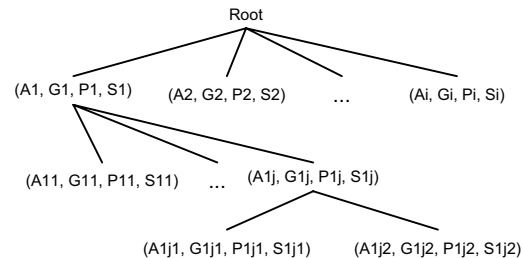

Fig. 1 An Example of Team Organization

The tuple (A, G, P, S) is used to record two critical team mental states: 1) a shared goal (note: a shared goal in our teamwork architecture is similar to joint persistent goal in [10]), and 2) a joint intention, denoted *JIntend(G, VT, TA, P)*, where *VT* (Virtual Team) is the set of roles in P (described in Sec. III), *TA* is the part of the team assignment in team structure S. To achieve their shared goal G, the agents in A jointly intend to perform plan P by delegating the roles in P to them according to team assignment *TA*. Moreover, the relationships between goals and their sub-goals are maintained. In Fig. 1, goal $G1j$ consists of sub-goals $G1j1$ and $G1j2$.

Each agent has a copy of the team organization. However it is not necessary that all copies be identical. Each agent may maintain its team organization in one of two different modes: 1) *relevant*, i.e., it just maintains the goals in which it is involved and the role-based plans and team structures to achieve these goals of performing the plans, and 2) *full tracking*, i.e., it maintains all goals, no matter whether it is involved in them, and the role-based plans and team structures to achieve these goals of performing the plans. These two modes result in different overlapping on the copies of team organization. With the relevant mode, any two copies of team organization may be different from each other, and less synchronization to maintain a team is required. With the full tracking mode, all agents have identical copies of a team organization, and more synchronization to maintain a team is required. To make the agents accomplish their teamwork, the

agents at least maintain the goals which they involve and corresponding role-based plans and team structures.

III. ROLE-BASED PLAN

In their role theory, Biddle and Thomas concluded that roles can be defined based on partitioning concepts for persons and their behaviors [5]. They also pointed out that pre-association with roles is too restrictive, i.e. the determination of which role actually performs which action being determined dynamically in a specific situation. Based on it, we define a position as all entities that can perform a set of primitive operations, and use this to characterize a collectively recognized category of persons who are able to exhibit a set of behaviors. We define a role as an entity of a position associated with a bag of temporally ordered actions. We define a role variable as an entity which is dynamically selected from a set of roles to be associated with a bag of temporally ordered actions. Generally speaking, a role variable stands for some role out of a set of roles. Depending on concrete situation, one role from the set is dynamically selected to fill the role variable. When this happens, the bag of operations of the assuming role is dynamically expanded.

A. Position

We define a concept of position to refer a collectively recognized category of entities (persons) that exhibit a set of behaviors.

Definition 1. An **entity** is an abstraction of any performer (e.g., agent or person) that is able to perform operations.

Definition 2. A **position** is a named set of all entities that are able to perform a set of primitive operations, denoted by operators. Given a set of operations O, a position based on the operations is

$$P(O)=\{e | e \in Entities \wedge \forall o \ (o \in O) \wedge Capable(e, o)\}, \quad (1)$$

where e represents an entity and $Capable(e, o)$ means that e is able to perform operation o, assuming the preconditions of o are true.

The purpose of defining position is to capture the capability requirements on agents. Every entity of a position is required to be capable to do the operators defined in the position. For example,

$$\text{(POSITION sniffer (talk move movein randmove sense selectTarget)).} \quad (2)$$

A position *sniffer* is defined by a set of operators, including *talk*, *move*, *movein*, *randmove*, *sense*, and *selectTarget*. Suppose e is an entity that takes on the position *sniffer*; e should be able to perform the above operators.

B. Role

We base our definition for a role on a position. In other words, any action (i.e., primitive operation) associated with the role must be in the operation set of the position. Let O be a bag of operations that must be performed by the same entity of a position, $COND$ be a bag of conditional operators where each action is contingent on a conjunction of conditions, and C_O be a bag of ordering constraints that impose a "temporal order" on O and the evaluation of the conditions in $COND$. We define a role r as an abstraction of the entity of position P that satisfies the constraints.

Definition 3. A **role** is an abstraction of an entity that performs a specific bag of operations and includes temporal constraints on the order in which the operations may be performed:

$$r = (id, P, O, COND, <_r), \quad (3)$$

where id is the name of r and refers to the entity of position P that must perform the operations in O.

$<_r$ is a set of temporal orders as (o_i, o_j) where $o_i, o_j \in O \cup COND \cup S$, and $S = \{o_s, o_e\}$. o_s is a dummy starting operation that can be performed by any entity of a position, and o_e is a dummy ending operations that can be performed by any entity of a position. Temporal orders are transitive. That is, if there are two temporal orders (o_i, o_j) and (o_j, o_k), then (o_i, o_k) is a true temporal order too. However, C_O is not a transitive closure. That means, even though (o_i, o_j) and (o_j, o_k) are in C_O, (o_i, o_k) may not be in C_O. And, cycles may exist in the set of temporal orders C_O; that is, there exist temporal orders as (o_1, o_2), (o_2, o_3), $\ldots$, (o_k, o_1), where $k \geq 2$. This does not mean a temporal conflict. Rather, it means that the operators may be executed more than once in accordance with the temporal orders.

An important characteristic of our notion of role is that it includes the specific operations that an agent filling the role must perform in a specific setting as well as their ordering constraints. The operations associated with a role will be derived as a plan. Thus, the specific operations and constraints in O and C_O are dynamically determined as the execution of system progresses. Roles are thus dynamically determined. However, the static nature of the position underlying a role allows constraints to be imposed that assure that any agent taking on a role will be capable of performing the operations of the role.

C. Role Variable

A role variable is similar to a role except that the role variable is dynamically selected from a set of roles. So, in the definition of a role variable, we do not explicitly require it to belong to any specific position. Rather, we require a selection constraint whose satisfaction will select one out of a set of roles to perform the operations of the role variable in accordance with the specified order.

Definition 4. A **role variable** is an abstraction of an entity that is dynamically selected from a set of roles to be associated with a bag of temporally ordered actions:

$$rv = (id, O, COND, <_{rv}, RS, SC), \quad (4)$$

where *id* is the name of role variable *rv*, *RS* is the selection scope, i.e., a set of roles, *SC* is the selection constraint, $<_{rv}$ is the ordering constraints.

D. Role-Based Plan

In RoB-SMM, processes are the place to specify team activities in terms of roles or role variables.

> *Definition 5.* Suppose the **process** of a role-based plan is composed by a set of roles $r_1, r_2, ..., r_m$, where $1 \le i \le m$, and a set of role variables $rv_1, rv_2, ..., rv_n$, where $1 \le j \le n$. The process, denoted *Proc*, is the union of the roles and role variables:

$$Proc= \bigcup_{i=1}^{m} O_{r_i} \cup \bigcup_{j=1}^{n} O_{rv_j}, \bigcup_{i=1}^{m} Cond_{r_i} \cup \bigcup_{j=1}^{n} Cond_{rv_j}, \bigcup_{i=1}^{m} <_{r_i} \cup \bigcup_{j=1}^{n} <_{rv_j}, \bigcup_{j=1}^{n} RS_{rv_j}, \bigcup_{j=1}^{n} SC_{rv_j} \qquad (5)$$

The difference between the actions associated with roles and actions associated with role variables is that action associations by roles are static (or direct) while action associations by role variables are dynamic (or indirect). In this way, the process of a role-based plan allows both static and dynamic action association. Moreover, the actions associated with role variables are dynamically associated with the roles which fill the role variables. Thus, even though there are both static and dynamic action associations, a role-based plan still can be viewed as being composed by roles, and role variables can be viewed as an intermediate to express dynamic action associations. We call the aggregation of the roles in a role-based plan the *virtual team* of the plan, denoted by *VT*. We call the aggregation of the role variables in a role-based plan the *role variable set* of the plan, denoted by *RVS*. We note that roles in *VT* may form a sub-team to invoke a sub-plan and the sub-plan contains a virtual team too.

To delegate roles to agents, the agents must be "qualified" to fill the roles (we call this process role assignment, see Sec. IV for detail). A necessary condition for this is that the agent must be able to perform operations required by the role. As well as the qualification of the roles, there can be a set of social norms (i.e., conditions) which specify the selection of agents for roles. The social norms may exist for different kinds of reasons, such as geographical reasons, workload balance, and cooperation requirement. We define social norms as a set of constraints on delegating roles to agents and denote these constraints by Γ.

> *Definition 6.* A **role-based plan** is defined as
> $$P = (\Phi, \theta, \Psi, VT, \Gamma, RVS, Proc) \qquad (6)$$

where Φ is a set of preconditions, θ is a set of effects, Ψ is a set of termination conditions, *VT* is a virtual team, Γ is a set of constraints specifying social norms on delegating roles to agents, *RVS* is a set of role variables to specify non-predetermined actions, and *Proc* is the process of actions of *VT* achieving the pursuing goal (i.e., effects) under similar situations (i.e., preconditions). By this means, team plans are reusable for different agents, i.e. they are to achieve the effects in

any situation in which the preconditions are satisfied, independent of the agents executing the plan.

Fig. 2 shows an example role-based plan. It is a multi-agent version of the classic wumpus world problem [24] (see Sec. V for details). It defines a role-based plan scanandkill and two positions (sniffer and fighter). The virtual team consists of three roles, r_1, a sniffer, and r_2 and r_3, fighters. Also, a role variable *?fi* is selected from *r2* and *r3* and the one closest to the wumpus to be killed is selected. The constraints on delegating agents to roles also are specified. To fill r_2 and r_3, agents must have arrows, and r_2 and r_3 must be filled by different agents.

```
(position sniffer (move sense selectTarget))
(position fighter (move shoot))
(plan scanandkill ()
  (Roleteam ((role (r1) sniffer) (role (r2 r3) fighter)))
  (Constraint (hasarrow r2) (hasarrow r3) (NE r2 r3))
  (Process
    (par
      (while (cond (eq 1 1))
        (seq
          (do r1 (selectTarget))
          (if (cond (targetloc r1 ?X1 ?Y1))
          (do r1 (senseandmoveto ?X1 ?Y1)))))
      (while (cond (eq 1 1))
        (if (cond (wumpus ?X2 ?Y2))
          (seq
            (select ?fi (r2 r3) (closestto ?X2 ?Y2))
            (do ?fi (movecloseto ?X2 ?Y2))
            (do ?fi (shoot ?X2 ?Y2)))))))))
```

Fig. 2 An Example Role-Based Plan

IV. TASK DECOMPOSITION AND TASK DELEGATION

A. Generating Team Responsibility Graph for Role-Based Plan

A responsibility is an obligation by an agent or a team of agents to perform actions, which forms an intention to do the actions. When an agent *ag* takes the responsibility of a role *r*, the agent forms an intention, denoted by Intend(*ag*, *r*, *P*), to execute the plans *P* associated with the role *r*. We define Intend(*ag*, *r*, *P*) on Cohen and Levesque's persistent goal (PG) [10] as follows:

- Agent *ag* has a PG relative to the negation of the termination condition of *P* of having done all the actions associated with role *r* (denoted by **Resp**(*r*, *P*)) in accordance with the temporal orders on the actions.
- Agent *ag* believes throughout that it is doing the actions in **Resp**(*r*, *P*) until all the actions are done or the termination condition of *P* is satisfied.

When a team of agents *T* takes the responsibility of the virtual team *VT* in a role-based plan *P* with a team assignment *TA*, the agents in *T* form a joint intention (denoted by JIntend(*T*, *VT*, *TA*, *P*)) to execute the actions associated with the roles (denoted by Resp(*VT*, *P*)). We define JIntend(*T*, *VT*, *TA*, *P*) based on Cohen and Levesques' joint persistent goal (JPG) as follows:

(1) The agents in *T* have a JPG, relative to the negation of the termination condition of *P,* of having done all actions in Resp(*VT, P*) in accordance with the temporal orders on the actions.

(2) Until all the agents in *T* mutually believe that all actions in Resp(*VT, P*) are done or that the termination condition of *P* is satisfied, the agents in *T* mutually believe that: a) they are doing the actions in Resp(*VT, P*) in accordance with the temporal orders on the actions; and b) agent *ag* in *T* takes the responsibility of the role *r* delegated to it.

Responsibility is represented as a graph $G(V, E)$. G is directed bipartite graph, where V is a set of nodes and E is a set of directed links connecting nodes. The nodes represent the entrance and exit of a plan P, the evaluation of conditions, operations, cooperation among roles, and flow control. The links represent various kinds of relations among nodes, including dependencies between two nodes, coordination connections, sub-plan connections, goal connections, and connections among roles for the selection of role variables. A graphical language has been developed to visualize actions and their temporal orders in a responsibility (either an individual or team responsibility). Moreover, a representation in this language facilitate translating the process of a role-based plan into a graph of a team responsibility based on the algorithm shown in Fig. 3, and decomposing the graph of the team responsibility to graphs of individual responsibilities (described later in Sec. III.B).

---

generateTeamResponsibility (*P*)

1: create a team responsibility, *Resp*;
2: build the syntax tree of *P*, *tree*;
3: let *Proc* be the root of *tree*;
4: create an entrance node, *StartNode*, in *Resp*;
5: create an exit node, *ExitNode*, in *Resp*;
6: Translate(*Proc*, *StartNode*, *ExitNode*, *Resp*);

Fig. 3 Generating Team Responsibility
Graph for Role-Based Plan *P*

We note that redundant connector nodes may be used for easy translations the algorithms in Fig. 3. The redundant connector nodes can be reduced. The details, however, are beyond the scope of this paper. Fig. 4 shows team responsibility graph of the plan scanandcollect (shown in Fig. 2) after reducing redundant connector nodes.
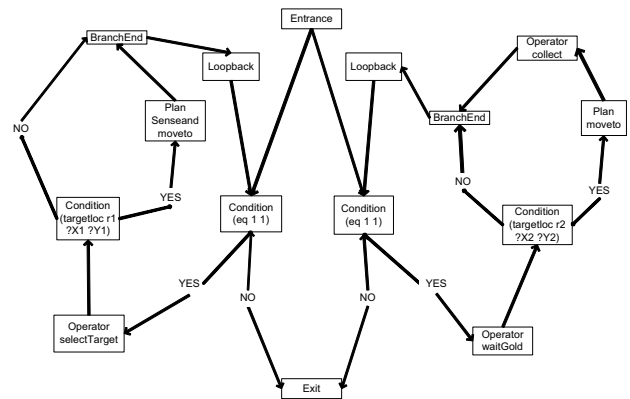

Fig. 4 A Reduced Team Responsibility Graph

B. Decomposing A Team Responsibility Graph to Individual Responsibility Graphs

We decompose a team responsibility graph into individual responsibility graphs by ownerizing the nodes and links in the team responsibility graph to the roles and role variables in the role-based plan. As a result, the individual responsibility graph of a role (or role variable) contains the nodes and links ownerized to the role (or role variable).

The decomposition needs to ensure two things: 1) every action in team responsibility is included into the responsibility of the role (or role variable) with which the action is associated; and 2) the temporal orders on the actions in team responsibility can be preserved so that the joint intention to the actions in the team responsibility can be realized by individual intentions to the actions in the individual responsibilities.

As described in previous Sec. III.A, when we translate a role-based plan into a team responsibility graph, we translate the actions in the process into corresponding nodes and ownerize the nodes to the roles or role variables with which the actions are associated. If an action is an invocation of an individual operator, the action is translated into an operator node and the operator node is ownerized to the role (or role variable) which invokes the operator. If an action is an invocation of a team operator, the actions are translated into individual operators corresponding to the team operator and one of the individual operators is ownerized to each involved role (or role variable). If an action is a complex action other than a team operator, such as a role selection, a sub-plan invocation, or a goal achievement, the action is translated to a set of nodes corresponding to the action and one of the nodes is ownerized to each involved role (or role variable).

Because a link connects two nodes and represents the relationship between two nodes, the owner of the link depends on what the owners of the two nodes are. If the owners of the two nodes are the same role or role variable, the owner of the link is consequently the same role or role variable. Otherwise, if the owners of the two nodes are different, the link implies communication between the owners of the two nodes and the source of the link is the initiator of the communication. It is thus

very natural to let the owner of the link be that of the source node.

---

decomposeTeamResponsibility (*Resp, Start, End*)

1: create array *visitedList*;
2: forwardOwnerize(*Resp, Start)*;
3: reset *visitedList* to empty;
4: backwardOwnerize(*Resp, End)*;

Fig. 5 Decomposing A Team Responsibility
To Individual Responsibility

Fig. 5 is the algorithm for decomposition. *Resp* is the team responsibility graph of *P* with entrance node, *Start*, and exit node, *End*. The algorithm visits the graph of team responsibility using Depth-First Search from two directions, following the directions of links and starting from the node *Start*, and following the reverse direction of links and starting from the node *End*. An array, *VisitedList*, is used to keep track of visited nodes.

V. ROLE-AGENT ASSIGNMENT

Our mechanism of task delegation is realized by delegating individual responsibilities to agents. Our mechanism of task delegation contains two levels of responsibility delegations. The first level is delegating the team responsibilities of all the roles in a role-based plan to the agents invoking the role-based plan before the agents start executing the role-based plan. The second level is dynamically delegating the responsibilities of role variables during the execution of the role-based plan. If a role is selected to fill a role variable, the responsibility of the role variable is dynamically included into the responsibility of the role. Given that a certain agent has intended to the responsibility of the role in the first level, the responsibility of the role variable is implicitly and dynamically delegated to the agent. Therefore, key issues in our mechanism of task delegation are the determining a team assignment for invoking a role-based plan and selecting a role for a role variable.

A. Role-Agent Assignment and Admissibility

We define a *role-agent assignment* to be a delegation of a role *r* to an agent *ag*, denoted by ***Assign(r, ag)***. Once a role *r* is delegated to an agent *ag*, agent *ag* commits to the responsibility of role *r*. Consequently, agent *ag* intends to perform the actions in the responsibility of role *r*, i.e. Intend(*ag, r, P*).

To actually execute the actions in the responsibility of role *r*, agent *ag* must be capable of the actions. Otherwise, agent *ag* is not "qualified" for role *r*, or *Assign*(*r, ag*) is not feasible. We use a notion of *admissibility* of role-agent assignment to characterize feasible role-agent assignments.

Definition 7. Assign (*r, ag*) is **admissible** iff

$$oprequirement(r) \subseteq capability(ag), \qquad (7)$$

where *oprequirement*(*r*) is the qualification of role *r*, and *capability*(*ag*) is the set of operators of which agent *ag* is capable.

One may argue that a role in a plan may not perform all operations in the qualification of its position. The reason why we require the agent capable of all operations in the qualification of the position is that, we want to separate role-agent assignment from the detail of the process of a plan. This is consistent with the way that people employ in reality. For example, to fill a job position, a manager makes offer decision based on its qualification and applicant's skills. The manager does not need to consider the details about what is going to be done by this position. For most tasks, the employee for this position only use part of skills required by the qualification of this position.

B. Team Assignment and Admissibility

Suppose that a team of agents *T* wants to invoke a role-based plan *P* with virtual team *VT*. We define a team assignment for *T* invoking the role-based plan *P* to be the aggregation of role-agent assignments of all roles in *VT* to the agents in *T*, denoted by *Assign(VT, T)*. It could happen that more than one role is delegated to a single agent. Once the roles in *VT* are delegated to the agents in *T*, the agents jointly commit to the team responsibility of *VT*. Consequently, the agents jointly intend to perform the actions in the team responsibilities of *VT*, i.e., JIntend*(T, VT*, Assign(*VT, T*), *P*). To perform the actions in the team responsibility of *VT*, every agent *ag* in *T* intends to perform the actions in the responsibility of role *r* delegated to the agent *ag* according to Assign*(VT, T)*, i.e., Intend*(ag, r, P)*.

Similarly, admissibility is required for team assignment.

Definition 8. Assign(VT, T) is admissible iff it satisfies the following conditions:
1. For every *Assign*(*r*, *ag*) in *Assign(VT, T)*, *Assign*(*r, ag*) is admissible;
2. Every constraint in the constraint set of plan *P* is satisfied under *Assign(VT, T)*;
3. For every sub-plan invocation, there is an admissible team assignment.

We note that, the above formalism is only to search for an admissible team assignment for *T* invoking *P*, and it occurs when the execution reach the point when *T* invokes *P*. The statements inside *P*, including condition evaluations and sub-plan invocations, have not been reached yet. Checking condition 3 is to ensure that with the found admissible team assignment for *T* invoking *P*, there will be admissible team assignments for sub-plan invocations of *P* during the future execution of *P* (the team assignment searching for sub-plan invocations occurs when the execution reach them).

One may question why not just decide which role is delegated to which agent during the execution of a role-based plan (i.e., at the first moment when a role invokes an action). The reason why we search for an admissible team assignment before actually executing a role-plan is that, we want to not only decide which role is delegated

to which agent, but also have agents form a joint intension to enforce the execution of the role-based plan to be a team effort; moreover, the joint intension ensures that every action in P will be executed in accordance with the temporal ordering on them unless the termination conditions of the role-based plan are satisfied.

C. CSP Algorithm of Searching for Admissible Team Assignments

Suppose a team of agents T wants to invoke a role-based plan, P. The virtual team VT of plan P is a set of roles, and the constraints of plan P is C_O. In the process of P, there is a set of sub-plan invocations Inv. The problem of searching for an admissible team assignment for T invoking P can be formalized as a constraint satisfaction problem (CSP) [18]:

- The set of variables of the CSP is VT of plan P.
- The nonempty domain of the CSP is T. The possible values for each variable (role) are the agents in T.
- The set of constraints of the CSP includes three types of constraints: 1) qualification for delegating roles to agents, i.e. for any role r_i in VT, $oprequirement(r_i) \subseteq Capability(ag)$, where ag is the agent assigned to r_i; 2) the constraints in C_O; and 3) an admissible team assignment for each sub-plan invocation in Inv.

Various algorithms, including backtracking search, local search and distributed search, have been developed to resolve CSP. Backtracking search functions as a depth-first search to choose values for a variable at a time and backtrack when a variable has no legal value left. An algorithm of backtracking search has been developed to search for an admissible team assignment for agents invoking a role-based plan.

D. Role Selection

A role selection is the select of a role r to fill a role variable $?r$, denoted by **Select**$(?r, r)$. By Definition 4, role variable declaration includes the selection scope RS and the selection constraints SC. On the other hand, the capability requirement of every role in RS must include all actions associated with the $?r$. However, the capability requirement may not include all actions associated with $?r$. Correspondently, if role r is selected to fill $?r$, the agent to whom r delegated may not be capable to do some action(s) associated with $?r$. To prevent this oddity, a plan can be statically checked to make sure all role variables in the plan are well defined. Also, the agents to whom the selected roles are delegated must be able to execute the actions associated with the role variables. For this reason, we only consider the selection constraints but not the capability issue when resolving role selections. Fig. 6 is the algorithm for role selection.

roleSelection (*?r, RS, SC*)

```
1: loop{
2:    if ∀ roles ∈ RS have been tested
3:         return failure;
4:    randomly select an untested role r from RS;
5:    let ag be the agent to which role r is delegated;
6:    if ∀ constraints ∈ SC are satisfied with ?r substituted by ag
7:         return r;
8:    mark r as a tested role;}
```

Fig. 6 Selecting A Role for A Role Variable

VI. EXPERIMENT

We choose an extension of the wumpus world described in [24], a multi-agent wumpus world, as the domain for our experiments. Agents have different capabilities, such as sense, kill wumpuses, and collect gold. There are three roles including r_1, sniffer, r_2, fighter, and r_3, carrier. Table 1 lists operators and their durations. A sniffer can perform sense, a fighter can perform shoot, and a carrier can perform collect. They all can perform move. The goal of the team is to kill wumpuses and collect gold. Before an agent moves to a square, a sniffer needs to sense that square and to decide if there is a wumpus in the square, and a fighter needs to shoot toward that square if there is. Before a carrier collects a piece of gold, a sniffer needs to sense a square and decide if there is a piece of gold in the square.

Table 1 The Durations of Operators

Operator	Duration (ms)
Move	200
sense	100
shoot	100
collect	100

We collect three data: 1) the number of wumpuses killed, 2) the amount of gold collected, and 3) time performance by the absolute execution time of a plan.

We report three experiments. The first explores plan reusability of RoB-MALLET. The second shows its flexibility in supporting simultaneity of invocation. The third evaluates its efficiency in supporting teamwork.

A. Evaluating plan reusability

Experiment 1 evaluates that RoB-MALLET is flexible enough to allow reuse of role-based plans by running a role-based plan with different formation of agents.

Table 2 defines five teams of agents that could invoke a role-based plan in the wumpus world. The randomly generated world is 10×10 squares and contains 20 wumpuses and 20 pieces of gold. Agents initially have no knowledge of the squares.

Table 2 Teams with Different Formations Executing the Plan

Team	Agent	Capability	Role(s)
T1	ag_1	move, sense, shoot, collect	r_1, r_2, r_3
T2	ag_2	move, sense	r_1
	ag_3	move, shoot, collect	r_2, r_3
T3	ag_4	move, shoot	r_2
	ag_5	move, sense, collect	r_1, r_3
T4	ag_6	move, collect	r_3
	ag_7	move, sense, shoot	r_1, r_2
T5	ag_8	move, sense	r_1
	ag_9	move, shoot	r_2
	ag_{10}	move, collect	r_3

Table 3 shows that all five teams have successfully completed the teamwork. Even though they have different time performances of accomplishing the teamwork, our model is flexible for different formations of teams to invoke the plan. We note that, each of teams T2, T3, and T4 consists of two agents, however, the two agents in different teams play different roles. The workload of the agent taking r_1 (sniffer) directly decides team performance. The faster that agent finds wumpuses and gold, the faster the team finishes the teamwork. In T2, the agent taking r_1 did not take another role. In T3 and T4, the agents taking r_1 also took another role (r_2 in T3 and r_3 in T4). Consequently, the team performance of T2 is better than T3 and T4. For the similar reasons, T3 killed wumpuses slower than T4 because the agent taking r_2 did not take other role in T3 while the agent taking r_2 took r_1; and T3 collected gold faster than T4 because the agent taking r_3 did not take other role in T3 while the agent taking r_3 took r_1.

Table 3 The Duration of Different Teams Executing the Plan

Team	Time to Kill All Wumpuses	Time to Collect All Golds
T1	387	429
T2	151.5	183
T3	285	241
T4	253	296.5
T5	139.5	163.5

B. Evaluating simultaneity of invocations

Simultaneity of invocation means that an agent may be involved in multiple plan invocations (at the top

level) dynamically and simultaneously. In experiment 2, we use a world of 20×20 squares which contain 40 wumpuses and 60 pieces of gold. Rather than randomly generate the world, it has been specifically constructed to contain more difficult scenarios therefore need close cooperative behaviors. These are 25 pieces of unreachable gold in the wumpus world. For example, a piece of gold in the corner is surrounded by two wumpuses. Such gold is unreachable unless a path is opened by killing one of the surrounding wumpus. Again, agents initially have no knowledge of the squares.

We define two teams pursuing different top level goals. Team T1 consists of two agents, ag_1 and ag_2, who scan a wumpus world and collect the gold found respectively. Team T2 only consists of agent ag_3 who is able to kill wumpuses but has a separate goal that is irrelevant to the main point of this example. We represent it simply by random motion through the world. Three plans, randommove, scancollect and kill, are separate team processes for T1 and T2. Two teams start plans scancollect and randommove simultaneously (at the top level). With proactive communication from ag_1 [4, 18], ag_3 dynamically finds plan kill for a goal and builds and executes a script to start it many times to help ag_2 while ag_3 is still involved in plan randommove. The team collects 54 pieces of gold finally. Fig. 7 shows the distribution of the number of pieces of gold collected over the execution time.

We note that T2 was considered a separate team from T1, rather than as a second sub-team of a larger team, to show the more general situation with separate teams. The entire scheme works within sub-teams of a team as well.
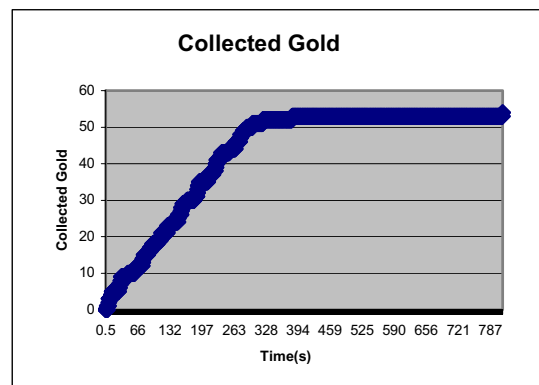

Fig. 7 The Distribution of Gold Collected Over the Execution Time

C. Evaluating Efficiency in Supporting Teamwork

Experiment 3 evaluates the efficiency of RoB-MALLET in supporting teamwork by comparing it with MALLET. We define a team consists of three agents which scan a wumpus world, kill wumpuses found, and collect gold found. The performance of the team is compared with that of a logically equivalent team supported by MALLET. We have run a set of experiments in four configurations: 1) distributed mode

and operators with durations, 2) centralized mode and operators with durations, 3) distributed mode and operators without durations, and 4) centralized mode and operators without durations.

According to Fig. 8, we found that, to kill various numbers of wumpuses, the time cost in RoB-MALLET is consistently much less than that in MALLET. The time cost is somewhat proportional to the number of wumpuses killed in both, however, the slope for RoB-MALLET is much smaller than that for MALLET.
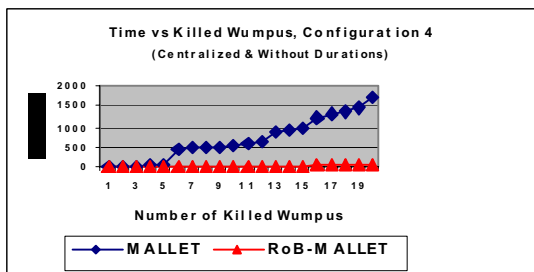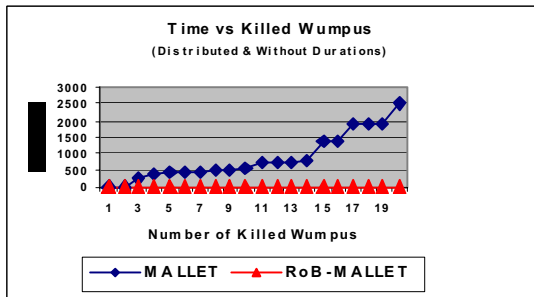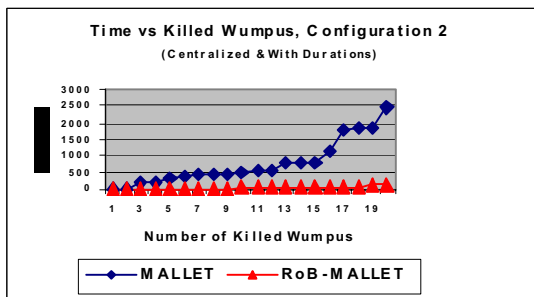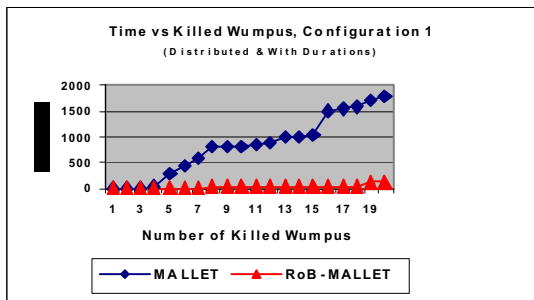
Time vs Killed Wumpus, Configuration 1
(Distributed & With Durations)

Time vs Killed Wumpus, Configuration 2
(Centralized & With Durations)

Time vs Killed Wumpus
(Distributed & Without Durations)

Time vs Killed Wumpus, Configuration 4
(Centralized & Without Durations)

Fig. 8 RoB-MALLET vs. MALLET with Configurations 1-4

VII. RELATED WORK

A. Phychological Interpretation of Roles

The concept of role is generally but loosely understood as: a character assigned or assumed, a socially expected behavior pattern usually determined by an individual's status in a particular society, a part played by an actor or singer, a function or part performed especially in a particular operation, or an identifier attached to an index term to show functional relationships between terms (from Webster's Dictionary). This understanding implies an attractive property of roles: a role stands for a functional abstraction, independent of the specific person who plays it.

In their role theory, Biddle and Thomas [5] discussed the properties of role and the relationships among roles from behavioral, social, cognitive, normative and legislative perspective. They also gave classificatory concepts of role, which deal with a limited set of phenomenal referents, by using partitioning conceptual operations and a criterion for subclass operation. The referents applied to partitioning conceptual operation include behaviors, persons, and persons and their behaviors. Their classificatory concepts of roles capture various behavioral aspects of role by the referent of behaviors and the referent of persons and their behaviors, while the referent of persons is not related to the behaviors aspects of role.

We ground our concepts related to role on Biddle and Thomas' role theory. Our interest is to define a formalism of role amenable to computation, particularly specifying team activities at abstract level independent of the specific agents that will execute them, decomposing team activities into individual tasks, and delegating the individual tasks to actual agents.

B. Roles Revlevent to Organizing Team Activities

Ferber et al. [13] proposed a generic meta-model of organizations based on the concepts of roles, groups and organizations. The meta-model uses organizational reflection to describe the relationships between interaction and organization. Moreover, a formal specification was proposed to define role behavior and interaction requirements. In some teamwork architectures described earlier in Sec. I, including STEAM [28], RETSINA [27] and JACK [6], team plans specify team activity in terms of roles. In these teamwork architectures, when a team of agents invokes a team plan to achieve a certain goal, the agents fill the roles in the team plan according to their skills and the capability requirements of the roles. In STEAM, agents can monitor and repair failures through team operators and roles. In RETSINA, agents can dynamically drop roles and other agents can reselect the dropped roles as necessary.

While the BDI (Belief, Desire and Intention) model [23] emphasizes practical reasoning based on beliefs, desires and intentions, Chainbi [9] proposed a Belief-Goal-Role theory (BGR) to stress the concept of organization and thereby cooperation. Chainbi divided actions into four categories: physical actions (i.e., interactions between agents and the environment), communicative actions (i.e., interactions between agents), private actions (i.e., internal functions of agents), and decision actions (i.e., actions that generate

communicative, physical and private actions). A role is defined as a sequence of actions. Chainbi gave a set of formalization about the condition under which an action of a role can be taken, and the belief changes of agents according to the category of the action. However, many issues remain unsolved, for example, how to specify a role, how to delegate roles to agents, how the delegation of a role to agent impacts on the agent's mental states. So his BGR is quite preliminary and no multi-agent architecture based on BGR has been seen.

Researchers in Australian Artificial Intelligence Institute, including Rao, Sonenberg, Tidhar, Georgeff, and et al, developed a team-oriented programming language [30, 25] (with different terms, including Team-Oriented Programming, Planned Team Activity and Social Plans), which semantics is formally defined on shared mental states, mutual beliefs, joint goals and joint intentions. For simplicity, we refer their team-oriented programming language as TOP. TOP is an implementation of BDI. In TOP, team activities are organized, in terms of roles, by a plan description language based on plan graphs. A plan graph contains START, AND, OR and END nodes representing synchronization. Social structures in TOP specify the levels of authority and responsibility that each team member possesses. To invoke a team plan, agents, guided by skills and capabilities, fill roles in the social structure of the plan. Our concepts of roles are distinguished from the roles in TOP and other teamwork architectures discussed later in this section in these aspects: 1) while their delegation of roles to agents is made based the agents' capabilities, our delegation of roles to agents is made based on the social norms (i.e., constraints) on delegating the roles to agents, as well as the agents' capabilities; 2) our roles and role variables distinguish static (by roles) and dynamic (by role variables) action associations; and 3) when delegating roles and role variables in a plan to agents, we have the agents form a mental state (joint intention) to enforce the execution of the plan as a team effort, particularly the sub-actions in the plan will be executed coherently. The feature of (2) and (3) are not provided by roles in existing teamwork architectures.

C. Roles Revlevent to Task Decomposition and Delegation

Stone and Veloso [26] used roles to decompose tasks for Robocup soccer team. In [26], a role consists of the specification of an agent's internal behaviors (i.e., update the agent's internal state based on its current internal state, the world state, and the agreement among team) and external behaviors (i.e., the agent's interaction with the world). Roles are defined in terms of positions in the field of soccer game, such as midfielder. Team tasks then are decomposed to sub-tasks for the roles. Rather than being fixed to certain positions, robots coordinate and dynamically decide the team formation by role assignments.

Vail and Veloso [31] continue the efforts on dynamic role assignment in robot soccer games. They develop an approach for sharing sensed information and effective coordination through the introduction of shared potential fields. The potential fields were based on the roles of the other robots on the team. The robots are assigned roles by following the gradient to a minimum of the potential field.

Castelfranchi and Falcone [8] defined delegation and adoption based on the relationships between agents: "in delegation an agent A needs or likes an action of another agent B and includes it in its own plan"; and "in adoption an agent B has a goal since and until it is the goal of another agent A". The action in delegation and adoption is viewed as a task. Castelfranchi and Falcone viewed a role as a social relation regarding to a task. If agent A delegates task *t* to agent B (i.e., Delegate(A, B, t)), A is a role client and B is a role contractor. Then, a task delegation is a creation of a role. Further, they extended a role to be an abstract agent with goals and plans for those goals. The tasks in the plans are delegated through the role delegation.

D. Roles Relevent to Coordination Among Individuals

Barbuceanu et al. [3] have developed a coordination language, called COOL, which specifies coordination among individuals in terms of roles. At the organization level, roles specify social constraints on agent behaviors, in particular, the obligation of an agent filling a role to another agent with respect to performing actions to achieve a goal. At the individual agent level, obligations and interdictions were viewed as mental states much like beliefs, desires, and intentions. Being more specific, an obligation is defined as a generic rule, which specifies the condition under the obligation becomes active (:condition clause), the role being obligated (:obligated clause), the role that obligates (:authority clause), and the goal of the obligation (:goal clause). The control architecture of COOL selects obligations and goals, and further coordination plans according to beliefs and generic obligation rules.

Ashforth [2] models an individual's role transactions in an organization. Such transactions include role entry, role exit, role learning, role innovations, role occupation and the attributes of role transitions. This process is similar to our dynamic role-agent assignment mechanism in terms of an individual agent can fit in different roles along with their teamwork processes.

Neale and Griffin [20] model individual role perceptions in the perspective of sitting the individual in an organization. The model is composed of three components: system requirements, which represent the demands made of the individual by the employing organization; role schemas, defined as individual perceptions of how the role is typically enacted within the broader society; and a role-specific self-concept. This model helps individuals adjust their perception, behavior, and abilities. This adjustment helps people change their working positions smoothly and adapt the new roles quickly.

E. Dynamic Role-Agent Assignment

Zhu and Zhou [35] formalize the problem of role-agent assignment as a matching between an agent's availability and capability and a number of ranges of a

role. For each role, they classify the possible agents who can play the role into two types – current agent and potential agent. The current agent means the agent who is currently playing that role, and the potential agent possesses the ability to play that role but is not currently playing it. All roles have an upper bound and a low bound to the number of agents that can be accommodated by this role. A team is workable if all roles in it have been filled by an appropriate number of agents.

Odell et al. [21] discuss a similar aspect of dynamic role assignment in modeling methodologies. In their work, they emphasize that dynamic role assignment reflects dynamic classification and activation. Classification affects the instantiation of agents and activation affects the current behaviors of agents. The concept of activation is similar to the concepts of converting potential roles into current ones in Zhu and Zhou's paper.

Dastani et al. [11] formalize the role assignment in a formal language 3APL for multi-agent systems. In their method, agents temporarily take up roles. This creates the need to define precisely what it means that an agent "takes up" a role and "enacts" it. They define possible relations between roles and agents and discuss architectural and functional changes that an agent must undergo when it enters an open agent system by playing different roles.

VIII. CONCLUSION

We have defined role and role variable and accommodated them to complex actions. Based on these conceptual notations, the knowledge of team processes can be specified without having to specific agents and agent variables. This will allow joint mental states to be formed and the teamwork knowledge to be reused by different teams of agents. Based on roles and role variables, we have developed mechanisms of task decomposition and task delegation, by which the knowledge of a team process is decomposed into the knowledge of a team process for individuals and then delegate it to agents. We have developed an efficient representation of joint mental states by which agents only maintain individual processes complementary with others' individual process and a low level of overlapping for team organizations.

In the future, RoB-MALLET can be extended to the area of team planning. In this paper, we have explained how role-based plans, together with operators, can be used in planning algorithms. However, we just presented an algorithm to search for a role-based plan to achieve a goal without considering operators or decomposing the goal to a sequence of states. In the future research, we can extend this algorithm to include operators and goal decomposition as forward/backward planning algorithms do. Moreover, planning with existing team plans (i.e., existing role-based plans in our case) is essentially related to new research issue emerging with multi-agents systems: planning for a team of agents, instead just for a single agent as traditional planning

algorithms do. The search space in traditional planning algorithms for a single agent is just one dimension: the set of operators. However, the search space in planning algorithms that use role-based plans becomes two dimensions: 1) the set of operators and role-based plans, and 2) the set of agents that execute operators/plans. Consequently, the total search space becomes the multiplicity of these two dimensions and thus its complexity increases dramatically.

Other future work is to extend RoB-MALLET to express the knowledge related to time, including when actions are taken and how long they last. For example, duration (or a range of duration) can be included in the specification of an operator to specify how long the execution of the operator takes. The hardest part of this issue is we need investigate all kinds of process knowledge related to time and then develop a set of constructs which allows us to express all the kinds of process knowledge related to time. As long as the syntax and semantics of the constructs are decided, corresponding extension of RoB-MALLET can be made to represent knowledge about time in individual process, such as by using Time Petri Nets and Timed Petri Nets [22].

REFERENCES

[1] de Antonio, A., Ramírez, J., Imbert, R. and Méndez, G., Intelligent Virtual Environments for Training: An Agent-Based Approach, Lecture Notes in Computer Science, 3690: 82-91, 2005.

[2] Ashforth, B. E., Role Transitions in Organizational Life: An Identity-based Perspective. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2001.

[3] Barbuceanu, M., The Role of Obligations in Multi-Agent Coordination, Applied Artificial Intelligence 13: 11-38, 1999.

[4] Becht, M.; Muscholl, M; Levi, P, Transformable Multi-Agent Systems: A Specification Language for Cooperation Processes, in Proceedings of the World Automation Congress, ISOMA'98.

[5] Biddle, B. J. and Thomas, E. J. (editors). Role theory: Concepts and research. John Wiley & Sons, 1966.

[6] Busetta, P., Ronnquist, R., Hodgson, A., Lucas, A., JACK intelligent agents - components for intelligent agents in Java, Technical Report TR9901, Agent Oriented Software, Australia, 1999.

[7] Cao, S., Volz, R. A., Ioerger, T. R., Miller, M. S., On Proactive Helping Behaviors in Teamwork. IC-AI, pp. 974-980, 2005.

[8] Castelfranchi, C. and Falcone, R., From Task Delegation to Role Delegation, in Lecture Notes in Artificial Intelligence, M. Lenzerini (Ed.), Springer-Verlag, Berlin, Germany, 1997.

[9] Chainbi, W., Ben-Hamadou, A. and Jmaiel, M., A Belief-Goal-Role Theory for Multi-Agent Systems, International

Journal of Pattern Recognition and Artificial Intelligence, 15(3): 435-450, 2001.

[10] Cohen, P. R. and Levesque, H. J., Teamwork. Nous, Special Issue on Cognitive Science and AI, 25(4):487-512, 1991.

[11] Dastani M., Dignum, V. and Dignum, F., Role Assignment in Open Agent Societies, in Proceedings of 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems, Melbourne, Australia, pp. 489-496, 2003.

[12] Fan, X., Yen, J., Miller, M. S., Ioerger, T. R. and Volz. R. A., MALLET - An Multi-agent Logic Language for Encoding Teamwork, IEEE Transaction on Knowledge and Data Engineering, 18 (1): 123-138, 2005.

[13] Ferber, J. and Gutknecht, O., A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems, in Proceedings of International Conference on Multi-Agent Systems (ICMAS 98), Paris, France, 1998.

[14] Grosz, B. and Kraus, S., Collaborative Plans for Complex Group Actions, Artificial Intelligence, 86 (2): 269-357, 1996.

[15] Jennings, N. R., On Being Responsible, in Decentralized Artificial Intelligence, E. Werner and Y. Demazeau (Eds.), North Holland Publishers, Amsterdam, Holland, pp. 93-102, 1992.

[16] Kishore, R., Zhang, H. and Ramesh, R., Enterprise integration using the agent paradigm: foundations of multi-agent-based integrative business information systems, Decision Support Systems, 42(1): 48-78, 2004.

[17] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., and Osawa, E., Robocup: The Robot World Cup Initiative, in Proceedings of IJCAI-95 Workshop on Entrainment and AI/AI-Life, 1995.

[18] Kumar, V., Algorithms for Constraint-Satisfaction Problems: A Survey, pp. 32-44, AI Magazine, Spring 1992.

[19] Lewis, W. and Rickel, J. W., Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. International Journal of Artificial Intelligence in Education 11: 47-78, 2000.

[20] Neals, M. and Griffin, M.A., A Model of Self-Held Work Roles and Role Transitions, Human Performance, 19(1): 23-41, 2006.

[21] Odell, J.J., Van Dyke Parunak, H., Brueckner, S. and Sauter, J., Changing Roles: Dynamic Role Assignment, Journal of Object Technology, 2(5): 77- 86, 2003.

[22] Ramchandani, C., Analysis of Asynchronous Concurrent Systems by Timed Petri Nets, Technical Report TR-120, Massachusetts Institute of Technology, Cambridge, MA, 1974.

[23] Rao, A. and Georgeff, M., BDI Agents: From Theory to Practice, Proceedings of the First International Conference on Multi-Agent Systems, 1995.

[24] Russell, S. and Norvig, P., Artificial Intelligence - A Modern Approach, Prentice Hall, Upper Saddle River, NJ, 2002.

[25] Scerri, P., Pynadath, D. V., Schurr, N. and Farinelli, A., Team Oriented Programming and Proxy Agents: the Next Generation, in Proc. of the 1st Inter. Workshop on Programming MAS at AAMAS'03.

[26] Stone, P, and Veloso, M., Task Decomposition and Dynamic Role Assignment for Real-Time Strategic Teamwork, Artificial Intelligence, 100 (2): 2441-273, 1999.

[27] Sycara, K., Paolucci, M., Van Velsen, M., Giampapa, J., The RETSINA MAS Infrastructure, Autonomous Agents and Multi-Agent Systems, 2003.

[28] Tambe, M., Towards Flexible Teamwork, Journal of artificial intelligence research, 7: 83-124, 1997.

[29] Tambe, M. Johnson, W. L., Jones, R., Koss, F., Laird, J. E., Rosenbloom, P. S. and Schwamb, K. Intelligent Agents for Interative Simulation Environments, AI Magazine, 1995.

[30] Tidhar, G., Team Oriented Programming: Preliminary Report, in Technical Report 41, AAII, Australia, 1993.

[31] Vail, D., Veloso, M., Multi-Robot Dynamic Role Assignment and Coordination through Shared Potential Fields, in A. Schultz, L. Parkera and F. Schneider (eds.), Multi-Robot Systems, Kluwer, pp. 87-98, 2003.Vazquez-Salceda, J., Dignum, V. and Dignum, F., Organizing Multiagent Systems, AAMAS, 11: 307-360, 2005.

[32] Vazquez-Salceda, J., Dignum, V. and Dignum, F., Organizing Multiagent Systems, AAMAS, 11: 307-360, 2005.

[33] Zhang, Y. and Yin, J, A Role-Based Modeling for Agent Teams, in Proceedings of the IEEE Workshop on Distributed Intelligent Systems (DIS'06), pp. 348-353, 2006.

[34] Zhang, Y., Ioerger, T. R. and Volz, R. A., Decision-Theoretic Proactive Communication in Multi-Agent Teamwork, in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'05), pp. 3903-3908, 2005.

[35] Zhu, H. and Zhou, M.C., Role-Based Collaborations and their Kernel Mechanisms, IEEE Trans. on Systems, Man and Cybernetics, Part C, 36 (4): 578-589, 2006.

Yu Zhang is an assistant professor in Computer Science at Trinity University. She received her Ph.D in Computer Science from Texas A&M University in 2005. Dr. Zhang's research falls within AI and Multi-Agent Systems. She is in the editorial board and program committee for over 30 journals, conferences and technical groups in this area and has regularly reviewed papers and proposals for journals, conferences and federal agencies. She is a member of IEEE, ACM and AAAI.