

Temporal Aspects of Dynamic Role Assignment

James J. Odell¹, H. Van Dyke Parunak², Sven Brueckner², John Sauter²

¹ James Odell Associates, 3646 West Huron River Drive, Ann Arbor
MI 48103-9489 USA
email@jamesodell.com
http://www.jamesodell.com

² Altanum Institute, 3520 Green Court, Suite 300, Ann Arbor,
MI 48105-1579 USA
{van.parunak, sven.brueckner, john.sauter }@altarum.org
http://www.erim.org/{~vparunak, sbrueckner}

Abstract. A helpful abstraction of a group of agents is a set of interacting roles, or sets of normative behaviors, that the agents can assume. An important characteristic of real-world agent systems is that the roles played by an agent may change over time. These changes can be of several different kinds. We describe an illustrative application where such role changes are important, analyze and classify the various kinds of role changes over time that may occur, and show how this analysis is useful in developing a more formal description of the application.

1 Introduction

The notion of *role* is fundamentally a thespian concept. As humans, we find the perspective and language of the theater a useful analogy for describing and understanding many of the same complex aspects of individual behavior [Odell, 2003]. Since we commonly employ the notion of role in real life for conceptualizing human behavior, it may also serve as a useful device for other kinds of individuals in a MAS—be they life forms, active software constructs, or hardware devices.

In an agent-based system, we define *role* as a class that defines a normative behavioral repertoire of an agent. Roles provide both the building blocks for agent social systems and the requirements by which agents interact.¹ Each agent is linked to other agents by the roles it plays by virtue of the system’s functional requirements—which are based on the expectations that the system has of the agent. The static semantics of

¹ Several possible implementation techniques exist for implementing programs that support social entities possessing multiple and changeable class-based roles, including class inheritance and aggregation. In this paper, we will not discuss program-level implementation options for treating role as a class. Instead, our emphasis will be at the analysis-level (i.e., a conceptual and implementation-independent approach).

roles, role formation and configuration, and the dynamic interactions among roles has been examined closely in recent years [Ferber et al. 2003][Castelfranchi, 2000][Destani, 2003][Odell, 2001][Parunak, 2001]. However, little work has been done on formalizing the temporal aspects of dynamic role assignment. As a result, role modelers refer only informally to actions such as “taking on a role,” “playing a role,” “changing roles,” and “leaving roles.” However, such terms can be interpreted ambiguously.

For example, consider the scenarios that follow. These six scenarios are considered to be “changes,” yet semantically they all have a different meaning.

1. *Classify* – Add the role of manager to the role of Employee as the result of a promotion.
2. *Declassify* - Remove the role of Manager as the result of a demotion.
3. **Reclassify* - Change from the role of Employee to the role of Unemployed Person.
4. *Activate* - Take up the behaviors of the Manager role as part of the day-to-day business activity.
5. *Suspend* – Stop any Manager behavior and take on just those of the Employee role as part of the day-to-day business activity.
6. **Shift* - Change from an Employee role to a Pet Owner role, where neither is played at the same time as the other. This is a combination of activating one role while suspending another.

(*These are composite roles, not primitives.)

These scenarios will be discussed in more depth in the following sections. Prior to that discussion, however, some fundamental notions need to be identified and defined.

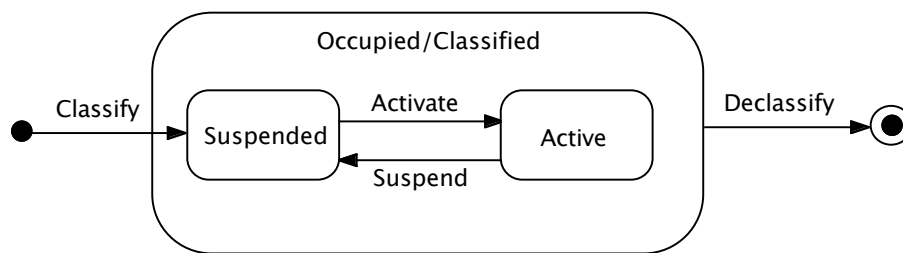


Figure 1 - Statechart depicting some of the permitted states and transitions of an agent in a role.

The statechart in Fig. 1 depicts four of the operations in the scenarios above. Here, an agent comes to *occupy* a role when it is *classified* and ceases to occupy the role when it is *declassified*. Furthermore, while an agent occupies a role, it can be either *active* or *suspended* in playing that role. Transitioning between those two states can be achieved via the *suspend* and *activate* operations. Definitions for these states and transition operations are as follows:

States

Occupied – The state of an entity that is an instance of a particular role.

Suspended - The state of an entity occupying a particular role, when the role has no processes executing.

Active - The state of an entity occupying a particular role that is executing some of all of its processes.

Operations

Classify – An operation that results in an agent being an instance of, or occupying, a particular role.

Declassify – An operation that results in an agent no longer being an instance of, or occupying, a particular role.

Reclassify – An operation that results in an agent being both declassified as one particular role and classified as another.

Activate – An operation that results in an agent entity that is active.

Suspend – An operation that results in an agent entity that is suspended.

Shift – An operation that results in an agent entity becoming both suspended within one occupied role and active in a different occupied role.

These “role-changing” operations can best be discussed by partitioning into categories of dynamics: *dynamic classification* and *dynamic activation*. The sections that follow will describe these notions in more detail. AUML notation will also be proposed.

For concreteness, we develop these ideas in the context of a specific multi-agent application, a swarm of unpiloted air vehicles (UAV’s) that must coordinate their actions in various ways to accomplish a collective sensing task. Section 2 describes this application informally. Sections 3 and 4 develop and refine two aspects of dynamic roles: dynamic classification and dynamic activation, and suggest AUML representations for them. Section 5 applies these refinements to the UAV application, and Section 6 summarizes our contribution.

2 An Example Application

We demonstrate the temporal role dynamics in MAS in an example of a swarm of small robotic air vehicles (UAV) that perform a sensing mission in an urban environment. The mission proceeds through the *deployment phase*, in which the individual units are initialized and released, the *target approach phase*, where the swarm moves through the urban environment and establishes a line-of-sight communications network that links the base with the target area, the *objective completion phase*, where

the swarm performs its sensing task and communicates the resulting data through the dynamic network to the base, and the *recovery phase*, where the swarm returns to the base or dissolves depending on the type of UAV (disposable or not). We chose the target approach phase for our detailed discussion of dynamic role assignment.

The mission as outlined in the previous paragraph cannot be performed by a single UAV (agent), because of the restricted communications and sensing capabilities of the units. The restriction to line-of-sight communications in the urban environment requires that the swarm dynamically forms a communications network made up of strategically placed perching or hovering units that route data to and from the deployment base. These stations in the network are assigned dynamically as the swarm proceeds towards its target while avoiding obstacles and threats. Thus, one role an agent may assume during the mission is that of a *Communications Node*. All UAVs in the swarm are physically capable of performing this function.

Those units that do not take on the role of a Communications Node eventually arrive at the target area, ready to perform the mission objective. At this point, the swarm enters the next phase of the mission as some units take on the *Sensing Node* role and coordinate with other Sensing Nodes to perform the sensing task specified in the mission. Only a subset of all the UAVs in the swarm have the more expensive sensor packages that are capable of performing this task. Data acquired by the Sensing Nodes is sent to the base through the network formed by the Communications Nodes.

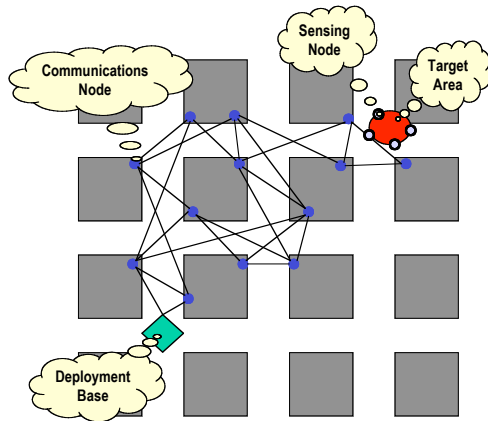


Figure 2 - A Mission Swarm Dynamically Assigning Communications and Sensing Node Roles.

The mission swarm operates in a hostile environment and individual units are prone to failure. Thus the communications network as well as the objective execution team has to cope with continuous attrition. An individual unit is aware of the current state of the communications/sensing team in its local environment and if it determines that attrition in its neighborhood threatens the performance of the mission, either by disrupting the information flow between the base and the target area or by removing required sensing capabilities, it seeks to recruit “uncommitted” units to fill the gap. This recruitment attraction draws units of the initially released swarm as well as additional units from the deployment base. The self-healing mechanism requires that agents are not only aware of the presence of other units in their neighborhood, but also that they know, which role they currently perform and which roles they are physically capable of performing.

3 Dynamic Classification

Dynamic classification refers to the ability to change the classification of an entity. While dynamic classification applies to both object classes and agent classes [Odell, 1998], in this paper we will discuss it in terms of agent roles. For example in Fig. 3(a), the “Alice” agent changes from being an instance of the roles Employee, Manager, and Salesperson to being an instance of the role Unemployed Person. In other words, with dynamic classification, an agent can be an instance of different roles from moment to moment.

Consistent with [Ferber et al. 2003], we insist that each agent have at least one role at all times. Dynamic classification deals with adding additional roles or removing roles beyond the minimum of one. This requirement is analogous with the notion that every human must play the “person” role, whatever other roles they may have. In the case of humans, this minimal role persists throughout the agent’s life. It is conceivable that an artificial agent might begin with the minimal role A, add role B, then remove role A, leaving it with the minimal role B. Whether such a fundamental redefinition of the agent is possible will depend on such features as physical equipment associated with the agent and the nature of the platforms on which the agent can run.

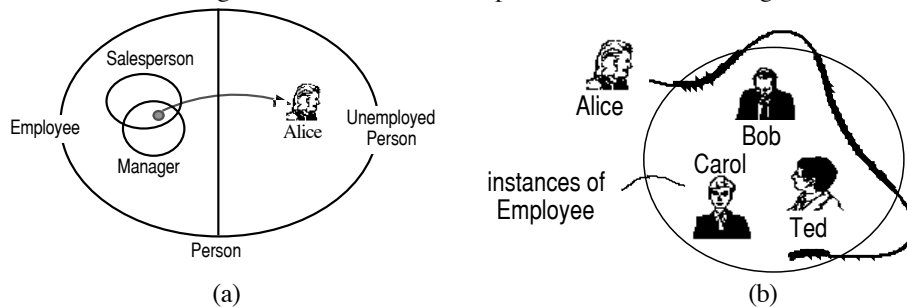


Figure 3 - (a) Dynamic classification refers to the ability to change the classification of an agent’s role. (b) The agent “Alice” moving in and out of being classified as an Employee over time.

To become an instance of a given role, the agent is *classified* as an instance of, or *occupies*, that role. Once classified, then, the agent occupies the new role and possesses all of its features. In the opposite process, if an agent is *declassified*, it is removed as an instance of a particular role—and no longer occupies the role nor possesses features unique to that role. Figure 3(b) portrays the “Alice” agent being classified and declassified in terms of the role Employee. At some point in her life, Alice is first classified as an Employee. Later, through some process, Alice is declassified as an Employee: she becomes unemployed. At another point, Alice may become re-employed, followed again by a period of unemployment. This behavior may continue until retirement is reached or the process of death takes place. And where both operations are used at the same time, an agent is said to be *reclassified* when it is both declassified in one role and classified as another.

Based on the descriptions above, we can now discuss “change” scenarios 1, 2, and 3 without ambiguity:

1. **Classify - Add the role of manager to the role of Employee as the result of a promotion.** In this situation, the person still remains an instance of Employee role. However, the person becomes a new instance of the Manager role. In other words, the person remains classified as an Employee but is now—in addition—classified as a Manager. This is called *classification*.
2. **Declassify - Remove the role of Manager as the result of a demotion.** Here, the person still remains an instance of Employee role, but is no longer an instance of the Manager role. *Declassification* has the opposite effect of classification in scenario 1, above.
3. **Reclassify - Change from the role of Employee to the role of Unemployed Person.** In this scenario, the person ceases to be an instance of the Employee role and becomes an instance of the Unemployed Person role. In other words, the person was *declassified* as an Employee, while simultaneously being *classified* as an Unemployed Person. This is called *reclassification*.

The table below summarizes the role assignments involving dynamic classification: classification, declassification, and reclassification operations.²

Operation	Pre-state	Post-state
Classify	A and not B	A and B
Declassify	A and B	A and not B
Reclassify	A	B

Both UML and AUML notations would express these three operators as illustrated in Fig. 4. Figure 4(a) indicates agent-1’s classification as role-*n*; in Fig. 4(b), agent-1 is declassified as role-*m*; in Fig. 4(c), agent-1 is being reclassified from role-*m* to role-*n*.

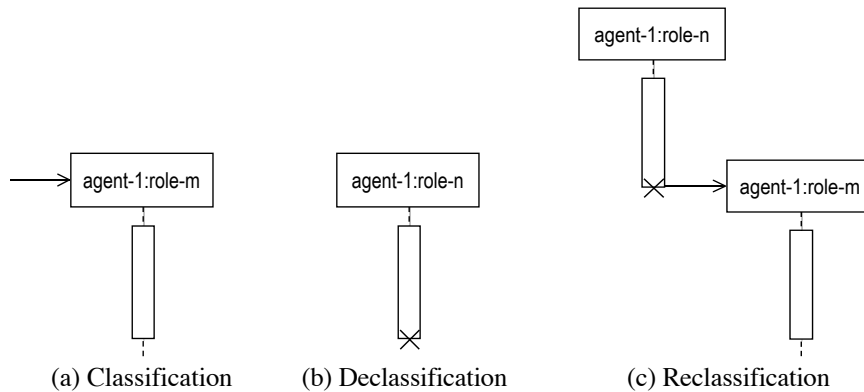


Figure 4 – AUML notation for classification, declassification, and reclassification.

² For completeness, arguably two more operators could be added: *create* and *terminate*. The create operator classifies entities that did not previously exist. Here, the prestate for an entity would be null, and the poststate a particular role. Termination is the opposite, where the prestate would be for an entity in some role and the poststate would be null.

4 Dynamic Activation

In the previous section, we have seen how “Alice” may be classified and declassified as Employee and Unemployed Person roles over time. Orthogonally, she may get married and become an instance of the role Married Person and Wife. Additionally, she may be confirmed as a Supreme Court Justice or buy a pet and become a Pet Owner. While she may be a Supreme Court Justice for life, she may later give the pet away and be removed from the Pet Owner set. Every agent, then, must always be classified in at least one role — where Agent can be thought of as a role in its own right.³

In her lifetime, Alice may be an instance of many roles. This means, first, that the roles that apply to an entity can change over time (dynamic classification). Second, it means that an entity can have multiple roles that apply to it at any one moment. When an entity is an instance of more than one role this is called *multiple classification* (not to be confused with multiple inheritance).

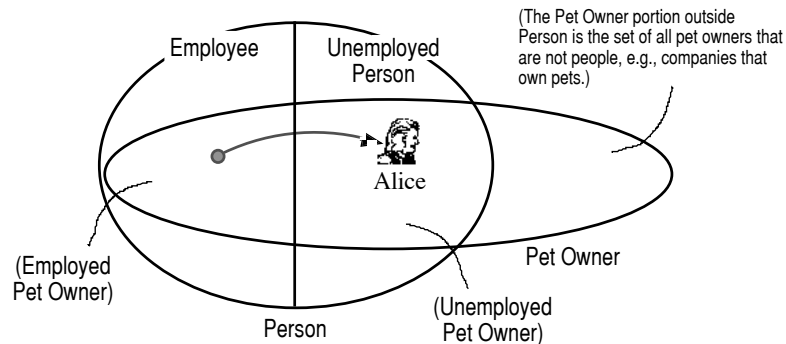


Figure 5 - “Alice” involved in both dynamic classification and multiple classification.

Multiple classification is a useful notion here, because in a society-based system an agent is likely to be active (or quiescent) in multiples roles at the same time. However, what does it mean for a role to be “active”? In some sense, a quiescent agent that is waiting for a message or some signal to awaken could be considered active in its role, because alertness can be thought of an activity. UML 2.0 [OMG, 2003] offers a useful refinement by distinguishing between user-defined actions (which are represented explicitly in sequence diagrams and activity diagrams) and fundamental system actions such as i/o, invocation, and data flow (which are not represented as actions in these diagrams). In UML, each activation, or *execution occurrence*, has some duration and is bounded by a start and stop point. We propose to take advantage of this refinement in the following unification:

- We adopt the UML 2.0 definition of action. Any unit of behavior that has started and has not yet ended is considered “active.” Otherwise it is “inactive.”

³ Having Agent as a role is a controversial point. However, in [Odell, 2003] we defined *role* as a class that defines a normative behavioral repertoire of an agent. The basic class called Agent defines the normative behavioral repertoire for agenthood.

- *We use the basic role of AgentId to specify primitive behavior.* (Id in AgentId is meant to suggest the Freudian sense of primal basic urges, not the sense of “Identity.”) Behavior such as controlling, handling data flows, and waiting for messages and signals can be thought of as “primitive” actions that all entities must possess to be agents. Therefore, any entity playing the role of AgentId can exhibit this basic behavior, deferring “higher-level” behavior to user-specified actions in more specialized roles. Furthermore, these basic behaviors are themselves actions. For example, actions that support listening for messages and signals, *by definition*, begin the moment an entity is classified an AgentId and cease when the entity is no longer an AgentId. This default role satisfies the criterion by Ferber and Gutknecht [Ferber, 2003] stating that “every agent plays at least one role,” in this case the AgentId role.
- *We consider roles other than AgentId to be active only when their user-defined actions are active.* Activity of primitive actions is attributed to the concurrently executing AgentId role, not to the user-specified role.

Dynamic activation involves the following operations: activate, suspend, and shift. These operations are discussed in “change” scenarios 4 through 6, as follows:

4. **Activate - Assume the behaviors of the Manager role as part of the day-to-day business activity.** Here, the person still remains an instance of both Employee and Manager roles. However, when dealing with the employee’s boss, the person is *active* in playing Employee role. Yet, when the employee starts addressing her subordinates, she also becomes active in the Manager role. In other words, the person begins the scenario as active only in the Employee role, and ends being active in both the Employee and Manager roles.
5. **Suspend - Change from the role of Manager to the role of Employee as part of the day-to-day business activity.** As with the previous example, the person still remains an instance of both Employee and Manager roles. In this scenario, the person might start the day as a Manager role by approving an expense report for a subordinate. However, if the person then reports to her boss, she *suspends* her role as Manager. In other words, the person begins the scenario as classified and active in both Employee and Manager roles, and ends with the Manager role suspended while the Employee role remains active. (This is the opposite effect of the scenario 3, above.)
6. **Shift - Change from an Employee role to a Pet Owner role, where neither is played at the same time as the other.** Here, the person still remains an instance of both Employee and Pet Owner roles. However, this differs from scenarios 4 and 5, because the person is not active in both roles at the same time. Here, the person suspends his Employee role and becomes active in the Pet Owner role—yet consistently remains classified in both Employee and Pet Owner roles.

The table below summarizes the role assignments involving activation-related classification: activate, suspend, and shift.⁴

Operation	Prestate		Poststate	
	Active	Suspended	Active	Suspended
Activate	A	B	A and B	
Suspend	A and B		A	B
Shift	A	B	B	A

Both UML and AUML notations would express these three operators are illustrated in Fig. 6. Figure 6(a) indicates agent-1 is activated as role-m; in Fig. 6(b), agent-1 is suspended as role-m. In Figs. 6(c) and 6(d), the role of agent-1 shifts from role-n to role-m. Asynchronous messages (indicated by the open arrowhead) do not require a response. Therefore, the shift would proceed from the end of an execution occurrence bar (the thin triangle over the lifeline) for one role to the beginning of the execution occurrence bar for another (Fig. 6(c)). In contrast, synchronous messages (indicated by the solid arrowhead) require a response before proceeding. Figure 6(d) depicts an agent in role-n sending a message that activates role-m. Since the message is synchronous, all role-n processing is suspended until control is returned from role-m (the dashed arrow).

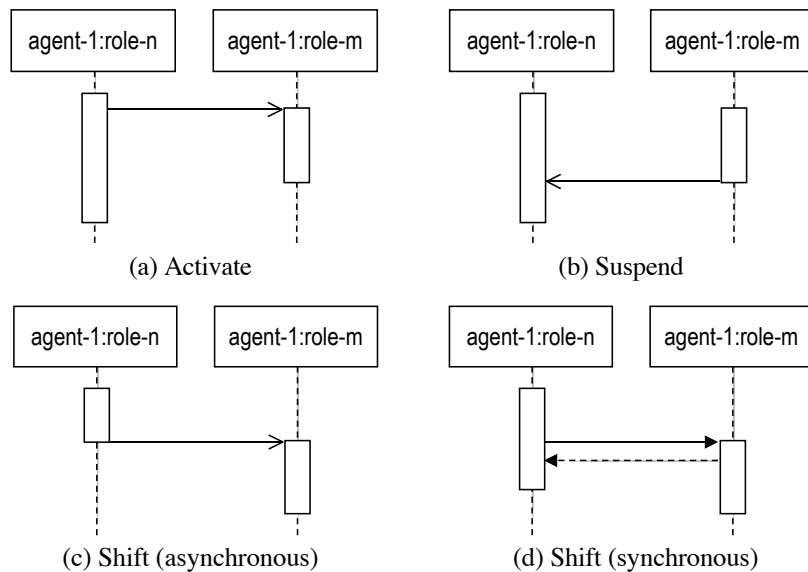


Figure 6 – UML and AUML notation for expressing activate, suspend, and shift.

⁴ Four combinations were omitted from this table. The situations where the pre- and poststates have only suspended roles and where the pre- and poststates have only active roles is not interesting here because there are no state changes. The prestate, where only active role(s) exist and become only suspended ones, is just two concurrent cases of suspension. The prestate, where only suspended role(s) exist and become only active ones, is just two concurrent cases of activation.

For graphical clarity, message lines can be supplemented with stereotypes. For the activate, suspend, and shift operations, the stereotypes would be «activates», «suspends», and «shifts».

5. The Application Analyzed

Let us now revisit the sensing mission example (Section 2) in the light of the formal specification of role dynamics. At the moment of deployment, the units with the sensor package occupy three roles (Task Controller, Flight Controller, and Sensor) and the units without the sensor package occupy three roles (Task Controller, Flight Controller, and Communications Node).

The *Task Controller* (TC) role is activated when the UAV is first turned on. It is responsible for dynamically determining the current tasks that the unit performs during the mission. It uses a set of role activation and transition rules to determine what task it should be performing.

The role of *Flight Controller* (FC) specifies behaviors needed to control the movement of the UAV such as path planning, collision avoidance, etc. The Task Controller will initially inform the Flight Controller of the target location and the Flight Controller will begin to move the UAV towards the target area.

The *Sensor* role manages the task of collecting the target data and transmitting the results back to the deployment base station. When it is activated, it may give new flight coordinates to the Flight Controller so it can perform its function.

The *Communications Node* (Comms) role, causes the UAV to maintain line-of-sight with nearby neighbors and relays communications packets between the base station and the rest of the swarm. It also sends movement commands to the Flight Controller.

The transition rules (in the Task Controller role) are context dependent (e.g., where is the agent, what is its state, what is the locally perceived demand for certain functions within the swarm) and constrained by the unit's physical capabilities (e.g., available sensors, remaining fuel) and individual preferences (e.g., risk aversion, susceptibility to peer-pressure). The rules combine the current context, constraints, and preferences to determine current active role(s) as well as whether to declassify current roles and classify new roles.

For instance the Task Controller role in a UAV may perceive a lack of UAVs with active Communications roles within its line of sight and choose to activate its Communications role⁵. Once the Communications role is activated it maintains line of sight with its nearest neighbors and relays messages. Other UAVs in the vicinity, perceiving the newly activated role, continue on their original flight plan towards the target area realizing that the communications relay function is sufficiently covered in that area now. The Communications role remains active until the mission is complete and the UAVs begin their egress back towards the deployment base. Once its position as a relay point is no longer required to maintain communications with the swarm, it

⁵ The Task Controller uses a probabilistic decision process to determine whether to activate the role. The exact nature of the decision process is beyond the scope of this paper.

suspends the Communications role and the Task Controller instructs the Flight Controller to head home. Figure 7 shows the AUMML diagram for this UAV (not all the interactions are shown on the diagram). A similar diagram could be developed for the nodes that activate their Sensor role when they reach a target area to coordinate with other UAVs collecting data on the target.

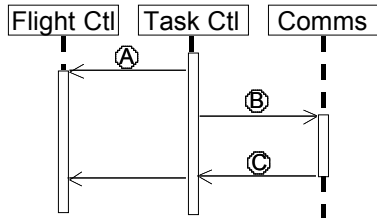


Figure 7 - AUMML for a Communications UAV. A: TC instructs FC to move to target area. B: Enroute, TC determines need to stop and serve as communications relay. C: Mission completes, Communications role is done, and TC tells FC to fly home

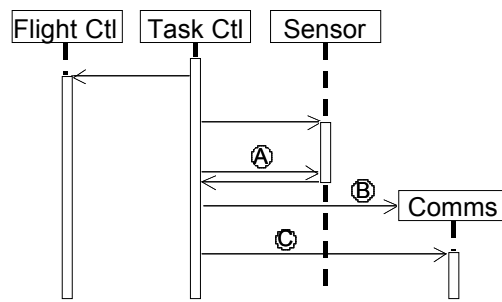


Figure 8 - Classification. A: TC determines a need to change to a Communications role, so it suspends the Sensor role, B: Classifies the Comms role, and C: activates the Comms role.

Since there are only a few UAVs with sensor packages, they would normally not be used to serve as communications relays. But in an emergency they can perform that function. Since they do not initially occupy the role of Communications Node, they must change their role through a classification operation. Figure 8 depicts one of these UAVs that was in the middle of a sensing role and determined that there was a higher priority need to fill a Communications Node role. It suspended the Sensor role, classified the new role of Communications Node, and activates that new role. Classification in this case is more than just activation of previously existing code in the UAV. A

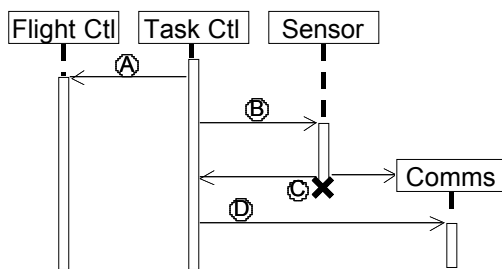


Figure 9 - Reclassification. A: TC instructs FC to fly to target area B: UAV reaches target and TC activates Sensor role C: Sensor HW failure, reclassify as Communications Node, D: TC activates a Comms role.

communications relay must have current routing information. The exact form of this information varies with the communications protocol in use, but the information is dynamic and in the nature of the case must be learned by a relay UAV before it can perform this function. Classification is only possible for a UAV that has learned this information (perhaps by lurking on the network for a while without relaying, or by requesting routing information from a nearby routing UAV and using this as a first approximation). Part of what

the Task Controller does in the process of classifying the UAV for the relay role is acquiring this routing information.

A last example demonstrates reclassification. In Fig. 9 the UAV performing its sensing task (active Sensor role) experiences a hardware failure in its sensor package so it can no longer perform the sensor function. It immediately reclassifies itself from the Sensor role to the Communications Node role which is later activated by the Task Controller when it determine a need for that role in the area. Similarly if the UAV loses its ability to move (e.g., through hardware failure, or loss of fuel) the Flight Controller role would be declassified and the Task Controller would probably activate the Communications role (if not already active) to serve the mission in its only remaining capacity. This may free up a nearby UAV serving as a communications node to move on to perform some other function elsewhere in the swarm.

6 Conclusion

Roles are increasingly recognized as a valuable abstraction for modeling groups of agents. In dynamic environments, an agent may change the roles it plays over time. Analysis of these changes show that they fall into two general categories.

1. The more conventional concept is Dynamic Activation. An agent may incorporate multiple roles but not be active in all of them at the same time. Varieties of Dynamic Activation describe the different patterns in how an agent activates or suspends the various roles that it possesses.
2. Dynamic Classification deals with the more fundamental binding between an agent and a role. Straightforward mechanisms for role assignment in contemporary programming languages (e.g., inheritance from a class defining the role's behaviors) are static and persist for the agent's lifetime. However, the concept of Dynamic Classification encourages us to conceive of roles being bound to an agent after the agent is instantiated, and unbound without terminating the agent.

We have demonstrated the usefulness of these concepts in formalizing the behaviors of a swarm of unpiloted air vehicles performing a cooperative sensing task.

The notion of roles invites a new approach to agent programming, in which the unit of agent invocation is the role rather than the individual behavior. In such a role-oriented programming environment (ROPE), invocation consists of passing an agent a role and an execution environment (compare the notion of Agent Coordination Context in [Omicini, 2002]), and it is up to the agent to carry out the role in that context. Development of ROPE is a future opportunity for this line of research.

Acknowledgements

This work is supported in part by DARPA, contract F30602-02-C-0196 to Altairum, under DARPA PM Vijay Raghavan. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official

policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

The authors are grateful to their colleagues Paul Chiusano, Bob Matthews, Mike Samples, and Peter Weinstein for insightful review and comments on earlier drafts of this paper.

7 References

Castelfranchi, Cristiano (2000) "Engineering Social Order," *Engineering Societies in the Agent World*, Springer, Berlin, pp. 1-18.

Dastani, M., V. Dignum, and F. Dignum (2003) "Role Assignment in Open Agent Societies" in *Proceedings of AAMAS'03, Second International Joint Conference on Autonomous Agents and Multi-agent Systems*. 2003. Melbourne, Australia.

Ferber, J., O. Gutknecht, et al. (2003). "Agent/Group/Roles: Simulating with Organizations." *Fourth International Workshop on Agent-Based Simulation (ABS03)*, Montpellier, France.

Martin, J. and J.J. Odell, *Object-Oriented Methods: A Foundation*. UML ed. 1998, Englewood Cliffs, NJ: Prentice Hall.

Odell, J.J., *Advanced Object-Oriented Analysis & Design using UML*. 1998, Cambridge, UK: Cambridge University Press.

Odell, J., H.V.D. Parunak, and B. Bauer (2001) "Representing Agent Interaction Protocols in UML," in *Agent-Oriented Software Engineering*, P. Ciancarini and M. Wooldridge, eds. 2001, Springer: Berlin. p. 121-140.

Odell, J., H.V.D. Parunak, and M. Fleischer (2003), "The Role of Roles in Designing Effective Agent Organizations," in *Software Engineering for Large-Scale Multi-Agent Systems*, A.F. Garcia, et al., eds. 2003, Springer-Verlag: Berlin.

Omicini, A. Towards a notion of agent coordination context. In D. Marinescu and C. Lee, Editors, *Process Coordination and Ubiquitous Computing*, 187-200. CRC Press, 2002.

Parunak, H. Van Dyke and James Odell (2001) "Representing Social Structure using UML," *Proc. of the Agent-Oriented Software Engineering Workshop, Agents 2001 Conference*, Paolo Ciancarini, Michael Wooldridge, and Gerhard Weiss eds., Agents 2001 conference, Montreal, Canada, Springer.