

Representing Social Structures in UML

H. Van Dyke Parunak

ERIM

PO Box 134001

Ann Arbor, MI 48113-4001 USA

+1 734 623 2509

vparunak@erim.org

James Odell

James Odell Associates

3646 West Huron River Drive

Ann Arbor, MI 48103

+1 (734) 994-0833

jodell@compuserve.com

ABSTRACT

From a software engineering perspective, agent systems are a specialization of object-oriented (OO) systems, in which individual objects have their own threads of control and their own goals or sense of purpose. Engineering such systems is most naturally approached as an extension of object-oriented systems engineering. In particular, the Unified Modeling Language (UML) can be naturally extended to support the distinctive requirements of multi-agent systems. One such requirement results from the increasing emphasis on the correspondence between multi-agent systems and social systems. Sociological analogies are proving fruitful models for agent-oriented constructions, while sociologists increasingly use agents as a modeling tool for studying social systems. We combine several existing organizational models for agents, including AALAADIN, dependency theory, interaction protocols, and holonics, in a general theoretical framework, and show how UML can be applied and extended to capture constructions in that framework.

Categories and Subject Descriptors

D.2.1 [Software Engineering]: Requirements/Specifications – languages, methodologies.

J.4 [Computer Applications]: Social and Behavioral Sciences – psychology, sociology.

General Terms

Documentation, Design, Standardization, Languages, Theory

Keywords

Agents, UML, AAML, Organizations, Dependencies, Protocols, Hierarchy, AALAADIN

1. INTRODUCTION

From a software engineering perspective, agent systems are a specialization of object-based systems, in which individual objects have their own threads of control and their own goals or sense of purpose. Elsewhere we have defined agents as objects that can say “go” (reflecting their separate threads of control and the resulting ability to execute without being

externally invoked) and “no” (reflecting the priority of their internal goals over external direction) [17]. Engineering such systems is most naturally approached as an extension of object-oriented systems engineering. In particular, the Unified Modeling Language (UML), a product of the OO community, is a natural starting point for developing requirements and designs for agent-based systems. It is widely known, and supported by a number of computer-aided software engineering platforms.

Some researchers have already called the attention to the potential of UML, in its unmodified form, for addressing many aspects of agent-based systems [2, 25]. As valuable as these efforts are, they cannot accommodate the additional functionality of agents over objects. Earlier work addressed one such area, the definition of interaction protocols between autonomous processes, and suggested constructs and conventions for Agent UML (AAML) [18, 19].

This paper addresses another area of agent functionality that goes beyond the capabilities of current UML. Sociological concepts have always been a source of inspiration for multi-agent research, and recently the agent community has been returning the favor by exploring the potential of agent-based models for studying sociological phenomena (e.g., [4, 8, 15, 24]). The result of this interaction has been the formalization of a number of sociological and psychological concepts with important applications in engineering agent systems, concepts that are not directly supported in UML.

This paper brings together a number of these concepts, including “group,” “role,” “dependency,” and “speech acts,” into a coherent syntax for describing organizational structures, and proposes UML conventions and AAML extensions to support their use in the analysis, specification, and design of multi-agent systems.

Section 2 outlines the syntax of group structure on which our representation is based. Section 3 describes a simple scenario and illustrates the use of the proposed UML constructs to model it. Section 4 summarizes our contribution.

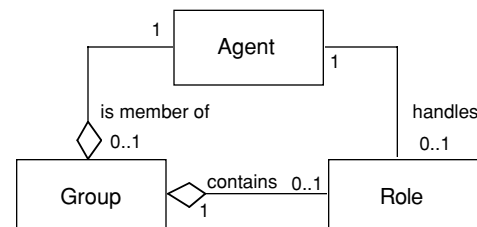


Figure 1: AALAADIN architecture expressed as a UML class diagram

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Autonomous Agents '01, May 28-June 1, 2001, Montreal, Canada.
Copyright 2001 ACM 1-58113-000-0/00/0000...\$5.00.

2. A GROUP SYNTAX

We begin with the AALAADIN architecture proposed by Ferber and Gutknecht [10]. Their model, expressed as a UML class diagram in Figure 1, involves three elements:

- An *agent* is an active communicating entity.
- A *group* is a set of agents.
- A *role* is an abstract representation of an agent's function, service, or identification within a group.

A fundamental insight of AALAADIN is that groups interact only through shared agents. An agent can be a member in several different groups at the same time, perhaps playing a different role (or roles) in each of them, and can move from one group to another dynamically. From a classical OO point of view, these capabilities distinguish the "agent-group" relationship from the "instance-class" relationship.

AALAADIN's notion of a group is naturally extended to organizations. We understand "group" as the generic term for a set of agents related through their roles. The term "organization" refers to groups distinguished by formality and stability, but the underlying model and formalisms are the same.

AALAADIN offers an attractive combination of parsimony and expressiveness. It is a straightforward basis for developing representational mechanisms for organizational concepts. We refine AALAADIN in three ways. First, we suggest that the notion of "role" can be defined in terms of two other notions current in the agent community, dependencies and action templates. Second, the ontology should be extended with the concept of an environment through which agents interact. Third, we confront the issue of aggregation as developed in the holonic model and its apparent tension with AALAADIN's flat network of groups.

2.1 Looking Behind Roles

AALAADIN defines a group as a set of agents. An agent's role mediates between it and the group. That is, it participates in the group not in its own right, but as an incumbent of a particular role, and the nature of the group is defined by the set of roles that it includes and their relations to one another. This observation encourages us to seek a more fundamental way to define roles, one that makes explicit their relations with one another. Two concepts from the theory of multi-agent systems contribute to this refined definition: dependency theory and speech acts. Both of these concepts define structured relations among interacting agents. Our insight is that a role is just a labeling of a recurring pattern of dependencies and (speech and other) interactions.

Dependency theory (e.g., [5, 6, 26], to cite only a few classic references) is based on the idea that interactions among agents result from an incompatibility between a single agent's goals and its resources and capabilities. Typically, an agent does not control all the resources or have the ability to execute all the actions required to achieve its objectives, so it must interact with other agents who do control those resources or who can perform those actions. For expository simplicity, we consider only action dependencies (since a dependency on an agent for a resource may be represented as depending on that agent to perform the action of providing the resource), and define a dependency as a three-tuple $\langle \text{Dependent}, \text{Provider}, \text{Action} \rangle$. The literature on dependency theory works out many

additional details that might be included in a dependency, including the specific goal that leads to the dependency, the specific plan leading to that goal that requires the specified resource or action, and which of the agents involved in the dependency recognizes its existence. Such details can readily be incorporated in our approach.

A pattern of dependencies is an important component of a role. For example, if agent A is a customer, there must be some agent B on whom A depends for goods and services, while B depends on A for money. However, dependencies alone leave many details of the role ambiguous. From dependencies alone, A and B might be thieves preying on one another. We can refine the definition of a role by incorporating more dynamic information, based on speech act theory.

Speech-act theory originates in the observation of [1] that utterances are not simply propositions that are true or false, but attempts on the part of the speaker that succeed or fail. The theory has become ubiquitous in the development of agent communication protocols such as KQML [12] and the FIPA Agent Communication Language [13]. Isolated utterances do little to characterize the relationships among interacting agents. However, the "speech-act" nature of utterances permits the definition of a set of relations that can obtain among successive utterances in a conversation [20]. For example, utterance B "responds" to utterance A if A caused B; it "replies" to A if it responds and in addition is addressed to the agent who uttered A, it "resolves" A if it replies and in addition causes A to succeed, and it "completes" A if it fulfills an earlier commitment. These relations may be extended to non-speech actions. For example, a request to close the door may be resolved by the physical action of closing the door. A formal construction based on these relations, the Dooley Graph [7, 20], consists of nodes corresponding to agents, with directed edges corresponding to utterances and other actions issued by the source of the edge and received by its sink. Each node in a Dooley Graph corresponds to a single agent, but an agent may occupy several different nodes. The significance of a single node is that it terminates speech acts associated with a single function of the agent. By a thespian analogy, this function has been called a "character" [20], but bears clear relationship to the notion of a role in AALAADIN. The characters disclosed through Dooley analysis are useful in engineering agent code, since they define reusable behavioral templates (cf. the modules in BRIC [9], AARIA's Actions [23], Contexts in Gensym's Agent Development Environment [21], and Singh's agent templates [27]). Such a template implements a particular protocol, defining an interrelated set of speech or other actions that an agent may undertake.

From the point of view of a protocol, A is a customer of B if the two participate in the pattern of actions outlined in Table 1. This pattern is not the only protocol that would be appropriate for a customer and supplier. For example, a customer might initiate the exchange with a request for quotation. The point is that a protocol can capture part of the semantics of a role. As

Table 1. A simple protocol characterizes the "customer" role.

Utterance #	Speaker	Utterance
1	B	Advertisement: Offers goods in exchange for money
2	A	Sends an order to B for the goods, accompanied by the payment
3	B	Sends the goods to A

with dependencies, however, the semantics are incomplete. The same protocol would be appropriate between two actors in a play, neither of whom depends on the other for either money or goods. Adding the dependency information provides a clearer and more robust picture of the roles of customer and supplier than either offers by itself. Such a protocol can be represented in UML at various levels of abstractions Previous work [18, 19].

From the point of view of system analysis, both dependency theory and interactions are attractive because they can be analyzed (at least at the level necessary for our purposes) without access to the internal state of the agents. Empirically, an agent's function in a group is defined by the network of dependencies and actions in which it is embedded. When a particular pattern of dependencies and protocols recurs, it is useful to summarize it as a role. However, the role is not primitive, but built up from dependencies and interactions. The dependencies and actions are what really matter. The role is simply a label that we attach to a recurring pattern to enable us to manipulate it with greater efficiency. This viewpoint implies (contrary to the original AALAADIN model exemplified in Figure 1) that the same role can appear in multiple groups, if they embody the same pattern of dependencies and interactions (Figure 2).

If an agent in a group holds multiple roles concurrently, it may sometimes be useful to define a higher-level role that is composed of some of those more elementary roles. For example, consider the three roles Customer, Vendor, and Employee, all within the group Market Economy. An end-user in such an economy is an agent who earns money in a job and uses that money to purchase goods. Thus the End User role is the composition of Customer and Employee roles. A supply chain link in such an economy is an agent who buys goods, transforms them into other goods, and then sells them. Thus Supply Chain Link is the composition of Customer and Vendor roles. Figure 2 expresses this composition of roles by the composition association loop at the top of the Role class.

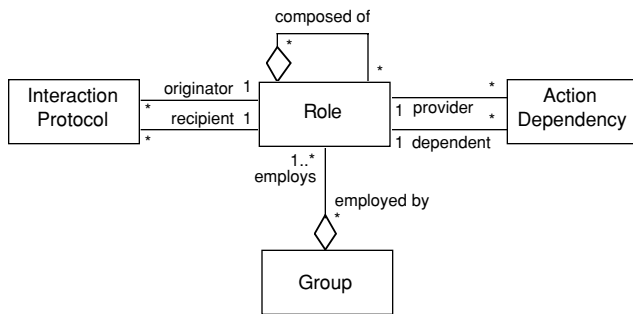


Figure 2. UML Class Diagram expressing the role associations described in Section 2.1.

2.2 The Importance of the Environment

Agents cannot be considered independently of the environment in which they exist and through which they interact [11, 16]. This insight is often ignored in work on computational multi-agent systems, where researchers consider the environment as a passive communications framework and everything of interest is relegated to one or another of the agents. Even such purely electronic environments often prove embarrassingly active when an application is scaled up for real-world use, and those

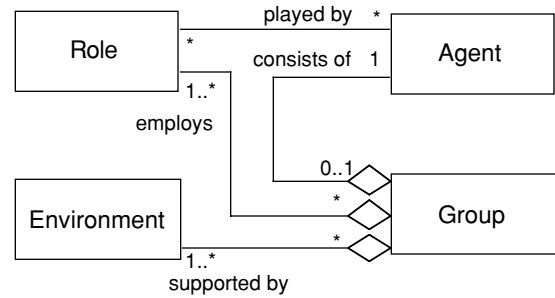


Figure 3. The environment and its involvement with groups, agents, and roles.

engineering agents for non-electronic domains (such as factory automation) must consider the environment explicitly.

Consider, for example, the active role of the environment in pheromone models of coordination [3, 22]. In natural insect societies and engineered systems inspired by them, the environment actively provides three information processing functions.

1. It *fuses* information from different agents passing over the same location at different times.
2. It *distributes* information from one location to nearby locations.
3. It provides *truth maintenance* by forgetting information that is not continually refreshed, thereby getting rid of obsolete information.

In modeling groups, it is important to recognize the role of the environment in supporting the dependencies and protocols that define the participants' roles. The exact form that this representation takes will vary widely, depending on the group in question. However, we should recognize that environment, along with the network of roles that it supports, is a defining component of a social group (Figure 3).

2.3 Groups as Agents

At first glance, AALAADIN contrasts strongly with the holonic view of agent organization. A holon is an agent that may be made up of smaller holons and that may join together with other holons to form higher-level holons. Thus the holonic model explicitly permits groups to be members of other groups. The resulting lattice of holons is called a holarchy, and is the dominant organizational metaphor in holonic systems [28]. In AALAADIN, groups are related only through shared members, and there is no provision for one group as a group to be a member of another.

Both models have their own attractions. The holonic model recognizes the need for some form of hierarchical aggregation in real-world systems, which must remain understandable while spanning a wide range of temporal and spatial scales. A modern automotive factory incorporates hundreds of thousands of individual mechanisms (each a candidate for agent-hood) in hundreds of machines that are grouped into a dozen or so lines. Engineers can design, construct, and operate such complexes only by shifting the focus from mechanism to machine or line, depending on the problem at hand, and recognizing the higher-level agents are aggregates of lower-level ones. Similarly, in e-commerce applications, a corporation is a legal entity that is independent of the individual people who make up its employees and directors.

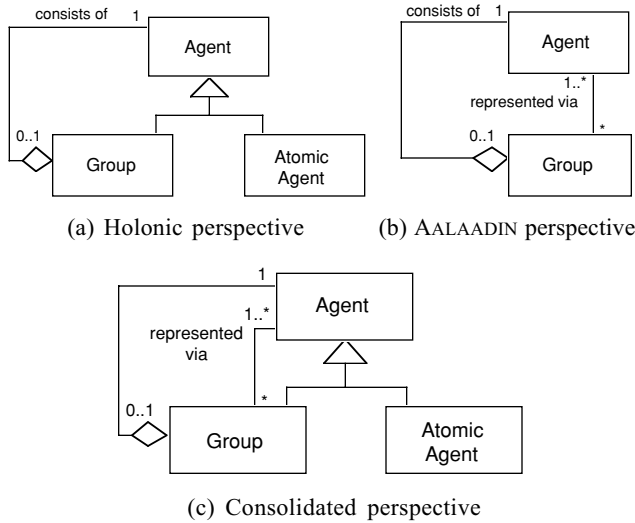


Figure 4. Three perspectives on the relation between Groups and Agents

Conversely, AALAADIN recognizes that when two groups (at any level) interact, they do so through the interactions of their components. Negotiations between two companies take place through individual people with the roles of representing their groups in the negotiation. Two machines in a factory interact by virtue of a process involving the sensors of one and the actuators of another.

As practicing engineers of agent-based systems, we recognize the need for both perspectives, and resolve the tension pragmatically. When we begin to analyze a group A , we identify the agents $\{a_1, a_2, \dots, a_n\}$ occupying its roles. Those agents may be individual persons, robots, or computer systems (atomic agents). They may also be other groups, $a_i = B$, which we treat as black boxes. We take this “holonic” perspective as long as our analysis can ignore the internal structure of the member groups (Figure 4a). However, subsequent analysis often requires us to open such a black box and look inside at its roles and their incumbent agents, analyzing $B = \{b_1, b_2, \dots, b_m\}$. At that point, we insist on identifying which of B ’s member agents is actually responsible for filling B ’s role in A , thus adhering to the discipline of AALAADIN (Figure 4b). Figure 4c is thus the consolidated model for our approach.

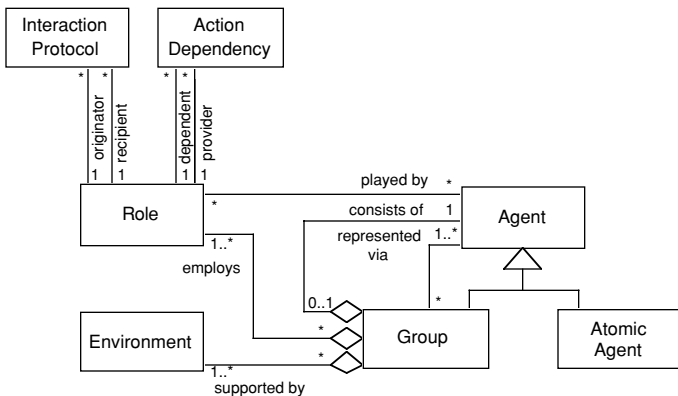


Figure 5. Consolidated ontology.

2.4 Summary

In sum, our model is based on AALAADIN, but with three extensions:

- Roles are not ontologically primitive, but are defined as recurring patterns of dependencies and actions.
- The definition of a group includes not just a set of agents occupying roles, but also the environment through which they interact.
- AALAADIN’s requirement that groups interact only through identified members is relaxed in the case of unanalyzed groups, which are permitted to occupy roles in higher-level groups following the holonic model.

Figure 5 consolidates the class views described in this section.

3. AN EXAMPLE

We illustrate the theory and its UML instantiation by modeling a simple example, a terrorist organization.

3.1 A Terrorist Organization

A national security organization might construct an agent-based model of terrorism, for use in contingency planning and modeling emerging threats. In this particular model, we envision interactions among three groups with associated roles. Individual agent A occupies roles in all three groups.

- The terrorist organization (TO) has roles
 - *Operative*, who actually deploys and operates the instrument of terrorism (e.g., plants and detonates the bomb, or shoots the gun) ($= A$)
 - *Ringleader*, who sets the vision for the organization and may bankroll it personally
- The weapons cartel (WC) has roles
 - *Customer*, who wishes to procure arms ($= A$)
 - *Supplier*, who delivers arms to the customer
 - *Negotiator*, who negotiates the deal with the customer and receives payment.
- Western society (WS) has roles
 - *Citizen*, whom the terrorist operative wishes to target
 - *Student*, a convenient cover for a foreign national ($= A$).

Informally, individual A procures financing and a mission from TO, while feeding back information that permits TO to expand its activities. A uses funds from TO to purchase weapons from WC, and then occupies a role in WS to deploy the weapons against citizens.

3.2 Its AUML Model

The OMG’s Unified Modeling Language (UML) version 1.3 already provides a wealth of diagramming elements [14]. However, extensions to UML are required to effectively model agents and agent-based systems. Within both OMG and FIPA, an effort is currently underway to define a UML for agents (AUML) that extends UML (<http://www.auml.org>). This section proposes usages of and extensions to UML to represent

groups, agents, and roles, as illustrated in the Terrorist Organization scenario, above.

3.2.1 Swimlanes as Groups

The scenario described in 3.1 involve three groups, each employing defined roles. Figure 6 illustrates these groups and roles using two UML techniques: the class diagram and swimlane. The class diagram here models the various terrorist roles and their relationships. (The slash in front of each name indicates that the name is a role name, rather than a class name.)

In UML, swimlanes graphically organize responsibility of activities within an activity graph. However, AAML proposes that the same device be used for *any* kind of UML diagram—in the case of Figure 6, a class diagram. For example, Figure 6 indicates that the Terrorist Organization involves two roles, Operative and Ringleader, where the Ringleader agent coordinates Operative agents. Figure 6 also depicts a second kind of swimlane based on agent instance. For example, agent *A* plays the roles of Operative, Customer, and Student. Multidimensional swimlanes are highly uncommon in the UML community, but are supported by UML version 1.3.

The UML 1.3 swimlane is only “syntactic sugar,” a graphical packaging technique. It cannot specify a swimlane's underlying semantics. Understanding swimlanes like those in Figure 6 could cause difficulty because vertical swimlanes specify group aggregation, while the horizontal swimlanes specify role instantiation. We propose that UML be extended to specify the swimlane's underlying relationship. In Figure 6, the vertical swimlanes are indicated as <<aggregation>> and the horizontal as <<instantiation>>.

3.2.2 Class Diagrams to Define Roles

As mentioned above, the vertical swimlanes define an aggregation relationship between groups and the roles that comprise the group. Another way to express these relationships within a group is to use a class diagram as depicted in Figure 7.

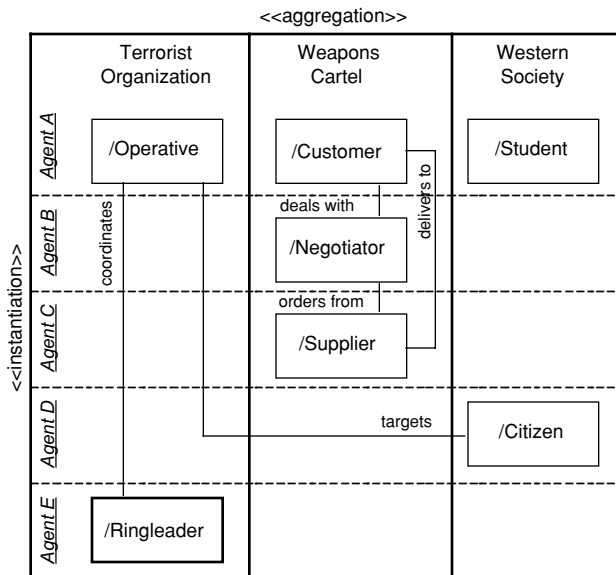


Figure 6. A class diagram with swimlanes depicts the interrelated roles with their agents and groups.

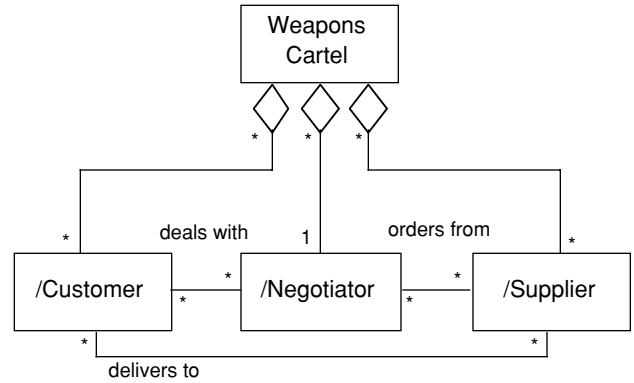


Figure 7. A class diagram depicting that Terrorist Organizations consist of agents playing several distinct roles.

The class diagram in Figure 7 and the Weapons Cartel swimlane in Figure 6 are basically equivalent. They both show that each Weapons Cartel group consists of agents playing the roles of Customer, Negotiator, and Supplier. Additionally, they both depict relationships among the roles. The only difference is that Figure 7 expresses the relationship cardinality constraints (multiplicity) between the Weapons Cartel and the various roles, while Figure 6 does not. For example, Figure 7 indicates that while each Weapons Cartel group may have multiple customers and suppliers, it may only have one negotiator. These constraints cannot be expressed by UML swimlanes.

3.2.3 Sequence Diagrams to Show Roles as Patterns of Interactions

Class diagrams model the kinds of entities that exist in a system along with their relationships. Modeling the interactions that may occur among these entities can be represented using a UML sequence diagram. For example, Figure 8 depicts the permitted interactions that may occur among Customer, Negotiator, and Supplier agents for a weapons procurement negotiation. The tabbed folder encompassing the sequence diagram indicates that the

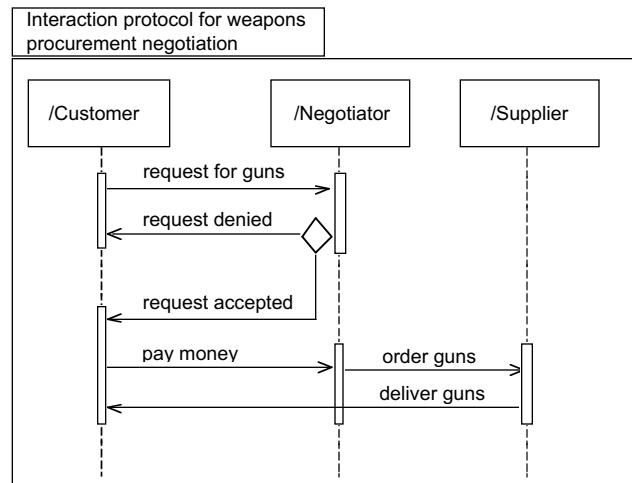


Figure 8. Sequence diagram depicting an interaction protocol for buying guns from a terrorist operation agent.

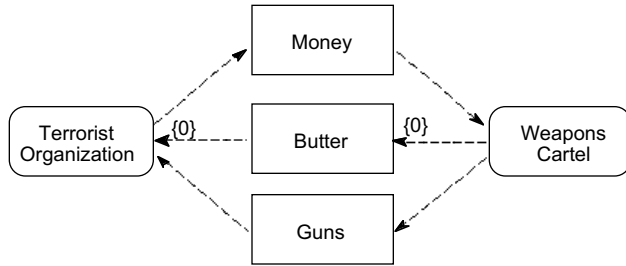


Figure 9. An object-flow activity graph specifies roles as patterns of activities along with the products that are produced and consumed by each activity.

interaction can be viewed as a unit called a *package*.

The only extension to UML is the addition of the diamond-shaped decision symbol. While branching decisions can be expressed in UML 1.3 using guard conditions, we recommend using the same symbol employed for the same purpose in activity graphs—the diamond. (For more AUML extensions to the sequence diagram, see [18, 19].)

3.2.4 Activity Graphs to Show Groups as Patterns of Dependencies

The object-flow activity graph represents activities and the kinds of objects that they produce or consume. The weapons procurement in Figure 9 states that a Terrorist Organization procures guns (and not butter) from a Weapons Cartel to which it pays money. Instead of representing the way in which roles relate or interact among groups (Figure 6, Figure 7, Figure 8), this diagram models *groups* as processing entities in their own right. For example, Figure 9 depicts two groups and their dependencies, without regard to the underlying roles. In other words, it expresses the pattern of dependencies between a Terrorist Organization and a Weapons Cartel.

Object-flow activity graphs, then, provide a way to model a system by removing some of the detail, providing a "higher-level" view of the system components. Figure 4c expresses the fact that all groups are agents; but until this point, the only agents we were modeling were as role-playing elements of a group. The activity graph allows us to model groups of agents as agents. In this way, we can express the kinds of dependencies that are best represented at a group level. When a more detailed view of the underlying interactions is required, a sequence diagram (e.g., Figure 8) can be used.

Such an approach requires a UML extension. UML requires all activities terminate in order to produce their output. However, agent groups such as terrorist organizations or order processing departments are typically thought of as continuous activities, not as processes with definable starts and completions. This proposed extension is important when modeling long-lived groups that produce output many things during their lifetime.

4. SUMMARY

Engineering of agent-based systems requires the availability of common languages for requirements analysis, specification, and design. The Unified Modeling Language (UML) has gained wide acceptance in the Object-Oriented community, and is supported by a number of computer-aided software engineering tools. The close relation between objects and agents has led numerous researchers to seek to apply it to

agents. Carrying out this agenda requires that we identify the distinctive constructions required in designing agent-oriented systems and develop conventions for using and extending UML to accommodate them.

In the case of social structures, insights from AALAADIN, dependency theory, and holonics can be fused into a single metamodel of groups as composed of agents occupying roles (defined as patterns of dependency and interaction) in an environment. Various UML constructs, including swimlanes, class diagrams, activity diagrams, and sequence charts, can capture the crucial distinctions in such a model.

5. ACKNOWLEDGMENTS

When the first author presented an invited talk on applications of UML to agents at MAAMAW'99, several participants, including Cristiano Castelfranchi and Yves Demazeau, raised the question of how a wide range of organizational structures could be captured in UML. Their challenge was the germ for this paper. Some of the ideas concerning the relation of dependencies and action relations to the definition of roles were stimulated by discussions with Catholijn Jonker, Gabriela Lindemann, and others at the MASHO'00 workshop, and by comments on a preliminary version with John Sauter, Mitch Fleischer, and others at ERIM.

6. REFERENCES

- [1] J. L. Austin. *How to Do Things with Words*. Oxford University Press, 1962.
- [2] F. Bergenti and A. Poggi. Exploiting UML in the Design of Multi-Agent Systems. In *Proceedings of Engineering Societies in the Agents' World*, pages 96-103, 2000.
- [3] S. Brueckner. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Thesis at Humboldt University Berlin, Department of Computer Science, 2000.
- [4] K. Carley and M. Prietula, Editors. *Computational Organization Theory*. Lawrence Erlbaum Associates, 1994.
- [5] C. Castelfranchi. Founding Agent's 'Autonomy' on Dependence Theory. In *Proceedings of 14th European Conference on Artificial Intelligence*, pages 353-357, IOS Press, 2000.
- [6] C. Castelfranchi, M. Miceli, and A. Cesta. Dependence Relations among Autonomous Agents. In Y. Demazeau and E. Werner, Editors, *Decentralized AI 3 (Proceedings of the Third European Workshop on Modeling Autonomous Agents in a Multi-Agent World)*, Elsevier, 1992.
- [7] R. A. Dooley. Appendix B: Repartee as a Graph. In R. E. Longacre, Editor, *An Anatomy of Speech Notions*, pages 348-58. Peter de Ridder, Lisse, 1976.
- [8] M. E. Epstein and R. Axtell. *Growing Artificial Societies: Social Science from the Ground Up*. Boston, MA, MIT Press, 1996.
- [9] J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Harlow, UK, Addison Wesley Longman, 1999.
- [10] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings of Third International Conference on Multi-*

- Agent Systems (ICMAS'98)*, pages 128-135, IEEE Computer Society, 1998.
- [11] J. Ferber and J.-P. Müller. Influences and Reactions: a Model of Situated Multiagent Systems. In *Proceedings of Second International Conference on Multi-Agent Systems (ICMAS-96)*, pages 72-79, 1996.
- [12] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. McKay, J. McGuire, R. Pelavin, S. Shapiro, and C. Beck. DRAFT Specification of the KQML Agent-Communication Language., vol. 2000, 1993. Available at <http://www.cs.umbc.edu/kqml/kqmlspec/spec.html>.
- [13] FIPA. FIPA Agent Communication Language Specifications., vol. 2000, FIPA, 2000. Available at <http://www.fipa.org/repository/aclspecs.html>.
- [14] M. Fowler and K. Scott. *UML Distilled: Applying the Standard Object Modeling Language*. Reading, MA, Addison-Wesley, 1997.
- [15] G. N. D. Gilbert, J. *Simulating Societies: the computer simulation of social processes*. London, UCL Press, 1993.
- [16] J. P. Müller. *The Design of Intelligent Agents*. Berlin, Springer, 1996.
- [17] J. Odell. Agents: Technology and Usage (Part 1). *Distributed Computing Architecture/E-Business Advisory Service*, 3(4):1-29, 2000.
- [18] J. Odell, H. V. D. Parunak, and B. Bauer. Extending UML for Agents. In *Proceedings of Agent-Oriented Information Systems Workshop*, pages 3-17, 2000.
- [19] J. Odell, H. V. D. Parunak, and B. Bauer. Representing Agent Interaction Protocols in UML. In *Proceedings of First International Workshop on Agent-Oriented Software Engineering*, Springer, 2000.
- [20] H. V. D. Parunak. Visualizing Agent Conversations: Using Enhanced Dooley Graphs for Agent Design and Analysis. In *Proceedings of Second International Conference on Multi-Agent Systems (ICMAS'96)*, pages 275-282, 1996.
- [21] H. V. D. Parunak. Workshop Report: Implementing Manufacturing Agents. Industrial Technology Institute, 1996. Available at <http://www.erim.org/~vparunak/paamncms.pdf>.
- [22] H. V. D. Parunak. 'Go to the Ant': Engineering Principles from Natural Agent Systems. *Annals of Operations Research*, 75:69-101, 1997.
- [23] H. V. D. Parunak, A. D. Baker, and S. J. Clark. The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design. *Integrated Computer-Aided Engineering*, Forthcoming, 1999.
- [24] M. J. Prietula, K. M. Carley, and L. e. Gasser. *Simulating Organizations: Computational Models of Institutions and Groups*. Menlo Park, CA, AAAI Press, 1998.
- [25] G. Satapathy and S. R. T. Kumara. Object Oriented Design based Agent Modeling. In *Proceedings of The Fourth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 143-162, The Practical Applications Company, 1999.
- [26] J. S. Sichman, Y. Demazeau, R. Conte, and C. Castelfranchi. A Social Reasoning Mechanism Based on Dependence Networks. In *Proceedings of 11th European Conference on Artificial Intelligence*, pages 416-420, John Wiley and Sons, 1994.
- [27] M. P. Singh. Developing Formal Specifications to Coordinate Heterogeneous Autonomous Agents. In *Proceedings of Third International Conference on Multi-Agent Systems (ICMAS'98)*, pages 261-268, IEEE Computer Society, 1998.
- [28] University of Hannover. Holonic Manufacturing Systems. 2000. Web Page, <http://hms.ifw.uni-hannover.de/>.