

AGENT ROLES AND ASPECTS

Elizabeth A. Kendall
Intelligent Business Systems Research, BT Research Laboratories
MLB1 / PP12
Martlesham Heath, Ipswich IP5 3RE ENGLAND
Email: kendall@info.bt.co.uk

ABSTRACT

Agent systems are highly distributed and feature widespread coordination and collaboration. Because of these characteristics, a traditional object oriented approach to software development is not adequate. This paper presents role models as valuable abstractions for specifying, modeling, and designing agent systems. It also discusses the relationship between role models and aspect oriented programming (AOP), and concludes that AOP appears to be a promising approach for agent system implementation.

1.0 ROLE MODELS

1.1 Overview

In object oriented software engineering, role models have been developed in response to the following needs:

- Role models emphasize how entities interact with each other. Classes stipulate the capabilities of individual objects, while the notion of a role focuses on the position and responsibilities of an object within an overall structure or system.
- Role models describe systems in terms of their patterns of interaction. With this, they provide a new abstraction that is orthogonal to classes and objects.
- Role models can be instantiated, generalized, specialized, and aggregated into compound models [1, 11]. This promotes reuse.
- Role models can be dynamic [9, 10], involving sequencing, evolution, and role transfer. This could be of value to models and implementations of mobility.

1.2 Example

The example is based on that found in [13], and the application is an object model for a factory. There are managers, including the plant manager. There are several functional groups, including assembly, QA (quality assurance), and a repair shop. There are customers of the plant who order components.

Using UML notation, the class diagram is shown in Figure 1, while a collaboration diagram appears in Figure 2. The class diagram does not reveal much about the system's dynamics; this is addressed in the collaboration diagram in Figure 2. A typical messaging sequence is shown. The sequence begins with a Customer's request for a new product. This leads to messages 2 through 5, where the plant manager delegates work to assembly and then to QA. The product is then delivered to the customer in message 6, but there is a problem that leads to messages 7 and 8. The pattern of interaction is clear; all messaging is done through the plant manager.

The collaboration depicted in Figure 2 is a pattern of general applicability; it is the Mediator pattern [4]. A role model representation [14] of it is shown in Figure 3, where a rounded box is a role, and the solid arrows indicate collaboration paths between the

roles. The direction of the arrow represents the direction of the path, and the solid circle on the link between the Mediator and the Colleague indicates that there can be more than one Colleague. The objects in the application (shown as rectangles as in Figure 1) play the various roles, as indicated by the dashed arrows. (The notation shown is based on a hybrid of [1] and [14].)

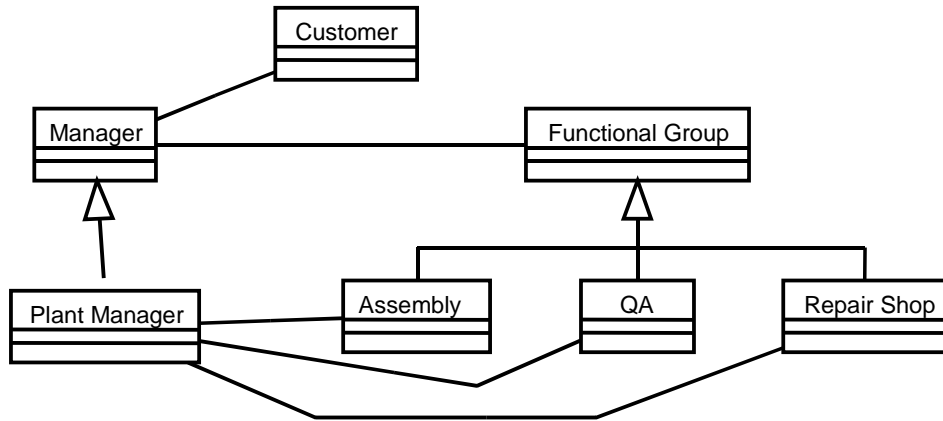


Figure 1: Class Diagram

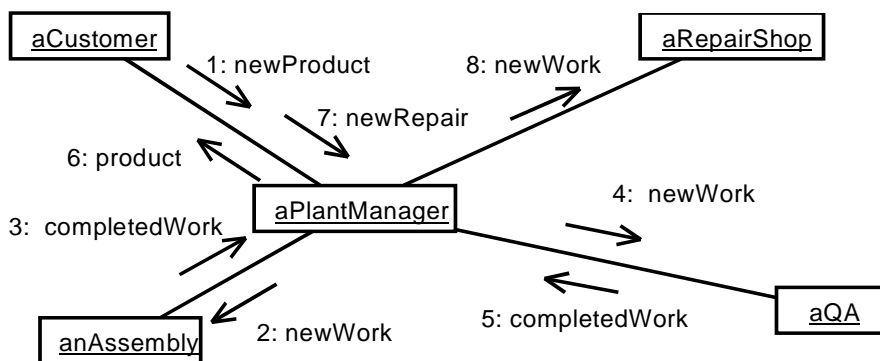


Figure 2: Collaboration Diagram

Role Model

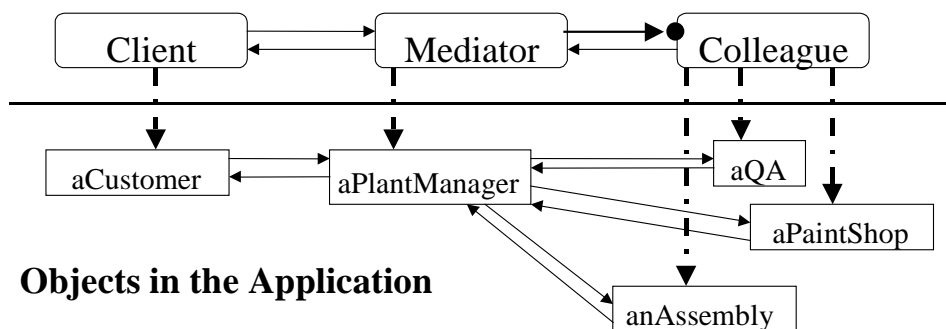


Figure 3: Role Model for the Mediator Pattern (based on [1] and [14])

The full semantics of the role model for the Mediator pattern can be detailed in additional views and notation provided in [1] and [11]. The important distinction between the collaboration diagram in Figure 2 and the role model in Figure 3 is that the role model is an abstraction; the collaboration diagram in Figure 2 is merely an instantiation of it.

1.3 Role Model Relationships

Once you have a base role model such as the one in the top half of Figure 3, you can build on it to form new models. One role model may be an aggregate of others. Also, a new role model may be derived from one or more base models, refining it; in this case, the derived role must be able to play all of the base roles. Synergy can make a combined role more than just the sum of the parts.

An example of this is the Bureaucracy pattern that is documented as a role model in [15]. This pattern features a chain of responsibility, a multilevel hierarchical organization, and centralized control at every level. This pattern can be constructed by bringing together the Composite, Mediator, Observer, and Chain of Responsibility patterns [4] [14, 15]. A simple combination of the four design patterns results in sixteen roles for the Bureaucracy pattern. However, there are in fact only five roles because the resulting compound pattern is more than a sum of the individual patterns. The resulting role model is shown in Figure 4. In the figure, a triangle indicates that one role refines another (the triangle points toward the more general role, as inheritance in UML).

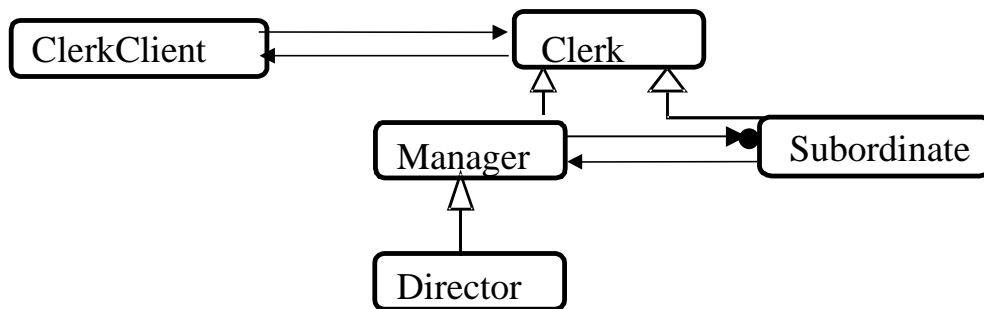


Figure 4: Role Model for the Bureaucracy Pattern [Riehle]

2.0 ROLES AND ASPECTS

Role modeling addresses software specification, analysis, and design. Use cases, the standard first step in requirements capture and analysis, focus on activities and interactions; they are also first class objects that can be generalized, refined, instantiated, and aggregated. Therefore, use cases map well to role models [5] [6], and role models are a logical second step in analysis. Once they have been formulated, role models provide specifications for the entities that play the various roles.

Many role models, including the two presented in section 2, are documented patterns, and they do have corresponding object oriented designs. However, a design and implementation that is solely class or component based has its problems and limitations. Role model implementations become unwieldy and difficult to manage, control, and debug when they are distributed across many components that are instances of different classes. Higher level, or alternative, language constructs are needed.

[8] presents activities as a language mechanism, mapping designs with activities onto high level object oriented constructs. The approach involves representing activities as relation classes, with member participants that are in fact roles. In [8], activities may be *initiating*, which means the activity is in charge and the participants must follow its instructions, or *reactive*, where the participants are in control. These seem to be analogous to centralized and decentralized control. An activity object's responsibilities

include monitoring progress, checking for the status of participants, synchronization, and similar tasks.

There are similarities between activity objects and the control objects introduced by Jacobson in his original work on use cases [5]. Jacobson proposes one control object per use case, which is an approach to centralized control that seems analogous to Kristensen's activity objects.

Aspects, like role models, are orthogonal to object or component definitions. Aspects cut across the units of a system's functional decomposition. Examples provided in the literature [7] are communication and coordination strategies, synchronization constraints, and exception handling. It appears that aspects, role models, and activities (per [8]), are very similar, and that Aspect Oriented Programming (AOP) can be used in a general sense to implement role models.

Role modeling and AOP are complementary, because role models can be used to specify and model aspects. Further, role models are "first class" objects, and this will allow aspect models to be abstracted and reused. This reuse may also carry over to the implementation, although this is obviously a question for further research.

3.0 AGENT ROLES AND ASPECTS

3.1 Overview

According to the FIPA standard, strong agents feature [3]:

- goal directed behavior
- autonomous determination of courses of action
- interaction by negotiation and delegation
- modeling of anthropomorphic mental attitudes, such as beliefs, intentions, desires, plans, and commitments
- flexibility in responding to situations and needs

Based on the FIPA definition, agents are proactive and autonomous objects with rich organizational or social behavior. At the lowest level, agent interaction follows a specified and structured protocol. But, more advanced social behavior in agents involves coordination and negotiation strategies, which may include strategies for exception handling and conflict resolution. Further, an agent can have knowledge of why other agents are related, such as the relevant collaborating goal(s) or task(s), and collaborating agents may share or exchange ontologies.

Role modeling and AOP offer a promising approach for agent analysis, design, and implementation because of the following:

1. *Social*: The emphasis is on social or interactive behavior.
2. *Proactive*: Roles in a role model work together to accomplish a goal
3. *Unified Model*: Agents, objects, and people can play roles.
4. *Design*: Role model synergy or synthesis may be valuable for agent design.
5. *Implementation*: AOP and activity- based implementations may facilitate agent implementation. Present implementations are very complex and unwieldy, due to the fact that agent systems are highly distributed and feature widespread interaction.
6. *Role Dynamics*: Agent organizations can take on various formations (role models)[2], and organizations, formations, and role assignments are dynamic [16].

7. *Documentation:* Role models provide documentation for agent frameworks and systems that is independent of the implementation. This helps to identify commonalities that can then be recognized as patterns.

3.2 Example

Research at BT has been identifying, consolidating, and documenting agent role models. The aim is to clarify the concept of an agent role, to identify patterns, and to determine what tools and techniques are needed. Many role models of documented design patterns are relevant; for example, Mediator, Broker, and Bureaucracy, among others, often appear in agent systems.

The example shown in Figures 5 and 6 is for a contract net [3]. Figure 5 provides the role model, while Figure 6 depicts the collaboration diagram. There are two high level roles: Contract Net (CN) Manager, and CN Participant. There is more than one CN Participant. Within the high level roles, there are role sequences (shown in Figure 5b); a role sequence is read from left to right, and the progression is indicated by a filled chevron. The CN Manager has the internal roles of a Publisher, a Proposal Manager, and a Client. Each CN Participant is a Subscriber, Bidder, and Server.

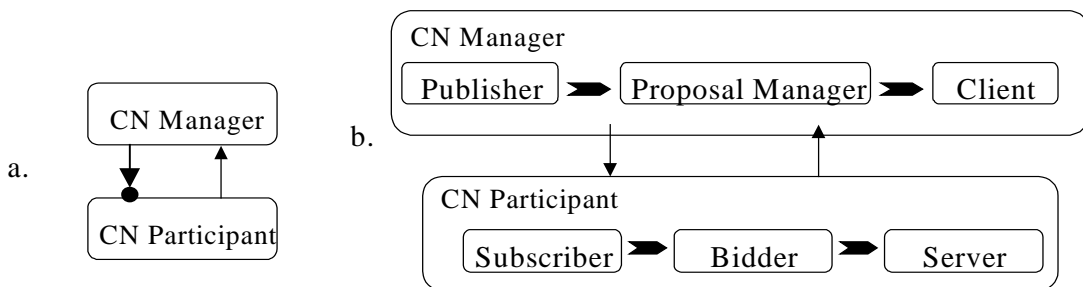


Figure 5: Role Model for a Contract Net

The collaboration diagram shows a message sequence for a contract net. In its Publisher role, the CN Manager releases a cfp (call for proposals). The Bidder considers it, and either formulates a bid or refuses the cfp. The rest of the sequence is as shown.

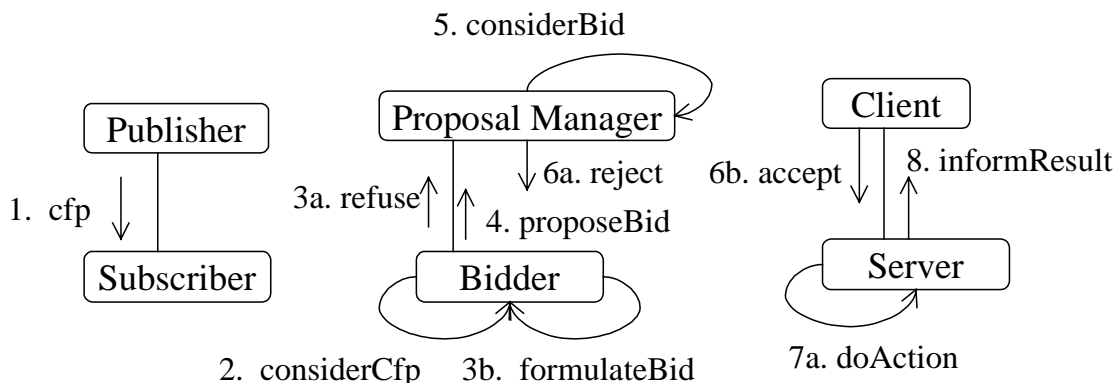


Figure 6: Collaboration Diagram for a Contract Net

4.0 SIGNIFICANCE AND SUMMARY

Any software system is comprised of patterns of interaction. A typical agent system, for example, will be comprised of many patterns of interaction; those shown in Figures 2 through 6 are a representative sample.

When a system is distributed and there is emphasis on collaborative or coordinated behavior (such as in found in agents), these patterns of interaction are of paramount importance. They capture the essential characteristics better than static relationships, such as containment and inheritance.

This paper discusses role modeling and aspect oriented programming as promising approaches to specifying, modeling, designing, and implementing distributed, collaborative systems. This work has just begun; it is anticipated that it will provide many benefits for agent based system analysis, design, and implementation

5.0 REFERENCES

1. Andersen, E. (Egil), *Conceptual Modeling of Objects: A Role Modeling Approach*, PhD Thesis, University of Oslo, 1997.
2. Barbuceanu, M., Gray, T., Mankovski, S., "Coordinating with Obligations," Agents '98, Minneapolis, May, 1998, p. 62 - 69.
3. Dickinson, I., "Agent Standards", Agent Technology Group, 1997. <http://drogo.csel.tstet.it/fipa>.
4. Gamma, E.R., R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.
5. Jacobson, I., et al., *Object Oriented Software Engineering: A Use Case Driven Approach*, Addison Wesley, 1992.
6. Kendall, E. A., U. Palanivelan, S. Kalikivayi, "Capturing and Structuring Goals: Analysis Patterns," EuroPlop'98, European Pattern Languages of Programming, Germany, July, 1998.
7. Kiczales, G., J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. - M. Loingtier, and J. Irwin, "Aspect Oriented Programming," Xerox Corporation, 1997. <http://www.parc.xerox.com/spl/projects/aop/>
8. Kristensen, B. C., D. C. M. May. "Activities: Abstractions for Collective Behavior". *Proceedings of the European Conference on Object-Oriented Programming (ECOOP'96)*, Linz, Austria, 1996.
9. Kristensen, B. B., Olsson, J., "Roles & Patterns in Analysis, Design and Implementation", OOIS'96, Proceedings of the 3rd International Conference on Object-Oriented Information Systems, London, England, 1996.
10. Kristensen, B. B., Osterbye, K., "Object-Oriented Modeling with Roles", OOIS'95, Proceedings of the 2nd International Conference on Object-Oriented Information Systems, Dublin, Ireland, 1995.
11. Reenskaug, T., Wold, P., Lehne, O. A., *Working with Objects, The OOram Software Engineering Method*, Manning Publications Co, Greenwich, 1996.
12. Reenskaug, T., "Role Modeling Enters the Main Stream," Object EXPERT, January, 1997.
13. Reenskaug, T., "The Four Faces of UML," www.ifi.uio.no/~trygve/documents/, May 18, 1998.
14. Riehle, D., "A Role-Based Design Pattern Catalog of Atomic and Composite Patterns Structured by Pattern Purpose", Ubilab Technical Report 97.1.1 Zurich, Switzerland: Union Bank of Switzerland, 1997. <http://www.riehle.org>.
15. Riehle, D., "Composite Design Patterns", OOPSLA '97, *Proceedings of the 1997 Conference on Object-Oriented Programming Systems, Languages and Applications*, ACM Press, Page 218-228, 1997.
16. Veloso, M., Stone, P., Han, K., "The CMUnited- 97 Robotic Soccer Team: Perception and Multiagent Control," Agents '98, Minneapolis, May, 1998, p. 78 - 86.