Analysis and Design of Multiagent Systems Using Hybrid Coordination Media

Scott A. DeLoach
Department of Computing and Information Sciences, Kansas State University
234 Nichols Hall, Manhattan, KS 66506 (785) 532-6350
sdeloach@cis.ksu.edu

ABSTRACT

Over the last few years, two advances in agent-oriented software engineering have had a significant impact. The first is the identification of interaction and coordination as the central focus of multiagent systems design and the second is the realization that the multiagent organization is distinct from the individual agents that populate the system. Also, the evolution of new, more powerful hybrid coordination models, which combine data-centered and control-centered coordination approaches, have given us the capability to model and implement the rules that govern organizations independently from the individual agents in the system. This paper investigates how to combine the power of these hybrid coordination capabilities with the concept of organizational rules using traditional conversation-based approaches to designing multiagent systems.

1. INTRODUCTION

Two recent advances in agent-oriented software engineering have had a significant impact on approaches to building multiagent systems. The first, identification of interaction and coordination as the central focus of multiagent systems design, lead to an initial focus on conversation-based agent oriented methodology such as COOL [1], MaSE [6], or MESSAGE [8]. However, recent advancements in coordination frameworks, such as *hybrid coordination media* [4], which combine data centered[1] and control centered[2] [5] approaches, have allowed us to perform more advanced forms of coordination than were possible with the previous conversation-based approaches. The second significant advancement impacting agent-oriented software engineering was the realization that the organization of a multiagent system is distinct from the individual agents that populate the system [15]. While the individual agents play roles *within* the organization, they do not constitute the organization. Organizations consist of *organizational structures* as well as *organizational rules* that define the requirements for the instantiation and operation of the organization as well as constraints on agent behaviors and interactions. Thus we can separate organizational responsibilities from agent responsibilities; the organization, not the individual agents, should be responsible for enforcing the organizational rules. Hybrid coordination models allow organizational rules to be embedded in the media instead of in the agents themselves [3].

While these advances are fairly recent, some initial suggestions on how to incorporate these concepts into current multiagent systems methodologies have been put forth. For instance, there is a proposal to modify the Gaia methodology to incorporate the notion of social laws and coordination media [14]. Unfortunately, these proposals are still at a high level and do not provide concrete guidance on how to use existing analysis and design abstractions with advanced coordination models and organizational concepts. Therefore, the goal of this paper is to provide detailed direction on integration of advanced coordination models and organizational rules into existing multiagent methodologies. Specifically, we consider current analysis and design models to investigate how to integrate the existing abstractions of goals, roles, tasks, agents, and conversations with organizational rules and advanced coordination models. We also look at how to use advanced coordination models to implement the resulting designs.

Extending existing conversation-based multiagent analysis and design approaches with these additional concepts is an important first step toward taking advantage of these advanced coordination models. While one might be tempted to simply throw out the concept of conversations altogether, we resist that urge for two reasons. First, conversation-based approaches are widely understood and provide an easily understandable metaphor for agent-to-agent communication. Second, conversation-based approaches have shown that they are verifiable and give designers some measure of system coherence [7]. Using the full power of hybrid coordination media without a solid engineering approach could lead to multiagent system designs that are not comprehensible, verifiable, or coherent. In particular, we will focus on modeling such capabilities using the Multiagent Systems Engineering (MaSE) methodology [6], as it provides a rich and detailed modeling language for the analysis and design of conversation-based multiagent systems.

In Section 2, we discuss the current state of the art in hybrid coordination media and provide a taxonomy of possible uses of their reactive capabilities in multiagent systems. Next, in Section 3, we demonstrate how advanced coordination models and organizational rules affect the MaSE analysis and design phases. We (1) analyze the analysis phase where we add the notion of organizational rules to the existing analysis models, (2) show how to map the various analysis artifacts, including organizational rules, into an enhanced design model, and (3)) show how environmental tasks can be mapped into reactions in hybrid coordination media. We end with a discussion of our results and conclusions in Section 4.

2. HYBRID COORDINATION MEDIA

As discussed above, hybrid coordination media have allowed us to focus on advanced forms of coordination as well the separation of organizational responsibilities from agent responsibilities. The hybrid coordination model is defined in terms of a *tuple space*, which is a shared repository of tuples (as shown in Figure 1), where agents can read and write tuples to help perform inter-agent coordination. While this is a simple model, it is very powerful and can be used, among other things, to implement simple message passing as well as more complex coordination patterns such as broadcasting or multiagent conversations, where all the agents involved in the conversation can see and respond to messages sent through the tuple space. Other advanced methods include *overhearing*, where one agent listens in on other agent's conversations and offers help when possible [2], and *observations*, in which the system informs an agent when information of interest to that agent becomes available [12].

[1] In data-centered approaches, entities interact by exchanging data via a shared data space, which determines how data is stored, accessed and consumed, but has no control over the coordinating entities.

[2] In control-centered approaches, entities interact with each other over specified ports using well-define protocols that are governed by the coordination media, such as in conversation-based systems.

Hybrid coordination models also include *reactions*, which allow additional, control-centered, supervision of the coordination process. A reaction is simply a function that, upon seeing a tuple that matches some predefined pattern, performs some action on the tuple space. These actions can inform agents that a specific type of tuple has been placed in the tuple space, perform processing based on a tuple, extract tuples, modify tuples, etc. Using reactions, the coordination media itself can play a part in determining how the tuples should be used. Examples of such hybrid coordination media include MARS [3], TuCSoN [10], and LIME [9]. Hybrid media have many advantages over typical control driven models including location independence, notifying agents when data changes occur, separating application specific roles from organization specific roles, and simplifying implementation of organizational and security rules. Although they provide a central repository logically, many hybrid coordination frameworks actually work in a distributed manner and are appropriate for multiagent systems.

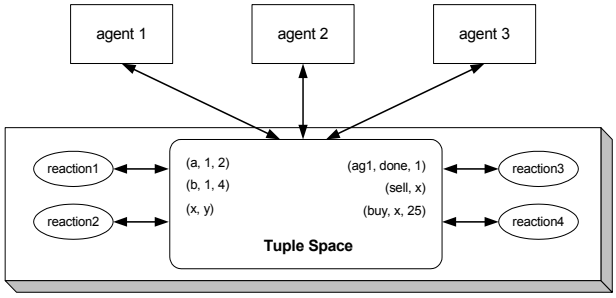

Figure 1. Reactive tuple space environment

Taxonomy of reaction types

Hybrid coordination media reactions (or *environmentally based reactions*) have been used for many purposes including implementing social tasks, implementing security policies, controlling membership, restricting selfish behaviors, modifying data formats, and enforcing social norms and conventions. In this section we describe a taxonomy of the of reactions types that are useful in multiagent systems. We base our taxonomy on the way reactions are modeled during analysis and design. In MaSE, the use of a role is central to the analysis and design and it seems clear that reactions should play some type of role within the system. Therefore, we begin our categorization of reactions based on whether they play explicit or implicit *roles* in the system. Figure 2 shows our taxonomy of environmentally based reactions. An *explicit* role is a role that exists in the system role model, fulfills a goal of the system, and interacts overtly with other system roles typically implemented as agents. There are two types of reactions that can be characterized as implementing explicit roles: reactive tasks and social tasks. A *reactive task* works on the behalf of an agent (or agents) by monitoring the environment tuple space for changes (additions, deletions, modifications) and notifying the agents when a specific change occurs. The other type of explicit role is a *social task*, which is a task that helps coordinate a group of agents in achieving a task that is not assigned to any single agent.

An example of a reactive task is a reaction that monitors the advertisement of items for sale at an auction. When a tuple is placed in the tuple space that matches its criteria, the reactive task would notify its agent and then go back to monitoring. In this example, an agent would typically create the reaction and would expect to be notified when specific changes occur. An interesting social task illustration is described in [3] where a reaction is assigned the responsibility of carrying out auctions in a multiagent system. In the example, agents place items up for sale by placing an appropriate tuple into the tuple space. At that point, a reaction is executed that takes the responsibility for collecting bids from bidder agents, selecting a winner, and notifying the seller as well as the winning and losing bidders of the results.

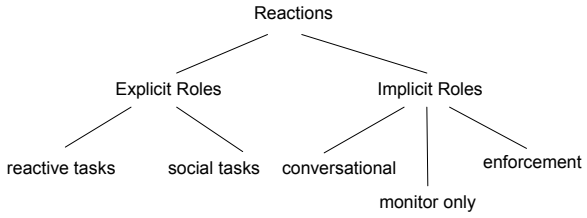

Figure 2. Taxonomy of environment-based reactive tasks

The second types of roles in our taxonomy are implicit roles. *Implicit roles* are roles or tasks that lie outside the main purpose of the multiagent system and are not typically associated with goals or roles in the system role model. Implicit roles are often used to ensure compliance with security or organizational rules, or as a means of monitoring system operation. We have discovered three types of implicit roles: conversational tasks, monitoring tasks, and enforcement tasks. *Conversational* tasks are reactions that communicate directly with agents in the system. They are different from reactive or social tasks in that they do provide direct functionality to the system; however, they can "interrupt" the natural flow of the system and take corrective action if needed. For instance, we might have a reaction that monitors negotiations between agents in the system for violations of the rule stating that all sellers must actually own the resources they are attempting to sell. If the reaction determined that a seller was attempting to violate the rule, it could notify both the seller and the buyer that something was amiss, thus interrupting the negotiation process. A *monitoring* task could be used in much the same way, only without directly intervening in the negotiation process. Generally, monitoring tasks are used to track the state of the system over time or to inform users of problems during system operations and are useful for debugging as well as machine learning. Finally, an *enforcement* task is a task that monitors the environment and can take corrective action, but does not directly intervene with the agents. Such modes of intervention might include modifying tuples, disconnecting agent's access to communication or resources, or even terminating agents. An example of an enforcement task would be a reaction that monitored the environment for agents attempting to sell "stolen" resources. Instead of directly dealing with the seller or bidders, it might simply delete such items from the environment, thus enforcing the organizational rule.

3. MODELING REACTIVE BEHAVIOR

In this section we show how to use the artifacts generated in the analysis phase of the MaSE methodology, enhanced with organizational rules, to generate designs that take advantage of the capabilities of hybrid coordination media. We discuss the MaSE analysis artifacts, look at designs choices, and consider thier implementation in the following subsections.

Throughout this paper, we will use the conference management example as defined in [15]. The conference management system is an open multiagent system supporting the management of various sized international conferences that require the coordination of several individuals and groups. There are five distinct phases in which the system must operate: submission, review, decision, final paper collection. During the submission phase, authors should be notified of paper receipt and given a paper submission number. After the deadline for submissions has passed, the program committee (PC) has to review the papers by either contacting referees and asking them to review a number of the papers, or reviewing them

themselves. After the reviews are complete, a decision on accepting or rejecting each paper must be made after which authors are notified of the decisions and are asked to produce a final version if their paper was accepted. Finally, all final copies are collected and printed in the proceedings. The conference management system consists of an organization whose members may change at each stage of the process. Also, since each agent is associated with a particular person, it is possible to imagine that the agents could be coerced into displaying opportunistic behaviors that would benefit their owner to the detriment of others. Such behaviors could include reviewing ones own paper or unfair allocation of work between reviewers, etc.

The analysis phase

Based on their definition above, we would expect that the explicit roles (both reactive tasks and social tasks) to appear as roles defined in the analysis phase. In fact, this is the case by definition. Since an explicit role performs part of the system function, it must appear as a role in the analysis phase role model. Therefore, this will not require changes to the analysis phase of any role-based agent oriented methodology such as MaSE, Gaia [13], MESSAGE [8], or SODA [11]. Almost any role identified in the analysis phase could become a reactive or social task and implemented as a reaction. The decision allocating roles to reactions is made during the design phase.

Figure 3 shows a MaSE role model depicting the relationships between the roles in the conference management system. In the diagram, a box denotes each role while a directed arrow represents a communication protocol between roles, with the arrows pointing away from the initiator to the responder. Notice that while we referred to the PC chair and PC members in the problem description, we have intentionally abstracted out the roles played by those typical positions into partitioning, assigning reviews, reviewing papers, collecting reviews, and making the final decision. As we will see later, this will provide us flexibility in the design phase. Basically, the system starts by having authors submit papers to a paper database (PaperDB) role, which is responsible for collecting the papers, along with their abstracts, and providing copies to reviewers when requested. Once the deadline has past for submissions, the person responsible portioning the entire set of papers into groups to be reviewed (the Partitioner role) asks the PaperDB role to provide it the abstracts of all papers. The Partitioner partitions the papers and assigns them to a person (the Assigner) who is responsible for finding *n* reviewers for each paper. Once assigned a paper to review, a Reviewer requests the actual paper from the PaperDB, prepares a review, and submits the review to the Collector. Once all (or enough) of the reviews are complete, the Decision Maker determines which papers should be accepted and notifies the authors.

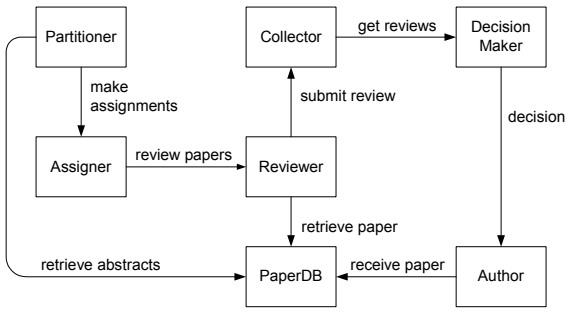

Figure 3. Role model for conference management system

Thus, for the conference management system we have identified seven explicit roles. However, in MaSE, we do not stop at simply identifying the roles; we also identify the tasks that the roles must perform. Therefore, a more detailed version of the conference management system role model is shown in Figure 4. Basically, we have extended the role model by adding the tasks (shown using ellipses attached to roles). Generally, each role performs a single task, whose definition is straightforward and documented in a concurrent task diagram (not included here for simplicity). However, some roles, such as the Paper DB or Reviewer role are more complex and require multiple tasks. For instance, the Paper DB role has three tasks: Collect Papers, Distribute Papers, and Get Abstracts. The Collect Papers tasks must accept papers and ensure they are in the right format and meet all the eligibility requirements while the Get Abstracts task must extract the abstract from submitted papers and send them to Partitioner. The Distribute Papers task simply distributes accepted papers to the appropriate Reviewers when requested.

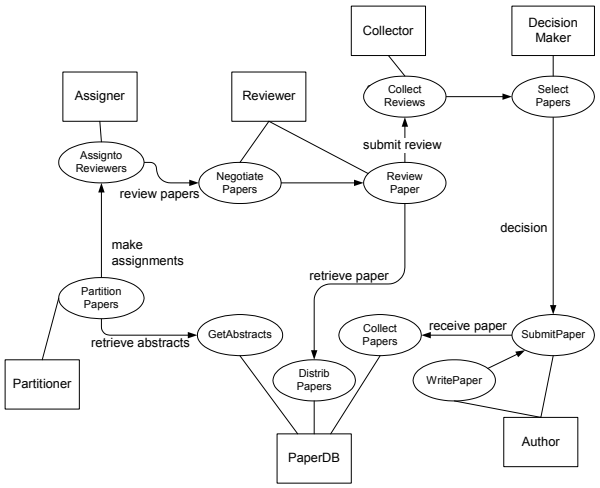

Figure 4. Expanded MaSE role model

While one of the main focuses in the analysis phase is the identification of roles and their associated tasks, we must also identify other requirements that are not tied explicitly to roles. These are often captured as organizational rules that correspond to implicit system roles. Several organizational rules have been identified for the conference paper review system in [15] as shown below, (temporal operators are defined in Table 1).

1. $\forall p : \#(reviewer(p)) \geq 3$
2. $\forall i, p : Plays(i, reviewer(p)) \Rightarrow \bigcirc\Box \neg Plays(i, reviewer(p))$
3. $\forall i, p : Plays(i, author(p)) \Rightarrow \Box \neg Plays(i, reviewer(p))$
4. $\forall i, p : Plays(i, author(p)) \Rightarrow \Box \neg Plays(i, collector(p))$
5. $\forall i, p : participate(i, receivePaper(p)) \Rightarrow \Diamond initiate(i, submitReview(p))$
6. $\forall i, p : participate(i, receivePaper(p)) \mathcal{B} initiate(i, submitReview(p))$
7. $\forall p : [submittedReviews(p) > 2] \mathcal{B} initiate(chair, decision(p))$

Basically, the first rule states that there must be at least three reviewers for each paper (# is the cardinality operator) while rule two keeps a reviewer from reviewing the same paper more than once. Rules three and four try to limit selfish agent behavior by ensuring that a paper author does not review or collect reviews of his or her own paper. The last three rules describe appropriate system operation. Rule five states that if a paper is received, it should eventually be reviewed. Rule six requires that a paper must actually be received before a review can be submitted on it while rule seven requires that there be at least two reviews before a paper can be accepted or rejected.

During the analysis phase, these organizational rules are collected; however, they are integrated into the design during the design phase. It is at this point that the designer must decide how to monitor or enforce these rules. As we will see, the rules can be assigned to a particular agent in the design or they can be implemented via conversational, monitoring, or enforcement tasks as part of the environment.

Table 1. Temporal Operators

○ φ	φ is true next
□ φ	φ is always true
◇ φ	φ is eventually true
φ $\mathcal{B}$ ϕ	φ is true before ϕ is true

The design phase

In this section, we take our analysis and show how it can be used to develop a number of different designs using reactions. The goal is to show different options that are available with the environmentally based reactions, not to advocate a particular approach as being necessarily better in all instances.

The initial step in the design phase is to define agents from the roles identified in the analysis phase; the end product is an Agent Class Diagram, which depicts the system organization as defined by agent classes and conversations between them. An *agent class* is a template for a type of agent in the system and is analogous to an object class in object-orientation. Agent classes are defined in terms of the roles they play and the conversations in which they participate. In an agent class diagram, boxes denote agent types (with the roles it plays listed under its name) while directed arrows represent conversations between agent types.

Design 1 – Traditional. Traditional multiagent design approaches, such as advocated in [6], might result in the design shown in Figure 5, which is appropriate for a small conference. In this design, various roles are combined into agents in the system. For instance, the PC Chair agent plays the Partitioner, Collector, and Decision Maker roles while the PC Member agent plays both the assigner and reviewer roles. Outside of author agents, the only other agent is the DB agent, which provides an interface to the database containing the papers, abstracts, and author information, etc.

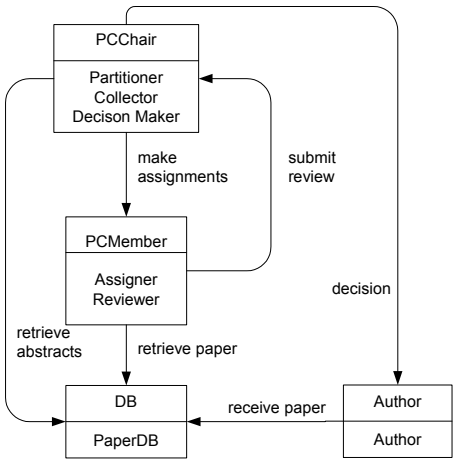

Figure 5. Traditional design

Unfortunately, the traditional multiagent design described above does not separate agent tasks from social tasks, which is desirable for extensible, open multiagent systems [5]. To ensure the enforcement of organizational rules, we must interleave the rules into the individual agents themselves. For example, the only place to ensure that at least two reviews were completed before the decision to accept or reject a paper was made (rule 7) is in the PC Chair agent itself. This forces us to rely on self-policing agents, which, in the presence of self-interested agents, is a less than desirable approach.

Design 2 – Explicit roles. As advocated by some [5], the appropriate place to monitor and enforce organizational rules is in the organization itself. Thus, using the same analysis, we created a new design using environment-based reactions for the PaperDB and Collector roles, as shown in Figure 6. In effect, their tasks become part of the organization and implemented as reactions in a hybrid coordination media. However, we should point out that a hybrid coordination media is not required to take advantage of this design approach. While not as efficient and requiring additional overhead, this design could also be implemented using traditional message oriented techniques. One approach would be an "environment" agent to handle all tasks normally assigned to hybrid coordination media reactions. Even though less elegant, the advantages of separating social tasks and organizational rules from the agents would remain.

By being assigned to reactions, the Get Abstracts, Distribute Papers, Collect Papers, and Collect Reviews tasks can more easily support the conference management organizational rules. This is because the information collected and used by these tasks can easily be shared through the tuple space (or common database). For instance, The Distribute Papers task can enforce rule 3 (an author cannot review his or her own paper) by simply checking the reviewer against the paper author. Likewise, the Collect Reviews task can monitor rule 5 (if a reviewer receives a paper, he or she must eventually submit a review) and send warnings if reviews are not submitted in a timely fashion. The role can also enforce rule 6 (the paper must be received by a reviewer before the review is submitted) by not accepting reviews until the paper has actually been requested, as well as rule 7 (there must be at least two reviews before the chair can make a decision) by only sending reviews once there are at least two of them. This design approach also allows the organizational rules to be updated without necessarily affecting individual agent designs.

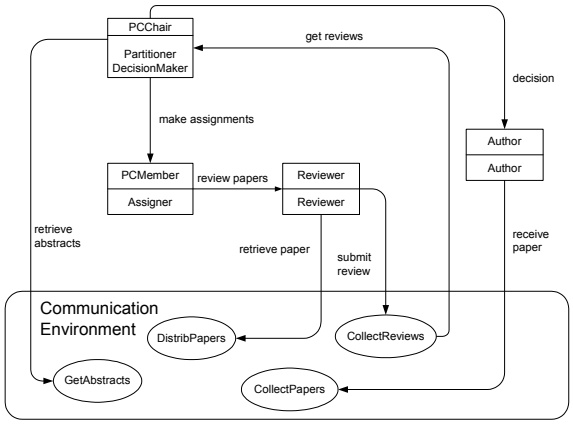

Figure 6. Design with explicit reactions

Design 3 – implicit roles. A third design that does not assign explicit roles from the role model to the environment is shown in Figure 7. In this design, we only use implicit roles to monitor and enforce the organizational rules. For instance, in Figure 7, there are three environmental tasks (Monitor Num Reviews, Monitor Decisions, and Monitor Reviewers) that monitor or enforce organizational rules 2, 3, and 7. The dashed line between the reactions and the conversations denote that the reactions *monitor* those conversations by executing when tuples that start the conversations are placed into the tuple space. In the case of monitoring-only reactions, the reactions simply monitor the tuples and either display or log the information of interest. For instance, the Monitor Decision task might simply

monitor the *decision* conversations and log only those decisions that are made without the required number of reviews, which it would have access to via tuples shared by the Monitor Num Reviews reaction.

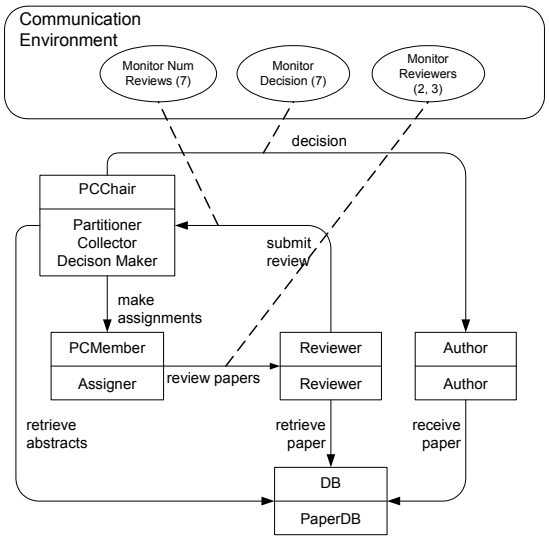

Figure 7. Design with monitoring/conversational reactions

An example of a monitoring-only reaction, Monitor Decision, is shown in Figure 8. Semantically, we assume that reactions receive messages before agents and must pass the message along to its intended recipient. In Figure 8 this is shown by the *receive* event that initiates the transition from the start state. Once the message is received, the Monitor Decision reaction validates it (in this case, that it has at least two reviews) and, if valid, passes the message along to the intended recipient. However, if the decision is not valid, the decision is simply logged and the message is passed along anyway.

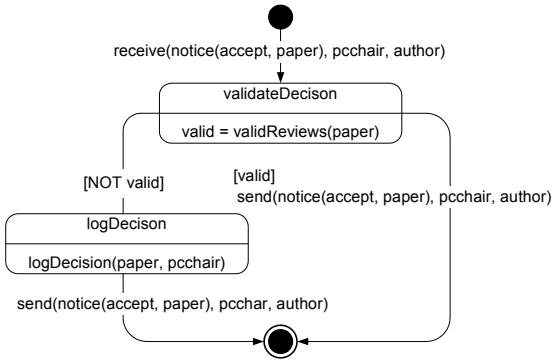

Figure 8. Monitor Decision reaction (monitor only)

We can use the same basic design as shown in Figure 7 but use either conversational or enforcement reactions as opposed to monitoring-only reactions. Basically, the difference is that conversational or enforcement reactions actually provide correction either directly or indirectly with the offending agents. For example, Figure 9 shows a conversational reaction that actually intercepts the *notice* message being sent to an author and, if the correct number of reviews have not been accomplished, sends the PC Chair a message stating that the decision was invalid instead of forwarding the notice message on to the author.

Of course, a conversational reaction requires that the agent with which it wishes to converse must be able to handle the communications. Thus, the original *decision* conversation (from the viewpoint of the PC Chair) must be modified to work with a conversational reaction. Specifically, the PC Chair's side of the conversation must be able to handle an *invalidDecision* message from the environment. Thus, in Figure 10, we have modified the conversation to accept the *invalidDecision* after sending the original notice. This is an example of the strength of using a conversation based design approach. Using conversations, it is possible to trace the exact sequence of possible message through the system and thus automatically verify that all conversations and reactions are consistent and do not cause unwanted side effects such as deadlock.

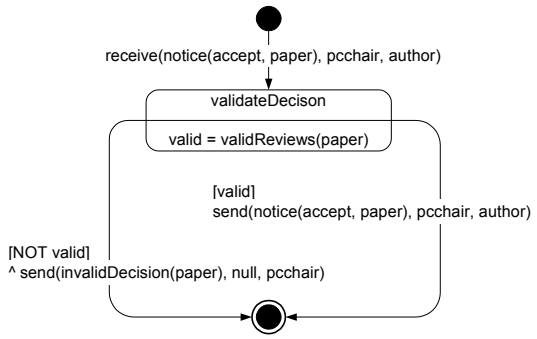

Figure 9. Monitor Decision reaction (conversational)

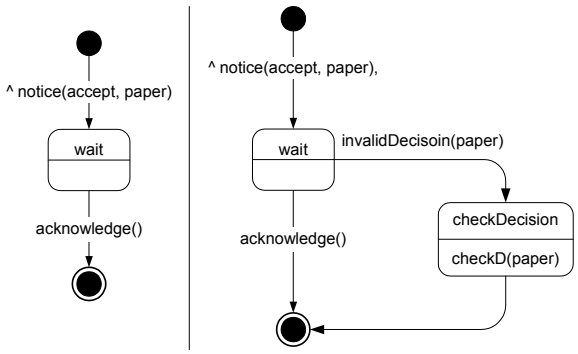

Figure 10. Decision conversation (original & modified)

Implementation

As implied above, reactive tasks are ideally implemented in hybrid coordination media as reactions. When a new tuple appears matching a predefined pattern, the reaction is invoked and, if other application specific criteria are met, the reaction notifies its agent. Because reactive tasks belong to specific agents, the agents themselves are responsible for placing the reactions into the environment.

Social tasks can also be easily defined as reactions in hybrid coordination media. The example of social task of performing an auction is a good example [3]. However, in case of social tasks, the reaction is not under control of an individual agent, but is part of the organization itself and thus is started at system initialisation. In the auction example, the reaction waits until it sees an appropriate tuple from a selling agent in the environment. Such a tuple might include the type of auction to use, the criteria for selecting a winner, and a minimum acceptable price. In this case, it is possible to have multiple reactions in the environment that implement different types of auctions (English, Dutch, etc.). We also foresee the use of social tasks in controlling the introduction of new agents in the system. If a new agent puts a tuple into the tuple space, an organizational reaction can verify certain properties about the agent before allowing it to join the system.

While we view conversational tasks as intercepting messages and forwarding them on if they are valid, the implementation in a traditional hybrid tuple space coordination media would be

quite different. In such an implementation, reactions would be executed when the appropriate tuple representing the message was placed in the tuple space. The reaction would then check the tuple and extract it if it was not valid and communicate back with the initiating agent by placing a new tuple in the tuple space.

A monitoring task could also be implemented as a reaction defined by the organization. However, instead of interacting with other agents, it would simply save the information it gleaned from the state of the tuple space to some data store or display. Monitor tasks could be used to monitor specific types of tuples, tuples from specific agents, or even all tuples.

The implementation of enforcement reactions is very similar to conversational reactions. However, instead of placing tuples in the tuple space as a means of communicating with an agent, the reaction simply modifies the tuple space to enforce the rule for which it was designed. For example, if authors are required to submit papers in PDF format, we could enforce this rule via an enforcement reaction that would automatically convert non-PDF formats to PDF; the reaction would simply extract any non-conforming paper tuples and replace them with the appropriate PDF version.

4. RESULTS AND CONCLUSIONS

The goal of this paper was to propose some initial approaches toward integrating organizational rules and advanced coordination models with more traditional, conversation-based agent-oriented software engineering models. To accomplish this task, we extended MaSE modeling notations to include organizational rules and the ability to explicitly model environment based tasks that can interact with other agents or simply monitor conversations. In all, we identified five types of tasks that that take advantage of the capabilities provided by advanced coordination media: reactive tasks, social tasks, conversational tasks, monitoring tasks, and enforcement tasks. While we do not believe this list of tasks to be an exhaustive, it provided us with the necessary concepts to model organizational rules. While the extensions to the MaSE models are a good start, we certainly see the need for additional constructs to be able to model more advanced coordination techniques such as multi-way conversations and the "overhearing" of other agent's conversations.

While the originally intended for the analysis and design of closed multiagent systems, the incorporation of organizational rules makes MaSE useful in the analysis and design of open systems as well. Organizational rules allow designers to specify how new agents can enter the system as well as the rules they must abide by once they are part of the system. While using a conversation-based approach still requires reliance on specific coordination protocols, designers no longer have to depend on self-policing agents (i.e., incorporating the organizational rules into the agents themselves) to ensure observance with organization rules. Social, monitoring, and enforcement tasks implemented via advanced coordination media provide the organization itself the mechanisms to allow new agents into the system, monitor their behavior, and guarantee compliance with organizational rules and protocols.

5. REFERENCES

[1] Barbuceanu, M., and Fox M. COOL: A Language for Describing Coordination in Multi Agent Systems. Proceedings of the First International Conference on Multi-Agent Systems, pp. 17-25, AAA Press, June 1995.

[2] Busetta, P., Serafini, L., Singh, D., and Zini, F. Extending Multi-Agent Cooperation by Overhearing. Proceedings of the Ninth International Conference on Cooperative Information Systems (CoopIS 2001), Trento, Italy, Sept 2001

[3] Cabri, G., Leonardi, L., and Zambonelli, F. Implementing Agent Auctions using MARS. Technical Report MOSAICO/MO/98/001.

[4] Ciancarini, P., Coordination models and languages as software integrators, ACM Computing Surveys 28 (1996), pp. 300--302.

[5] Ciancarini, P., Omicini, A., and Zambonelli, F. Multiagent System Engineering: the Coordination Viewpoint. Intelligent Agents VI. Agent Theories, Architectures, and Languages, 6th International Workshop (ATAL'99), Orlando (FL), May 1999, Proceedings. LNAI 1757, Springer-Verlag, 2000.

[6] DeLoach, S.A., Wood, M.F., and Sparkman, C.H. Multiagent Systems Engineering, The International Journal of Software Engineering and Knowledge Engineering, Volume 11 no. 3, June 2001.

[7] Lacey, T.H., and DeLoach, S.A. Automatic Verification of Multiagent Conversations. in Proceedings of the Eleventh Annual Midwest Artificial Intelligence and Cognitive Science Conference, pp. 93-100, AAAI Press, Fayetteville, Arkansas, April 2000.

[8] MESSAGE: Methodology for Engineering Systems of Software Agents. Deliverable 1. Initial Methodology. July 2000. EURESCOM Project P907-GI.

[9] Murphy, A.L., Pietro Picco, G., and Roman, G.C. LIME: A Middleware for Physical and Logical Mobility. In Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21), Phoenix, AZ, USA, April 16-19 2001, pp. 524-233.

[10] Omicini, A., Denti, E. From Tuple Spaces to Tuple Centres. Science of Computer Programming 41(3). Elsevier Science B. V., November 2001.

[11] Omicini, A. SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems. Agent-Oriented Software Engineering, 1st International Workshop (AOSE 2000), Limerick (Ireland), June 2000, Revised Papers. LNCS 1957, Springer-Verlag, 2001.

[12] Viroli, M., Moro, G., and Omicini, A. On Observation as a Coordination Paradigm: Ontology and a Formal Framework. 16th ACM Symposium on Applied Computing (SAC 2001), Las Vegas (NV), March 2001.

[13] Wooldridge, M., Jennings, N.R., and Kinny, D. The Gaia Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and Multi-Agent Systems. Volume 3(3), 2000.

[14] Zambonelli, F., Jennings, N.R., Omicini, A., and Wooldridge M.J. Agent-Oriented Software Engineering for Internet Applications. Coordination of Internet Agents: Models, Technologies, and Applications, Chapter 13. Springer-Verlag, March 2001.

[15] Zambonelli, F., Jennings, N.R., and Wooldridge, M.J. Organisational Rules as an Abstraction for the Analysis and Design of Multi-Agent Systems. International Journal of Software Engineering and Knowledge Engineering. Volume 11, Number 3, June 2001. Pages 303-328.