An Agent-Environment Interaction Model

Scott A. DeLoach and Jorge L. Valenzuela

Department of Computing and Information Sciences, Kansas State University
234 Nichols Hall, Manhattan, KS 66506
{sdeloach, jvalenzu}@cis.ksu.edu

Abstract. This paper develops a model for precisely defining how an agent interacts with objects in its environment through the use of its capabilities. Capabilities are recursively defined in terms of lower-level capabilities and actions, which represent atomic interactions with the environment. Actions are used to represent both sensors and effectors. The paper shows how the model can be used to represent both software and physical agents and their capabilities. The paper also shows how the model can be integrated into the Organization-based Multiagent Systems Engineering methodology.

1 Introduction

There is widespread agreement that the environment in which a multiagent system is situated is of fundamental importance in the analysis, design, and operation of the system. However, even with this agreement, few multiagent methodologies include the modeling of the environment or the agent's interactions with it as first class entities [10]. In situated multiagent systems, the *environment* is the entity in which agents exist and communicate [6]. Communication is a critical factor that enables agents to interact and coordinate. Typically, this interaction and coordination is modeled using direct communication through the social environment; however, it can also be modeling indirectly through the physical environment. A *social environment* is the entity that provides the principles, processes and structures that enable the agents to communicate while the *physical environment* provides principles and processes that affect objects within an environment [6]. In [4], Ferber defines a multiagent system as having six basic entities:

- An environment, E
- A set of objects, O, that exist in E
- A set of agents, A, which are active objects (i.e., a subset of O)
- A set of relations, R, that define relationships between objects in O
- A set of operations, O, that agents can use to sense and affect objects in O
- A set of universal laws that define the reaction of the environment to agent operations

Based on Ferber's definition, we have identified *five requirements* for specifying agent-environment interaction model. Essentially, an AEI should define:

L. Padgham and F. Zambonelli (Eds.): AOSE 2006, LNCS 4405, pp. 1–18, 2007.

1. A unique entity called the environment
2. The set of objects in the environment (which includes agents)
3. Specific types of relations that may exist between objects in the environment
4. The set of operations that agents may perform upon objects in the environment
5. The laws that govern the effect of those operations on objects in the environment

While capturing these elements is essential, we believe it is also critical that these concepts be captured using a model that shows direct relations between the objects, agents, and actions as well as specifies the intended effect of each action unambiguously. We believe it is also important to provide a model that allows these concepts to be specified and viewed at the appropriate level of abstraction.

While most current multiagent methodologies provide some notion of the environment or the agent's interactions with it, no major methodologies possess a detailed agent-environment interaction model that explicitly defines how the environment is affected by agents or how the agent perceives the environment. Including such an agent-environment interaction model is important because it allows us to explicitly identify (1) how agents directly interact/coordinate with each other, (2) how agents indirectly interact/coordinate with each other, and (3) the effect of agents on objects in the environment, which in situated multiagent systems often determines whether the system has accomplished its goals. In addition, agents in situated multiagent systems also generally require some representation of the environment in order to effectively communicate with other agents and to achieve their goals. By including a well-defined model of the environment in the agent-environment interaction model, the analysis, and design of these agents should be clearer and thus improved over implicit approaches.

The goal of this paper is to present an Agent-Environment Interaction model (AEI) that can be integrated into appropriate multiagent systems methodologies. Specifically, we will integrate the AEI Model into the Organization-based Multiagent Systems Engineering methodology (O-MaSE) [1]. To make the notation as clear and unambiguous as possible, we use standard UML notation with liberal use of keywords to denote specific concepts in the model. Obviously, if our AEI Model is integrated into other methodologies and modeling approaches, the notation can be adapted as needed.

The paper is organized as follow. In Section 2, we discuss how some current multiagent methodologies address environmental issues and provide an overview of O-MaSE [1]. In Section 3, we present our AEI Model and integrate our AEI Model into O-MaSE. In Section 4, we present a detailed example of the AEI Model using a robotics Weapons of Mass Destruction (WMD) simulation system. Finally, in Section 5, we present our conclusions and areas for future work.

2 Related Work

In this section we review four prominent multiagent systems methodologies and how they model interactions with the environment: Gaia, Message, Prometheus,

and O-MaSE. We also analyze how well each of these methodologies meets the agent environment interaction requirements stated above.

2.1 Gaia

The extended version of Gaia [12] adds some basic concepts and organizational abstractions to the original version of GAIA [11]. Among these additions is an Environment Model, which is introduced during the analysis phase. Because the authors believe that "it is difficult to provide a general modeling abstraction and general modeling techniques because the environment for different applications can be very different in nature" [12], they model environmental entities in terms of abstract computational resources. These resources are modeled as tuples that the agents may read, (sense), effect, (change), or consume, (remove). Thus the Gaia Environment Model can be viewed as a list of resources that can be accessed using an associated name and acted upon based on the type of action associated with them. An example of a Gaia Environment Model is shown below [12].

```
reads var1 // readable resource of the environment.
var2 // another readable resource.
change var3 // a variable that can be also changed by the agent.
```

Analyzing the Gaia Environment Model using our five AEI model requirements shows that, while it does include a limited notion of objects, it does not include any notion of agents (requirement 2). In addition, the Gaia Environment Model severely limits the types of relations (requirement 3) and actions (requirement 4) that can be performed on those objects. Finally, the Environment Model has no notion of environmental laws that affect the environment objects independently of the agents (requirement 5). A more general notion of an AEI could be of benefit in the Gaia methodology.

2.2 MESSAGE

In the MESSAGE methodology [5], the MESSAGE modeling language defines some knowledge-level-concepts like *Concrete-Entity*, *Activity*, and *MentalStateEntity*. One of the concrete entities defined is *Agents*, which are autonomous entities that can perform actions that affect resources. The *Actions/Activities* are concrete entities and include *Tasks* and *Interaction Protocols*. Agents can also perceive information entries that describe the state of a resource. Another concrete entity is a *Resource*, which represents a non-autonomous entity that agents can access/use.MESSAGE builds five views of the Analysis Model: Organization, Goal/Task, Agent/Role, Interaction and Domain views. The *Organization view* shows the concrete entities in the system, the environment and the relationship among them.

Based on our requirements, we see that MESSAGE defines elements of its environment as containing objects (both agents and resources) that can interact using actions and messages. However, MESSAGE does not include the notion of environmental laws that affect the objects in the environment (requirement 5).

Even though MESSAGE captures most of the required information, it does not explicitly define an agent-environment interaction model and does not provide a flexible way to represent or define actions at an appropriate level of abstraction.

2.3 Prometheus

The aim of the Prometheus System Identification Phase is to identify the basic functionality of the system along with the inputs, outputs, and important data structures [7]. Prometheus models these inputs as *percepts* and defines them as raw data coming from the environment. Outputs are modeled as *actions*, which are defined as the agent's way to modify the environment. Scenarios are used in Prometheus to describe how the system operates nominally. Each scenario consists of a set of steps that can include goals, actions, percepts, scenarios, or "other" for special types of steps.

The architectural design phase focuses on identifying the agents in the system and their interaction. Once the agents are identified, the next step is to define the percepts each agent reacts to and the actions it may perform. Agent interaction is specified by defining messages and the different repositories to be used. All these items are depicted in the system overview diagram. The Detailed Design Phase focuses on defining the capabilities, which are defined in terms of internal events, plans, and detailed data structures of the agents. Each capability is described by a descriptor, which includes the definition of its percepts, actions, data read or written, interaction with other capabilities, and sub-capabilities.

Our analysis reveals that Prometheus does not explicitly define the environment. It does not define the objects in the environment (requirement 1), the relationships between them (requirement 2), or the laws that govern the effect of agent's actions on the environment (requirement 5). However, Prometheus does capture the operations that it uses to get percepts from the environment and perform actions on the environment. Thus Prometheus too could benefit from an explicit AEI Model.

2.4 Organization-Based Multiagent Systems Engineering

The Organization-Based Multiagent System Engineering (O-MaSE) [1] methodology extends the original MaSE [3] methodology to allow the design of organizational multiagent systems. Some of the weaknesses of MaSE addressed by O-MaSE include the tendency to generate static organizations, the inability to model sub-organizations/systems, and the lack of explicit concepts for modeling interactions with the environment. To model interactions with the environment, O-MaSE represents both the sensing and manipulation of the environment as a type of *Capability*, which is defined as an "atomic entity that defines the agents' abilities; these abilities include *soft* abilities such as access to resources or computational algorithms, as well as *hard* capabilities such as sensors and effectors [1]. We use this notion of capabilities as the foundation for our AEI Model, extending it to allow capability composition as well as to model direct interaction with the environment. The current version of the O-MaSE metamodel is

shown in Fig 1. The important elements for this paper center on the Capabilities and the Domain Model. In O-MaSE, Capabilities can be either plans or actions, where plans are algorithmic entities that use actions for low-level operations. The domain model is relatively simple, consisting of a set of Environment Objects. Actions interact directly with the Environment Objects.

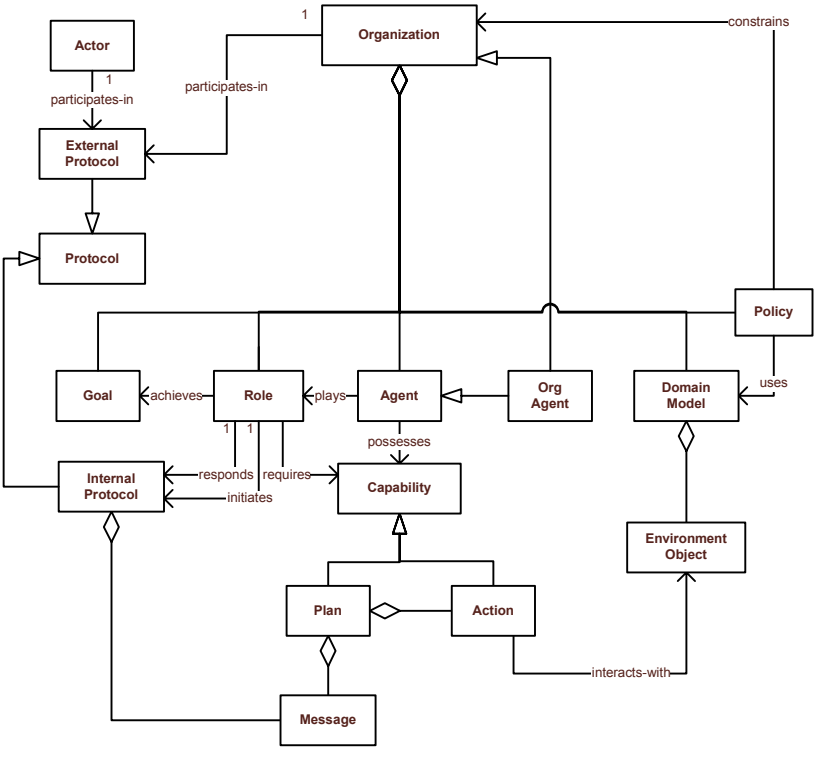

Fig. 1. O-MaSE Metamodel

However, even though O-MaSE does provide the notion of capabilities and a domain model, it does not meet all our requirements for an AEI Model. Specifically, the domain model does not include agents as objects in the environment (requirement 2) or the definition of the laws that govern the effect of those operations on objects in the environment (requirement 5). In the next section, we present our AEI Model that ties together the O-MaSE elements to provide a complete environment interaction model that can be used with O-MaSE and easily adapted to other main stream multiagent systems methodologies.

3 Agent-Environment Interaction Model

Our proposed AEI Model is composed of three main elements: the Capability Model, the Environment Model, and a set of Interactions between capabilities

and environment objects. Essentially, agents possess capabilities that sense and act upon objects in the environment via interactions. Fig 2 depicts the integration of these three parts into our AEI Model. The top part of the figure represents the *Capability Model*, which defines capabilities as consisting of a set of actions, each of which has a single operation that interacts with environment objects. The bottom part of the figure captures the *Environment Model*, which includes an explicit environment that contains a set of environment objects (that includes agents) and their relationships. The environment objects are governed by processes that implement specific environmental principles. *Interactions* are defined by the intended effect of operations on environment objects.

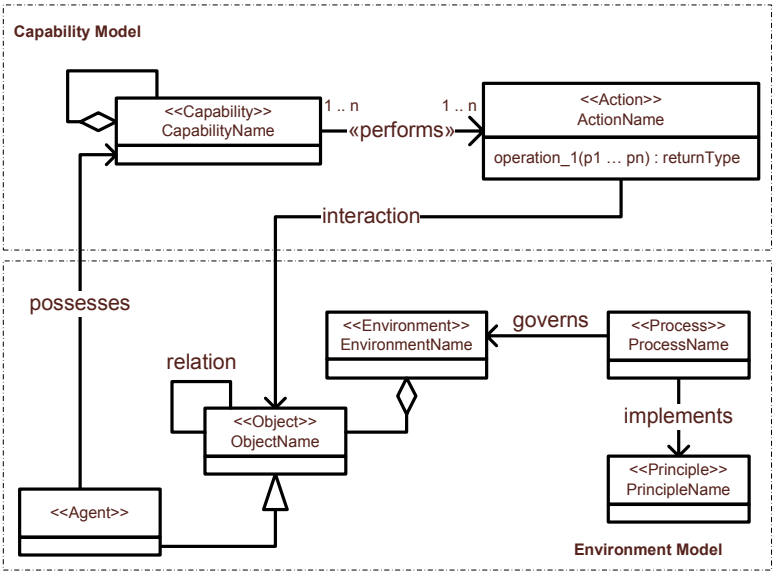

Fig. 2. Agent-Environment Interaction Model

Note that this general model captures all the items advocated by Ferber as described in Section 1. It also satisfies each of our specific requirements for an AEI Model: there is a unique entity called the environment, the environment consists of a set of objects that includes agents, it includes the notion of relations that may exist between objects in the environment, it defines a set of operations that agents may perform upon environment objects, and it captures the notion of laws that govern the effect of those operations on objects in the environment. The three main entities of our AEI Model are described in more detail in the following sub-sections.

3.1 Environment Model

In order to define the actions that an agent may perform upon then environment, it is critical that we understand exactly what types of objects may be in the

environment and the attributes of those objects. While environments have been widely touted as important in multiagent systems design, there is not a well accepted representation for them. Odell et. al. define the environment as the entity that provides the principles and processes for agents and objects to exist and communicate [6] while Russell and Norvig define an environment as an entity with which agents interact, with properties defined by concepts such as accessibility, determinism, dynamism, and continuity [9].

In our AEI model we use a simple Environment Model to model the objects upon which agents perform the basic actions of sensing and affecting the environment through interactions. Basically, the *Environment* is a container of *Objects*, which can include *Agents* situated in the environment. All the objects in the environment are affected by physical *Principles* that are implemented by *Processes* as depicted in Fig 3. Objects are defined simply via a name and a set of attributes. Here, environment objects are actually more closely related to object-oriented classes or types than true object instances.

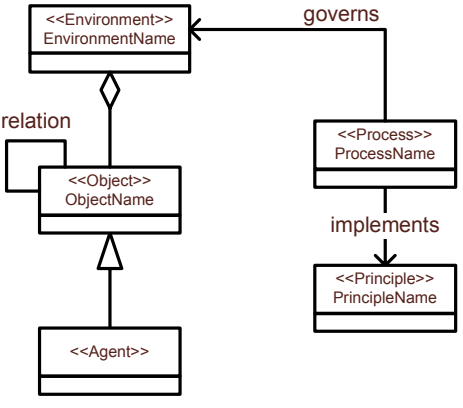

Fig. 3. Simple Environment Model

Fig 4 shows an example Environment Model for a Weapons of Mass Destruction Search (WMD) cooperative robotic search system. In this model, we are concerned with modeling the types of objects that can be found while doing a search of an area (office building, etc.) for suspicious boxes that can be classified as possible chemical, nuclear, or biological weapons based on signatures produced by the weapons. All objects in the environment have a location within the environment as well as a size (which we abstractly identify using a type PhysicalDimension). Robots also have four additional attributes: add, q, R, and G. The *add* attribute represents the address of the robot for communication purposes while the *q* attribute represents the message queue of incoming messages. As we will see later, the communication capability requires both these attribute values for proper operation. The *R* and *G* attributes represent the current set of roles and goals assigned to the robot. There are four types of inert objects:

doors, inert boxes, chairs, and tables. Each of these has a zero values for the rad, bio, and chem attributes. Finally, there are the three types of weapons that can exist: RadWeapons, BioWeapons, and ChemWeapons. Each of the weapon types are boxes. However, the exact type of weapon (or whether the box is inert) can be determined only by using special sensors mounted on specific robots.

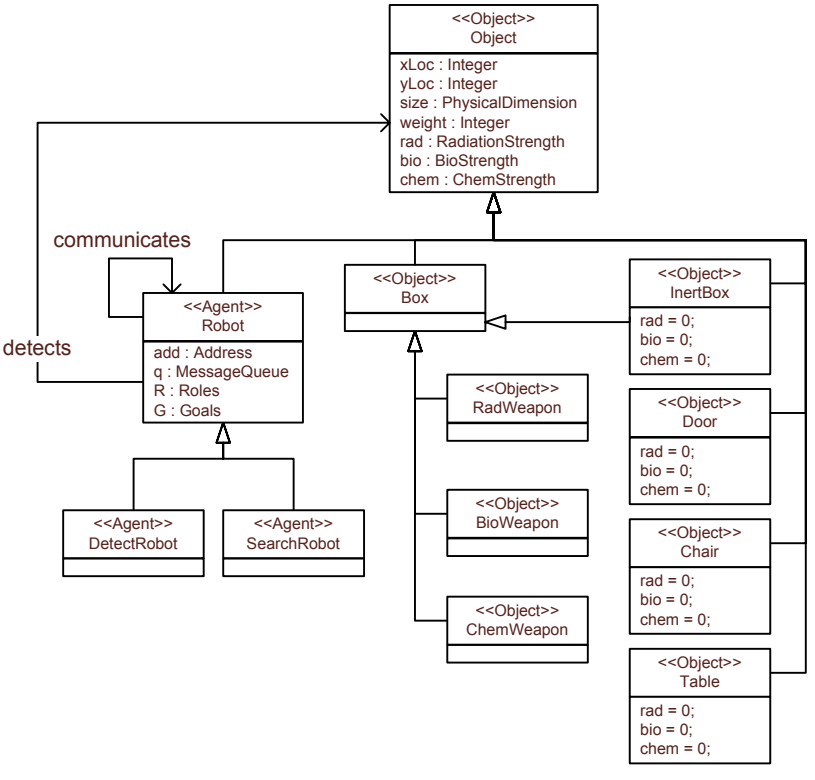

Fig. 4. WMD Environment Model

There are two relationships shown between objects in the Environment Model. The *communicates* relation is shown between two Robots. The communicates relation is critical in allowing Robots to use their communication capability (as described in Section 3.2) to send and receive messages. The *detects* relation is shown between Robots and all other objects in the environment (including other Robots). The detects relation allows Robots to use their Search and Pickup capabilities (see Section 3.2) to detect objects in the environment.

Besides identifying objects and attributes, it is also important to identify the principles and processes that govern the environment. For instance, in detecting radiation, there is the well known principle that the amount of radiation intercepted varies as the square of the distance between the source and the sensor. Thus, we must define a process that determines the amount of radiation detected

at any location in the environment. To do this, we must add the amount of radiation produced by all radiation sources in the environment, regardless of how little they add to the total. Thus, the process that is in play can be defined by the following equation.

$$radiation(x, y) = \sum_{\forall o:Box} \frac{o.rad}{\sqrt{(o.x - x)^2 + (o.y - y)^2}}$$

The WMD Environment Model is developed using traditional domain modeling or domain analysis techniques common to most object oriented development methodologies. Essentially, the goal of the domain modeling is to capture the objects, relationships, and behaviors that define the domain [8]. For our purposes, the domain is the environment of the multiagent system under development. A good Environment Model is critical in the definition of the interactions between the capabilities and the environment objects as the precise definition of the operations is based on the object attributes.

3.2 Capability Model

As defined by Russell and Norvig, an agent is anything that can sense and perform actions upon its environment [9]. As described above, most multiagent systems methodologies represent these sensors and effectors in some way, either implicitly or explicitly. In our AEI Model, we abstract the notion of sensors and effectors as agent *capabilities*. To keep in line with our O-MaSE definition, we assume capabilities can be either hardware or software based capabilities.

In our AEI Model, we represent the sensors, effectors, or a combination of both as *capabilities*. A capability can be defined at different levels of abstraction like sense, move, jump, etc. The *Jump* capability can be accomplished by sensing an obstacle and then passing (moving) over it. We define *Capability* as an entity that can *perform* one or more *actions* (e.g. sense or move) and can be composed of other sub-capabilities.

An *Action* is defined as an entity that represents the agent's actual sensor or effector. Specification of the execution of an action is defined via a single accessible *operation*. Each action's operation has a set of preconditions that determine whether or not the operation can be executed. If the preconditions hold, the operation may be executed. If the preconditions do not hold, the operation may not be executed. Operations also have a set of post-conditions that specify the desired state of the world after completion of the operation. However, because operations are assumed to be performed in a dynamic environment with external influences, the post-condition do not guarantee that the desired state will hold. In reality, the processes that implement the environment principles determine the actual state of the world after an operation is performed. For instance, if a robot performs an operation to move forward one meter, the robot may or may not actually move exactly one meter. Wheel slippage and wind conditions are environmental processes that help determine exactly how far the robot will actually move. When such environmental processes are expected by the agent,

the agent may predict its own performance or at least sense to determine the exact result of its operations.

The Capability Model provides support for reusability and modularity by encapsulating each sensor and effector operation individually in actions. Our model also provides support for constructing a capability using other capabilities, which we call a *composed* capability. We depict our Capability Model in Fig 5. By defining capabilities in terms of other capabilities as well as atomic actions, the model is very flexible and allows designers to capture sensor and effector operations at a continuum of granularity levels based on the application or designer preference.

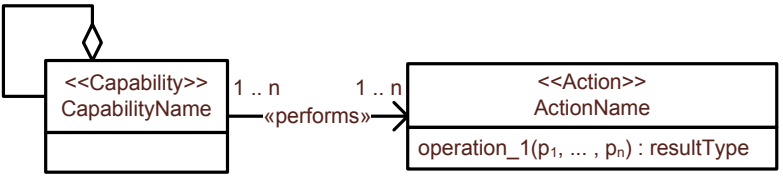

Fig. 5. Capability Model

The flexibility of capability definition using this model is shown in the following examples. In the example shown in Fig 6, the Search capability shows an agent's capability to scan and detect items in a particular location as a single high level action, SearchLocation. This level of abstraction may be appropriate during the initial stages of analysis or when the agent is using a predefined package that provides higher level services.

Fig. 6. Search Capability Example

In the second example shown in Fig 7, the PickUp capability is shown as being carried out by performing three lower level actions: Detect, Grab, and Lift. By defining actions at a lower level than that shown in Fig 6, the definitions of the actions in Fig 7 could be more easily reused when defining other capabilities such as Search (which could be defined using Detect) or Transport (which could use all three along with a Move action). Clearly, the level of abstraction or refinement should be left to the designer and, thus, our model allows a wide variety of choices.

Another feature of our Capability Model is the ability to capture the capability of an agent to send and receive messages in a single, consistent style. Fig 8 shows a simple definition of the Communicate capability, which is carried out by

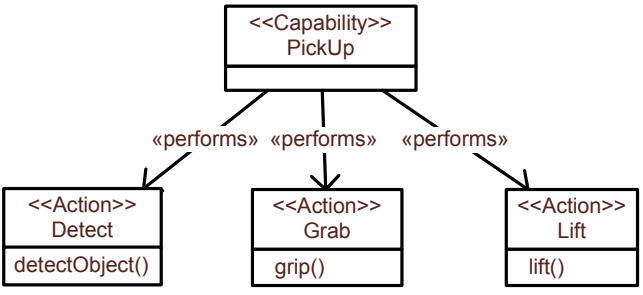

Fig. 7. Carry Capability Example

performing one of three actions: P2PTransmit, Broadcast or Receive. While most multiagent modeling techniques use special notation for sending and receiving of messages, they are actually special forms of actions. By allowing designers to specify communications in the same way as other actions, it actually allows the designer to specify exactly how communication can be performed.

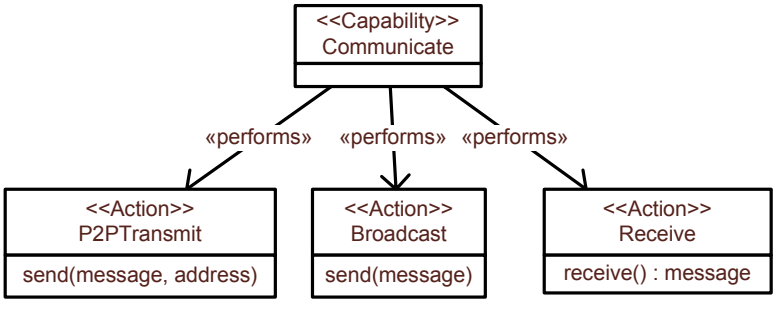

Fig. 8. Communication Capability Example

The AEI Model also allows the designer to create new capabilities out of existing capabilities. Thus, in Fig 9, the *Rescue* capability is an example of a composed capability that uses the previous defined capabilities of Search, Pickup, and Communicate. In essence, the Rescue capability has access to all the actions defined as part of the Search, Pickup, and Communicate capabilities.

Complete specification of the Capability Model requires defining pre- and post-conditions for each of the operations. These pre- and post-conditions completely and unambiguously define the interactions between agent capabilities and environment objects, which are defined in the next section.

3.3 Interactions

By executing an operation defined in an action, an agent can sense or manipulate its environment. If this action is to sense, the agent receives information

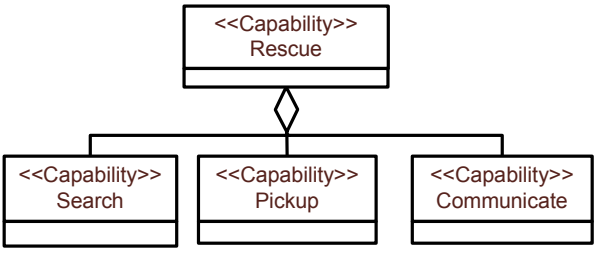

Fig. 9. Composed Rescue Capability Example

regarding the environment. If the action modifies an object in the environment, the environment will change and the agent representation of the environment will change as well.

As described above, each operation is defined by a pre-condition that determines whether or not the action can be executed and a post-condition that specifies the desired state of the world after the operation is performed. Again, the post-condition does not specify the *actual* state of the world after the operation is performed, but only the *desired* state since the environment is governed by the principles and processes defined in the Environment Model.

An example of the specification of an interaction via operation pre- and post-conditions is shown below in the definition of the send(message, address) operation from the P2PTransmit action. For consistency, the notation used is UML's Object Constraint Language (OCL).

P2PTransmit: send(message, address)
Pre: not(address = null)
Post: self.possesses.communicates->select(add = address).q->includes(message)

The semantics for this send operation state that if the address given as a parameter is not null, then the message (also given as a parameter) is added to the message queue (q) of the agent whose add parameter is equal to the address parameter. Since 'self' refers to the capability, the reference 'self.possesses' follows the possesses relation between capabilities and the agents that possess them. Thus 'self.possesses.communicates' refers to the set of agents with which the agent possessing the P2PTransmit capability can communicate.

3.4 AEI Model and O-MaSE

The AEI Model fits nicely into the O-MaSE metamodel due to the fact that O-MaSE already possesses the main concepts used in our AEI Model. While capabilities and actions existed in the original O-MaSE metamodel, their relationship had to be adjusted slightly to fit the AEI Model. The integration of the AEI Model into the O-MaSE metamodel is shown Fig 10; the bold lines represent the new/modified entities and relations. First, we had to allow capabilities

to be composed of lower level capabilities. Next we had to change the semantics of capabilities being either a plan or an action. To maintain the semantics of the AEI Model presented, we have also added the constraint that capabilities may be composed of either a single plan or a set of lower level capabilities and actions. The Domain Model is used to capture the AEI Environment Model. We added the concept of environmental properties as a component of the domain model, where an Environmental Property specifies the principles and processes that govern the environment. We also added the ability for agents to be represented as environment objects.

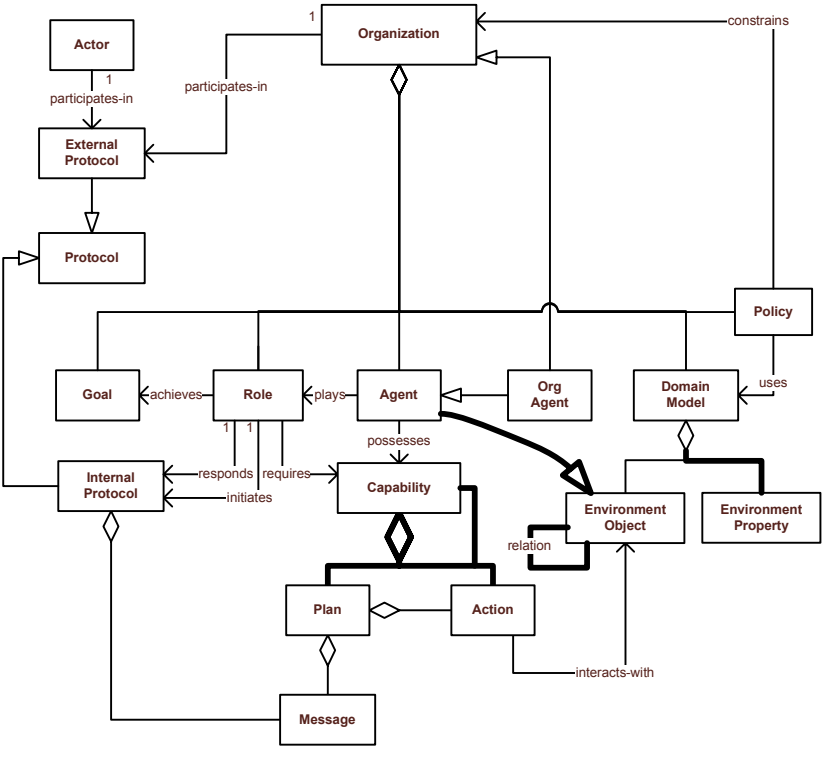

Fig. 10. O-MaSE Metamodel Extended with AEI Concepts

4 Example

To illustrate the use of the AEI Model integrated into the O-MaSE methodology, we chose to model a Weapon of Mass Destruction (WMD) Search system as an example of a cooperative robotic agent system. The goal for the robot team is to search a specified area for possible chemical, radioactive or biological weapons and remove such weapons once they have been positively identified. Each of the robots on the team have multiple capabilities that allow them to sense various

weapon types, navigate, and locate themselves using Global Positioning System (GPS), and transport the weapons to a safe location.

Based on this system description, we defined an O-MaSE Agent Model in Fig 11. In our current notation, we annotate agents with the "Agent" keyword and capabilities by the "Capability" keyword. A base robot has three capabilities: Communication, GPS, and Move. SearchRobots have a Sonar capability to aid in searching while the RemovalRobots are equipped with Transport capabilities for removing any identified WMD objects. Finally, there are three types of DetectRobots: BioDetectRobots, ChemDetectRobots, and RadDetectRobots. All the DetectRobots are equipped with the capability to detect a specific type of WMD: BioDetector, ChemDetector, or a RadDetector. The filled-headed arrows between agents represent protocols used by the robots. The *divideArea* protocol is used at system initialization to determine which SearchRobots will be assigned which specific areas to search. When a SearchRobot detects a suspicious object, it uses the *detection* protocol to find an appropriate DetectRobot to investigate. If a positive detection is made by the DetectRobot, the SearchRobot uses the *positive* protocol to find an available RemovalRobot to remove the WMD object.

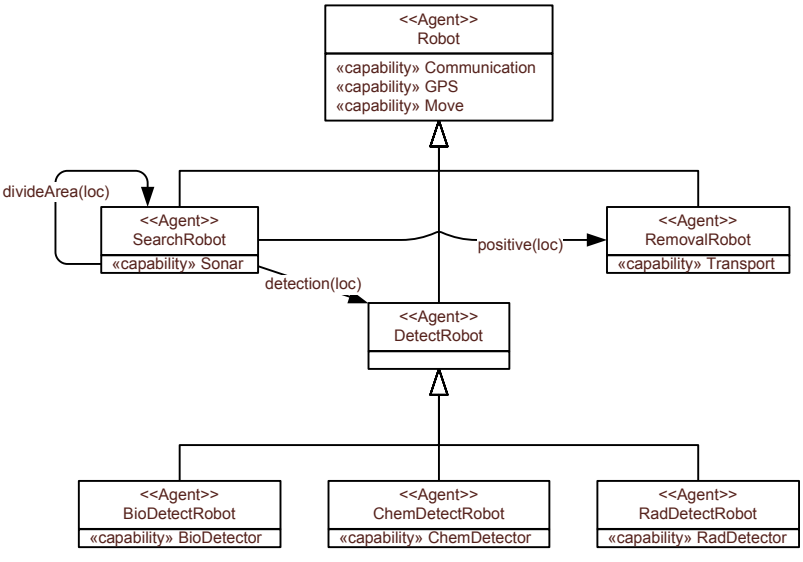

Fig. 11. WMD Agent Model

To complete the AEI model, we modeled the WMD environment using an O-MaSE Domain Model (as shown in Fig 4) and defined the set of capabilities to sense and manipulate that environment in the O-MaSE Capability Model as shown in Fig 12. As discussed previously, the level of abstraction for each capability may be different. For example, the *Communication* capability is implemented by two actions, P2PTransmit and Receive. The model also includes

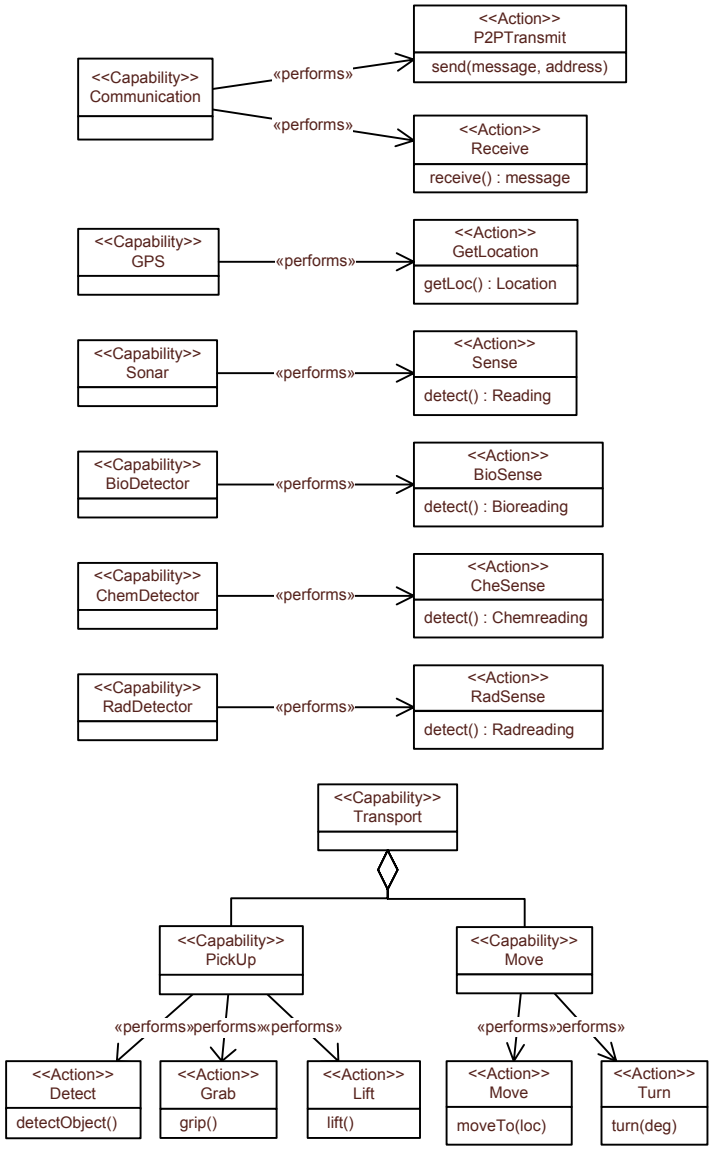

Fig. 12. WMD Capability Model

a composed capability *Transport* that uses the Pickup and Move capabilities. Each base capability is implemented by actions that interact directly with the objects in the environments.

Assuming the pre- and post-conditions for each operation are defined, the agent designer may use the operations to design the agent plans. A simple (algorithmic) plan for the ChemDetectRobot is shown below. It uses the Communication, GPS,

Move, and ChemDetector capabilities and their associated actions to define the plan. For simplicity, we describe the plan using a simple pseudo code approach. The notation used to access the operations assumes that names are unique and follows the form *CapabilityName.ActionName.OperationName*.

ChemDetector Plan

```
loop
  m = Communication.Receive.receive()
  if (m.performative = ‘‘possible’’)
    Communication.P2PTransmit.send(acknowledge(loc), m.sender)
    repeat
      Move.Move.moveTo(loc)
      currentLoc = GPS.GetLocation.getLoc()
    until (currentLoc = loc)
    result = ChemDetector.CheSense.detect()
    if (result = positive)
      Communication.P2PTransmit.send(positive(loc),m.sender)
    else
      Communication.P2PTransmit.send(negative(loc),m.sender)
    end if
  end if
end loop
```

When the ChemDetector robot receives a *possible* message, it sends an acknowledge message and then moves to the location specified. Once at the appropriate location, the robot uses its chemical sensor and returns the result, either positive or negative.

5 Conclusion and Future Work

In this paper we have described an approach for modeling a multiagent system's interactions with its environment. The key concepts in our approach were capabilities (and the actions they perform) and a model of the environment. We defined a set of requirements for an Agent Interaction Model based on Ferber's definition of a multiagent system. We suggest that our AEI Model captures all these requirements since it contains a unique entity called the environment (requirement 1) and that is modeled as a set of objects/agents (requirement 2) and a set of relations between those objects/agents (requirement 3). Through a set of capabilities possessed by the agents, the agents have access to a set of operations that they may perform upon environment objects (requirement 4) whose effect are governed by environmental laws (requirement 5).

We showed how our AEI Model could be integrated into the O-MaSE methodology with only slight modifications to the O-MaSE metamodel. Finally, we presented a WMD Search simulation system design using O-MaSE and the AEI

Model. We showed how the model captured the relationship between the agent's capabilities and their affect on the environment and how to use those definitions to model low-level agent plans.

It is our contention that all multiagent systems methodologies should provide a robust way to define the interaction of agents with their environments. While most methodologies provide some mechanism for describing these details, most do not provide a sufficient modeling capability. Thus, we believe that integrating the concepts of our AEI Model, or elements thereof, into existing methodologies is not only possible, but would be a positive step toward more complete system models.

Finally, we are working to fully integrate our AEI Model into the O-MaSE methodology and into agentTool III (aT^3) [2]. We are continuing to evolve O-MaSE to provide a flexible methodology that can be used to develop both traditional and organization-based systems. A long term goal is to provide a tailorable methodology that is fully supported by automated tools. $aT^3$is being developed as an Eclipse plug-in that will give the agent system designer unprecedented flexibility while providing enhanced verification capabilities between models. Eventually, aT^3will provide predictive performance metrics to allow the designer to make intelligent tradeoffs and will generate code for FIPA compliant frameworks. As defined in this paper, the AEI Model is a key step in fully defining the design of agents to the point where a higher degree of low-level code generation is possible.

References

1. DeLoach, S.A. Engineering Organization-based Multiagent Systems. LNCS Vol. 3914, Springer, (2006) 109-125
2. DeLoach, S.A. Multiagent & Cooperative Robotics Laboratory. "agentTool III Home Page," http://macr.cis.ksu.edu/projects/agentTool/agentool3.htm (2006)
3. DeLoach, S.A., Mark F. Wood and Clint H. Sparkman, Multiagent Systems Engineering, The International Journal of Software Engineering and Knowledge Engineering, 11(3) (2001) 231-258
4. Ferber, J. Multi-Agent Systems - An Introduction to Distributed Artificial Intelligence. Addison-Wesley, Harlow (1999)
5. MESSAGE: Methodology for Engineering Systems of Software Agents. Deliverable 1. Initial Methodology. EURESCOM Project P907-GI (2000)
6. Odell, J., Parunak, H., Fleischer, M., Bruckner, S. Modeling Agents and their Environments. LNCS Vol. 2585, Springer (2002) 16-31
7. Padgham, L. and Winikoff, M. Prometheus: A Methodology for Developing Intelligent Agents. LNCS Vol. 2585, Springer (2003) 174-185
8. Pressman, R. Software Engineering: A Practitioner's Approach (6 ed.), McGraw-Hill (2004)
9. Russell, S. and Norvig, P. Artificial Intelligence: A Modern Approach. Prentice Hall; 2nd ed. (2002)

10. Weyns, D., Parunak, H., Michel, F., Holvet, T., and Ferber, J. Environments for Multiagent Systems State-of-the-Art and Research Challenges. LNAI Vol. 3373, Springer (2005) 1-47
11. Wooldridge, M. Jennings, N. and Kinny, D. The Gaia methodology for agent-oriented analysis and design. Journal of Autonomous Agents and Multi-Agent Systems, 3(3), (2000) 285-312
12. Zambonelli, F., Jennings, N. R., and Wooldridge, M.J. Developing Multiagent Systems: The Gaia methodology. In ACM Transaction on Software Engineering Methodology 12(3), (2003) 317-370