BEHAVIOR AND SECURITY EXTENSIONS TO FIPA
ARCHITECTURE AND SPECIFICATIONS

V. S. ALAGAR
JOANNE HOLLIDAY
Department of Computer Engineering
Santa Clara University
Santa Clara, California, CA 95053
{valagar, jholliday}@cse.scu.edu

In this paper we provide two extensions to the FIPA standard. One extension
provides a generic behavior model for agent types. Another extension is the pro-
vision of a security model for agent applications. We illustrate the correspondence
between the abstract behavior models and the FIPA architectural standard.

1. Introduction

Agent technology is perceived to have the potential for seamless integra-
tion of business services over the Internet. Active media technology revolves
around web applications, and agent technology can provide viable solution
for its large scale industrial application. Currently, several agent platforms
are available in the market. However, due to lack of rigour and standards
in their development, their deployment in building a large system requires
a considerable amount of re-engineering of agents. Re-engineering may
sometimes require reconstructing agents from scratch, thus robbing away
precious time and cost. As a solution to this problem, the Foundation for
Intelligent Physical Agents (FIPA) has come up with several standards 4,5,6,
which are abstract architecture specifications for agents. However, FIPA
specifications are only informally described. Its goal is to aid developers in
creating concrete specifications of different elements of FIPA agent system.
It does not provide any hint on how and where an agent may be used.
In this paper we provide methods to strengthen FIPA architecture so that
application engineers can develop concrete and detailed agent descriptions
and assemble them by understanding their behavior descriptions. Conse-
quentially, the behavior of agent communities and their collaborations can be
simulated and proved correct before they are deployed in an application. The two significant contributions made in this paper are:

- FIPA architecture extension to include formal behavior models for
generic agents, and
- security model within FIPA standard for agent applications.

2. FIPA Architecture - a brief summary

The Agent Message Structure (AMS) contains fields for providing information required to accurately transfer data between agents. All fields in the message structure except the performative are optional. FIPA defines languages with different formal basis. FIPA Content Language Library defines the requirements of an acceptable language.

The FIPA-Semantic Language (FIPA-SL) consists of a set of production rules for representing semantic information. Each unit of expression can be categorized into three types: Identifying expression, Action expression or Proposition. Identifying expressions are used to declare variables and make assertions. Action expressions describe some action that is either already performed, intended to be performed in the future or is currently being performed. Proposition is used to represent the behavioral aspects/mood of the agent like goal, intent, belief, uncertainty. The SL provides a way to represent semantics. The representation does not inhibit incorrect semantic representations.

The FIPA Constraint Choice Language (CCL) is a query and assertion language. It is primarily designed to represent constraint satisfaction problem (CSP), where a set of attributes are involved, each having their domain and range. The valid CCL element can have three types of elements, Object, Action or Proposition. This language is similar to SL, but in CCL it is not possible to represent modal operations like belief, intent, goal or uncertainty. CCL may be encoded using XML constructs and paradigms. Communicative acts (performatives in speech-act theory) are atomic action expressions and are building blocks for protocols. FIPA has a public library to register all communicative acts. We use the following communicative acts in this paper: agree, failure, inform, not-understood, request, and refuse.

3. Extensions to FIPA Architecture

In this section we propose two additions to FIPA standards: (1) a language for behavior description, and (2) a security model for agent platforms, se-
curity agents as a new agent type, and secure agent communication.

3.1. A Language for Behavior Description

FIPA architecture does not define formally what behavior means, nor does it prescribe a behavior description language for agents. It only sets criteria for content and communication languages, and message types in protocols. We add to FIPA architecture behavior models, by defining the behavior of agent types as extended state machine diagrams.

The language for describing the behavior of agents is a finite state machine, extended with state hierarchy, transition specifications, and clocks for enforcing time-dependent actions. A state is an abstraction of a property held by the agent at an instant of its lifecycle. It is possible to associate with each state an invariant defined on some of the attributes of the agent. A state may be simple or compound. A compound state can be decomposed into states which are either simple or compound. This refinement ability adds expressive power to the agent behavior description. Each transition from a state to another state is specified as a pair \([g/a,e]\), where \(g\), called guard, is a predicate, and \(a\), called action, is another predicate, and \(e\) is a parameterized event. The semantics of the transition specification \([g/a,e]\) labeling the transition from state \(s_i\) to state \(s_j\) is that if the event \(e\) occurs when the agent is in state \(s_i\) and \(g\) becomes true at that instant, the agent changes its state to \(s_j\), and at state \(s_j\) the predicate \(a\) is true.

The behavior of an agent is given by the trace semantics of timed labeled transition systems, the formalism underlying the state machine description of the agent. The behavior depends on the context, defined by the set of messages that can be received from or sent to other agents in a specific application. A message is either an event received from another agent (input) or an event sent to another agent (output). Input and output messages occur at the ports of the agent. An internal event corresponds to an internal activity that the agent has to perform by itself; it does not interfere with another agent. A message can be acted upon only when the agent is at a state where the guard of at least one transition specification can be satisfied. The result of acting upon a message includes a state change, a computation as required by the action predicate \(a\), and the enabling of one or more future actions, as required by the time constraints in force.

A computational step of an agent is an atomic step which takes the agent from one status to the next status as defined by the specification of the transition taken at that instant. A computation \(c\) of an agent \(A\) is a
sequence of statuses such that two successive status in the sequence are related by a computational step. Formally, the behavior of an agent $A$, denoted $\text{Comp}(A)$, is the set of all computations. The complete interacting behavior of two agents is described by the set of computational paths in the synchronous product machine of the machines that describe the individual behavior of the agents. We can extend the definition of synchronous product to a collection of interacting agents and define the behavior of the agent system.

### 3.2. FIPA Extensions for Security

Security is an important issue for agent-based E-Commerce transactions. We propose two levels of security, one afforded by agent management system, and another provided by a security agent, an autonomous agent which monitors agent interactions and triggers remedial actions.

#### 3.2.1. Agent Management System

FIPA architecture makes no mention of security aspects. Several risks associated with the current FIPA standard are given in $^1$. Hence, a FIPA compliant agent may not be secure and should not be used in security-critical applications. To remedy this situation, we have the following recommendations.

*Recommendations to Agent Platform:* The agent platform shall validate the owner registering an agent and log the source address of the request. The log information is stored on a per security domain basis, meaning, e.g., an organization before initializing the agent platform would specify the destination of the log information. This is to protect the sensitivity of the actions performed by the agents. As a part of agent platform initialization, parameter validation module/validation rules must be specified. The validation information must be able to qualify all the allowed values that could be sent to the agents. In addition to platform validation of parameters, the agents could specify their own validation rules to protect themselves from other agents. During initialization/runtime an agent can supply filters for various input stimuli. Agents should be able to define those agents they wish to hear from, agents that must be blocked, and a list of addresses that can be allowed/blocked. Such information is fed to the FIPA MTS, which processes them and performs routing.

*FIPA architecture recommendations:* The description of agents in Agent Management System (AMS) must include the domain of origin and
signature attribute. These attributes can have an optional presence, but must be queriable from the Directory Facilitator (DF). When an agent interaction takes place, the receiver may know the sender of the information in most cases. For premium or restricted services, although the receiver may know from the incoming data who the sender is, this information may not be enough to qualify the user for a given service. The receiver should also know other information, such as the security clearance level of the sender, the scheme of approval the sender went through, and the time of expiry.

We propose several additional measures to strengthen the FIPA architecture and its execution framework. An agent description must have a parameter specifying the validation rules for parameters. It could also specify a method for accessing the validation routine by specifying a Uniform Resource Identifier (URI). The RFC 2396 describes the format of URL.

Logging is a function supported by AMS. Domain of logging is "AMS-agent-description" and properties are log level and log string. Log levels are defined as a part of the AMS. To support existing standard logging levels, MESSAGE, WARNING, ERROR may be supported as the range of log level. In the function register, one parameter that must be added is the AMS-agent-addresses, an abstract data structure that defines two components action and addresses. The action can have either of the two binary values, accept or reject. The addresses field could be specified either as a range or enumerated as multiple values. AMS processes them and provides the MTS with this information. When a request arrives, MTS determines the agent that needs to be contacted and checks the list of deny addresses. If there is a match, the MTS ignores the request and logs the action as a warning. Multiple entries logged from the same source may be determined as an attack based on factors like frequency, time interval, and number of attempts. Otherwise, the request is forwarded to the agent. This is used to protect against denial of service attacks.

Agent platform specification may mandate standard validation rules, such as: "Source IP Addresses must be from the internal network", and "Agents may not provide their forwarding addresses other than ones approved by the administrator". During agent registration, it is optional for agents to pass the name of the interface exposed by it that performs attribute validation. This would be a part of register method parameter. Agents must be partitionable in terms of domains. Also each agent platform may be specified as to which of the domains it services. This would ensure accountability of each domain for their agents. Further, the logging information provided by each agent may be routed to their specific domain...
log server/file.

3.2.2. **Security Agents**

Security agents should be added to the FIPA specifications. Security agents are task agents that remember events, draw inferences, and plan actions to achieve security goals \(^{1,3}\). They are autonomous, evolve through learning, and have the ability to make predictions based on logical inference. E-business communications need to be guarded by security agents that provide the security services requested for the conduct of e-business. Agents can be directed to collect commercial data only from trusted agencies and controlled sources. Agents support for secure e-business can be segmented into five distinct categories: authentication, authorization, data integrity, confidentiality, and non-repudiation \(^7\).

![Diagram of Security Agent Class Structure](image)

Figure 1. Security Agent Class Structure

Figure 1 shows the context of functioning of security agent during an interaction session of two participants. Security agents are supported by a knowledgebase of security and business rules. The inference engine that is part of the knowledgebase, enables the agent to detect activities that violate business workflow policies, and security policies. The security agent is a *watchdog* who will monitor the activities of all agents that function in the system. It has access to history of agent states, and knowledge of events that must be triggered to access sensitive information storage. Based on the knowledge base and current system state it draws inferences, and plan actions to achieve security goals.

It is impossible to conduct an exhaustive logical analysis in real time. It is better to have methods that train security agents using probabilistic rules and data mining. The method will assign a confidence and support
level for every rule. The history maintained by the agent is a valuable tool
to derive such rules. With predictive capability, it should be possible to
stop information leakage and violation of policies before they are actually
committed.

Activities of the task agents will be authenticated by the security and
business policies encapsulated in them. Since each task agent has expertise
in a specific domain, and security policies are domain specific, it is appro-
piate to encapsulate them in the task agents. A business transaction can
be modeled in a workflow diagram and security policies can be enforced
at each atomic step of the workflow activity.

Figure 2 shows the behavior of a task agent engaged in fulfilling a client’s
request. Since a security agent is a task agent, the behavior of a security
agent includes the behavior shown. We can refine this behavior description

Figure 2. State Transition Diagram for Task Agent
to include anonymously performed activities, and inference based on data mining.

4. Conclusion

In this paper, we have extended FIPA by adding security attributes, and have given guidelines for incorporating security policies in an agent platform. We used extended finite state machines to model the behavior of agent types. The semantic basis for behavior is the labeled transition system semantics. Knowing the characteristics, the context where agents of a type can function, and their behavior, the developer is given precise guidance for the development of cooperative agents for E-Commerce transactions. For security-critical applications, we should be able to analyze the system before deployment and ensure that security is not violated in any transaction. This will be possible if the recommended security and behavioral extensions are incorporated into the architecture. The technical report ¹ includes a discussion on the rationale for agent classification, agent types based on the classification scheme, the formal behavior descriptions of agents, and an illustration of E-Commerce architecture based on our approach.

References


