

# Participatory Simulation for Designing Evacuation Protocols

Yohei Murakami  
Department of Social Informatics  
Kyoto University  
Kyoto, 606-0801, Japan  
+81 75-753-5398  
yohei@kuis.kyoto-u.ac.jp

Toru Ishida  
Department of Social Informatics  
Kyoto University  
Kyoto, 606-0801, Japan  
+81 75-753-4820  
ishida@i.kyoto-u.ac.jp

## ABSTRACT

In evacuation domain, there are evacuation guidance protocols to make a group of evacuees move smoothly. Each evacuee autonomously decides his/her action based on the protocols. However, the protocols sometimes conflict with evacuees' goals so that they may decide to violate the given protocols. Therefore, protocol design process has to consider human's decision making on whether or not to follow the protocols so as to control every evacuee more smoothly. To address this problem, we introduce participatory simulation where agents and human-controlled avatars coexist into protocol design process. It allows us to validate protocols at lower cost than demonstration experiments in the real world, and acquire decision making models from log data. In order to refine the protocols based on the acquired models, we have designed and implemented the agent architecture separating decision making from protocol execution.

## Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence – *multiagent systems*.

## General Terms

Design

## Keywords

protocol design, multi-agent simulation, evacuation simulation

## 1. INTRODUCTION

Disaster-prevention hardware, such as fire protection equipments and refuge accommodations and so on, is improved based on the lesson learned from frequent disasters. In contrast with the hardware, disaster-prevention software such as evacuation methods has no advancement. This is because a lot of subjects are necessary to validate designed evacuation methods and the cost to conduct the demonstration experiments is very high. Moreover,

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

ATDM'06, May 8, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...\$5.00.

such experiments are too complex to reproduce the results. The non-reproducibility causes difficulties in analyzing problems occurring in the experiments.

One of the approaches to solving these problems is multi-agent simulation [9]. Multi-agent simulation is a simulation to monitor macro phenomena emerging from interactions between agents which model actors such as humans. Once models are acquired from existing documents and past research data, we can reproduce simulation results as many times as needed. Besides, multi-agent simulation enables us to estimate the effectiveness of designed evacuation methods by assigning the methods to agents as their interaction protocols. That is, we view the protocols as behavioral guidelines of the agents during evacuation.

However, even when the simulation results show the effectiveness of the evacuation protocols for simulated evacuees, the problem of validating whether they are effective for real evacuees still remains. In order to solve the problem and develop more effective evacuation methods, we need participatory approach which is a method of bringing potential evacuees and leaders into the simulation. Therefore, we aim to design evacuation methods by using participatory simulation where agents and human-controlled avatars coexist. In this simulation, we can check the effectiveness of the designed evacuation methods by providing the protocols for not only agents but also potential evacuees controlling avatars. In order to accomplish our goal, we set up the following research issues.

- Establishment of protocol design process: Humans may not follow the given protocols, since they are more autonomous than agents. Thus, we need protocol refinement process considering human's decision making about whether or not to follow the given protocols. To construct more valid protocols, we have to modify protocols after verifying human's decision making, internal models, obtained from participatory simulations.
- Realization of autonomy under social constraints: To simulate human autonomous behavior under a given protocol, which is a social constraint, agent architecture needs a decision making mechanism independent of a given protocol. This mechanism coordinates a proactive action and an action prescribed by a given protocol, and realizes the selfish behavior that the agent violates its given protocol.

This paper will first describe what an interaction protocol is, and then clarify the target protocol we try to design. Next, we will

propose our protocol design process and agent architecture necessary to realize the proposed process. Finally, we will check usefulness of our approach by applying it to refine “Follow-me method,” a new evacuation method proposed by a social psychologist.

## 2. Interaction Protocols

In the area of multi-agent, an interaction protocol is often employed as a means to control interactions among agents for any purpose. Interaction protocols can be roughly divided into two groups based on the goal of the protocol; interaction protocol for coordinating agents and interaction protocol for avoiding conflicts among agents.

The former is given to agents having joint goal or intention. By strictly following the prescribed protocols, agents can exchange a sequence of messages without considering the details of implementation of other agents. This conversation leads joint actions of multiple agents, such as consensus building and coordination. A Contract net protocol is representative of such coordination protocols in multi-agent domain. Foundation for Intelligent Physical Agents (FIPA), standards body for multi-agent technology, tries to develop specifications of standard coordination protocols in order to construct multi-agent systems in open environment like the Internet.

On the other hand, the latter is given to the society consisting of agents with different goals. The protocols define the minimum behavioral guidelines which constituent agents should obey. By sharing the protocols, every agent can expect other agents’ behavior and avoid the conflicts between agents. For example, left-hand traffic is this type of protocol. The protocol plays an important role that prevents collisions of cars. Such a protocol acts as a social constraint, so it often conflicts with agents’ goals. As a result, agents sometimes violate social constraints. That is, we can not always force agents to follow social constraints. For instance, a car with a pregnant woman can move the right side of a road to avoid a traffic jam.

In evacuation domain, there are humans with different goals; evacuee’s goal is to go out the nearest exit as early as possible and leader’s goal is to guide evacuees toward a correct exit. Hence, this paper focuses on the latter, and we try to design evacuation protocols. In this section, we introduce the existing protocol description languages and point out the problems in applying them to describe the latter protocol. Then, we explain scenario description language  $Q$  which we employ in order to describe evacuation protocols.

### 2.1 Protocol Description Language

There are some representative protocol description languages such as AgenTalk [7] and COOL [1] based on finite state machine, and IOM/T [3] equal to interaction diagrams of AUML [12] which is modeling language focusing on sequence of message exchanges between agents. AgenTalk presents clear interfaces between agents and the given protocols, called *agent-related functions*, for specifying general protocols to adapt to each application domain. Functions specific to each agent are implemented as call-back functions which are invoked from protocols using the agent-related functions. COOL provides continuation rules that say the subsequent protocols for agents to

handle several protocols given to them. IOM/T generates skeleton classes from described protocols.

As mentioned above, these protocol description languages provide just three types of choices for agents; whom to interact with, what protocols to employ, and what message content to send to other agents. This is because these languages are designed so that protocols described in them can define every interaction between agents and completely control the agents for coordination among them. That is, the protocols described in these languages limit agent’s autonomy.

Therefore, we employ scenario description language  $Q$ , a protocol description language to request agents to do something. This language delegates to agents decisions about whether or not to follow requests. Therefore, it enables agents to do actions other than protocols and violate protocols.

### 2.2 Scenario Description Language $Q$

Scenario description language  $Q$  is a protocol description language to define the expected behavior of agents [6]. In  $Q$ , protocols are represented by finite state machine and called *scenario*. Scenarios consist of *cues*, requests to observe environment, and *actions*, requests to affect environment.

$Q$  scenarios are interpreted by  $Q$  interpreter which is designed to connect with legacy agents. All  $Q$  interpreter can do is to send request messages to agents and receive the result messages from the agents. It does not consider how to execute the requests. If protocol designers respect the agent autonomy, the level of abstraction of cues and actions is high. On the other hand, if they require the degree of preciseness of protocols, the vocabularies are defined concretely.

In fact,  $Q$  is employed in realizing evacuation simulation [9] and socio-environmental simulation [14], both of which need complex social interactions.

## 3. Protocol Design Process

The existing protocol development process consists of the following five steps: analysis, formal description, validation, implementation, and conformance testing. In this process, correctness of the designed protocol is assured by checking whether a deadlock exists in the protocol or not, and whether the protocol can terminate or not [5][8]. This process is realized by assuming that every agent complies with interaction protocols and the protocols describe every interaction between them.

On the other hand, validation of protocols as social constraints takes more than verifying the correctness of the protocols, such as deadlock freeness, liveness, and termination, since they describe only partial interaction between humans, and delegate decision making on what to do to humans. In order to check the validity of the protocols, we have to also consider human decision making. This is why participatory simulation where agents and human-controlled avatars coexist is effective to refine the protocols. Moreover, by acquiring agent’s internal model about decision making from participatory simulation results, we can modify the protocols efficiently without conducting participatory simulation many times. However, the effectiveness of the protocols strongly depends on the internal models, so we have to verify the acquired internal models whenever we obtain them. The protocol refinement process defining criteria about verification of the internal models is needed.

In this section, we describe overview of our protocol refinement process referring to Figure 1. The section 5 provides details of each step while designing evacuation protocols

**Step1: Creating Protocols**

- (1) Extract agent's action rules from the existing documents and data of previous experiments, and construct agent's internal models (M1).
- (2) Describe the initial protocols (P1) by using existing documents and experts knowledge.
- (3) Conduct multi-agent simulation. The system designers check if its result (R1) satisfies their goal (G). If it does not satisfy the goal, they repeatedly modify both of the agent's internal models and the protocols until the result of simulation is closely similar to the goal. In addition, let S be a simulation function whose arguments are agent's internal models and protocols.

**Step2: Validating Protocols**

- (1) Replace some of the agents with human-controlled avatars given the same protocols as in step 1 (P1). This participatory simulation enables us to store log data which are impossible to record in the real experiments.
- (2) Compare the result of the participatory simulation (R2) and the result in step1 (R1). System designers check if the protocols (P1) are valid for the real users.

- (3) Finish this protocol refinement process if R2 is similar to R1. Otherwise, go to the step 3.

**Step3: Modifying Agent Models**

- (1) Modify the agent's internal models (M3) using log data obtained by participatory simulation.
- (2) Conduct multi-agent simulation using the modified agent's internal models and the protocols (P1).
- (3) Compare the result of the multi-agent simulation (R3) and that of the participatory simulation (R2). System designers verify the modified agent's internal models. If they check the verification of the models, go to step 4. Otherwise, they repeatedly modify the agent's internal model until R3 is closely similar to R2.

**Step4: Modifying Protocols**

- (1) Modify the protocols (P2) in order to efficiently control a group of agents based on the agent's internal model (M3) and satisfy the goal.
- (2) Conduct multi-agent simulation using the modified protocols (P2).
- (3) Compare the result of multi-agent simulation (R4) and the ideal result (R1). The system designers verify the modified protocols. If they check the verification of the modified protocols, go to step 2 again in order to confirm if the modified protocols are valid for the real users. Otherwise, they repeatedly modify the protocols until R4 is closely similar to R1.

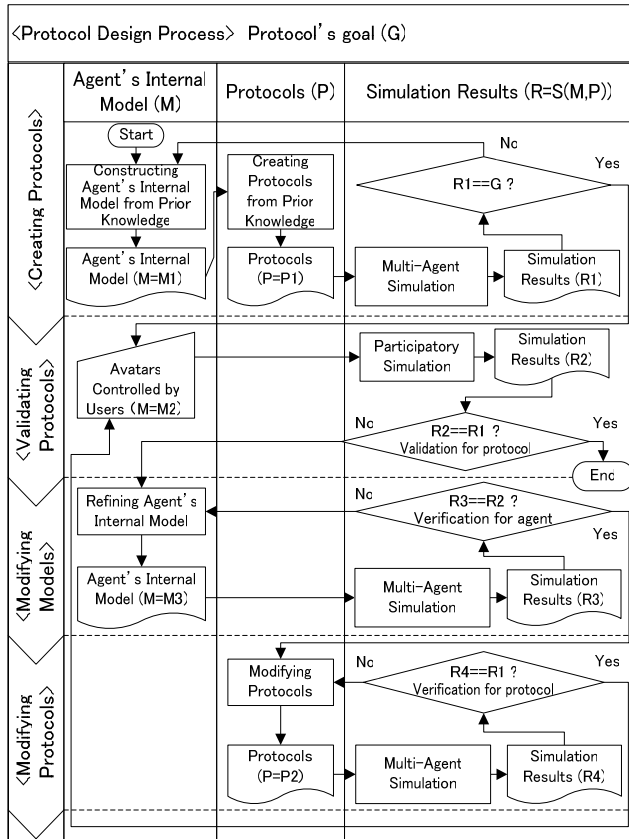


Figure 1. Protocol Design Process.

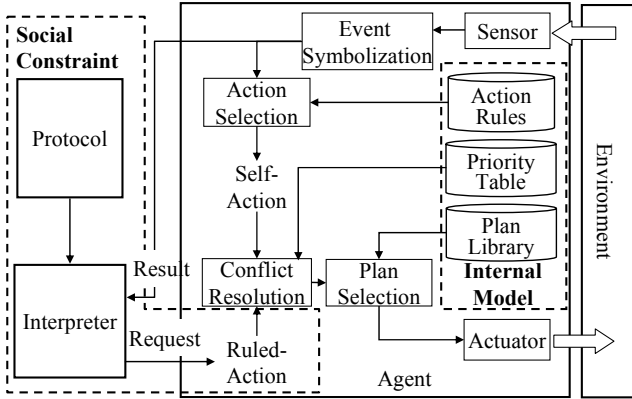
**4. Agent Architecture with Social Constraints**

In contrast to the existing interaction protocols whose goals are to realize joint actions by multiple agents, our target protocols are to accomplish individual actions without conflicts with other agents. Such a difference of attitudes towards the protocols changes agent architecture. In the existing agent architecture, the given protocols are embedded so that the behavior of traditional agents can be strictly controlled by the protocols. To construct such agent architecture, there are two approaches; implementing the protocols as agent's behavior [2], and deploying a filtering function between each agent and its environment in order to control interactions [4].

On the other hand, agent architecture with social constraints has to realize decision making on whether an agent follows the constraints so that it can achieve its own goals. Therefore, agent's decision making needs to be separated from interpretation of protocols. This architecture enables agents to deviate from the protocols by dealing with requests from the protocols as external events like their observation. In the following sections, we discuss design and implementation of agent architecture with social constraints.

**4.1 Design of Agent Architecture**

In this research, we need agent architecture that enables agents to select either proactive action or action described in the protocols. Implemented based on this architecture, agents can autonomously decide to perform next action according to their local situation. Figure 2 shows the agent architecture with social constraints, which we design.



**Figure 2. Agent Architecture with Social Constraints.**

In this architecture, observations which an agent senses are symbolized as external events, which are passed into the action selection. At the action selection, an executable action rule is chosen from a set of action rules depending on the received events. Action declared in an action rule is sent to the conflict resolution as proactive action the agent wants to perform.

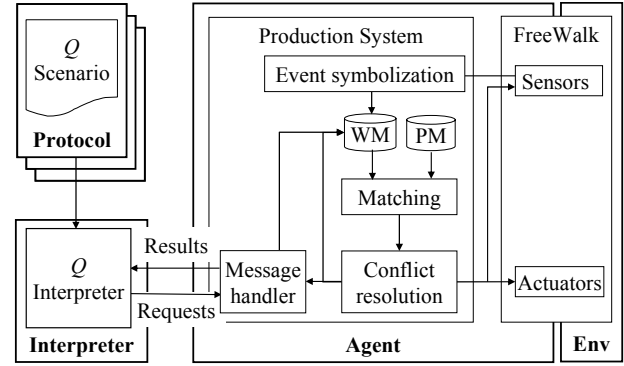
On the other hand, a protocol given to the agent is interpreted by the interpreter that also sends a request of sensing and acting to the agent. If the agent observes an event as described by the protocol, the external event is passed into the interpreter outside the agent as well as action selection within the agent. Interpreter interprets the given protocol and requests the agent to perform an action subsequent to the observation. The action prescribed in the protocol is also sent to conflict resolution.

The conflict resolution chooses only one action from the set of actions received from both of the action selection and the interpreter, depending on priorities of the actions. The chosen action is realized by employing the corresponding plans in the plan library. The effect of executing plans is given to the environment by its actuators. Information concerning the chosen action is kept in conflict resolution until the action is completed. This is used to compare priorities between the ongoing action and the new received action. If the priority of the new received action is higher than one of the ongoing action, the agent stops the ongoing action and starts to execute the new action instead of it.

In this way, the agent's behavior depends on a set of action rules leading proactive action, a table of priorities between actions used at conflict resolution, and a plan library storing how to realize the agent's action. Therefore, we define these three components as *agent's internal model*. Especially, a table of priorities between actions is the most important component to determine the agent's personality; social or selfish. If the proactive action is superior to the action prescribed in the protocol, it means a selfish agent. On the contrary, if these priorities are reversed, it means a social agent.

## 4.2 Implementation of Agent Architecture

To implement agent architecture with social constraints, we employ scenario description language  $Q$  and production system for description of a protocol and construction of decision making, respectively. The merit of the separation between protocol description and model description is that protocol designers and a-



**Figure 3. Implementation of Agent Architecture.**

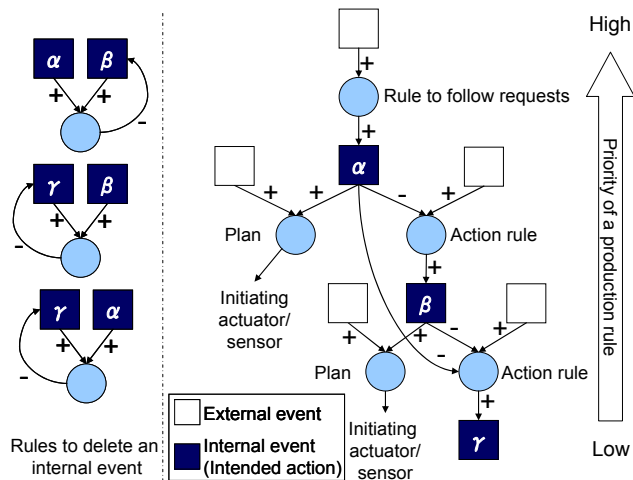
gent modelers have only to describe objects of their own interest. This implementation is shown in Figure 3.

In this implementation, three components indicating agent's internal model, a set of action rules, a table of priorities, and a plan library, are represented as prioritized production rules. All the rules are stored in production memory (PM). However, execution of plans depends on an intended action as well as external events, so it is necessary to add the following condition into the precondition of plans: "if the corresponding action is intended." For example, a plan like "look from side to side and then turn towards the back" needs the condition: whether or not it intends to search someone. By adding another condition concerning intention into the precondition of plans, this implementation controls condition matching in order not to fire unintended plans. Therefore, when any action is not intended, only the production rules representing action rules are matched against the stored external events. The successfully matched rule generates an instantiation to execute the rule, and then it is passed into the conflict resolution.

On the other hand, a  $Q$  scenario, a protocol, is interpreted by  $Q$  interpreter. The interpreter sends a request to the agent according to the given  $Q$  scenario. The request is passed into the agent through the message handler which transforms the request message to a working memory element in order to store it in the working memory (WM). Prepared in the PM, the production rule denoting "a requested action is intended to perform" can make an instantiation of the rule and send it to the conflict resolution.

Finally, the conflict resolution selects the action whose priority is highest. Thus, if the above production rule "a requested action is intended to perform" is superior to other production rules representing action rules, the agent socially behaves complying with the protocol. Conversely, if production rules corresponding to action rules are superior to the above action rule to follow the request, the agent selfishly behaves ignoring the request.

However, although such priorities enable an agent to resolve a conflict between concurrently applicable rules, it is impossible to control instantiations generated while executing another instantiation. For example, this architecture cannot avoid executing action whose priority is lower than ongoing action. Therefore, we need to design the production rules considering data dependency between the rules. Especially, we focus on intention, because every plan execution depends on generated intention. We have to consider the case where the intention to do



**Figure 4. Data Dependency for Social Agent Model.**

action whose priority is lower than ongoing action is generated, and the reverse case.

In the former case, we add a new condition; “if there is no intention generated by the more-prioritized rules in WM”, into the precondition of the less-prioritized rules. Because intention to perform action is kept in WM until the action is completed, the new condition blocks generating instantiation whose action is less prioritized than the ongoing action, while performing the ongoing action.

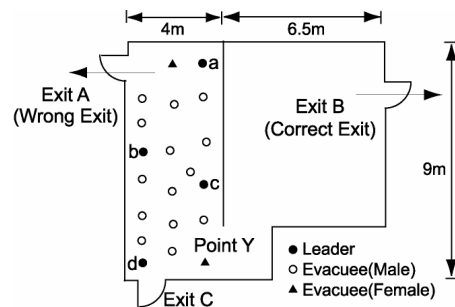
In the latter case, the problem that various intentions are in WM occurs. This state enables every plan to realize these intentions to fire all the time. Therefore, the production rule that deletes the WME denoting intention, whose action is less prioritized than others, is necessary.

Figure 4 shows the data dependency among production rules which compose social agents. Circles and squares mean production rules and WMEs, respectively. Arrow lines represent reference and operation towards WMEs. Specifically, an arrow line from a square to a circle represents reference to the data, while a reverse arrow line represents operation of the data.

## 5. Design of Evacuation Protocols

In disaster-prevention domain, simulations can contribute to evaluation of contingency planning and analysis of second disaster, since it is difficult to conduct experiments in the real world. Traditional simulations ignore the differences between people and treat everyone as uniform bits with the same simple behavior. These simulations are employed in order to evaluate construction of a building. Human action is, however, predicted from a numerical analysis of just spatial position without considering social interactions such as guidance although social interaction is extremely common and strongly influences the responses seen in real-world evacuations. Therefore, we conduct evacuation simulations considering social interactions in order to design evacuation protocols.

As a first step to addressing the problem of designing evacuation protocols, we simulated the controlled experiments conducted by Sugiman [13]. He established a simple environment with human subjects to determine the effectiveness of two evacuation methods: the “Follow-direction method” and the “Follow-me met-



**Figure 5. Ground Plan of the Experiment and Initial Position of Subjects [13].**

hod.” In the former, the leader shouts out evacuation instructions and eventually moves toward the exit. In the latter,

the leader tells a few of the nearest evacuees to follow him and actually proceeds to the exit without verbalizing the direction of the exit. Sugiman used university students as evacuees and monitored the progress of the evacuations with different number of leaders.

The experiment was held in a basement that was roughly ten meters wide and nine meters long; there were three exits, one of which was not obvious to the evacuees as shown in Figure 5. The ground plan of the basement and the initial position of subjects are also shown in the figure. Exit C was closed after all evacuees and leaders entered the room. At the beginning of the evacuation, Exit A and Exit B were opened. Exit A was visible to all evacuees, while Exit B, the goal of the evacuation, was initially known only by the leaders. Exit A was treated as a danger. Each evacuation method was assessed by the time it took to get all evacuees out.

### 5.1 Step1: Creating Protocols

In our past research, we succeeded in double-checking the result of the previous fire-drill experiment by multi-agent simulation [9]. However, the previous simulation employed the simplest agent's internal model, which only followed the given protocols. That is, every interaction was described as interaction protocols. Therefore, we have to redesign interaction protocols with an appropriate degree of abstraction so that participants can easily understand them in the next step.

At first, we redescribe interaction protocols the same as those employed in the real experiments, in the form of finite state machine. In “Follow-me method” condition, each instruction given a leader and an evacuee in the real experiments is as follow.

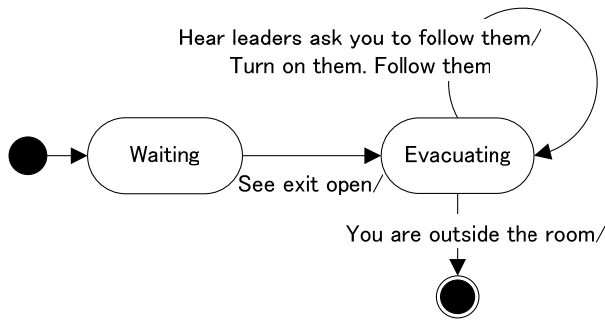
**Leader:** While turning on emergency light, put his white cap on. After a while, when the doors to this room are opened, say to an evacuee close to him “Come with me”, and subsequently move with the evacuee to Exit B.

**Evacuee:** When the doors to this room are opened, escape from the room while following direction from leaders with a white cap on.

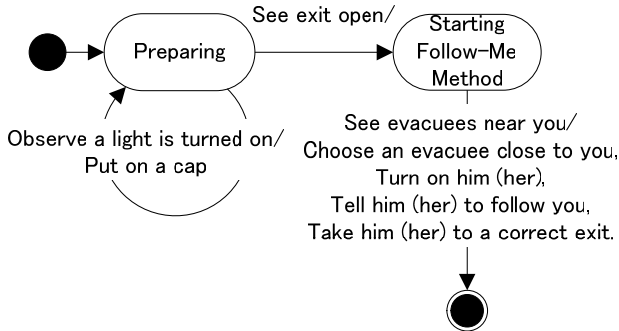
We try to extract action rules from the above instructions, and construct finite state machine by assigning each state to concurrently applicable rules. The generated finite state machines for a leader and an evacuee are shown in Figure 6 and Figure 7, respectively.

**Table 1.** Rules for Evacuation Scenarios.

Agent	Rule
Leader (Follow-me)	(Plan) When the leader goes out from the room, he checks if the target evacuee also goes out from it.
	(Plan) If the target evacuee is out of the room, the leader goes to the next exit.
	(Plan) If the target evacuee is within the room, the leader walks slowly so that he/she can catch up with him.
	(Plan) If the target evacuee goes out from the room, the leader picks up the pace and moves toward the next exit.
Evacuee (M1)	(Action rule) The evacuee looks for a leader or an exit.
	(Action rule) If the evacuee sees the exit open, he goes to the exit.
	(Action rule) If the evacuee sees a leader walk, he follows him.
	(Action rule) If the evacuee sees another evacuee close to him move, the evacuee follows him.
	(Plan) In order to look for a leader or an exit, the evacuee looks from side to side.
	(Plan) If the evacuee observes that someone he follows goes out from the room, he walks towards the exit.
Evacuee (M3)	(Plan) If the evacuee also goes out from the room, he follows the same target again.
	(Action rule) If the evacuee sees the people around him walk, it also walks towards the same direction.
	(Action rule) If the evacuee sees congestion in the direction of his movement, he looks for another exit.
	(Plan) In order to look for a leader or an exit, the evacuee turns towards the back.
	(Plan) In order to look for a leader or an exit, the evacuee turns on the same direction as the people around him.



**Figure 6.** Protocol for “Follow-me method”.



**Figure 7.** Protocol for Evacuees.

On the other hand, the difference between the previous protocols and the redescribed protocols is an agent's internal model. An agent's internal model consists of a set of action rules, which generates intention to perform a proactive action, and a set of plans, which realize the intention to do an action. Hence, we have to classify the left rules into two sets; action rules and plans. The criterion to classify the rules is what purpose the rule is used for. The rule that realizes the same goal as the given protocol is an action rule, while the rule that realizes other behavior is a plan. In the case of an evacuee, the following rules are action rules to realize evacuation; “go to the exit in his view,” “look for a leader,” and “follow someone close to him.” These action rules

generate intention to perform actions; “go,” “look for,” and “follow.” The means to realize these actions is a plan. Action rules and plans in “Follow-me method” condition are shown in Table 1. Note that leaders have no action rules since they completely obey their protocol.

Next, we conduct multi-agent simulation with the acquired protocols and agent's internal models. By simulating in three-dimensional virtual space like FreeWalk [11], it is easy to realize participatory simulation in the next step.

## 5.2 Step2: Validating Protocols

In the second step, we conduct participatory simulation by replacing some agents with human-controlled avatars. Participatory simulation enables us to record various data impossible to collect in the real experiments.

The purpose of participatory simulation is validation of the protocol described in the previous step. To accomplish this purpose, we instruct subjects on the evacuation protocol before participatory simulation, and then check if the result of participatory simulation satisfies the original goal. If it does not satisfy the goal, we have to modify the agent's internal model to more valid one by noting the difference between results of simulations.

In fact, we conducted participatory simulation by replacing twelve evacuee agents with subject-controlled avatars and instructing the subjects on the evacuation protocol. The other eight agents including four leaders and four evacuees were still the agents having been used in the previous step. We collected the results in the four leader “Follow-me method” condition and the four leader “Follow-direction method” condition, respectively.

In consequence, only the result of participatory simulation in the four leader “Follow-me method” condition was different from the result of multi-agent simulation. Figure 8 shows the situation reproduced on two-dimensional simulator. As shown in the figure, the circled evacuee avatar looked around it in the early stage, and after emergence of congestion, it walked towards the wrong exit in order to avoid the congestion.



