

# Scenario Description for Multi-Agent Simulation

Yohei Murakami<sup>1</sup>, Toru Ishida<sup>1,2</sup>, Tomoyuki Kawasoe<sup>1</sup> and Reiko Hishiyama<sup>1</sup>

<sup>1</sup>Department of Social Informatics, Kyoto University

<sup>2</sup>JST CREST Digital City Project

Kyoto, 606-0801, Japan

+81 75-753-5398

{yohei, kawasoe, hishiyama}@kuis.kyoto-u.ac.jp, ishida@i.kyoto-u.ac.jp

## ABSTRACT

Making it easier to design interactions between agents and humans is essential for realizing multi-agent simulations of social phenomena such as group dynamics. To realize large-scale social simulations, we have developed the scenario description languages  $Q$  and IPC (Interaction Pattern Card); they enable experts in the application domain (often not computing professionals) to easily create complex scenarios. We have also established a four-step process for creating scenarios: 1) defining a vocabulary, 2) describing scenarios, 3) extracting interaction patterns, and 4) integrating real and virtual experiments. In order to validate the scenario description languages and the four-step process, we ran a series of evacuation simulations based on the proposed languages and process. We successfully double-check the result of the previous controlled experiment done in a real environment.

## Categories and Subject Descriptors

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

**General Terms:** Design, Languages

## Keywords

Multi-agent Simulator, Scenario Description, Social Interaction, Evacuation Simulation

## 1. INTRODUCTION

Various physical models such as the magnetic model and the liquid model have been used to simulate social systems (economic phenomena, traffic flow and so on). With a large number of “entities,” those models can produce behaviors that well mirror real situations. However, since there is no difference between the entities, these types of simulations cannot treat “atoms” as individuals; this is inherently limiting as humans behave in quite different ways.

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.

*AAMAS '03*, July 14–18, 2003, Melbourne, Australia.

Copyright 2003 ACM 1-58113-683-8/03/0007...\$5.00.

A multi-agent simulation models each individual as an agent instead of modeling them as a physical system. This method has been used to analyze social systems and to synthesize realistic situations. Moreover, once simulators become accessible to humans via the Internet, multi-agent simulations allow humans to join experiments with software agents.

Multi-agent simulations can be applied to various areas, and they are currently being used to simulate social systems like traffic, urban planning, and politics. The evacuation simulation discussed in this paper is another direction [6][9]. When simulating social phenomenon in any detail, we need to describe various social interactions including verbal communication. For experts who work in the application domain (hereafter scenario writers), writing a scenario to control a group of agents is not easy, because most are not computing professionals. As the agents’ functions and environments become more complex (imagine a large number of human-like agents in a three-dimensional virtual space), the difficulty of writing scenarios causes a serious problem. In order to remove this problem and to help scenario writers with multi-agent simulations, we set up the following three research objectives.

- *Develop scenario description languages:* We need a general purpose scenario description language to describe social interaction between humans and agents that employs the vocabulary used in the application domain. We also need to provide a domain dependent language to capture the patterns of interaction observed in the application domain.
- *Establish a scenario description process:* Unlike computer programming, no specification is given in advance for a scenario description. We need to establish a process for scenario development that determines vocabulary, describes scenarios, extracts interaction patterns, and integrates real world observations with virtual world simulations.
- *Validate technologies using real-world problems:* We need to validate the scenario description languages and scenario description process by examining how well the scenario-based simulation accurately reproduces real life situations.

This paper will first explain what a multi-agent simulation is, and then clarify the application domain of our approach. Next, we will discuss the scenario description languages  $Q$  and IPC and the four-step process for scenario description. Finally, we will validate our approach in an evacuation domain by comparing the results of our simulation with those of the real-world experiment conducted by Sugiman in 1988 [13].

## 2. MULTI-AGENT SIMULATION

Multi-agent simulations can be roughly divided into two groups based on the granularity of the agent model; a) multi-agent simulation with a simple internal model (hereafter referred to as Fine-Grain) and b) multi-agent simulation with a complex internal model (hereafter referred to as Coarse-Grain).

### a) Fine-Grain:

This type of simulation is used in the analysis of complex social systems. Here, the KISS principle (Keep It Simple, Stupid) is often applied [1]. The KISS principle states that agent modeling should be simple even though the observed phenomenon is complex, and that complexity should be a result of the simulation. Hence, agents are expressed using a simple computational model that incorporates limited parameters. This approach is mainly used in the analysis of the relationship between the macro properties of the entire system and the micro properties of the agents constituting the system [4].

### b) Coarse-Grain:

This type of simulation is used for the reproduction of reality-in situations. Agent models reflecting the real world are created to make the simulation as realistic as possible [5]. This approach is used in an early stage of system development (examples include robots and plants) [2][7][10], in the examination of strategies for decision making [3][14], in education or training, and so on.

While the latter requires a scenario to describe the interaction between agents and humans, the former does not since it uses a simple agent model. This paper focuses on the latter, and assumes running simulations that focus on virtual experiences like evacuation drills.

## 3. SCENARIO DESCRIPTION LANGUAGE

### 3.1 Interaction Design Language $Q$

$Q$  is an interaction design language that describes how an agent should behave and interact with its environment involving humans and other agents. For details see [8]. The salient features of  $Q$ 's language functionality are summarized as follows.

- Cues and Actions

An event that triggers interaction is called a *cue*. Cues are used to request agents to observe their environment. No cue is permitted to have any side effect. Cues keep on waiting for the event specified until the observation is completed successfully. Comparable to cues, *actions* are used to request agents to change their environment.

- Scenarios

Guarded commands are introduced for the situation wherein we need to observe multiple cues simultaneously. A guarded command combines cues and actions. After one of the cues becomes true, the corresponding action is performed. A scenario is used for describing state transitions, where each state is defined as a guarded command. Scenarios can be called from other scenarios.

- Agents and Avatars

Agents, avatars and a crowd of agents can be defined. An agent is defined by a scenario that specifies what the agent is to do. Even if a crowd of agents executes the same scenario, the agents exhibit different actions as they interact with their local environment (including other agents and humans). Avatars controlled by humans do not require any scenario. However, avatars can have scenarios if it is necessary to constrain their behavior.

By introducing cues and actions, we can fluently describe the interaction between agents and their environment. We can also describe agents' behaviors according to their states by allowing the description of state transition.  $Q$  is currently implemented on the top of Scheme<sup>1</sup> so its language specifications are easy to extend.

### 3.2 Interaction Pattern Card (IPC)

Various agent interactions can be described using the  $Q$  language, but scenario writers may not be familiar with the syntax of Scheme, a mother language of  $Q$ . Furthermore, since  $Q$  is a general-purpose language, it exhibits too much freedom when writing scenarios for specific domains. To scenario writers, ease of use is often more important in a language than its scope of coverage. To support scenario description, we developed Interaction Pattern Card (IPC). Note that IPC is not merely a user interface for  $Q$ , it is a language to express interaction patterns.

The "card grammar" of this language describes the fundamental syntax and semantics of cards. The "card grammar" consists of state transitions expressed as columns, and cues (sensing) and actions expressed as rows. Labels like row name and column name are of great importance, since they indicate states and cues/actions respectively.

Figure 1 shows the translation process from IPC to  $Q$  scenario. For translating an interaction pattern card into a  $Q$  scenario, one can assign semantics to the structure of the card in the form of a macro definition. The assignment of semantics is described as the definition of an interaction pattern card (IPC definition), and then sent to the IPC translator, which generates  $Q$ . The card is filled using Excel and sent to It PC translator as data in CSV format. The IPC translator generates a  $Q$  scenario from the IPC card by expanding the macros based on the IPC definition given in advance.

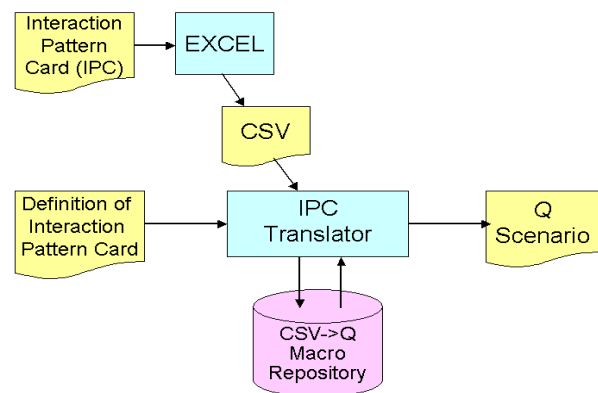


Figure 1. IPC/ $Q$  Translation Process.

<sup>1</sup> Scheme is a dialect of the Lisp programming language invented by Guy Lewis Steele Jr. and Gerald Jay Sussman.

To define a card, scenario writers must extract an interaction pattern from each domain. If several already written scenarios are available, it is not too difficult to extract an interaction pattern from the scenarios. Once the card is defined, the scenario writers can use IPC to write scenarios without needing to be aware of the details of the  $Q$  language.

The introduction of IPC not only provides a simple means of writing scenarios, but also facilitates dialogs between scenario writers and agent system developers. As a result, IPC becomes an interface to promote clear understanding of a simulation from both two sides.

#### 4. EVACUATION DOMAIN

An evacuation simulation is a good example of a social simulation. Traditional simulators ignore the differences between people and treat everyone as uniform bits with the same simple behavior. Human action is predicted from a numerical analysis of just spatial position without considering social interactions such as what other people are saying or doing. Social interaction is, however, extremely common and strongly influences the responses seen in real-world evacuations. For example, if the evacuees hear someone scream, they tend to avoid that area.

As a first step to addressing the problem of simulating evacuation situations that include social interaction, 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 method.” 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 2. 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.

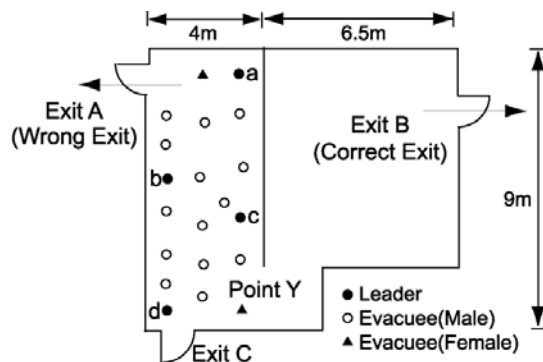


Figure 2. Ground Plan of the Experiment and Initial Position of Subjects [13].

The experiment was run four times with new evacuees and leaders each time. The two methods were tested with two and four leaders each; there were sixteen evacuees in every run. The results are plotted on the left-hand side of Figure 6. All sixteen evacuees were successfully guided to Exit  $B$  in all runs except for the “Follow-me method” with two leaders. In that case, as shown in Figure 6, eleven evacuees left the room via Exit  $A$  (the wrong exit), and five evacuees via Exit  $B$  (the correct exit).

### 5. SCENARIO DESCRIPTION PROCESS

#### 5.1 Overview of Process

In order to run a successful multi-agent simulation, scenario writers must provide appropriate scenarios to agents. A scenario, however, differs from a program in that no explicit specification is given in advance. Hence, the process for describing software cannot be applied to scenario description. It is necessary to create a process that models agents at an appropriate level of abstraction by observing the real world. We propose the following four-step process for creating scenarios.

##### STEP1: Defining a Vocabulary

A scenario writer and an agent system developer agree upon cues and actions as the interface between them. Note that cues and actions are not provided a priori but are defined for each application domain.

##### STEP2: Describing Scenarios

The scenario writer describes scenarios using the  $Q$  language, while the agent system developer implements cues and actions.

##### STEP3: Extracting Interaction Patterns

Patterns of interaction are extracted from several already written scenarios, and are used to define interaction pattern cards. The scenario writers describe scenarios by using the cards, and use their experience to improve the definition of the cards.

##### STEP4: Integrating Real and Virtual Experiments

The scenario writer conducts experiments in both real and virtual environments. The scenario writer utilizes the knowledge obtained from the experiment in the real world to refine the original scenarios. Real experiments also benefit from simulations. Examining the outcome of virtual experiments can improve the design of the real experiments.

The following subsections provide details of each step using our experiments on the evacuation domain.

#### 5.2 STEP1: Defining a Vocabulary

In the first step, cues and actions in the given domain are defined as follows through a dialog between a scenario writer and an agent system developer.

1) The appropriate level of abstraction of cues and actions depends on two independent factors: the level of agent autonomy, and the degree of preciseness required by the scenario writer. The selection of cues and actions depends on the application. For example, an agent that introduces Web pages needs to have cues and actions for conversation rather than those for physical gestures. On the other hand, in an evacuation simulation, physical gestures are more important than conversation.

**Table 1. Sample Cues and Actions for Evacuation Simulation.**

	Cue	Action
Motion	? <b>position</b> (get location and orientation) ? <b>observe</b> (observe gestures and actions) ? <b>see</b> (see objects)	Movement ! <b>walk*</b> (walk along a route) ! <b>approach*</b> (approach to other agents) ! <b>block*</b> (block other agents)
		Rotation ! <b>turn*</b> (turn a body) ! <b>face*</b> (turn a head)
		Gesture ! <b>point*</b> (point objects) ! <b>behave*</b> (perform some gesture)
		Appearance ! <b>appear</b> (show up) ! <b>disappear</b> (erase itself)
Conversation	? <b>hear</b> (hear voice) ? <b>receive</b> (receive text messages) ? <b>answer</b> (receive an answer to questions)	! <b>speak*</b> (speak by voice) ! <b>send*</b> (send text messages) ! <b>ask*</b> (ask questions)
Others	? <b>finish</b> (finish asynchronous actions) ? <b>input</b> (key input by users)	! <b>change</b> (change agent's image) ! <b>finish</b> (stop asynchronous actions) ! <b>output</b> (output logs)

2) Defined cues and actions are classified according to functions. The actions that return after completion are commonly called *synchronous actions*. *Asynchronous actions*, however, allow other actions to be executed in parallel. Note that asynchronous actions classified into the same group are not executed concurrently.

3) The determined cues and actions form an interface between the scenario writer and the agent system developer. The scenario writer can describe a scenario without knowing the details about of agent system implementation. That is, if multiple agent systems have the same cues and actions, the same scenario can be used in all systems.

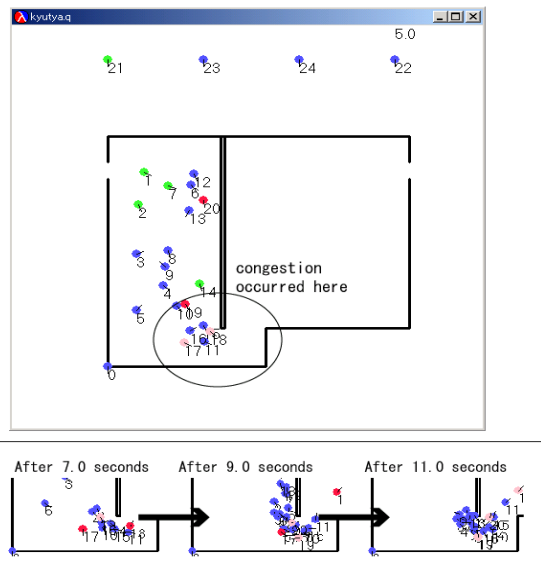
After studying Sugiman's experiment, we developed a vocabulary suitable for evacuation simulations; Table 1 summarizes the cues and actions. Actions with an asterisk can be used as both synchronous and asynchronous actions.

Cues and actions roughly fall into three groups: motion, conversation, and others. With regard to action, motion can be subdivided into movement, rotation, gesture, and appearance. By employing asynchronous actions, multiple actions can be concurrently executed. Actions within the same group cannot, however, be executed concurrently.

Once the vocabulary is determined, the agent system developer implements cues and actions. Two simulators were used in our evacuation simulation; one for two-dimensional space and the other for three-dimensional space. These two different simulators can use the same scenario, since they share the same cues and actions. The two-dimensional simulator is shown in Figure 3.

We use FreeWalk as the platform for the three-dimensional simulation [12]. It enables agents to interact with nonverbal cues; for example, gestures like pointing at something can be used. An example of a FreeWalk screen is shown in Figure 4.

One big difference between the two-dimensional and three-dimensional simulators is that the latter allows humans to project their avatars into the virtual space. Namely, three-dimensional simulators provide people with a vicarious experiential learning environment wherein evacuation or other emergency drills can be experienced.



**Figure 3. 2-D Evacuation Simulation.**



**Figure 4. 3-D Evacuation Simulation.**

**Table 2. Rules for Evacuation Scenarios.**

Agent	Rule
Leader (Follow-direction)	When Exit <i>A</i> is opened, the leader puts his cap on.
	If the leader is standing near the wrong exit, he proceeds to that exit.
	If he arrives at the wrong exit, he prevents other evacuees from escaping through that exit.
	When the leader sees someone and is in the left room, he indicates the direction of point <i>Y</i> with a loud voice.
	When the leader sees someone and is in the right room, he indicates the direction of Exit <i>B</i> with a loud voice.
	When the leader spots someone heading towards the wrong exit, he shouts out a warning.
	When the leader finds that all the evacuees around him are correctly evacuating, he joins them.
Leader (Follow-me)	When the leader finds an evacuee who is not moving, he directly encourages him to move to the exit.
	When Exit <i>A</i> is opened, the leader puts his cap on.
	When the leader spots an evacuee in front of him at the beginning of the evacuation, he walks up to him.
	If the leader cannot spot any evacuee, he looks around.
	When the leader approaches the nearest evacuee, he instructs the evacuee to escape by saying, “follow me.”
	After instructing the evacuee, the leader passes point <i>Y</i> and proceeds to Exit <i>B</i> .
	When the leader notices that the evacuees following him are falling behind, he waits for them to catch up.
Evacuee	When the leader sees that the evacuees have caught up, he resumes heading toward the evacuation point (Exit <i>B</i> ).
	When the leader finds that all evacuees following him are around Exit <i>B</i> , he stops guiding.
	When Exit <i>A</i> is opened, the leader looks at the exit.
	If the evacuee sees an exit near him, he heads toward it.
	If the evacuee hears the leader shouting out the direction, he looks at the leader.
	If the evacuee hears the leader instructing him to follow him, he follows the leader.
	If the evacuee sees where the leader is pointing at, he goes in the direction indicated by the leader.
	If the evacuee sees that the leader is walking, he follows the leader.
	If the evacuee is separated from the leader, he looks around.
	If each evacuee sees leaders indicate a direction within a fixed distance, he follows their directions.
If each evacuee hears any shouting of directions from several leaders simultaneously, he ignores their directions.	
An evacuee moves toward the nearest group of evacuees.	
An evacuee does not move until the crowd around him moves.	
An evacuee follows the people around him.	

### 5.3 STEP2: Describing Scenarios

In the second step, scenario writers examine the scenarios required for the simulation by analyzing various pieces of information, and describe agents’ scenarios as shown below:

- 1) From prior knowledge and documents, the scenario writer examines what kind of agents with what roles are needed in the simulation. Once the roles are determined, the scenario writer extracts rules and state transitions according to the role to construct each scenario using the agreed cues and actions. If cues and actions are incomplete, the scenario writer discusses the omissions with the agent system developer, and modifies the definition of cues and actions as appropriate.
- 2) Extracted rules are assigned to sets of concurrently applicable rules. As states are assigned to these sets, the scenario begins to take form. Since scenarios can call other scenarios, we can extract the shared part of the scenario from several scenarios and independently describe it to avoid redundancy.
- 3) A scenario for each role is assigned to the agent taking that role.

With regards to the evacuation simulation, we reviewed the technical papers written by Sugiman, and designed scenarios for three roles, which are a scenario for leaders employing the “Follow-me method”, a scenario for leaders employing the “Follow-direction method”, and a scenario for evacuees. Table 2 shows a list of all the rules forming each scenario.

These extracted rules use cues and actions shown in Table 1. For example, the scenario of the leader agent employing the “Follow-me method” is displayed on the right-hand side of Figure 5. The contents of the scenario are as follows.

Initially, the leader waits for Exit *A* to open. When Exit *A* is open, he puts on a cap to distinguish himself from other evacuees. If he spots any evacuee in front of him, he approaches the evacuee. If he cannot spot any evacuee, he looks around to see if there is any and approaches the nearest evacuee. Once he approaches an evacuee, he instructs the evacuee to escape by saying, “follow me” and passes point *Y* on the way to Exit *B*. If the evacuee falls behind during the guided evacuation, the leader stops and waits for the evacuee to catch up. Once the evacuee catches up, the leader resumes his movement toward the exit. If the evacuee arrives at Exit *B*, he leaves the room and the guided evacuation terminates.

In this figure, cues and actions start with “?” and “!” respectively. To represent asynchronous actions, we use the notation “!!”.

### 5.4 STEP3: Extracting Interaction Patterns

In the third step, we extract interaction patterns specific to the intended application domain from the accumulated scenarios as shown below:

Card ID	Card Name	Object of Observation	State	Card Type	Guiding Actions
1	Follow-me			Guidance Card	
2					
3					
4	Initiate Guidance	Exit A	Open		Put on a cap Choose an evacuee (one closest to me) Approach (Evacuee) Speak (Follow me) Start to walk (Point Y, Exit B)
5					
6					
7					
8	Guidance [Condition]	Evacuee's position/direction	My position/direction		Guiding Actions
9					
10		Evacuee's position/direction	My position/direction		Guiding Actions
11		More than 3.0m apart from me			Stop Turn direction (Evacuee)
12	Guidance [Repeat]	Within 1.5m from me	Positioned at the left room		Start to walk (Point Y, Exit B)
13			Positioned at the right room		Start to walk (Exit B)
14		Exit B			[Terminate Repeat]
15					
16	Terminate Guidance	Guiding Actions			
17		Walk (Outside the room)			

```
(defscenario Follow-Me-Method ()
  (let ((target '()))
    (scene1
      ((?observe :name Exit A :state Open)
        (!change :image Putting-on-a-Cap)))
      (set! target (Choose-Closest-Evacuee))
      (!approach :to target)
      (!speak :to target :sentence "Follow me")
      (!!walk :route (list Point-Y Exit-B))
      (go scene2)))
    (scene2
      ((?position :name target
        :distance_range '(3.0 10.0) :from me)
        (!finish :action "walk")
        (!!turn :to target)
        (go scene2)))
      ((?position :name target
        :distance_range '(0.0 1.5) :from me)
        (guard
          ((?position :name me :at Left-Room)
            (!!walk :route (list Point-Y Exit-B))
            (go scene2)))
          ((?position :name me :at Right-Room)
            (!!walk :route (list Exit-B))
            (go scene2))))
      ((?position :name target :at Exit-B)
        (go scene3)))
    (scene3
      (#t
        (!!walk :route (list Outside))))))
```

Figure 5. An Interaction Pattern Card (EXCEL Interface) and its Corresponding Q Scenario.

1) We seek a common scenario structure from several scenarios. A common scenario structure is defined as a control structure for common state transition or states that use common cues and actions.

2) Based on the scenario structure that has been discovered, we determine the card structure following the IPC “card grammar.” Once the card structure is determined, rows and columns are labeled to support our understanding of the card structure.

3) Finally, we assign semantics to the card structure in the form of macro definitions, which are used to generate Q scenarios by the IPC translator. The scenario writer describes a scenario using the card in Excel, and translates it into a Q scenario using the translator. IPC reduces the scenario writer’s burden in using the Q language, and improves the productivity of scenario writing. Moreover, the scenario writer can utilize the experience describing scenarios with IPC to further modify interaction pattern cards. Once the card is defined, it triggers a dialog between the scenario writer and agent system developer to exchange ideas on modeling agents.

Let us try to generate an interaction pattern card for the leader agents that employ the “Follow-me method” and the “Follow-direction method.” Four structures were discovered from already written scenarios; they are “initiate guidance” “determine a guidance method,” “repeat guidance,” and “terminate guidance.”

Next, we arrange and order these structures to form a card. In order to show the scenario writer what can be filled in and where, the rows and columns are labeled.

Figure 5 shows the IPC of a leader agent composed in this fashion. The blanks in the IPC shown in the figure were filled to describe the scenario of a leader agent employing the “Follow-me

method.” This IPC is equivalent in meaning to the scenario given on the right. By generating an IPC that focuses on the domain of evacuation, we could provide a simple means of writing scenarios.

### 5.5 STEP4: Integrating Real and Virtual Experiments

In the last step, we attempt to refine the scenario by comparing the results of real and virtual experiments. We do this because observing real-world experiments is vital to correctly modeling agents and to write suitable scenarios:

1) We run a simulation based on the scenario constructed using prior knowledge and documents, and compare it to the result of a real-world experiment. If any disparity is found, the result of the real-world experiment is analyzed. In order to eliminate the disparity, new rules obtained in the analysis are inserted into the scenario.

2) The simulation is then run again using the modified scenario. Again we compare the result of the simulation with the result of the real-world experiment. We verify which specific rule most influences the result of the real-world experiment. By repeating this cycle, the scenario is effectively refined to produce simulation results that reflect the real-world situation.

3) By changing the values of the parameters in the scenario, we can assess an environment not constructed in the real world, and simulate that environment.

In practice, we first ran an evacuation simulation using scenarios that were constructed using prior knowledge and documents. The simulation results yielded by the preliminary scenarios deviated significantly from those of the experiment. We analyzed each evacuation video recorded by Sugiman and found some other

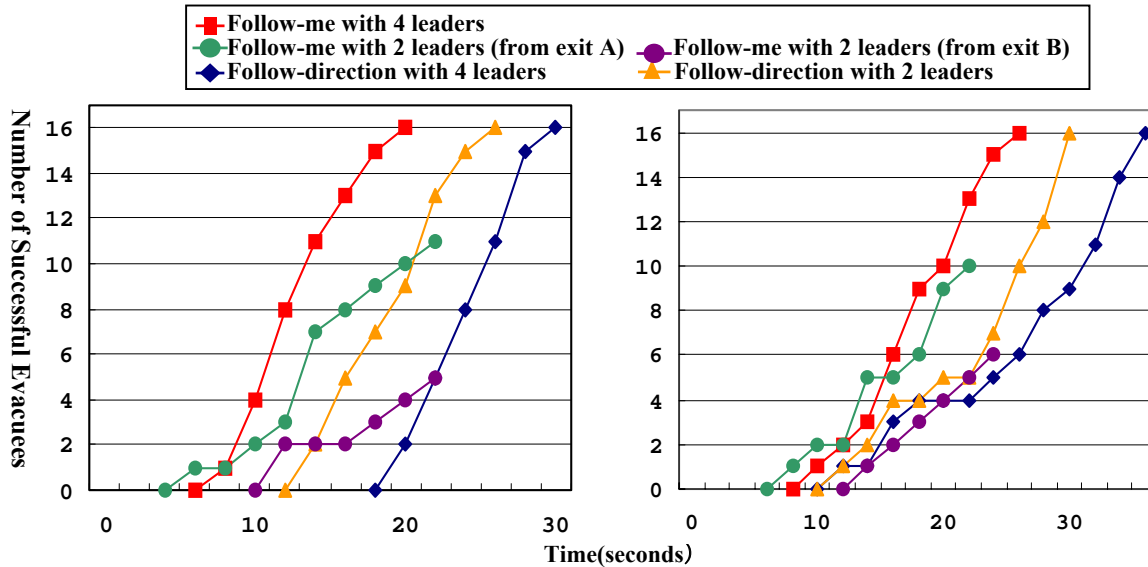


Figure 6. Comparison between Control Experiment by Sugiman (left) and Evacuation Simulation (right).

interesting behavior [11]. The evacuees' acceptance of the leader's guidance can be overruled if many of the surrounding evacuees move in a different direction, or the leader's instructions are overlaid by those of another leader. Moreover, the distance over which a leader can exert influence is not great, particularly if the room is dark or filled with smoke. Therefore, we added five rules to the evacuees' scenario. Table 2 shows the five additional rules.

We conducted the evacuation simulation again using the modified scenario, the results of which are shown in Figure 6. This simulation exhibited a similar difference between the simulated "Follow-me method" and "Follow-direction method" as that seen in the real world experiment. This confirms that the five additional rules are strong factors in the real-world experiment. Furthermore, since the simulation results closely parallel those recorded in the controlled experiment and the scenario incorporates the results of the experiment, we could conclude that the agents were reasonably modeled.

Once an appropriate model is acquired, it becomes possible to simulate an experiment that has not conducted in the real world. For example, Sugiman's experiment used only two and four leaders, but we can vary the number of leader agents in the simulation to more clearly show the relation between the number of leaders and the time required to complete the evacuation. Moreover, we can also examine the most effective evacuation method by combining the "Follow-me method" and the "Follow-direction method" in various ratios.

## 6. CONCLUSION

Describing the social interactions between agents and humans is essential if we are to conduct large-scale multi-agent simulations. With the assumption that experts in a given application domain (they are often not computing professionals) actually write scenarios for simulations, we tackled the following three problems.

- *Develop scenario description languages:* We developed the scenario description language  $Q$  which allows scenario

writers to easily describe interactions between agents and humans. We also developed IPC (Interaction Pattern Card), which provides a card interface to describe interaction patterns extracted from each application domain.

- *Establish a scenario description process:* We proposed the four-step process that consists of 1) defining a vocabulary, 2) describing scenarios, 3) extracting interaction patterns and 4) integrating real and virtual experiments. This process triggers dialogue and facilitates cooperation between computer professionals and application designers.
- *Validate technologies using real-world problems:* We obtained a result close to the one obtained in a real fire drill experiment, by using the scenario description languages and the description process that we developed.

To date, our research has used only software agents, but we plan to run a user-participating simulation on the Internet using a three-dimensional virtual space, FreeWalk [12], as the next step. By conducting simulations where agents and humans coexist, we can allow humans to vicariously experience disaster situations close to reality, making it possible to collect experience that would be difficult to obtain in the real world. Through cooperatively developing interaction scenarios by application and computing professionals, we can increase the effectiveness of multi-agent simulation in domains like crisis management.

## ACKNOWLEDGMENTS

The authors would like to thank H. Cho, H. Ito, H. Nakanishi, K. Tsutsuguchi, and A. Yamamoto for making this work possible. FreeWalk and  $Q$  have been developed by Department of Social Informatics, Kyoto University and JST CREST Digital City Project.

## REFERENCES

- [1] Axelrod, R. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton Univ. Press, pp. 4-5, 1997.

- [2] Bradshaw, J. M., Sierhuis, M., Gawdiak, Y., Jeffers, R., Suri, N., and Greaves, M. Adjustable Autonomy and Teamwork for the Personal Satellite Assistant. In *Proceedings of IJCAI-01 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*, 2001.
- [3] Clancey, W. J., Sachs, P., Sierhuis, M., and Hoof, R. V. Brahms: Simulating Practice for Work Systems Design. *International Journal of Human-Computer Studies*, 49, pp. 831-865, 1998.
- [4] Epstein, J. M., and Axtell, R. L. *Growing Artificial Societies: Social Science from the Bottom Up*. MIT Press, 1996.
- [5] Hanks, S., Pollack, M. E., and Cohen, P. Benchmarks, Testbeds, Controlled Experimentation, and the Design of Agent Architectures. *AI Magazine*, 14(4), pp. 17-42, 1993.
- [6] Hareesh, P. V., Shibano, N., Nakanishi, H., Kashiwagi, M., and Sawada, K. Evacuation Simulation: Visualisation Using Virtual Humans in a Distributed Multi-User Immersive VR System. In *Proceedings of International Conference on Virtual Systems and Multi-Media (VSMM2000)*, pp. 283-289, 2000.
- [7] Horling, B., Lesser, V., and Vincent, R. Multi-Agent System Simulation Framework. In *Proceedings of 16<sup>th</sup> IMCS World Congress 2000 on Scientific Computation, Applied Mathematics and Simulation*, 2000.
- [8] Ishida, T. *Q*: A Scenario Description Language for Interactive Agents. *IEEE Computer*, Vol.35, No. 11, pp. 54-59, 2002.
- [9] Ishida, T. Digital City Kyoto: Social Information Infrastructure for Everyday Life. *Communications of the ACM (CACM)*, Vol. 45, No. 7, pp. 76-81, 2002.
- [10] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., and Osawa, E. RoboCup: The Robot World Cup Initiative. In *Proceedings of the Agents 1997*, 1997.
- [11] Milgram, S., Bickman, L., and Berkowitz, L. Note on Drawing Power of Crowds of Different Size. *Journal of Personality and Social Psychology*, Vol.13, No.2, pp. 79-82, 1969.
- [12] Nakanishi, H., Yoshida, C., Nishimura, T., and Ishida, T. FreeWalk: A 3D Virtual Space for Casual Meetings. *IEEE Multimedia*, Vol.6, No.2, pp. 20-28, 1999.
- [13] Sugiman, T., and Misumi, J. Development of a New Evacuation Method for Emergencies: Control of Collective Behavior by Emergent Small Groups. *Journal of Applied Psychology*, Vol. 73, No. 1, pp. 3-10, 1988.
- [14] Sycara, K., Paolucci, M., Velsen, M. V., and Giampapa, J. The RETSINA MAS Infrastructure. To appear in *Special Joint Issue of Autonomous Agents and MAS*, Vol.7, Nos.1 and 2, July, 2003.