

Protocol Design Using Participatory Simulation

Yohei Murakami

Department of Social Informatics

Kyoto University, Japan

Protocol Design Using Participatory Simulation

Doctoral Thesis Series of Ishida Laboratory
Department of Social Informatics
Kyoto University

Copyright © 2005 Yohei Murakami

Abstract

With advent of ubiquitous society, problems for constructing socially-embedded systems have changed from technological difficulties, such as network infrastructure, to how to ensure that they yield appropriate social behavior. That is, success of the systems depends on interaction design between users and them. The most common way to design how to interact with users is a participatory design, which is a method of bringing users into development process. However, specifying problems of interaction design is a difficult task for system designers even if they employ participatory design. This is because results of demonstration experiment cannot be reproduced.

In this thesis, our goal is to develop the method to enable system designers familiar with application domain to design interaction between systems and users while receiving the feedback from users with relatively low costs. To accomplish this goal, we adopt multi-agent simulation consisting of agents representing users or systems. Once the agent models are constructed appropriately, it provides us reproducibility of results. Moreover, conducting the multi-agent simulation in three-dimensional virtual space allows us to participate in the simulation by controlling avatars and record the operation history.

In this thesis, we address following issues on supporting participatory design of socially-embedded systems by multi-agent technology.

1. Establishment of a methodology of multi-agent simulation

There are two types of multi-agent simulations depending on their purposes; a) *analytic multi-agent simulations* with a simple internal

model of agents and b) *synthetic multi-agent simulation* with a complex internal model of agents. The purpose of the former approach is to know principles of society, and the latter one is to reproduce reality-based situations. In this research, we need synthetic multi-agent simulation to design real systems. However, it is difficult for system designers to construct agents with complex internal models, because they are not computing professional. Their interests are how systems and users interact with each other rather than how systems work and users act. Therefore, we contrive a new method that designs interaction between agents without knowing the details of their implementation. This method enables us to describe interaction protocols separately from agent system implementation by delegating how to execute the interaction protocols to agents. Moreover, we present a domain dependent language to capture the patterns of interaction protocols. The language can reduce the system designer's burden to describe interaction protocols. We validate the proposed method by double-checking the result of the previous controlled experiment done in a real environment.

2. Refinement of protocols using participatory simulation

By repeating multi-agent simulation, it become possible to estimate how well the designed interaction protocols work. However, even when the simulation results show the effectiveness of the protocols for simulated users, the problem of validating whether they are effective for real users still remains. One solution to such a problem is to conduct participatory simulations where agents and human-controlled avatars coexist. In this simulation, the protocols are provided for users controlling avatars as well as agents. This is why the protocols need to be redesigned with an appropriate degree of abstraction so that users can easily understand them. In improving the protocols, we have to consider the human behavior of violating protocols because it is impossible to enforce every rule on humans. Therefore, we propose the process that refines not only protocols but also human internal mod-

els to make decisions about whether they follow the protocols. This process also enables us to verify and validate the protocols and the internal models using results of multi-agent simulations and participatory simulations. To realize this process, we develop new agent architecture with decision making about whether it comply with given protocols. We check the usefulness of the proposed method by applying it to improving evacuation methods.

3. Acquirement of users' internal models from log data

There are two constraints in acquiring users' internal models from log data obtained by participatory simulations. One is quantitative restriction of log data. The number of times to conduct participatory simulations is limited because they need several human subjects. The other is that we have to extract models specific to each subject for designing protocols available for diverse users. These constraints often breed arbitrary interpretation of results of data analysis. To avoid such a problem, computational support like machine learning is required. Therefore, we present a new algorithm to model human behavior based on hypothetical reasoning whose domain knowledge is the set of action rules obtained by interviews with some human subjects after participatory simulations. This algorithm can choose the action rules which explain a subject's operation history without inconsistency. The verification of the selected rules are assured because this method is a reasoning based on logic. We apply the algorithm to the acquisition of an evacuee's model in a fire-drill experiment. Finally, we validate the algorithm by comparing the subject's model with one obtained from the results of an interview with the subject.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Professor Toru Ishida at Kyoto University, for his continuous guidance, valuable advice, and helpful discussions. His stern attitude towards research is exemplary and he is an outstanding role model for me. I gratefully acknowledge valuable comments of other members of my thesis committee at Kyoto University, Professor Hajime Kita and Professor Osamu Katai.

Lecturer Hirofumi Yamaki and Assistant Professor Hideyuki Nakanishi at Kyoto University always gave me kind advice and helpful comments. I wish to thank Professor Toshio Sugiman, who advised an important role in the system and experiment design.

I am grateful to the other adviser, Professor Toyoaki Nishida at Kyoto University, for his valuable advice and discussions.

I also wish to express my special thanks to Akishige Yamamoto at Mathematical Systems Inc. for developing Q interpreter and advising scheme programming.

Brief mention should be made of the work by my colleagues at Ishida laboratory. Tomoyuki Kawasoe developed FlatWalk, two-dimensional multi-agent simulator, and prepared participatory simulation experiments. Yuki Sugimoto implemented learning algorithm which enables us to acquire human behavior models from log data obtained by participatory simulation.

Thanks to the members of the Ishida laboratory, especially Daisuke Torii. He gave me indispensable technical comments in writing my thesis and offered consultation for private life as well as research. I could not have completed the doctoral course and my thesis without him.

Finally, I would like to make the most cordial acknowledgment to my dear family, my old brother, Yoshiharu Murakami, my old sister, Keiko Murakami, and my parents, Yoshiaki and Michiyo Murakami. They gave me sincere support and encouragement indispensable to finish my doctoral course.

This research was supported in part by followings:

- The Japan Science and Technology Corporation for the Core Research for Evaluational Science and Technology Universal Design of Digital City Project.
- Japan Society for the Promotion of Science (JSPS), Grant-in-Aid for Scientific Research (A)(15200012, 2003-2005).

Contents

1	Introduction	1
1.1	Background	1
1.2	Approach	2
1.3	Structure of the Thesis	5
2	Interaction Protocols	9
2.1	Protocols for Coordination	10
2.1.1	Protocol Description Language	10
2.1.2	Protocol Engineering	13
2.2	Protocols as Social Norms	15
2.2.1	Social Law	17
2.2.2	Social Commitments	19
2.2.3	Social Conventions	20
2.3	Example: Evacuation Guidance Protocol	21
3	Scenario Description for Multi-Agent Simulation	25
3.1	Introduction	25
3.2	Multi-Agent Simulation	27
3.3	Scenario Description Language	28
3.3.1	Scenario Description Language Q	28
3.3.2	Interaction Pattern Card	31
3.4	Scenario Description Process	32
3.4.1	Overview of Process	32
3.4.2	Step1: Defining a Vocabulary	33

3.4.3	Step2: Describing Scenarios	37
3.4.4	Step3: Extracting Interaction Patterns	40
3.4.5	Step4: Integrating Real and Virtual Experiments	44
3.5	Summary	47
4	Protocol Refinement Process Using Participatory Simulation	49
4.1	Introduction	49
4.2	Participatory Simulation Approach	51
4.3	Agent with Social Constraints	52
4.3.1	Design of Agent Architecture	53
4.3.2	Implementation of Agent Architecture	54
4.4	Protocol Design Process	57
4.4.1	Overview of Process	58
4.4.2	Step1: Creating Protocols	61
4.4.3	Step2: Validating Protocols	63
4.4.4	Step3: Modifying Agent Models	65
4.4.5	Step4: Modifying Protocols	67
4.5	Summary	68
5	User Modeling by Participatory Simulation	71
5.1	Introduction	71
5.2	Overview of Modeling Process	73
5.3	Formalizing the Problem	76
5.3.1	Definition of Technical Terms	76
5.3.2	Domain Knowledge	77
5.3.3	Description of Observation	79
5.4	Acquisition of Operation Model	81
5.5	Application to Evacuation Domain	85
5.5.1	Evacuation Domain	85
5.5.2	Application of Modeling Process	86
5.6	Summary	92
6	Conclusion	95

6.1 Contributions	95
6.2 Future Directions	98
Bibliography	101
Publications	111

List of Figures

2.1	Ground Plan of the Experiment and Initial Position of Subjects[Sugiman 88]	22
3.1	Cues and Actions	29
3.2	Scenarios	29
3.3	Definition of Agents and Avatars	30
3.4	IPC/ <i>Q</i> Translation Process	31
3.5	2-D Evacuation Simulation	36
3.6	3-D Evacuation Simulation	37
3.7	An Interaction Pattern Card (Excel Interface)	42
3.8	<i>Q</i> Scenario Corresponding to IPC in Figure 3.7	43
3.9	Comparing Results between Real and Virtual Experiments	46
4.1	Participatory Simulation	51
4.2	Agent Architecture under Social Constraints	53
4.3	Implementation of Agent under Social Constraints	55
4.4	Data Dependency Graph of Production Rules	57
4.5	Protocol Design Process	60
4.6	Follow-Me Method (Leader’s Protocol)	61
4.7	Follow-Me Method (Evacuee’s Protocol)	62
4.8	Results of the Participatory Simulation	64
4.9	Interviews with Subjects	66
4.10	Modified Follow-Me Method (Leader’s Protocol)	67

5.1	Modeling Process (This chapter focuses on the area in the dotted frame)	75
5.2	Agent John's View at T-1	80
5.3	Proof Tree of $h_{T-2} \cup \Sigma \cup \{S_{T-1}\} \vdash Do(\text{walk})$	82
5.4	Participatory Simulation by FreeWalk	86
5.5	Priorities among Operation Rules in P_1	89
5.6	Operation History of the Subject Operating Agent11 (former)	90
5.7	Operation History of the Subject Operating Agent11 (latter)	91

List of Tables

2.1	Fields in Requirements Specification for Protocols.	13
3.1	Sample Cues and Actions for Evacuation Simulation.	35
3.2	Rules for Guidance Scenarios	39
3.3	Rules for Evacuation Scenarios	40
4.1	Agent Internal Model during Fire Drills	63
4.2	Evacuee's Internal Model Acquired from the Participatory Simulation	66
5.1	Operation Rules of Subject Operating Agent 11	88

Chapter 1

Introduction

1.1 Background

The era of ubiquitous computing/networking is coming against of the backdrop of development of the Internet, mobile computers, and electronic devices. The network infrastructure gradually enables us to construct large scale socially-embedded systems. As a result, future systems are expected to support multiple users simultaneously wherever they need [Nakashima 03, Yamashita 05]. This is different from existing information systems which have served individual users, such as ATMs in banks. This shift of the concept of the systems forces them to adapt their users. Some papers describe large scale multi-agent systems consisting of personal agents corresponding to each user [Sugawara 04, Kurumatani 03].

The problem in constructing such a system is how to ensure that they yield adequate behavior: “adequate behavior” does not merely mean computationally correct behavior, but appropriate social behavior when embedded in human societies. For example, a car navigation system that shares destinations of users has to provide route assignment favorable for every user.

However, it is known that the behavior of socially-embedded systems is hard to predict because the systems are exposed to unpredictable interactions with users. In the earlier example of the car navigation system, a

driver's deviation of his/her route given by it can result in that other drivers get caught in a traffic jam. To prevent such an inappropriate behavior of the systems, participatory design, which is a method of bringing users into development process, is more and more important [Ishida 04].

Participatory design has been employed to design user interfaces and contributed to improvement of usability and accessibility of systems. In this context, it is also called *user-centered design*. This approach focuses on how to design user interfaces, for instance, where and what information is displayed on a screen. In contrast to user-centered design, how socially-embedded systems behave in human society is most important to design the systems. That is, the key success factor of socially-embedded systems is interaction design, which describes how to interact with users.

However, there are the two following problems to implement participatory design. One is that it imposes a burden to assemble more subjects for a demonstration experiment than *user-centered design* approach. The other is that the environment surrounding socially-embedded systems is too dynamic and open to reproduce the results of demonstration experiments. This makes it difficult to investigate the cause of troubles that occur in the experiments. Even if we can locate the cause of the troubles, the effects of the resolutions are hard to predict. Both problems cause the situation that most of the demonstration experiments are conducted for testing system behavior rather than reflecting subjects' feedback in system design.

1.2 Approach

The objective of this research is to explore methods for resolving difficulties in participatory design of socially-embedded systems addressed in the previous section. We propose methods for designing interactions between users and systems that enable experts in an application domain to reflect users' feedback in their system design with relatively low costs.

To accomplish the goal mentioned above, we employ multi-agent simulations consisting of agents representing individual users. Multi-agent sim-

ulation is a method of observing phenomena emerging from interactions among agents which model each entity with decision making, such as humans. This method is different from the existing one in that it does not need governing equations but each model of entities based on prior knowledge and documents. This method allows us to understand and analyze the phenomena by comparing phenomena observed in the simulation and the real world. Recently, it has draw attention as solutions to analysis of social and economic systems.

Using multi-agent simulation not as analysis tools but as design tools brings the following two benefits to participatory design. One is to provide reproducibility of phenomena observed in the real world. This feature permits us to investigate the cause of the phenomena and estimate how effective various interaction designs are without limiting the number of times of experiments. The other is to enable us to participate in the simulation, by reproducing the entire process of the simulation in three-dimensional virtual space and introducing human-controlled avatars into it. We call the simulation where human-controlled avatars and agents interact with each other in virtual space *participatory simulation*. Participatory simulation can realize participatory design by the less number of subjects less than demonstration experiments in real space. Moreover, by using computational method like machine learning and data mining, we can acquire subjects' feedback from log data obtained by participatory simulation.

In order to apply multi-agent simulation including participatory simulation to participatory design of socially-embedded systems, we have taken the following steps in this thesis.

Firstly, we developed a general-purpose protocol description language to describe interactions between users and systems in multi-agent simulations. We view interaction protocols as behavioral guidelines of agents, while agents keep certain degree of autonomy under the given constraints. This language enables us to compose interaction protocols of application vocabularies. System designers are often not computing professionals, but experts in application domain where the systems work. This is why they are interested in what the expected behaviors of the systems are rather than how

they behave. They do not need logic and programming languages to explain and implement internal mechanism of agents, but languages to define the expected behavior of the systems by using vocabularies in each application domain. That is, they need to describe how the systems and users are expected to interact with each other. How changing this definition affects the entire systems is monitored through the process of simulations. In this way, the language allows system designers to design appropriate interactions between systems and users.

The change of viewpoint from an internal mechanism to interaction design transforms methodologies to develop systems. In the existing methodology established in agent-oriented software engineering, implementation phase of agents consisting of agent internal models and interaction protocols follows design phase of those models serially. This process is not suitable to redesign protocols because implementation of agents needs to be modified whenever protocols are improved. As the number of modifying protocols increases, the efficiency of development of the systems decreases. The development process deploying implementation phase and design phase simultaneously is necessary for refining protocols. Therefore, the protocol description language also has to be the interface between system designers and system developers.

Secondly, we established protocol refinement process to reflect results of participatory simulation in interaction protocols. Participatory simulation is defined as multi-agent simulation conducted by replacing some of agents with human-controlled avatars. By making humans participate in simulations, we can estimate the effectiveness of designed protocols on humans as well as agents. In fact, results of participatory simulations can be different of that of multi-agent simulations, even when results of multi-agent simulations show the expected behavior. This is because humans are too autonomy to force them to obey the given protocols. Thus, using results of participatory simulations, we have to modify not only interaction protocols but also agent internal models. The research issue to realize this process is to construct new agent architecture with decision making of whether it complies with given protocols.

Finally, we developed a new method of acquiring subjects' models from results of participatory simulations by machine learning. This method extracts individual models by generating explanation of operation history of each subject. However, amount of log data obtained by participatory simulation is not enough to apply inductive learning algorithms since the number of times to conduct participatory simulations is limited. Therefore, we employ deductive approach using domain knowledge. By applying machine learning to acquirement of subjects' models, the success of participatory design can be independent of the operator's ability.

The significant feature of user modeling by participatory simulation is to acquire an internal model specific to each participants. This method is different from inductive machine learning approaches that can extract a common model from operation histories of some participants. However, our approach is natural to apply to participatory design of socially-embedded systems. This is because unpredictable behavior of a user can affect other users in socially-embedded systems that provide diverse users with their service concurrently. Hence, it is necessary to acquire the individual model corresponding to each subject since system designers have to design interaction protocols robust for various users.

1.3 Structure of the Thesis

This thesis consists of six chapters, including this chapter as the introduction.

Chapter 2 is dedicated to introduce interaction protocols addressed in multi-agent domain. Interaction protocols are defined as written behavioral guidelines of agents, while agents keep certain degree of autonomy under the given constraints. There are two types of interaction protocols depending on their target agents; a) *coordination protocols* given agents with a joint goal and intention and b) *social norms* imposed on agents with individual goals and intentions. In order to clarify interaction protocols employed for interactions between users and socially-embedded systems, we explain a

summary of each protocol. Finally, we show a summary of the fire-drill experiment which is an example used throughout this thesis, and introduce two different types of evacuation guiding protocols.

Chapter 3 introduces a new method to conduct synthetic multi-agent based simulations. We developed a protocol description language for multi-agent systems, scenario description language Q , which enables us to define agents' expected behavior. Q focuses on interaction design between agents, compared with the existing approaches that focus on agent internal mechanisms. That is, Q tries to control a group of agents not by implementing how to coordinate with each other, but by requesting them to do something. This allows experts in application domains to describe interaction between agents without knowing the details of agent internal mechanism and conduct multi-agent simulation. Moreover, we 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 and discuss the effectiveness of our method.

Chapter 4 presents a framework to reflect results of participatory simulation in designing interaction protocols. In participatory simulation, two types of participants' behaviors are observed; social behavior complying with given protocols, and selfish behavior violating them. In order to reproduce such decision making about protocols, we propose new agent architecture with interpretation of protocols and decision making of protocols separately. Furthermore, we explain protocol refinement process defining both bases of verification and validation of protocols and decision making models.

Chapter 5 explains a new method for acquiring participants' internal models from log data obtained by participatory simulation. It is true that we can manually refine agent models by interviewing with participants and analyzing log data, but we explore a method to computationally obtain them. This is because manual approach costs us highly, for instance, it takes us from thirty minutes to almost one hour to interview with a participant. Here, we view an agent internal model as a set of prioritized condition-action rules. Based on this definition, we formalized user modeling as a search

to find combinations of action rules and their priorities that can explain an operation history consistently. To solve this problem, we apply hypothetical reasoning whose domain knowledge is a set of action rules collected through interviews with some participants after participatory simulation. Finally, we discuss validation of the model acquired by our method.

Chapter 6 concludes the thesis summarizing the results obtained through this research. We also address the prospect of the future research about protocol design.

Chapter 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. The existing methodologies for developing multi-agent systems produce an interaction model, defining what kind of interaction protocols are employed, in analysis and design phase [Zambonelli 03, Wook 00, Kendall 00, Caire 01, Padgham 02, Juan 02]. The objective of this research, interaction design between socially-embedded systems and users, is viewed as interaction protocol design, once the systems and users are modeled as agents.

There are two types of interaction protocols. One is for realizing coordination between multiple agents. We call it *coordination protocol*. Much research has ever been devoted to the coordination of the agents. Many papers have been written about coordination protocols (like Contract-Net) that allow agents to negotiate and cooperate. Most of the cooperation between agents is based on the assumption that they have some joint goals or intentions. Such a joint goal enforces some type of cooperative behavior on all agents.

The other is for avoiding interference between agents who have individual goals. We call it *social norm*. There are some kinds of social norms to make society composed of diverse agents more efficient. The significant feature of social norms is that the protocols often conflict with agent's goals. As a result, agents sometimes violate the given protocols and then

the protocols may collapse. However, social norms play an important role in reasoning behavior of other agents under incomplete information.

In this chapter, we explain each protocol regulating interactions between agents and then clarify the target of our approach in this research. Finally, we show a summary of the fire-drill experiment which is an example used throughout this thesis, and introduce two different types of evacuation guidance protocols.

2.1 Protocols for Coordination

A coordination protocol is a behavioral guideline describing every interaction between agents in order to accomplish coordination of multiple agents. The interaction described in the protocol is to exchange messages based on speech-act theory. The messages are described in KQML(Knowledge Query and Manipulation Language) or FIPA ACL(Agent Communication Language)[Finin 97, Labrou 99]. 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.

Contract net protocol is a representative of interaction protocols in multi-agent domain. This protocol is applied to task scheduling in distributed systems[Lesser 99], allocation of trucks[Sandholm 93], meeting scheduling[Sen 94], and so on. In addition, Foundation for Intelligent Physical Agents (FIPA), standards body for multi-agent technology, developed specifications of standard interaction protocols in order to construct multi-agent systems in open environment like the Internet.

In this section, we introduce languages to describe coordination protocols and explain the existing process to design them.

2.1.1 Protocol Description Language

Protocol description languages are divided into two types depending on viewpoints on protocols. One is for writing protocols from the view of each

agent and role. The other is for writing them from the view of an entire coordination.

The examples of the former language are COOL and AgenTalk, both of which are based on finite state machine. COOL is a language to integrate description of message contents in Knowledge Interchange Format (KIF) and description of agent communication in Knowledge Query Manipulation Language (KQML) by using a structured conversation framework representing a coordination mechanism between agents [Barbuceanu 95]. It provides continuation rules that say the subsequent protocols for agents to handle several protocols given to them. On the other hand, AgenTalk introduces the concept of inheritance into protocol description, which enables us to construct a new protocol by reusing the existing protocols at low cost [Kuwabara 96]. This approach is motivated by the assumption that various protocols specific to an application domain are needed. Moreover, it presents clear interfaces between agents and the given protocols, called *agent-related functions*, for specifying general protocols to adapt each application domain. Functions specific to each agent are implemented as callback functions which are invoked from protocols using the agent-related functions.

While, an example of the latter language is Cool (Cooperation Language) which is based on a multi-agent plan [Kolb 95]. In contrast to the above protocol description languages to describe each role's protocol based on finite state machine, Cool is a language to represent protocols as an entire conversation flow. That is, protocols described in Cool are composed of behavior of every agent. Therefore, it is possible to dynamically change agents' roles on runtime without assigning roles to every agent in advance.

Unlike the above approach that develops executable protocol description languages, there is another approach that describes the model of a protocol in a modeling language and then implements each agent according to the model. One of the representative modeling language for protocols is AgentUML (AUML) [Odell 00]. This is an extended UML (Unified Modeling Language) which is diffused in object-oriented software engineering. By extending UML, commonly used in software development, the cost of using

AUML is reduced. For example, interaction diagrams representing message sequences between agents and state chart diagrams representing state transitions of each agent are used to describe the model of a protocol. However, it is pointed out that there are gaps between design and analysis of models and implementation of them by taking this approach. Those gaps lead the high cost of implementing the models.

An approach that seamlessly connects a protocol model described in AUML and source codes described on JADE [Bellifemine 99], a standard framework for developing multi-agent systems is proposed to fill in the gaps [Dinkloh 04]. In this approach, a protocol model described in interaction diagram of AUML is transformed into a finite state machine model written in cpXML, and then state transition classes which JADE provides are generated from the model. All system developers have to do is to specify the generated skeleton classes and implement the body of the classes. Additionally, interaction description language IOM/T [Doi 05] with expressiveness equivalent to AUML is also developed as the same approach that connects AUML description and agent implementation. It is possible to easily generate the skeleton codes described in IOM/T from AUML description since IOM/T has the same expressiveness as AUML. Interactions described in IOM/T can produce programs executable on a target multi-agent system platform by IOM/T compiler which support the platform. Other than the solutions previously mentioned, there are some researches to generate executable codes of agents from a protocol model designed by graphical tools. For example, generating skeleton codes of Bee-agent from finite state machine described by IPEditor [Tahara 01, Kawamura 99].

Although there are various approaches to design and implement protocols as mentioned above, the most of their representations are based on a finite state machine model. This current situation shows that a finite state machine model is suitable to represent a protocol. Paurobally *et al.* summarized that other representations such as Petri Nets are less expressive and readable than a finite state machine [Paurobally 04].

Table 2.1: Fields in Requirements Specification for Protocols.

Name	the name of the protocol.
Keywords	a list of keywords for characterizing this protocol: auctions ,secured, etc.
Agents	the list of agents or roles which are involved in this protocol.
Initiator	the agent or the role which initiates the protocol.
Prerequisite	required conditions which must be true for initiating the protocol.
Function	a short description of the meaning of this protocol.
Behavior	a detailed description of the protocol with all the messages used, the meaning of the message contents and what messages are possible for any interaction state.
Conditions	conditions which must hold during the execution of the protocol: deadlock freeness, liveness, termination, etc.
Termination	normal exit interaction states. For example, the customer has the products and the clerk has the money.

2.1.2 Protocol Engineering

A coordination protocol is a behavioral guideline describing every message exchange between agents. Hence, the development methodology of the protocols is inspired by that of communication protocols [Holzmann 91]. The development process consists of the following five steps: analysis, formal description, validation, implementation, and conformance testing [Huget 03a].

Step1: Analysis

In the first step, an informal document which gathers all the features a protocol has to provide is defined. Such a description is done through a natural language description. Table 2.1 describes these feature [Huget 03b]. Field like name, keywords and function are not used during this step of the development process but they are in the protocol reuse and maintenance stage.

Step2: Formal Description

In the second step, a natural language specification of the requirements described in the previous stage needs to be translated into the formal description not to make ambiguous protocols. There are various ways to formally represent a protocol as mentioned in the previous section and each way has advantages and disadvantages. For instance, a finite state machine has advantages that it is easy to read and understand it, that it has formal semantics, and that it can be validated by some algorithms, and a disadvantage that it can not represent concurrency. On the other hand, although Petri Nets can describe concurrency, it is hard to describe and understand it. To resolve these disadvantages of Petri Nets, Mazouzi *et al.* tried to reduce a burden to describe Petri Nets by translating description in AUML, a modeling language without complete formal semantics, into description in Colored Petri Nets (CPN) [Mazouzi 02].

Step3: Validation

In the third step, the protocol formally described in previous step needs to be validated to check if structural properties and semantic properties are satisfied. The former validation technique uses the notion of graph and defines the property to be checked as a property on a graph; this is called reachability analysis. Such properties are also checked in validation of communication protocols [Holzmann 91]. On the other hand, the validation of the latter properties is typical for coordination protocols in multi-agent systems. Fornara *et al.* validated semantic properties as well as structural properties about topology of interaction protocols. They checked if every speech act can be performed at each state, and if all commitments are completed at each termination state by canceling, violating, or fulfilling them [Fornara 03].

Step4: Implementation

In the fourth step, it is time to implement a protocol according to

the formal specification of the protocol validated in the previous step. The purpose of this phase is to realize an agent program which conforms to the formal description. How to implement protocols depends on implementation of agent systems which follow them. In the case of using JADE, a multi-agent system platform compliant with FIPA specifications, protocols are implemented as state transition classes provided by JADE.

Step5: Conformance Testing

The previous steps of interaction protocol engineering offer the possibility to formally design, to check some properties and to synthesize an operational version of the protocols. Final step checks whether the operational version of the protocols still verifies the properties defined in the specification of requirements. This is because the translation from a formal description of a protocol into an operational one can insert errors and some features described in the specification of requirements can be absent. The conformance testing step is different from the validation one in that it checks properties on an operational version whereas the validation stage checks properties on a formal version of the protocol.

2.2 Protocols as Social Norms

As mentioned in the previous section, a coordination protocol is a behavioral guideline specifying every interaction between agents so that multiple agents with a joint goal can cooperate with each other. Unlike this, there is a protocol to prevent conflicts between agents with individual goals in society. Every agent can expect other agents' behavior by sharing it. This protocol is called a social norm. For example, left-hand traffic is a social norm. The protocol plays an important role that prevents collisions of cars. Such a social norm acts as a social constraint, so it often conflicts with an individual goal. 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.

This social norm is generally modeled by the following seven elements [López 02, López 03].

1. A set of normative goals
2. Addressee agents which are directly responsible for the satisfaction of the normative goals
3. Beneficiary agents that benefit from the satisfaction of normative goals
4. Context where addressee agents must follow norms
5. Exception states representing situations in which addressee agents cannot be punished when they have not complied with norms
6. Rewards to be given when normative goals become satisfied
7. Punishments to be applied when normative goals are not satisfied

Social norms are divided into social laws, social commitments, and social conventions [López 03]. We can distinguish them by three elements; norm-creating process, persistence, and rewards and punishments.

Regarding norm-creating process, norms might be created by the agent designer as built-in norms, they can be the result of agreements between agents, or can be elaborated by a complex legal system. Concerning their persistence, norms might be considered during different periods of time, such as until an agent dies, as long as an agent stays in society, or just for a short period of time until its normative goals become satisfied. Finally, both of rewards and punishments might not exist, some norms do not include either punishments or rewards, even though they are complied with. By these differences, social norms are classified into social laws, social commitments, and social conventions.

Social Law

A social law is defined as a norm which qualified agents and system designers create to make an entire system work efficiently in top-down style. Hence, addressee agents do not participate in its creation. Once a social law is created, the social law is imposed on the whole system uniformly. The imposed norm is valid until it is modified by the designers. A significant feature is to apply punishments to agents violating the norm.

Social Commitments

A social commitment is a norm which is produced by results of agreements between agents. Thus, addressee agents actively participate in its creation. Once the normative goals of a social commitment are satisfied, a reward can be claimed. Contrary to social laws, social commitments are temporary, because they may disappear once the normative goals become satisfied.

Social Conventions

A social convention is a norm which is accepted as general principles by the members of society. This norm is not enforced by rewards and punishments, but motivated to be fulfilled because of the empathy or sympathy that addressee agents have towards other agents, or because addressee agents want to express their social conformity. Thus, this norm is gradually formed within organization and society, diffused by following the norm which agents meet, and finally emerges. In this way, agents voluntarily comply with social conventions, so the norms do not have both punishments and rewards to compel them to obey.

2.2.1 Social Law

A social law is a norm assigned to society. Once an agent participates in the society, it has to adopt the norm absolutely. This is why agent's individual

goal can conflict with the given regulations. Therefore, the framework to enforce compliance is needed.

The simplest way is to incorporate the regulations into agents. Shoham *et al.* defined an agent as an entity repeatedly performing actions with state transition from a state to the next state, and regulations as prohibition of specific actions at each state [Shoham 92b, Shoham 95]. Hence, we have only to design a state transition function that can choose actions other than the actions prohibited by a norm, in order to construct an agent given the norm.

Under a social law introduced by Shoham, every agent strictly complies with the norm since it is incorporated into the agents. This is why the norm does not need punishments when an agent violates it. Such a strong constraint by a social law contributes to communicating with other agents and reducing problem space, while it causes the problems that agents can not react to unpredictable changes in the environment and sometimes accomplish their own goals. A solution to these problems is to relax the constraints by removing the given rules one-by-one until agents can accomplish their goals [Briggs 95].

While such derogation from a given social law enables agents to adapt changes in the environment flexibly, much derogation leads an inefficient situation that their behavior conflict with each other. Therefore, it is necessary to enhance legal force by introducing punishments when they violate a social law [López 02, López 03]. In addition, it is effective to add defender agents to monitor fulfillment of a social law, and crack down on violation of the norm [Boella 03]. However, note that introduction of such defender agents causes an infinite chain of norms because a norm and other defender agents to enforce strict regulations on them are needed again [López 03].

As mentioned above, there are diverse approaches to enhance legal force. The most important thing is a balance between flexibility and enforceability in order to keep individual utility and an entire system sound [Boella 05].

2.2.2 Social Commitments

A social commitment is a norm that is agreed on not by entire society but by party concerned. In contrast to a social law imposed by system designers in a top-down style, a social commitment is created by authority to make it or agreement among party concerned including addressee agents. Obligations which arise from such a social commitment are introduced by deontic logic.

Deontic logic is a form of modal logic and deals with propositions regarding obligations. It presents a notation to represent obligations approved between two agents, $O_{ij}(\phi)$. This means that agent i is committed by agent j to satisfy ϕ . On the other hand, $O_{ij}(\alpha)$ means that agent i is committed by agent j to perform action α . In the latter case, agent i has only to execute action α , while agent i has to make a plan to satisfy ϕ state in the former case. Dignum *et al.* described a commitment model to bring obligations by using deontic logic and Speech Act Theory [Dignum 99].

For example, if both of them are not authorized to generate obligations, either of them generates an obligation by promising to do something. This situation is represented by a commissive act of COMMIT meaning a promise, as follows. Note that $[]$ means having performed a speech act. The following description says that agent i is committed by agent j to do action α after agent i promises agent j to do action α .

$$[COMMIT(i, j, \alpha)]O_{ij}(\alpha)$$

In the case that either of them is authorized to generate obligations, an obligation arise through an authorized command. This situation is represented by a directive act DIRECT and a predicate *auth* meaning that an agent has an authorization to perform an act. The following description says that agent j is committed by agent i to do action α after agent i , which is authorized to tell agent j to do α , exercises the authority.

$$auth(i, DIRECT(i, j, \alpha)) \rightarrow [DIRECT(i, j, \alpha)]O_{ji}(\alpha)$$

Furthermore, Castelfranchi *et al.* believes that authority is derivational

concept arising from a social commitment [Castelfranchi 95]. In his paper, a social commitment is defined as below:

$$(S - COMM\ x\ y\ a\ z)$$

The above description means that agent x in front of witness agent z is committed by agent y to do action a . The social commitment generates some agent y 's authority of enforcement of agent x 's acting a , protest and sanction against nonperformance of the commitment.

2.2.3 Social Conventions

A social convention is a norm that is formed through interactions among agents in a bottom-up style and accepted as general principles by the members of a society. Strategies taken by individual agents in society gradually converge and then a social convention emerges from the society. This norm is not enforced by rewards and punishments, but motivated to be fulfilled because of the empathy or sympathy that members of society have towards other members. Therefore, formation process of social conventions strongly depends on agents' social conformity.

In order to find out relations between them, Shoham *et al.* proposed the following four methods for updating strategies taken by an agent. They are based on interaction history with another agent. In their paper, interaction means that a randomly selected pair of agents meet and observe each other's strategy [Shoham 92a, Shoham 97].

Internal Statistics:

Adopt the strategy with probability that each strategy agrees with other agent's strategy.

External Statistics:

Adopt the strategy with probability that is the proportion of each strategy encountered in other agents so far.

Internal Majority:

Adopt the strategy that most often agrees with other agent's strategy so far.

External Majority:

Adopt the strategy if so far it was observed in other agents more often than the other strategies.

To evaluate formation process of a social convention, they presented two assessment criterions; *convergence time*, how long it takes to reach the convergence that 80 percent or 90 percent of the agents choose the same strategy, and *convergence frequency*, how many times the convergence occurs. Shoham *et al.* conducted three types of simulations depending on the given constraints; no constraint, on frequency to update strategies, and on amount to memory interaction history.

Results of these simulations showed that the external majority update method had performed best in the normal test. In the test with a constraint on frequency to update, it was confirmed that the times of convergence had decreased as the constraint had been enhanced. This is because agents rarely adapted the dynamic environment as the frequency to update decreased. On the other hand, in the test with a constraint on amount to memory, it was confirmed that the times of convergence had decreased as the constraint had been relaxed. Moreover, Walker *et al.* succeeded in improving the convergence time and convergence frequency by considering communication between agents [Walker 95].

2.3 Example: Evacuation Guidance Protocol

As mentioned above, interaction protocols have two aspect; a behavioral guideline to accomplish joint actions, and a social constraint to make social behavior efficient. The former is a coordination protocol for cooperative agents strictly obeying the given protocol and corresponds on interaction

design within socially-embedded systems. On the other hand, the latter is a social norm for agents with individual goals and corresponds on interaction design between users and systems. This research focuses on the latter, and assumes protocols like evacuation guidance imposed by authority in a top-down style.

Through this thesis, we employ the fire-drill experiment conducted by Sugiman [Sugiman 88] in the real world as an example to apply our proposed methods. Evacuation guidance by leaders is viewed as a socially-embedded system to provide information like escape routes with evacuees.

In the experiment, he firstly 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

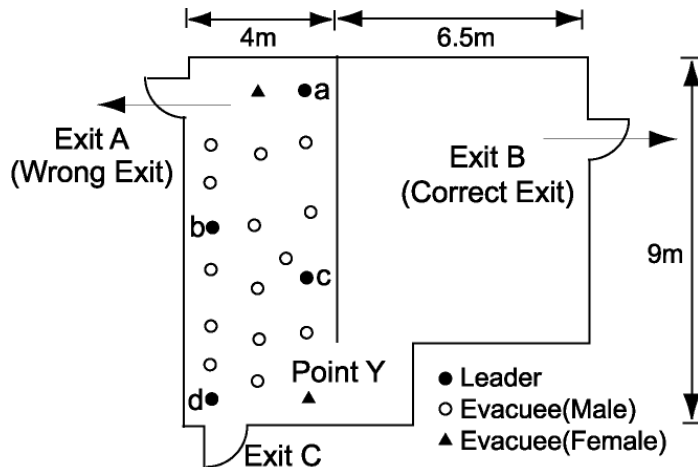


Figure 2.1: Ground Plan of the Experiment and Initial Position of Subjects[Sugiman 88]

and nine meters long; there were three exits, one of which was not obvious to the evacuees as shown in Figure 2.1. 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.

Chapter 3

Scenario Description for Multi-Agent Simulation

3.1 Introduction

Our purpose is to enable system designers to design interaction between socially embedded systems and users while simulating effects of the design. By estimating how the interaction design works well in society through simulations, they can effectively refine the interaction design.

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) so far. 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. Therefore, this type of simulation methods is not suitable for designing interaction between users and socially embedded systems.

This chapter proposes a new method for realizing a multi-agent simulation which is based on interaction design between agents. 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. We call it participatory simulation.

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 taken for instance in this thesis is another direction[Hareesh 00, Ishida 02a]. When simulating social phenomenon in any detail, we need to describe various social interactions including verbal and nonverbal 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 chapter 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[Sugiman 88].

3.2 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 as Coarse-Grain).

Fine-Grain: Analytic approach

This type of simulation is used in the analysis of complex social systems. Here, the KISS principle (Keep It Simple, Stupid) is often applied[Axelrod 97]. 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 [Epstein 96].

Coarse-Grain: Synthetic approach

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[Hanks 93]. This approach is used in an early stage of system development (examples include robots and plants)[Bradshaw 01, Vincent 01, Kitano 97], in the examination of strategies for decision making [Clancey 01, Sycara 03], 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 research focuses on the latter, and assumes running simulations that focus on virtual experiences like evacuation drills.

3.3 Scenario Description Language

The existing research of control of agents in multi-agent systems has focused on modeling an agent's internal mechanism to realize coordination between agents. For example, STEAM, study of reasoning how to coordinate with other agents based on a general-purpose teamwork model, and TAEMS, study of task scheduling by multi-agent systems [Decker 93, Tambe 97].

On the other hand, we focus not on an agent's internal mechanism, but on interaction between an agent and its environment. Our proposed scenario description language is a language to naturally describe the interaction consisting of action on its environment and observation of it. That is, we try to control agents by requesting them not how to coordinate but what to do.

3.3.1 Scenario Description 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 [Ishida 02b]. 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. Examples of a cue and an action are shown in Figure 3.1. The following description requests an agent to reply “Hi” if it hears “Hello” from Hanako.

```
(?hear "Hello" :from Hanako)
(!say "Hi" :to Hanako)
```

Figure 3.1: Cues and Actions

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 cor-

```
(defscenario conversation (&pattern ($x #f))
  (greeting ((?hear "Hello." :from $x)
             (!say "Hi." :to $x)
             (go asking))
            ((?hear "Good-bye" :from $x)
             (!say "See you tomorrow." :to $x)
             (go walking)))
  (asking ((?hear "Hello." :from $x)
           (!say "What's the matter?" :to $x)))
  (walking ...))
```

Figure 3.2: Scenarios

responding 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. An example of a scenario is shown in Figure 3.2. In the following description, a scenario consists of three state. At the initial state, an agent replies “Hi” if it hears “Hello” from someone and, at the next state, it replies “What’s the matter?” if it hears “Hello” again. In this way, an agent can do diverse actions depending on states even when it observes the same event.

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. Examples are shown in Figure 3.3. In the following description, agent Ken is assigned the “conversation” scenario described in Figure 3.2.

```
(defagent Ken :scenario conversation)
(defavatar Hanako)
```

Figure 3.3: Definition of Agents and Avatars

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* so its language specifications are easy to extend.

*Scheme is a dialect of the Lisp programming language invented by Guy Lewis Steele Jr. and Gerald Jay Sussman.

3.3.2 Interaction Pattern Card

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 IPC 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 3.4 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 pat-

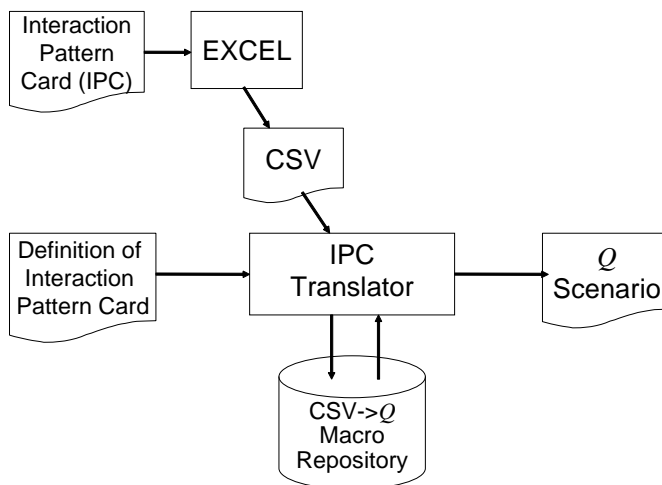


Figure 3.4: IPC/ Q Translation Process

tern 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.

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.

3.4 Scenario Description Process

3.4.1 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.

This process gradually advances development from the initial step while each step gives the previous step its feedback. Fixedness of vocabularies for application scenarios in the early stages makes parallel work between scenario writers and system developers more efficient.

The feature of our proposed scenario description process is to extract interaction patterns as a step of entire simulation process. While the existing processes focus only on simulating, defining IPC enables us to reuse the patterns of interactions and reduces a burden to describe scenarios in the same domain. That is, we can share interaction patterns with application experts in the same domain.

The following subsections provide details of each step using our experiments on the evacuation domain introduced in section 2.3.

3.4.2 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. These vocabularies are decided with relatively little effort because they are examined based on the existing models.
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 based on Sugiman’s papers; Table 3.1 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

Table 3.1: Sample Cues and Actions for Evacuation Simulation.

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

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 3.5.

We use FreeWalk as the platform for the three-dimensional simulation [Nakanishi 99]. 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 3.6.

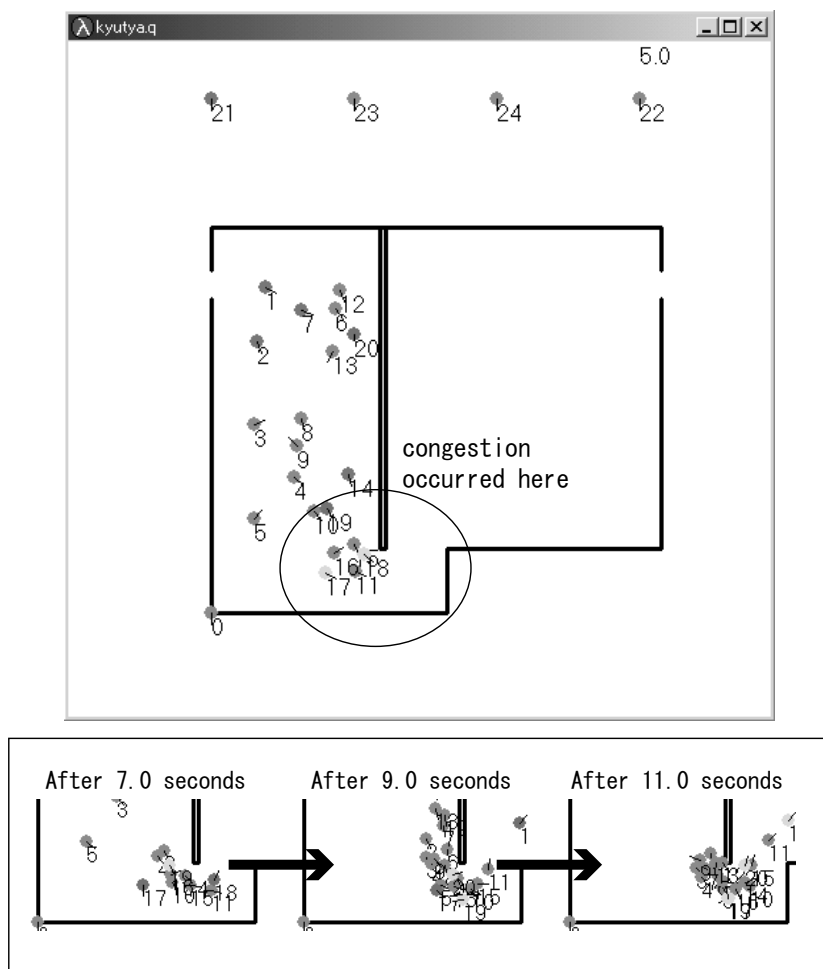


Figure 3.5: 2-D Evacuation Simulation

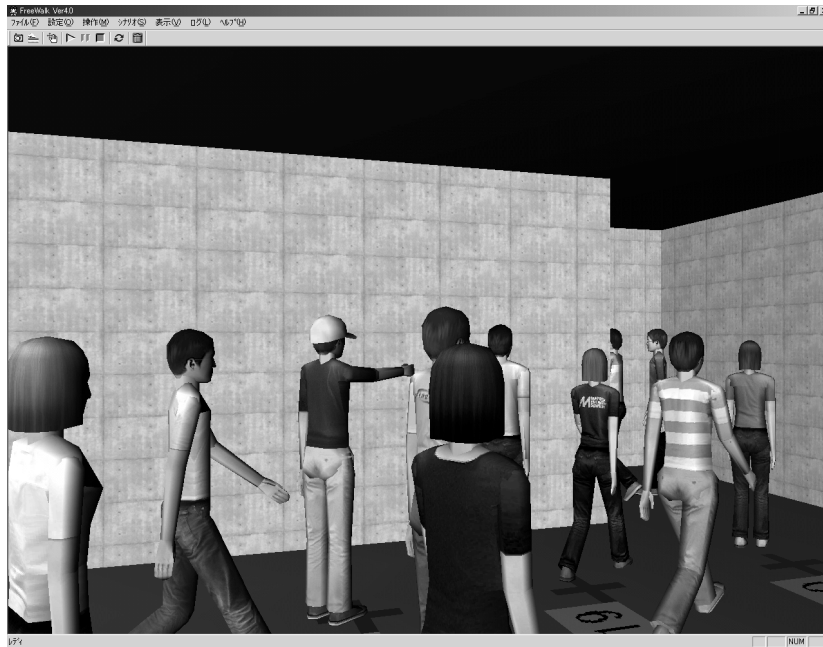


Figure 3.6: 3-D Evacuation Simulation

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.

3.4.3 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 3.2 and Table 3.3 shows a list of all the rules forming each scenario.

These extracted rules use cues and actions shown in Table 3.1. For example, the scenario of the leader agent employing the "Follow-me method" is displayed on the Figure 3.8. 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 "!!."

Table 3.2: Rules for Guidance Scenarios

Agent	Rule
Leader (FD)	When Exit A 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 (FM)	When Exit A 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.
	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.

Table 3.3: Rules for Evacuation Scenarios

Agent	Rule
Evacuee	When Exit A 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.

3.4.4 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:

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 3.7 shows the IPC labeled for evacuation domain. This IPC has four labels indicating each state; "Initiate Guidance," "Guidance [Condition]," "Guidance [Repeat]," and "Terminate Guidance," and also five labels indicating each cue and action; "Object of Observation," "State," "Guiding Actions," "Evacuee's position/direction," and "My position/direction."

Finally, we assign arranged card structures to the semantics of the card. Figure 3.7 shows the IPC of a leader agent composed in this fashion. In the IPC in Figure 3.7, each row is assigned the semantics as follows.

Initiate Guidance If a leader observes that an object filled in "Object of Observation" column becomes the state written in "State" column, it performs actions in order indicated in "Guiding Actions" column.

	A	B	C	D	E	F
1	Card ID	1	Card Name	Follow-me	Card Type	Guidance Card
2	Initiate Guidance	Exit A	Object of Observation	State	Guiding Actions	
3			Open		Put on a cap	
4					Choose an evacuee (one closest to me)	
5					Approach (Evacuee)	
6					Speak (Follow me)	
7					Start to walk (Point Y, Exit B)	
8			Guidance [Condition]	Evacuee's position/direction	My position/direction	Guiding Actions
9						
10	Guidance [Repeat]	Within 1.5m from me	Evacuee's position/direction	My position/direction	Guiding Actions	
11			More than 3.0m apart from me		Stop	
12					Turn direction (Evacuee)	
13			Positioned at the left room	Positioned at the right room	Start to walk (Point Y, Exit B)	
14					Start to walk (Exit B)	
15	Exit B		[Terminate Repeat]			
16	Terminate Guidance	Guiding Actions				
17		Walk (Outside the room)				
18						

Figure 3.7: An Interaction Pattern Card (Excel Interface)

Guidance (Condition) If an evacuee is at the position shown in “Evacuee’s position/direction” column and the leader stands on the position shown in “My position/direction” column, the leader performs actions in order indicated in “Guiding Actions” column.

Guidance (Repeat) If an evacuee is at the position shown in “Evacuee’s position/direction” column and the leader stands on the position shown in “My position/direction” column, the leader performs actions filled in “Guiding Actions” column sequentially. The leader repeat-

```

(defscenario Follow-Me-Method ()
  (let ((target '()))
    (waiting
      ((?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)))
    (guiding
      ((?position :name target
                  :distance_range '(3.0 10.0) :from me)
       (!finish :action ``walk``)
       (!!turn :to target)
       (go guiding))
      ((?position :name target
                  :distance_range '(0.0 1.5) :from me)
       (guard
         ((?position :name me :at Left-of-Room)
          (!!walk :route (list Point-Y Exit-B))
          (go guiding))
         ((?position :name me :at Right-of-Room)
          (!!walk :route (list Exit-B))
          (go guiding))))
      ((?position :name target :at Exit-B)
       (go evacuating)))
    (evacuating
      (#t
       (!walk :route (list Outside))))))

```

Figure 3.8: *Q* Scenario Corresponding to IPC in Figure 3.7

edly executes this rule until it reads the “Terminate Repeat” tag.

Terminate Guidance The leader performs a sequence of actions filled in “Guiding Actions” column sequentially.

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 in Figure 3.8. By generating an IPC that focuses on the domain of evacuation, we could provide a simple means of writing scenarios.

3.4.5 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:

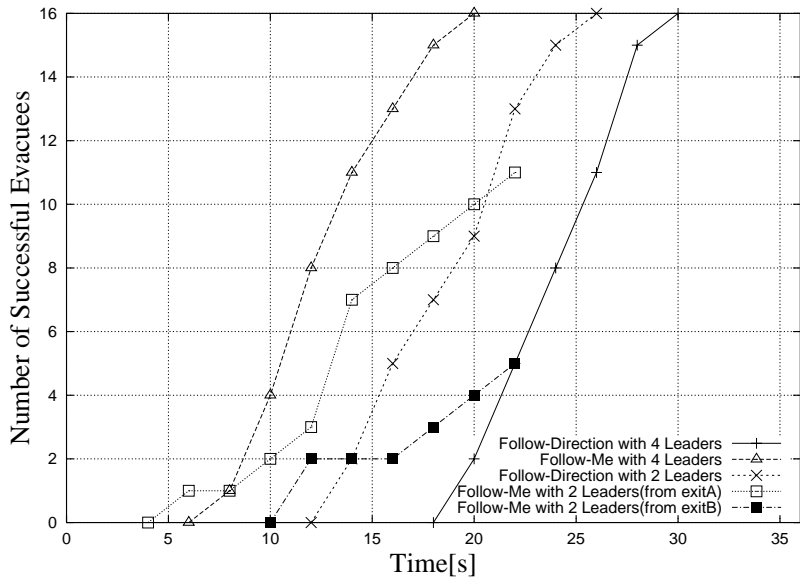
1. We run a simulation based on the scenario constructed using prior knowledge and documents, and compare it to the result of a real-world experiment. If any disparity is found, the result of the real-world experiment is analyzed. In order to eliminate the disparity, new rules obtained in the analysis 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 experiments.
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 interesting behavior[Milgram 69]. 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 3.3 shows the five additional rules.

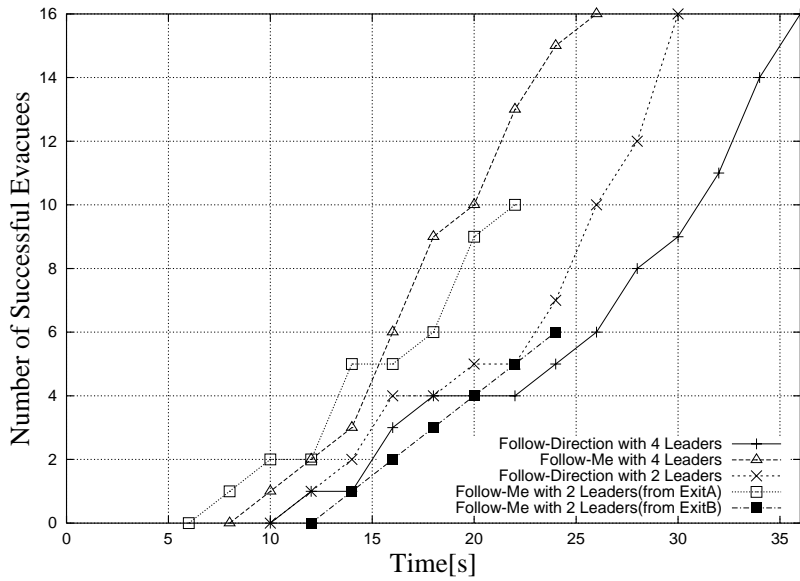
We conducted the evacuation simulation again using the modified scenario, the results of which are shown in Figure 3.9. This simulation exhibited a similar difference between the simulated "Follow-me method" and "Follow-direction method" as that seen in the real world experiment. These differences are as follows:

- In conditions when four leaders were present, the time required for every evacuee to escape in the "Follow-me method" condition was shorter than in the "Follow-direction method" condition. The difference was about 10 seconds the same as results of real experiments.
- In conditions when two leaders were present, "Follow-me method" fails to guide every evacuee to the correct exit, while "Follow-direction method" took amount of time required in the middle of "Follow-me method" and "Follow-direction method" with four leaders.

To sum up, the simulation showed the same results as real experiments that the leader-to-evacuee ratio reversed the superiority of one method to the other when evacuation time was concerned. This result 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



(a)Control Experiment by Sugiman



(b)Evacuation Simulation

Figure 3.9: Comparing Results between Real and Virtual Experiments

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 been 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.

3.5 Summary

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.

In this chapter, our research has used only software agents, but we ran a user-participating simulation using a three-dimensional virtual space, FreeWalk[Nakanishi 99], as shown in the next chapter. 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. Moreover, user-participating simulations contribute to validating new protocols such as new evacuation method, and refining agent models such as evacuee agents. Through cooperatively developing interaction scenarios by application and computing professionals, we can increase the effectiveness of multi-agent simulation in domains like crisis management.

Chapter 4

Protocol Refinement Process Using Participatory Simulation

4.1 Introduction

In the previous chapter, we have dealt with the problem of how to design interaction protocols between simulated-users and socially embedded systems on multi-agent simulation. To solve the problem, we have developed a scenario description language to describe the interaction that employs the vocabulary used in the application domain. This language enables application experts to describe the interaction without knowing the details about of agent system implementation. Results of the simulation using the description can estimate how the interaction protocol of the systems would work in society. However, even when the effectiveness of interaction protocols is confirmed by simulated-users, the problem of whether it is effective for real-users still remains. In this chapter, we introduce participatory simulation, where autonomous agents and human-controlled avatars coexist, as a solution to the problem.

Participatory simulation is a form of multi-agent simulation and realized by replacing some of the simulated users by human-controlled avatars. The simulation is performed in virtual space, and the avatars are controlled by the humans sitting in front of their desktop computers. Although par-

ticipants are given their protocols in advance by instructing them what the expected behavior is, the results of the simulation are often different from those of multi-agent simulation consisting of only scenario-controlled agents. This is because humans not only follow the given protocols, but also autonomously make decisions on what to do the next based on their local information of environment. In order to refine interaction protocols, it is necessary to modify both of user's internal models and interaction protocols while considering the difference between their results.

However, the problem that agents can not behave completely autonomously occurs because the existing agent architecture makes interaction protocols control every interaction. Therefore, our goal is to realize agent's decision making separate from its given protocols. As an approach to this goal, we try to add agent's internal model to make decisions on next action based on its local information of environment as well as the protocols. In order to accomplish the goal and to refine interaction protocols through multi-agent simulation, we address the following two research issues.

Realization of autonomy under social constraints

In order to go on agent architecture that separates decision making from interpretation of given protocols, social constraints, we need a behavior control function to coordinate actions; proactive actions and actions prescribed in the given protocols. It has to reproduce social behavior and selfish behavior of agents.

Establishment of protocol design process

Various types of interactions occur in the real world because humans behave completely autonomously even under social constraints. Therefore, the existing validation by formal description of protocols, such as finite state machine and Petri Nets, is not sufficient. Iteration of simulation including participants as well as agents is necessary to check whether the refined protocols are valid.

This chapter will first explain what participatory simulation is. Next, we will discuss the agent architecture that separates decision making from

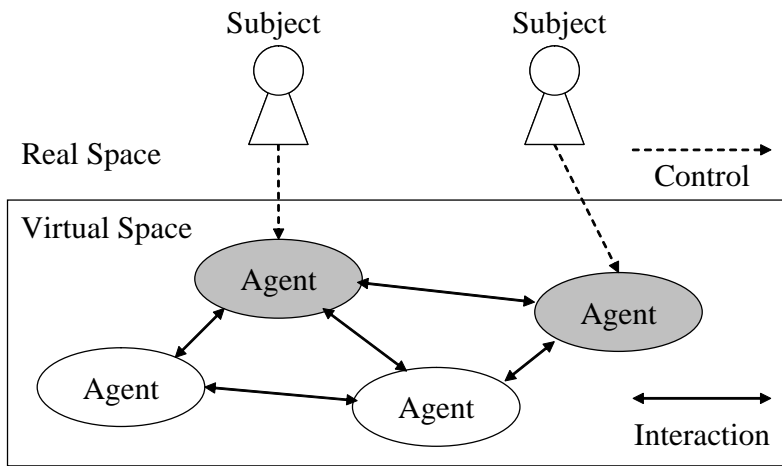


Figure 4.1: Participatory Simulation

interpretation of given protocols. Then, we propose protocol refinement process using participatory simulation and multi-agent simulation based on the designed agent architecture. Finally, we will validate our approach in an evacuation domain by improving an evacuation method proposed by Sugiman in 1988[Sugiman 88].

4.2 Participatory Simulation Approach

A participatory simulation is a multiagent simulation in which humans control some of the agents. The humans observe the virtual space from the view of their agents, and operate their agents by using controllers such as keyboards in front of desktop PCs. The coexistence in virtual space of autonomous agents and agents controlled by humans can realize indirect interactions between agents and humans (Figure 4.1).

Participatory simulation has been employed as one of approaches to virtual training systems. Virtual training systems for individual tasks, such as the acquisition of flight techniques, are already in use, and are becoming popular for teamwork training, such as leadership development[Rickel 99,

Traum 02]. This is because several of the human participants necessary to conduct group training can be replaced by software agents. These agents, however, must offer highly realistic and compelling social interactions with humans because they have to interact with the trainee [Johnson 00, Cavazza 02].

Therefore, the central issue in virtual training domain is to reproduce visually realistic interactions between agents and human-controlled avatars. Agents that provide realistic interactions have been realized by adding human-like expressions to agents, such as speech and nonverbal and emotional behavior [Cassell 00, Marsella 02, Lester 00, Pelachaud 02], since these are needed for the face-to-face interactions that occur in collaborations.

On the other hand, our research goal is to validate designed protocols and acquire participant's behavior models while observing interactions between agents and human-controlled avatars in multi-agent simulations. Drogoul *et al.* proposed a methodological process for developing multiagent-based simulations, and introduced the idea of participatory simulation [Drogoul 03]. However, they took consideration into refinement of agents, but not protocols. The final goal of this research is to reflect results of participatory simulation in system design by refining interaction protocols as well as participant's behavior models.

4.3 Agent 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. Such a difference of attitudes towards the protocols change 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 [Bellifemine 00, Bellifemine 99],

and deploying a filtering function between each agent and its environment in order to control interactions [Esteva 04].

On the other hand, agent architecture with social constraints has to re-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.

4.3.1 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 4.2 shows the agent architecture with social constraints, which we design.

In this architecture, observations which an agent senses are symbolized as external events, which are passed into the action selection. At the action

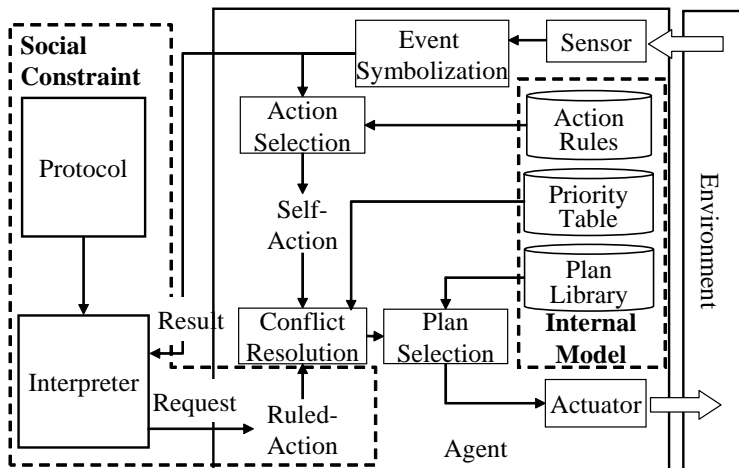


Figure 4.2: Agent Architecture under Social Constraints

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.

4.3.2 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. This implementation is shown in Figure 4.3.

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 han-

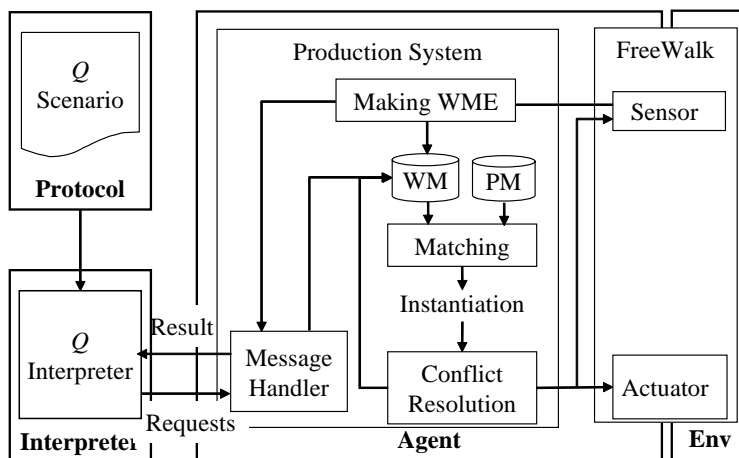


Figure 4.3: Implementation of Agent under Social Constraints

dler 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 given 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 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 4.4 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

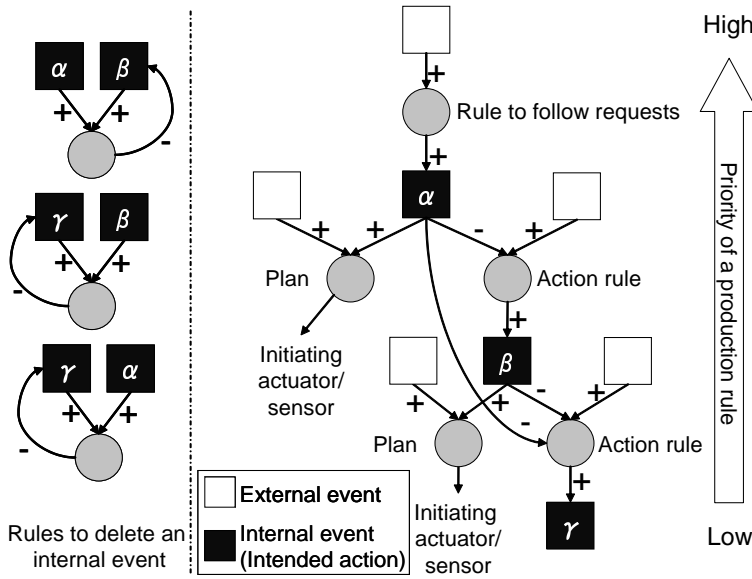


Figure 4.4: Data Dependency Graph of Production Rules

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.

4.4 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 [Mazouzi 02, Huget 03a, Fornara 03]. This process is realized by assuming that every agent complies with interaction protocols and the protocols describe every interaction between them.

Unlike this, interaction protocols in human society describe only partial interaction between humans, and humans and systems, and delegate deci-

sion making on what to do to humans. This is why participatory simulation where agents and human-controlled avatars coexist is effective to refine interaction protocols between socially-embedded systems and their users. However, we have to gather some subjects to conduct participatory simulation, which are high costs. Therefore, we have to explore the efficient process to introduce participatory simulation into protocol refinement.

4.4.1 Overview of Process

In this subsection, we describe overview of our protocol refinement process referring to Figure 4.5. The following subsections provide details of each step using our experiments on the evacuation domain introduced in section 2.3.

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.

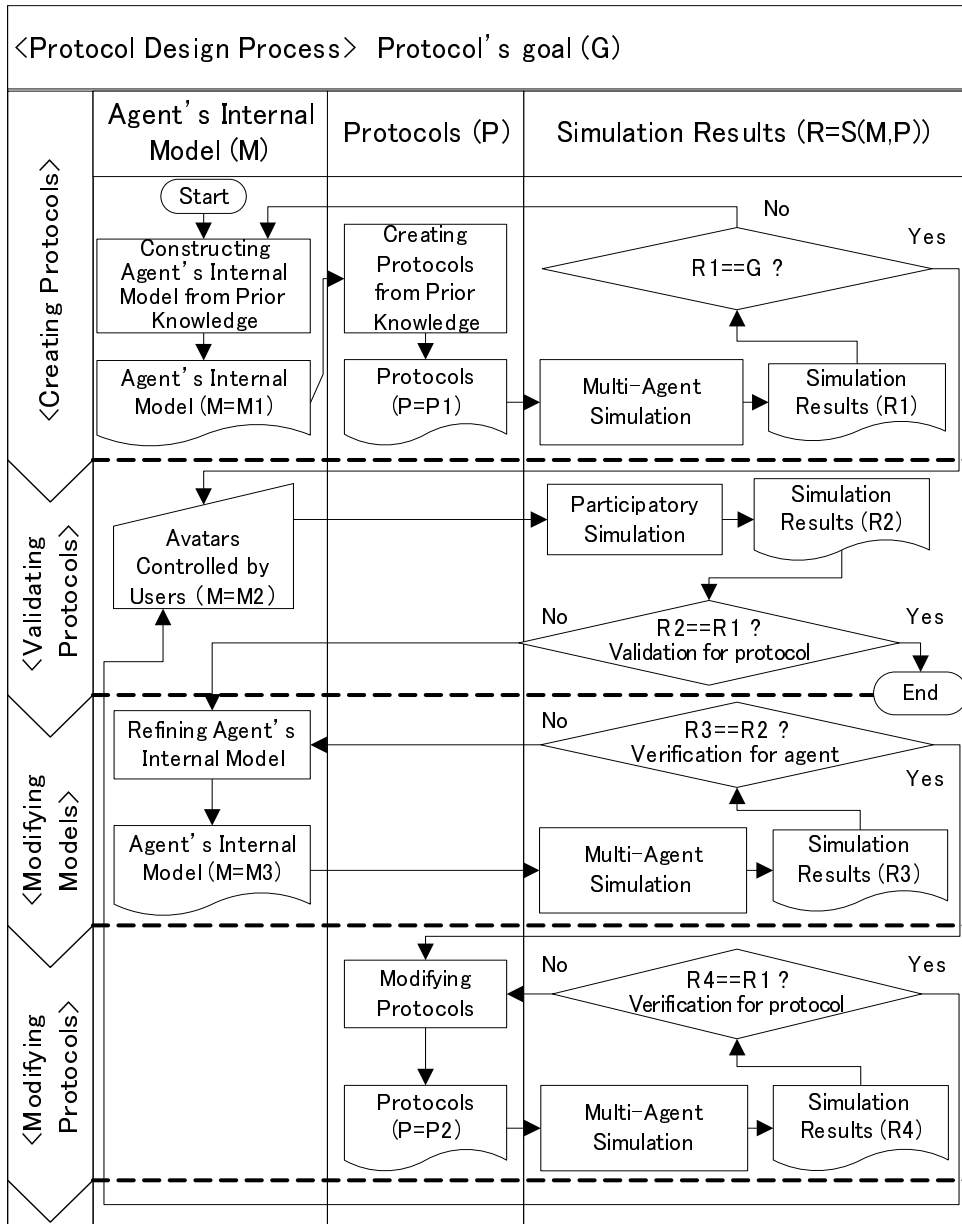


Figure 4.5: Protocol Design Process

4.4.2 Step1: Creating Protocols

As mentioned in the previous chapter, we succeeded in double-checking the result of the previous fire-drill experiment by multi-agent simulation [Murakami 03]. 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.

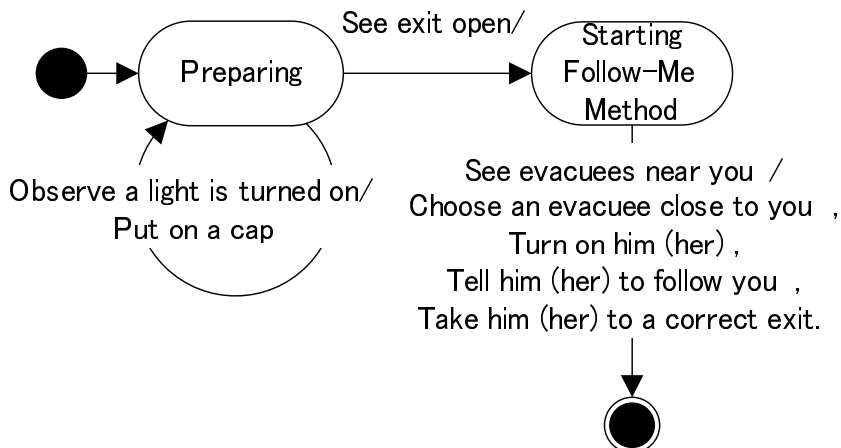


Figure 4.6: Follow-Me Method (Leader’s Protocol)

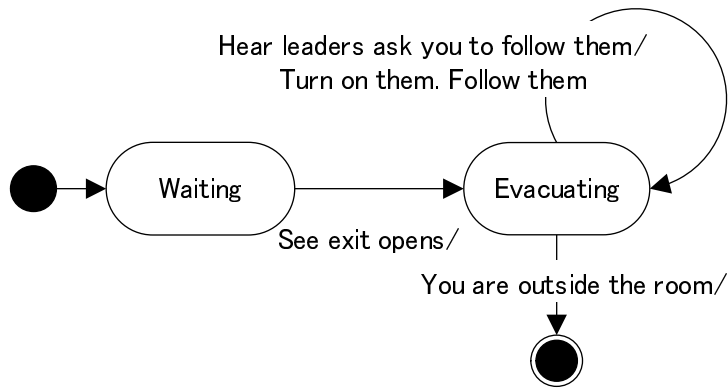


Figure 4.7: Follow-Me Method (Evacuee’s Protocol)

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 4.6 and Figure 4.7, respectively.

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 4.1. Note that

Table 4.1: Agent Internal Model during Fire Drills

Roles	Rules
Leader(M1)	(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 towards 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.
	(Plan) If the evacuee also goes out from the room, he follows the same target again.

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 [Nakanishi 99], it is easy to realize participatory simulation in the next step.

4.4.3 Step2: Validating Protocols

In the second step, we conduct participatory simulation by replacing some agents with human-controlled avatars. Participatory simulation enables us

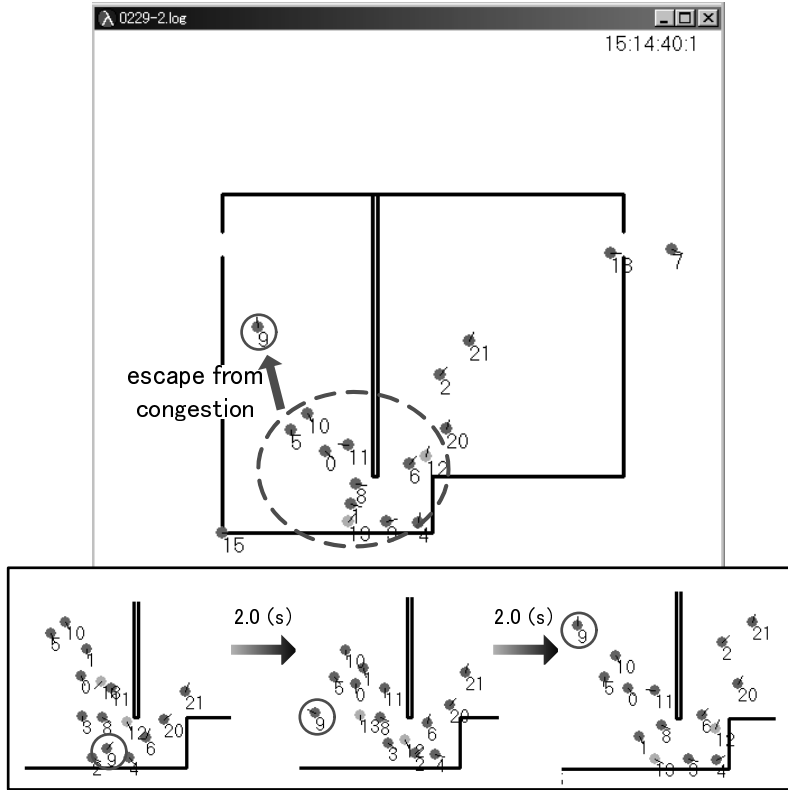


Figure 4.8: Results of the Participatory Simulation

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 4.8 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.

The above result implies that there is another agent’s internal model other than the model constructed so far. In the next step, we try to extract the new internal model from log data obtained by participatory simulation.

4.4.4 Step3: Modifying Agent Models

In third step, we modify the agent’s internal model using log data obtained by participatory simulation. Specifically, we refine the internal model of the avatar taking unpredictable behavior, by interviewing with the subject while showing him his captured screen, acquiring an internal model by machine learning, and reproducing the situation in participatory simulation by log data. Validity of the modified internal model is checked by comparing the result of multi-agent simulation with the modified agent model and one of participatory simulation. Modifying the agent model is repeated until the result of participatory simulation is reproduced by multi-agent simulation with the modified agent model.

In participatory simulation, we actually captured two subject’s screens on videotape and then interviewed with them while showing the movie to them. Figure 4.9 shows the interview with the subject. Showing the movie enables the subjects to easily remember what they focused on at each situation, and how they operated their avatars. At the interview, we asked the subjects three questions; “what did you focus on?” “what did you want to do?” and “what did you do?” Table 4.2 classifies the acquired rules into action rules and plans depending on the same criterion as in step 1.

However, it costs us high to interview in such a style, and so it is unreal-

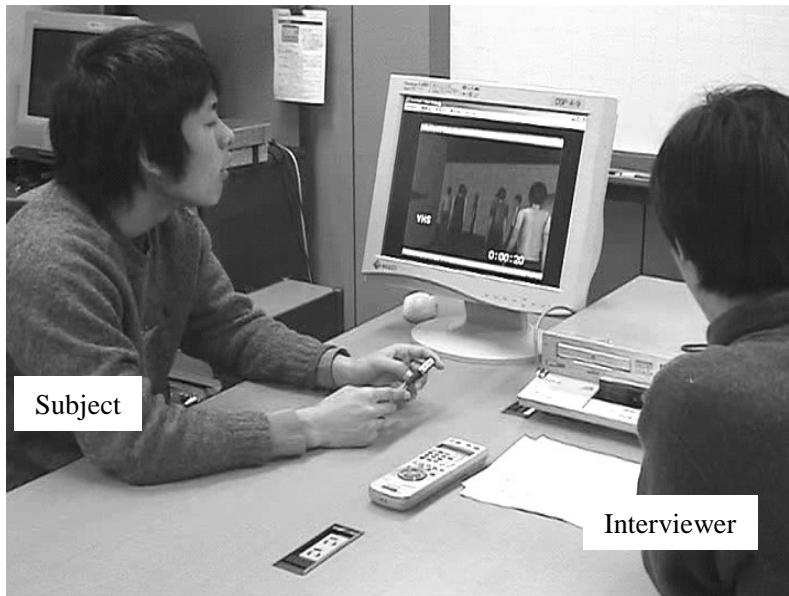


Figure 4.9: Interviews with Subjects

Table 4.2: Evacuee’s Internal Model Acquired from the Participatory Simulation

Roles	Rules
Evacuee(M3)	(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.

istic to interview with every subject. Therefore, we also propose the method that can support acquirement of agent models by applying a kind of machine learning [Murakami 05]. See also chapter 5 for the details.

4.4.5 Step4: Modifying Protocols

In the fourth step, we modify the protocols in order to accomplish system designer’s goal that the protocols can control the modified agent model correctly. Specifically, modifying the protocols is repeated until the result of multi-agent simulation consisting of the agent models acquired in the previous step satisfies the system designer’s goal.

In fact, we modified “Follow-me method” protocol by adding new state with the following transition rules; “if the leader finds an evacuee walk towards the reverse direction, he tells the evacuee to come with him.” The correctness of this protocol was checked if the simulation satisfied the goal that every evacuee goes out from the correct exit. The modified protocol is shown in Figure 4.10.

Finally, we will conduct participatory simulation using the refined protocols in order to validate the protocols.

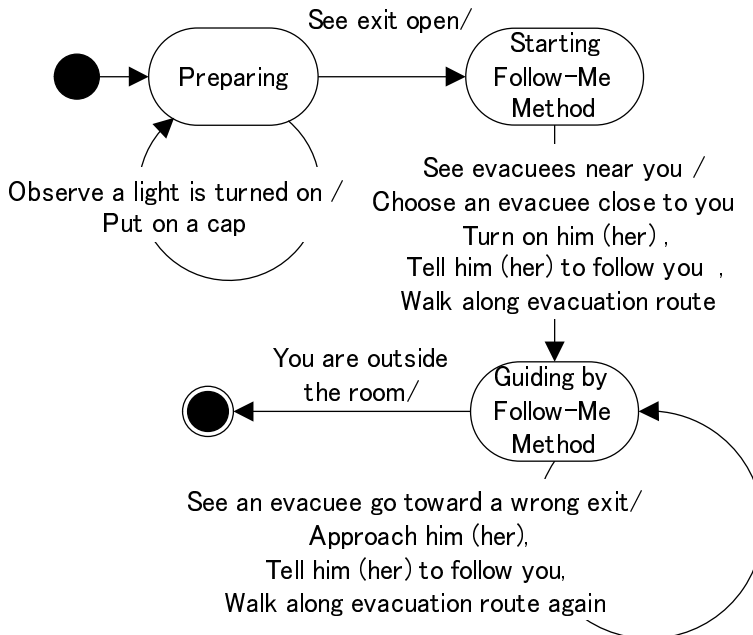


Figure 4.10: Modified Follow-Me Method (Leader’s Protocol)

4.5 Summary

In designing interaction protocols between socially-embedded systems and their users, we can examine the more realistic behavior of the system by participatory simulation, where agents and human-controlled avatars coexist, as well as multi-agent simulation, where only agents exist. The objective of this research is to employ participatory simulation in order to reflect subjects' feedback in protocol design. To accomplish this goal, we address the following issues.

Realization of autonomy under social constraints

Unlike multi-agent simulation, subjects controlling avatars are so autonomous that they sometimes violate the given protocols depending on their situation if they justify the violation. This kind of autonomy is also important to examine an appropriate degree of abstraction of the protocols. Therefore, we developed agent architecture separating decision making from interpretation of the given protocols by using scenario description language Q and a production system. Considering priorities of production rules and data dependency between the rules, we can realize social agents strictly complying with the given protocols and selfish agents sometimes violating the protocols.

Establishment of protocol design process

Although participatory simulation is very effective to validate protocols, it costs us to conduct it because of the fact that it needs several subjects. Therefore, we proposed the protocol refinement process that can not only validate the protocols but also acquire the models of participants for protocol refinement. This process defines criteria of verification and validation for agent models and protocols so that any system designers can reflect subjects' feedback in protocol design regardless of their ability. In fact, we applied our proposed method to improving evacuation guidance protocols and validated its usefulness.

The proposed protocol refinement process focuses on development of protocols based on careful consideration of the subjects' decision making. This methodology differs from the existing methodologies [Zambonelli 03, Wook 00, Kendall 00, Caire 01, Padgham 02, Juan 02], which design multi-agent systems from their goals in a top-down style. Our methodology is a middle way that designs protocols in top-down style while constructing agent models by participatory simulation in bottom-up style.

Chapter 5

User Modeling by Participatory Simulation

5.1 Introduction

In the previous chapter, we have presented the protocol refinement process to design more effective protocols by using participatory simulation where human-controlled avatars and scenario-controlled agents coexist. To more appropriately learn lessons from results of the simulations, we should carefully analyze the data. Machine learning can help this analysis, and so we present a method to acquire human behavior model with one of machine learning techniques in this chapter.

Diverse interactions between socially-embedded systems and their users occur because they are used in a dynamic and open environment. For instance, when using a car navigation system in the real world, some users follow directions given by a car navigation system, while other users sometimes turn off the route directed by it even when they are located in the same situation. Modeling such diverse human behavior as agents is critical for designing interaction protocols between users and systems. As the first step in developing diverse agents, we set our goal as acquiring diverse user operation models from the logs of agents controlled by human subjects in participatory simulations. Models to decide agent behavior can be approx-

imated by the operation models because the agents in the simulations are fully operated by the human subjects. In this research, operation models are defined as sets of prioritized operation rules, which represent how the subjects operated their agents in the simulations. In order to accomplish our goal, we address the following two research issues.

Extracting operation models that offer personality

To design interaction protocols adaptable for diverse users, the agent behavior must exhibit distinct personalities in multi-agent simulations. Our approach is to extract personal operation models from the subjects' operation history and the logs of agents who participated in simulations. This is different from inductive approaches to acquire a general user model from all subjects' log data.

Forming consistent operation models

We must combine a wide variety of operation rules to realize the diversity of operation models. However, many of these rules will be inconsistent. When developing a operation model, we have to combine several operation rules while maintaining the consistency of the model.

In this research, we apply a framework of hypothetical reasoning to acquirement of individual operation models. Hypothetical reasoning is a practical technique which is employed to diagnose system faults and diseases so far. It is a reasoning technique which infers that assumptions are true if observations are explained as logical consequence using the rule "System faults and causal factors of a disease cause their symptoms" on the assumptions. Based on the human decision rule "Human executes the action rule whose priority is the highest of the applicable rules on each state," subjects' behavior is determined by the set of operation rules and their priorities they have. Therefore, acquirement of each subject's model is formalized as a search problem that find the combination of operation rules and their priorities which can explain the observed behavior of a subject consistently, the same as diagnosis problems.

Here, the consistency is defined as an operation model includes different operation rules whose preconditions are the same, but whose effects are different. This means an inconsistent behavior that a subject with the model operates his/her agent in a different way even at the same state. In hypothetical reasoning, we can describe a constraint about such an inconsistent behavior. By checking whether a chosen operation rule is consistent with an model acquired by that time, hypothetical reasoning assures that a consistent operation model is finally acquired.

If we even collect subjects' behavior and operation rules available for subjects, application of hypothetical reasoning enables us to acquire diverse operation models. Although it is possible to deduce the operation rules used by a subject from his/her observed behavior by using expert knowledge concerning social psychology, the knowledge is difficult to acquire because of the fact that most knowledge is experimental, informal, and implicit. Furthermore, for the purpose of designing interaction protocols for socially-embedded systems, we need diverse agent models specific to the service domain and the service environment in order to reproduce a realistic simulation. This is why we have to acquire domain-specific knowledge and the knowledge acquirement become more difficult in this research. Therefore, we believe that hypothetical reasoning is effective to acquire subjects' operation model from results of participatory simulations.

This chapter will first clarify what a participatory simulation is and the modeling process that it permits. Next, we define the technical terms used in this chapter, and then formalize our target problem by using the terms for hypothetical reasoning. Finally, we validate our method in the evacuation domain by developing an operation model of a subject operating an evacuee agent.

5.2 Overview of Modeling Process

The use of participatory simulations (refer to section 4.2 in details) in the modeling process brings the following three benefits.

- It enables us to extract observations of each human subject and their operation history from the log data.
- It enables us to capture a subject's screen, which is then shown to the subject at the interview stage to acquire his/her operation rules effectively.
- It enables us to employ scenario-controlled agents in order to compensate for the lack of participants.

In this research, we acquire a subject's operation model, a set of operation rules, by explaining the subject's behavior observed in one or more participatory simulations. However, the operation rules so gained may exhibit some incompatibility. Therefore, we hypothesize whether each operation rule is employed by the target subject, and choose the assumptions that pass hypothetical reasoning, which offers the consistent selection of hypotheses. The result of hypothetical reasoning is a set of compatible operation rules employed by the target subject. We propose the following modeling process (Figure 5.1).

1. Multi-Agent Simulation

Conduct multi-agent simulation with agents modeled using prior knowledge, documents, and interviews with experts. Agent models are validated by comparing results of real experiments and simulation.

2. Participatory Simulation

Conduct participatory simulations with agents controlled by human subjects and record the log data of them.

3. Symbolizing Observation

Obtain a target subject's behavior from log data consisting of coordinate values and orientation of his/her agent, and describe the observations in predicate logic.

4. Interview with Subjects

Collect operation rules constituting domain knowledge through interviews with some of the human subjects.

5. Explanation of Observation with Hypothetical Reasoning

Generate explanations of the target subject's operation by using the domain knowledge and the observations.

6. Question and Answer

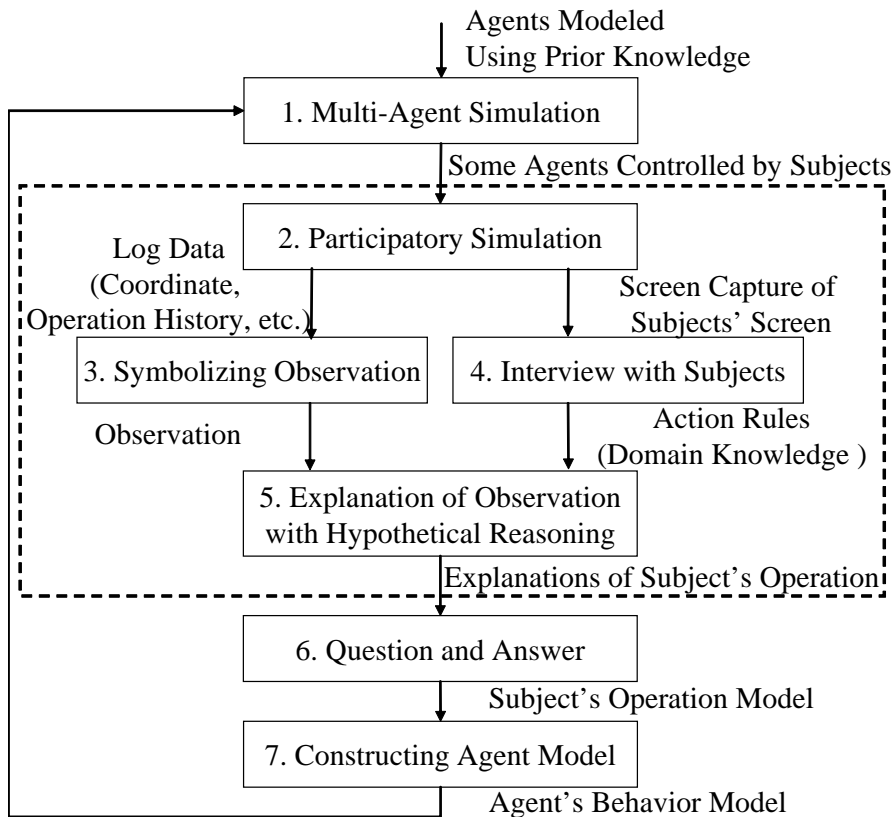


Figure 5.1: Modeling Process (This chapter focuses on the area in the dotted frame)

Winnnow the explanations generated in the previous step by questioning the target subject. The hypotheses in the last explanation represent the subject's operation model.

7. Constructing Agent Model

Construct an agent model through expert analysis of the winnowed explanation of the target subject's behavior.

In this research, we focus on using hypothetical reasoning to acquire the operation models shown in the dotted frame.

5.3 Formalizing the Problem

In this section, to permit the application of hypothetical reasoning to the acquisition of subject operation models, we formally define the domain knowledge and observations.

5.3.1 Definition of Technical Terms

In this research, we assume that a subject decides his/her next operation based on the world as observed from the view of his/her agent, and his/her operation model. The world observed by the subject is indicated by S . S consists of conjunctions of literals about the world. We describe this piece of information at time t as S_t . Time is a discrete value incremented by one as the world observed by the subject changes. The operation model is a set of prioritized operation rules $\langle P, \preceq \rangle$. P is a set of operation rules that could be employed by the subject, and \preceq represents the priorities of the elements of P . P is a subset of $Rules$, a set of operation rules obtained through interviews with subjects. \preceq is a subset of the Cartesian product $Rules \times Rules$. If each operation rule in $Rules$ is indicated by $rule_i$ ($0 \leq i \leq j \leq |Rules|$), $\langle rule_i, rule_j \rangle \in \preceq$ can be described as $rule_i \preceq rule_j$.

For applying hypothetical reasoning to the modeling of subjects, we define an operation selection mechanism and operation rules as domain

knowledge Σ . Each element of domain knowledge is indicated by σ_k ($0 \leq k \leq |\Sigma|$). We hypothesize which operation rules are employed by the target subject ($rule_i \in P$), and which rules take priority ($rule_i \preceq rule_j$). A set of these hypotheses, an inconsistent set of assumptions, is indicated by H . Moreover, we describe the subject's behavior from the beginning of the participatory simulation, 0, to the end of the simulation, *end*, as observation G . The solution h of hypothetical reasoning represents the an operation model of the subject. In addition, h is a subset of H .

5.3.2 Domain Knowledge

Domain knowledge consists of operation rules obtained through interviews with subjects, the subjects' operation selection mechanisms, and constraints to prevent the inconsistency caused by the inappropriate combination of hypotheses.

We describe operation rules as condition-action rules. Subjects operate their agents following the action part of the rule whose conditions are satisfied. Example 1 shows a description of operation rules.

Example 1 (Description of operation rules).

*rule*₁ : if *Near*(x , self), *Noop*(x), *Noop*(self) then Initiate(walk)

*rule*₂ : if *Near*(x , self), *Walk*(x), *Noop*(self) then Initiate(walk)

*rule*₃ : if *Near*(x , self), *Noop*(x), *Noop*(self) then Initiate(turn)

*Rule*₁: the subject executing this rule makes his/her agent pass agent x if both are standing next to each other. *Rule*₂: the subject executing this rule makes his/her agent follow x if x walks by the standing agent. *Rule*₃: the subject executing this rule makes his/her agent look around if both are standing next to each other. In these examples, self indicates the agent controlled by the target subject.

Next, we define the operation selection mechanism.

Definition 1 (Operation selection σ_1).

$$\begin{aligned} & (\exists rule_i (rule_i \in P \wedge rule_i = \max_{\leq} \{rule | Applicable(rule, S_t)\})) \\ & \Rightarrow Do(action(rule_i)) \end{aligned}$$

A subject employs $rule_i$ whose priority is the highest among all operation rules applicable in S_t , the world observed by the subject at time t^* . As a result, the subject starts the operation described in the action part of $rule_i$ at time t . *Applicable* and *Do* are predicates meaning that the precondition of a rule is satisfied, and that the subject initiates an operation, respectively. In addition, function *action* returns the agent's action initiated by the subject when executing the rule which is an argument of the function.

Definition 2 (Continuation of operation σ_2).

A subject can continue his/her current operation. This is represented by the predicate Continue.

However, if *Continue* always is true, it is possible to explain whether every subject's operation is *Do* or *Continue*. Therefore, we define an operation a subject takes as either *Do* or *Continue*. That is, σ_1 and σ_2 are exclusive knowledge.

Finally, we introduce the following constraint to prevent the inconsistency caused by the erroneous combination of hypotheses.

Definition 3 (Constraint: σ_3).

$$\begin{aligned} & \forall rule_i, rule_j (rule_i, rule_j \in P \wedge (condition(rule_i) = condition(rule_j))) \\ & \Rightarrow (action(rule_i) = action(rule_j)) \end{aligned}$$

P does not include operation rules whose preconditions are the same but whose actions are different. Function *condition* returns the precondition of the operation rule which is an argument of the function.

*In this research, we use the simple decision making rule in order to understand how to acquire an operation model by hypothetical reasoning, but we can also use other decision making rules as long as they are tautologous.

5.3.3 Description of Observation

The number of observations increases every time S changes. We define G and G_t , observations at time t , as below.

Definition 4 (Observation G).

$$G \equiv (G_0 \wedge \dots \wedge G_t \wedge \dots \wedge G_{end})$$

Definition 5 (Observation G_t).

$$G_t \equiv (S_t \Rightarrow A_t)$$

Observation G_t describes what situation the target subject observes from his/her agent's view, and how the subject operates his/her agent. The subject's operation at time t is indicated by A_t . Specifically, A_t is either the literal represented by predicate *Do*, meaning the subject initiates an operation, or one represented by predicate *Continue*, meaning the subject continues an operation.

These observations, in time-series form, are obtained from the log data of participatory simulations, and are described in predicate logic. Given the nature of the virtual space used, the log data consists of agent coordinate values and orientation, and subject's operation history. For example, we use observation G_{T-1} to indicate that the subject's operation at time $T-1$ results in John, see Figure 5.2, starting to walk at time T after standing at time $T-1$, as below.

Example 2 (Description of observation G_{T-1}).

$$\begin{aligned} &Near(\text{Mike}, \text{John}) \wedge Near(\text{Paul}, \text{John}) \wedge Far(\text{Tom}, \text{John}) \wedge Noop(\text{Mike}) \\ &\wedge Walk(\text{Paul}) \wedge Walk(\text{Tom}) \wedge ToLeft(\text{Paul}) \wedge ToLeft(\text{Mike}) \wedge Forward \\ &(\text{Tom}) \wedge InFrontOf(\text{Paul}, \text{John}) \wedge InFrontOf(\text{Mike}, \text{John}) \wedge InFrontOf \\ &(\text{Tom}, \text{John}) \wedge \dots \wedge Noop(\text{John}) \Rightarrow Do(\text{walk}) \end{aligned}$$

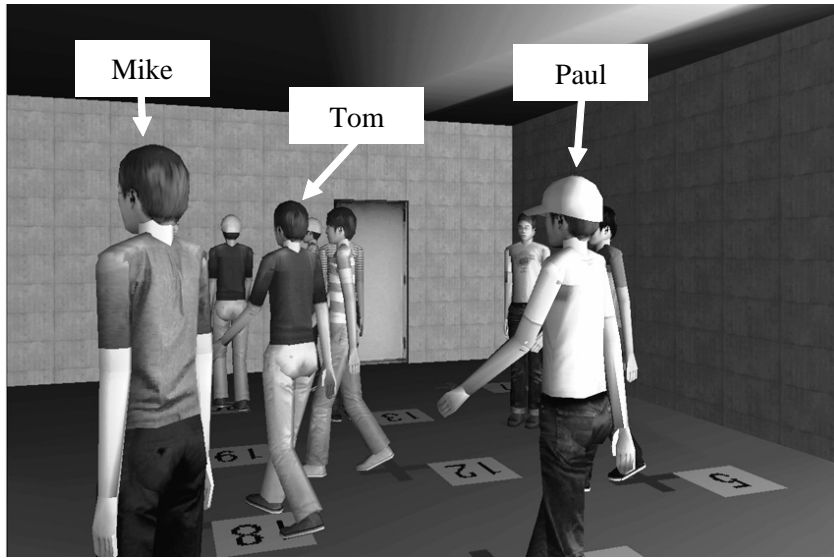


Figure 5.2: Agent John's View at T-1

The predicates constituting the above observation are classified into the following four types.

Distance

Near(x, y)/Far(x, y): Agent x is [near/far from] agent y .

Positional relation

InFrontOf(x, y)/Behind(x, y)/OnRightOf(x, y)/OnLeftOf(x, y): Agent x [is in front of/behind/on right of/on left of] agent y from the target subject's view.

Orientation

Forward(x)/Backward(x)/ToLeft(x)/ToRight(x): Agent x faces the [same/reverse] direction as the agent controlled by the target subject. Agent x faces to the [left/right] from the target subject's view.

Action

Noop(x)/Walk(x)/Turn(x): Agent x is [standing/walking/turning].

5.4 Acquisition of Operation Model

In this section, we explain how to acquire operation models through the use of hypothetical reasoning [Poole 87, Poole 94]. Specifically, we generate an explanation of G , behavior of the target subject, with Σ , which consists of operation rules and an operation selection mechanism, and H , whether each operation rule is executed by the agent and which rule has priority. The solution h fulfills the following three conditions and is the behavior model of the agent.

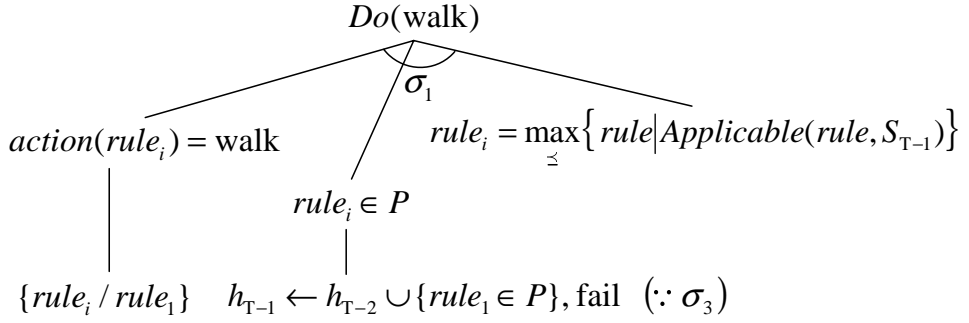
1. $h \cup \Sigma \vdash G$
2. $\Sigma \cup h$ is consistent.
3. A subset of h does not fulfill the above conditions.

First, the condition is transformed into $h \cup \Sigma \vdash G_0, \dots, h \cup \Sigma \vdash G_t, \dots, h \cup \Sigma \vdash G_{end}$ by applying “Rule T”, transitive law of provability, since G consists of conjunctions of G_t as indicated by definition 4. Additionally, $h \cup \Sigma \vdash G_t$ is transformed into $h \cup \Sigma \cup \{S_t\} \vdash A_t$ by applying the deduction theorem to the condition since G_t is equivalent to $S_t \Rightarrow A_t$ as indicated by definition 5. For example, the proof of G_{T-1} , the observation shown in Example 2, is transformed into $h_{T-2} \cup \Sigma \cup \{S_{T-1}\} \vdash Do(\text{walk})$. This proof is illustrated by the proof tree in Figure 5.3. Note that we assume *Rules* to be $\{rule1, rule2, rule3\}$, which are illustrated in Example 1, and h_{T-2} , the operation model acquired from G_0, \dots, G_{T-2} , to be $\{rule3 \in P\}$. G_{T-1} is proved in the following process.

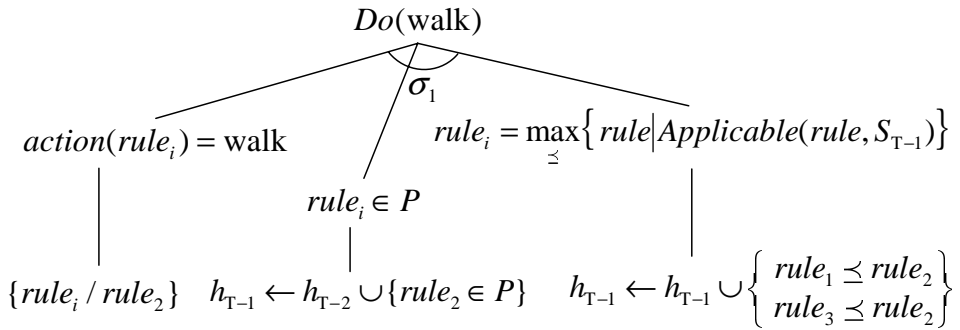
1. According to σ_1 , the proof of $Do(\text{walk})$ needs to prove $action(rule_i) = \text{walk}$, $rule_i \in P$, and $rule_i = \max_{\leq} \{rule \mid Applicable(rule, S_{T-1})\}$ to be true.
2. $Rule_1$ and $rule_2$ can satisfy the first condition, $action(rule_i) = \text{walk}$, since their consequents are $Initiate(\text{walk})$. There are two alternatives to generate the proof tree.

$$\begin{aligned}
S_{T-1} &= \left\{ \text{Near}(\text{Mike}, \text{John}) \wedge \text{Near}(\text{Paul}, \text{John}) \wedge \dots \wedge \right. \\
&\quad \left. \text{Noop}(\text{Mike}) \wedge \text{Walk}(\text{Paul}) \wedge \dots \wedge \text{Noop}(\text{John}) \right\} \\
A_{T-1} &= \text{Do}(\text{walk}), \quad h_{T-2} = \{ \text{rule}_3 \in P \} \\
\text{Rules} &= \left\{ \begin{array}{l} \text{rule}_1 : \text{if } \text{Near}(x, \text{self}), \text{Noop}(x), \text{Noop}(\text{self}) \text{ then Initiate}(\text{walk}) \\ \text{rule}_2 : \text{if } \text{Near}(x, \text{self}), \text{Walk}(x), \text{Noop}(\text{self}) \text{ then Initiate}(\text{walk}) \\ \text{rule}_3 : \text{if } \text{Near}(x, \text{self}), \text{Noop}(x), \text{Noop}(\text{self}) \text{ then Initiate}(\text{turn}) \end{array} \right\} \\
H &= \{ \text{rule}_1 \in P, \text{rule}_2 \in P, \dots, \text{rule}_1 \preceq \text{rule}_2, \text{rule}_2 \preceq \text{rule}_1, \text{rule}_3 \preceq \text{rule}_2, \dots \}
\end{aligned}$$

(a) Domain Knowledge and Hypothese



(b) Substitute $rule_1$ for $rule_i$



(c) Substitute $rule_2$ for $rule_i$

Figure 5.3: Proof Tree of $h_{T-2} \cup \Sigma \cup \{ S_{T-1} \} \vdash Do(\text{walk})$

3. Substitute $rule_1$ for $rule_i$

- (a) Choose the assumption, $rule_1 \in P$, from H to prove $rule_1 \in P$ true. However, $rule_3$ and $rule_1$ in P are incompatible according to σ_3 , and thus we are forced into back-tracking.

4. Substitute $rule_2$ for $rule_i$

- (a) Choose the assumption, $rule_2 \in P$, from H to prove $rule_2 \in P$ true.
- (b) Choose the assumptions, $rule_1 \preceq rule_2$ and $rule_3 \preceq rule_2$, from H to prove $rule_2 = \max_{\preceq} \{rule \mid Applicable(rule, S_{T-1})\}$ true.
- (c) $h_{T-1} = \{rule_2 \in P, rule_3 \in P, rule_1 \preceq rule_2, rule_3 \preceq rule_2\}$ is acquired.

By repeating the above process until G_{end} can be explained, we can acquire the operation model of the subject operating John. Algorithm 1 illustrates the algorithm used to acquire the operation model.

The goal of function `OperationModel` is to acquire operation model h such that $h \cup \Sigma \vdash G$, $\Sigma \cup h$ is consistent, and h is a minimal explanation. Consistency means that P does not include rules that have the same preconditions but different consequents. Therefore, we have only to confirm whether P is consistent by using Algorithm 2 every time P is updated (step 18 and step 19). P is updated by adding an operation rule non-deterministically chosen by the `choose` function. If an inconsistency occurs, another rule is chosen from the remaining rules. If there are no other candidates, the `choose` function returns false.

To acquire a minimal explanation, function `OperationModel` tries to explain G_t using P acquired by the explanation of G_{t-1} (step 13-16). If it fails, it chooses a new operation rule not in P (step 17-24). If no candidate can explain G_t , the function restarts the explanation of G_{t-1} to acquire another P .

```

1:  $t$  /* time */
2:  $P$  /* A set of operation rules employed by the target subject
   ( $P \subseteq Rules$ ) */
3:  $\preceq$  /* Priorities among the elements in  $P$  */
4:  $S_t$  /* The world observed by the target subject at time  $t$  */
5:  $A_t$  /* The target subject's operation at time  $t$  */
6: if  $t < 0$  then
7:   return ( $\{\}$ ,  $\{\}$ )
8: end if
9:  $(P, \preceq) \leftarrow \text{OperationModel}(t - 1)$ 
   /* If the subject continues the same operation as time  $t - 1$  */
10: if  $A_t = \text{Continue}$  then
11:   return ( $P, \preceq$ )
12: end if
   /* If the subject initiates a new operation at time  $t$  */
   /* Choose an operation rule from  $P$  */
13: if choose  $p \in \{rule \mid \text{Applicable}(rule, S_t),$ 
    $A_t = \text{Do}(\text{action}(rule))\} \cap P$  then
14:   Update  $\preceq$ 
    $s.t. \forall r \in \{rule \mid \text{Applicable}(rule, S_t)\} \Rightarrow r \preceq p$ 
15:   return ( $P, \preceq$ )
16: end if
   /* Choose a new operation rule not in  $P$  */
17: if choose  $p \in \{rule \mid \text{Applicable}(rule, S_t),$ 
    $A_t = \text{Do}(\text{action}(rule))\} \setminus P$  then
18:    $P \leftarrow P \cup \{p\}$ 
19:   if  $\text{Inconsistent?}(P)$  then
20:     fail
21:   end if
22:   Update  $\preceq$ 
    $s.t. \forall r \in \{rule \mid \text{Applicable}(rule, S_t)\} \Rightarrow r \preceq p$ 
23:   return ( $P, \preceq$ )
24: end if
25: fail

```

Algorithm 1: $\text{OperationModel}(t)$ **return** (P, \preceq)

```

1: if  $\exists p_i, p_j (p_i, p_j \in P \wedge (condition(p_i) = condition(p_j)) \wedge$ 
    $\neg(action(p_i) = action(p_j)))$  then
2:   return true
3: end if
4: return false

```

Algorithm 2: Inconsistent?(P) **return** *boolean value*

5.5 Application to Evacuation Domain

To validate our method, we used it to model a subject operating an evacuee agent in an evacuation drill.

5.5.1 Evacuation Domain

We used our modeling method to create the evacuee agents needed to develop effective evacuation methods. Specifically, we used a controlled experiment conducted in the real world[Sugiman 88] (See also section 2.3 in details). 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 2.1. 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. All evacuees were guided to Exit B by the leaders.

At first, we reproduced the same situation in FreeWalk[Nakanishi 04], 3D virtual space platform. Next, we conducted a participatory simulation by replacing 12 scenario-controlled agents with agents controlled by human subjects (Figure 5.4). The remaining agents were controlled by their scenarios described in scenario description language Q [Ishida 02b, Murakami 03].



Figure 5.4: Participatory Simulation by FreeWalk

5.5.2 Application of Modeling Process

We applied our method to model the subject operating agent 11, circled in Figure 5.6 and Figure 5.7. The set of operation rules in domain knowledge consisted of the following 18 operation rules obtained through interviews with 6 human subjects.

Rule1: If the evacuee sees other escaping evacuees, he looks to the direction towards which they go.

Rule2: If the evacuee cannot see a leader, he looks for them by looking around and stepping backward.

Rule3: If the evacuee sees a lot of evacuees, he goes towards the direction to which they look.

Rule4: If the evacuee cannot see the exit, he looks for it by looking around and stepping backward.

- Rule5:** If the evacuee finds a leader, he approaches to him/her.
- Rule6:** If the evacuee sees a leader near him, he follows him/her.
- Rule7:** If the evacuee meets congestion, he goes towards it.
- Rule8:** If the evacuee finds a leader, he pays attention to him/her.
- Rule9:** The evacuee moves in the same direction as the leader.
- Rule10:** The evacuee moves in the same direction as a lot of evacuees.
- Rule11:** If the evacuee meets congestion ahead of his walk course, it stops.
- Rule12:** If the evacuee sees another evacuee heading toward the same direction begin to walk again, he steps forward.
- Rule13:** If the evacuee sees another evacuee heading toward the same direction stop, he also stops.
- Rule14:** The evacuee follows someone in front of him.
- Rule15:** If the evacuee encounters congestion, he walks through it.
- Rule16:** If someone in front of the evacuee walks slowly, he passes him/her.
- Rule17:** If the evacuee sees the exit, he goes toward it.
- Rule18:** The evacuee looks around before going to the exit.

Table 5.1 shows the sets of operation rules chosen from all rules through explanation of evacuee agent11's operation history. It is verified that each set of operation rule can explain the operation history because they are acquired by logical consequence. On the other hand, the validation of them is checked by comparing them with operation rules obtained by an interview with the evacuee. In this example, we confirmed that the set of operation rules obtained by the interview with the subject controlling agent11 is the

Table 5.1: Operation Rules of Subject Operating Agent 11

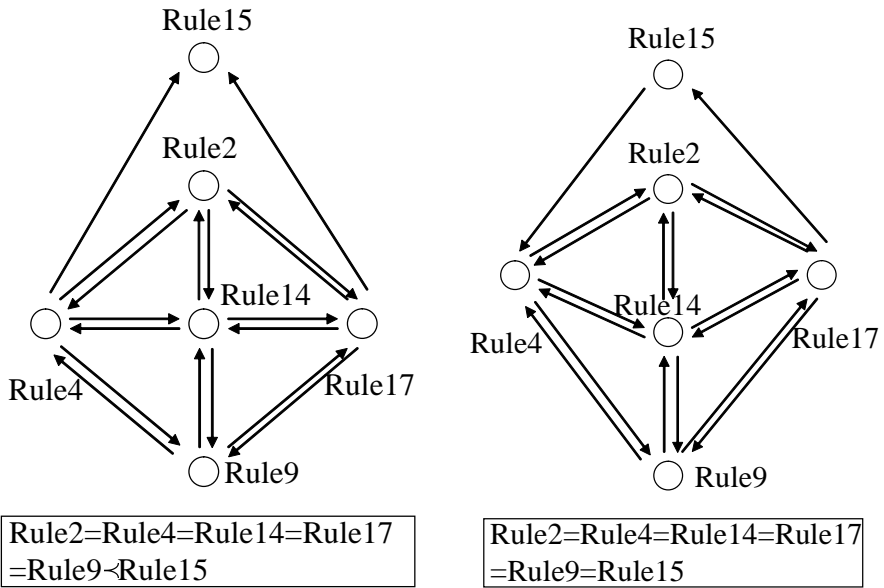
P	Set of Operation Rules
P_1	$P = \{\text{Rule2, Rule4, Rule9, Rule14, Rule15, Rule17}\}$
P_2	$P = \{\text{Rule2, Rule4, Rule9, Rule14, Rule16, Rule17}\}$
P_3	$P = \{\text{Rule2, Rule4, Rule14, Rule15, Rule17, Rule18}\}$
P_4	$P = \{\text{Rule2, Rule4, Rule14, Rule16, Rule17, Rule18}\}$

same as P_1 . However, it is unrealistic to validate every acquired model in this way because it takes about one hour to interview once.

Therefore, we propose the way to winnow acquired operation models in order to choose valid one from them by a question and answer system. In the proposed way, we divide the operation rule sets into two groups, and generate new observation which can be explained by either groups. A valid operation model is chosen by asking the target subject if the observation is true for him/her. For example, in the case of Table 5.1, asking the subject whether he goes towards the exit or looks around him when he sees a leader go to the exit enables us to select $\langle P_1, P_2 \rangle$ or $\langle P_3, P_4 \rangle$. By repeating such a process, it is possible to finally winnow a valid operation model from a set of verifiable operation models obtained by hypothetical reasoning. Note that additional interviews are needed to collect operation rules enough to explain every subject's behavior if we finally acquire no operation model.

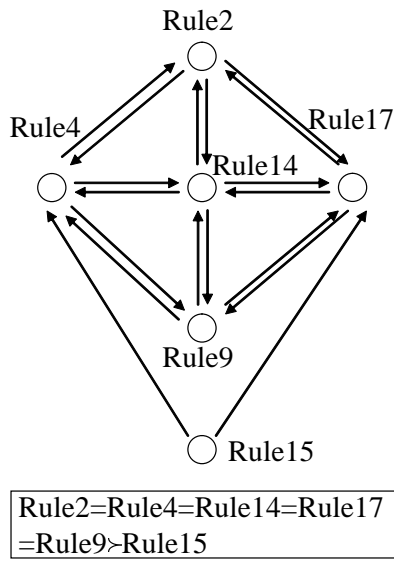
Regarding as properties among operation rules in P_1 , we can acquire three kinds of relations as shown in Figure 5.5. Arrow lines indicate priority between operation rules and the direction of the arrow corresponds to order of priority. In addition, when both $\text{Rule1} \preceq \text{Rule2}$ and $\text{Rule2} \preceq \text{Rule1}$ are satisfied, they are equal to $\text{Rule1} = \text{Rule2}$.

Each explanation generated by three kinds of priorities corresponds to results of the interview mentioned above. Specifically, we can calculate the following sets of applicable rules from the target subject's operation history in Figure 5.6 and Figure 5.7; $\{\text{Rule2, Rule14, Rule17}\}$, $\{\text{Rule4, Rule9, Rule14}\}$, $\{\text{Rule2, Rule4}\}$, $\{\text{Rule2, Rule4, Rule14}\}$, $\{\text{Rule9, Rule14,}$



(a)

(b)



(c)

Figure 5.5: Priorities among Operation Rules in P_1

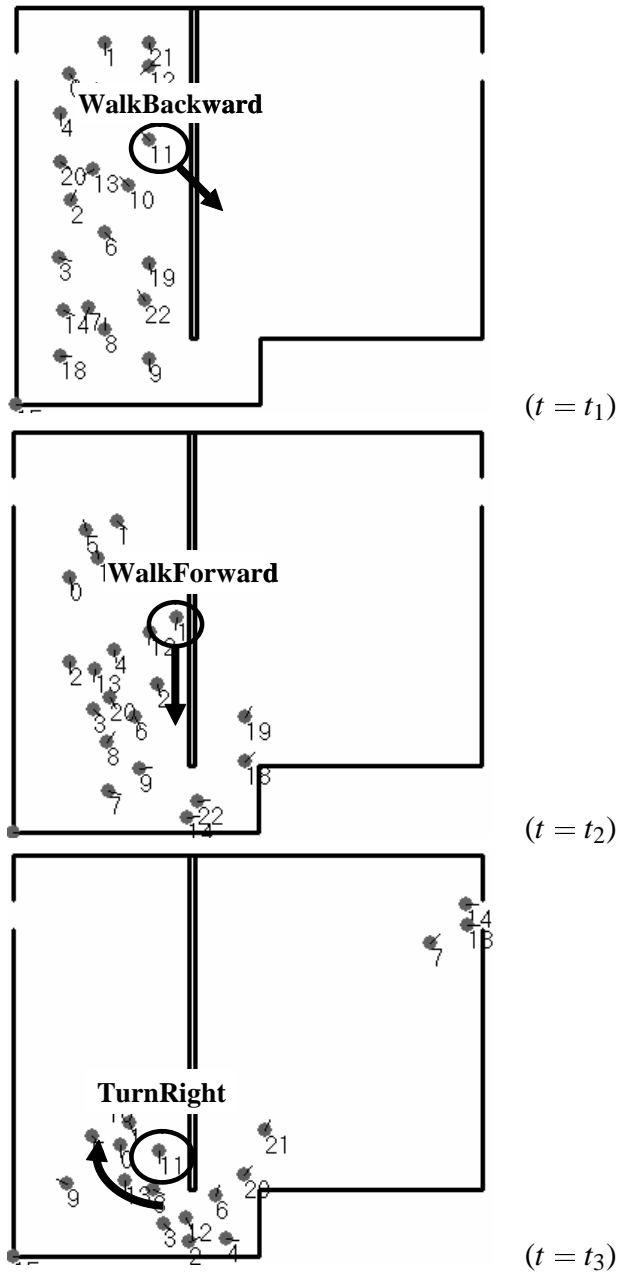


Figure 5.6: Operation History of the Subject Operating Agent11 (former)

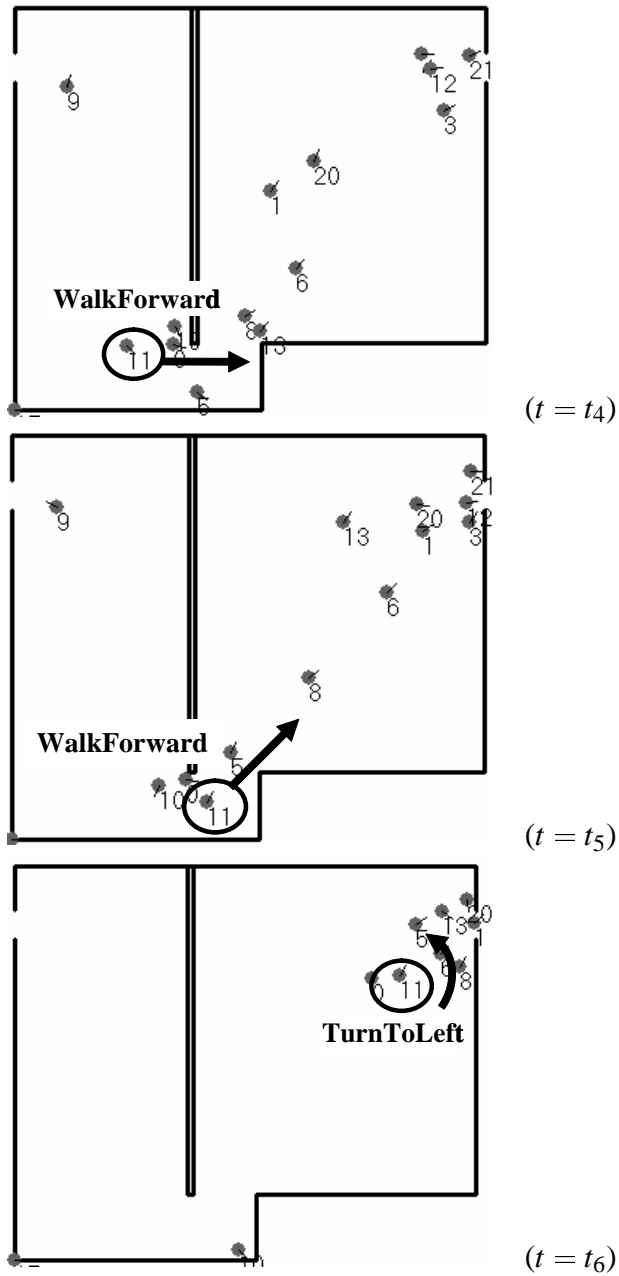


Figure 5.7: Operation History of the Subject Operating Agent11 (latter)

Rule17}, and {Rule15} are applicable at S_{t_1} , S_{t_2} , S_{t_3} , S_{t_4} , S_{t_5} , and S_{t_6} , respectively. The priorities choose Rule2, Rule9, Rule4, Rule14, Rule17, and Rule15 from each set of applicable rules. These choices are the same as the execution order of operation rules obtained by the interview. However, they may generate not only the correct execution sequence of operation rules but also other sequences because the priorities shown in Figure 5.5 include a lot of the same priorities. That is, precision of the execution sequences is low. Therefore, we need to specify the precondition of each operation rule in more detail in order to determine only one execution sequence.

5.6 Summary

To render multiagent simulations about socially embedded systems more realistic, a diversity of simulated-user models must be realized. In this research, we addressed the following two issues in realizing the needed diversity of operation models of users participating in the simulation.

Forming various consistent behavior models

We established a modeling process, based on hypothetical reasoning, that offers the consistent selection of assumptions. This allows us to combine operation rules in various ways while avoiding incompatibilities.

Extracting behavior models that offer personality

We extracted a personal operation model from log data by generating explanations of each subject's operation. This enables us to characterize each subject as combinations of operation rules constituting the explanation.

In this research, we have confirmed that there are some operation models that can explain correct execution sequence of operation rules, corresponding to one obtained by the interview, in models generated by our proposed method. For future works, we have to develop the method to determine only

one valid operation model from several acquired models. A specified model is implemented as an agent, and then it is employed to refine interaction protocols.

Chapter 6

Conclusion

6.1 Contributions

In developing socially-embedded systems, participatory design is needed in order to predict systems' behavior after introduction of them. Participation of their users in development process enables system designers to specify actual interactions between the systems and the users, and examine the expected behavior of the systems in society. In this research, we have proposed a new method that can realize participatory design at low cost by simulated-users, and support interaction design based on technological foundations. In this chapter, we summarize our contributions in accomplishing this approach as follows.

- Scenario description language for describing interaction between agents

We developed a general-purpose scenario description language Q to describe interaction protocols between agents. This enables us system designers, experts in an application domain, to define the expected behavior of the systems and the users by using the vocabularies in the application domain. The existing languages for multi-agent systems focus on agent's internal mechanisms, and so interactions is caused by agent's decision making implemented in them. On the other hand,

Q has no concern with the details of agent's internal mechanisms, and is a language to request agents to affect and observe their environment. It allows us to design interaction protocols and control agents' behavior while keeping their autonomy in the sense that Q delegates how to carry out the described requests to the agents. In addition, we developed IPC that can represent patterns of interactions in table form and describe interaction protocols by filling in the blanks in IPC cards. We can reuse interaction patterns by defining IPC and describe interaction protocols at low cost.

- Process for realizing coordination between system designers and system developers

We established new simulation process to enable system designers and system developers to cooperate with each other in order to realize multi-agent simulations. In contrast to development process established in object-oriented software engineering where model construction and implementation are connected serially, they are deployed in parallel. Once vocabularies to describe an interaction protocol are decided, they form an interface between system designers and system developers. System developers implement the vocabularies according to their specifications, while system designers describe interaction protocols using the vocabularies. Protocol description is independent of agent implementation, and so we can change interaction protocols without affecting agent implementation. That is, we can describe various types of interaction protocols even after starting to implement agents, and estimate how well the described protocols work through multi-agent simulation.

- Agent architecture for making decision under social constraints

We proposed new agent architecture with decision making under interaction protocols defined by system designers. Interaction protocol is a behavioral guideline that leads interactions between agents. By each agent's complying with protocols, their coordination is realized.

However, in participatory simulation, participants often violate the protocols designed by system designers in advance. This problem is caused by the fact that the goals accomplished by protocols are different from participants' goals. Therefore, system designers have to take consideration into users' decision making as well as consistency within interaction protocols, in order to construct more robust interaction protocols. To conduct multi-agent simulation with this purpose, we developed new agent architecture that can deal with requests described in protocols as external events the same as other events in environment. This architecture enables agents to decide whether they should comply with the given protocols or not.

- Process for improving interaction protocols

Through this thesis, we focus on the method to design interaction protocols with considering agent's autonomy. The existing methods in protocol engineering are not suitable for designing protocols imposed on agents whose goals sometimes conflict with those of protocols since those methods are based on the assumption that there are joint goals among agents. Therefore, we established the method that can refine interaction protocols using participatory simulation where agents and human-controlled agents coexist. In this process, we modify users' internal models and the interaction protocols alternatively in incremental steps until results of participatory simulation come to an ideal state. Moreover, this process presents two criteria to check whether modifications of agent's internal models and protocols are valid or not.

- Algorithms for acquiring a model of an user participating in simulations

We formalized acquiring participants' operation model from the log data obtained by participatory simulation as a search problem and presented a solution to the problem based on hypothetical reasoning. Although it costs lower to conduct participatory simulation than demon-

stration experiments in the real world, the number of times to conduct participatory simulation is limited for the reason that they also need several human subjects. In addition, we have to acquire not general operation model but diverse types of models so that we can design interaction protocols effective for socially-embedded systems that provide a lot of users with their services. Therefore, we can not take inductive approach that can acquire a general model from a lot of data, but deduce individual operation models by explaining log data with domain knowledge, a set of given operation rules. However, there can be incompatible operation rules in the domain knowledge. To avoid generating an explanation that consists of incompatible rules, we employed hypothetical reasoning.

6.2 Future Directions

It is expected that the number of socially-embedded systems increases more and more. That means our behavior will be constrained by several interaction protocols given by the system providers and we will have to handle several protocols. If we interact with more systems, it is a hard task to handle them. One of the solutions to this problem is introduction of personal agents that can support users personally. As a result, we believe that socially-embedded systems will evolve into socially-embedded multi-agent systems. Therefore, we conclude this thesis with the list of possible future research issues of protocol design for socially-embedded multi-agent systems.

- Coordination of several interaction protocols

Improvements of interoperability of information-providing systems by Web services and developments of semantic Web will enable us to easily construct a composite socially-embedded system. In this situation, we need personal agents that can support users directly. The problem to realize such a personal agent is to develop agent mechanism to resolve conflicts between the given protocols.

- Deliberate agents under social constraints

In this research, an agent architecture under social constraints is very simple one that performs actions described in the given interaction protocol if individual conditions to execute them are satisfied. However, in the real world, we reason about effects of the cases when we violate and follow the protocol in order to decide whether to comply with the protocol or not. Especially, what the subsequent actions are is an important factor to determine whether to obey the protocol. We need an agent architecture that can consider the potential deontic action described in the given protocol, for developing personal agent.

- Enforcement of interaction protocols by social pressure

There are various methods to force agents to comply with protocols. A fine is one of the methods. It is true that a fine is effective to make personal agents obey their protocols, but it is not effective for users because they have a choice that they do not use their personal agent. Therefore, it is necessary not to impose an explicit sanction but to motivate agents to comply with their given protocols. For example, if agents are equipped with decision making to cooperate with other agents based on compliance rate, they are motivated to follow the protocol as much as possible. We need socially-embedded multi-agent systems with such social pressure.

Bibliography

- [Axelrod 97] Axelrod, R.: *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*, pp. 4–5, Princeton Univ. Press (1997)
- [Barbuceanu 95] Barbuceanu, M. and Fox, M.: COOL: A Language for Describing Coordination in Multi Agent Systems, in *Proc. of the First International Conference on Multi-Agent Systems*, pp. 17–24 (1995)
- [Bellifemine 99] Bellifemine, F., Poggi, A., and Rimassa, G.: JADE - A FIPA-compliant agent framework, in *Proc. of the Fourth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*, pp. 97–108 (1999)
- [Bellifemine 00] Bellifemine, F., Poggi, A., and Rimassa, G.: Developing Multi-agent Systems with JADE, in Castelfranchi, C. and Lesperance, Y. eds., *Intelligent Agents VII. Agent Theories Architectures and Languages*, pp. 89–103, Springer-Verlag (2000)
- [Boella 03] Boella, G. and Torre, van der L. W. N.: Norm Governed Multi-agent Systems: The Delegation of Control to Autonomous Agents, in *Proc. of the IEEE/WIC International Conference on Intelligent Agent Technology*, pp. 329–335 (2003)
- [Boella 05] Boella, G. and Torre, van der L. W. N.: Enforceable social laws, in *Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 682–689 (2005)

- [Bradshaw 01] Bradshaw, J., Sierhuis, M., Gawdiak, Y., Jeffer, R., Suri, N., and Greaves, M.: Adjustable Autonomy and Teamwork for the Personal Satellite Assistant, in *Proc. of the IJCAI-01 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents* (2001)
- [Briggs 95] Briggs, W. and Cook, D. J.: Flexible Social Laws, in *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 688–693 (1995)
- [Caire 01] Caire, G., Coulier, W., Garijo, F., Gomez, J., Pavón, J., Leal, F., Chainho, P., Kearney, P., Stark, J., Evans, R., and Massonet, P.: Agent Oriented Analysis Using Message/UML, in Wooldridge, M., Weiß, G., and Ciancarini, P. eds., *Agent-Oriented Software Engineering II*, pp. 119–135, Springer-Verlag (2001)
- [Cassell 00] Cassell, J., Bickmore, T., Campbell, L., Vilhjalmsson, H., and Yan, H.: Human Conversation as a System Framework: Designing Embodied Conversational Agents, in Cassell, J., Prevost, S., Sullivan, J., and Churchill, E. eds., *Embodied Conversational Agents*, pp. 29–63, MIT Press (2000)
- [Castelfranchi 95] Castelfranchi, C.: Commitments: From Individual Intentions to Groups and Organizations, in *Proc. of the First International Conference on Multiagent Systems*, pp. 41–48 (1995)
- [Cavazza 02] Cavazza, M., Charles, F., and Mead, S.: Interacting with Virtual Characters in Interactive Storytelling, in *Proc. of the First International Conference on Autonomous Agents and Multi-agent Systems*, pp. 318–325 (2002)
- [Clancey 01] Clancey, W., Sachs, P., Sierhuis, M., and Hoof, R.: Brahms: Simulating Practice for Work Systems Design, *International Journal of Human-Computer Studies*, Vol. 49, pp. 831–865 (2001)

- [Decker 93] Decker, K. and Lesser, V.: Quantitative Modeling of Complex Computational Task Environments, in *Eleventh National Conference on Artificial Intelligence (AAAI93)*, pp. 217–224 (1993)
- [Dignum 99] Dignum, F.: Autonomous Agents with Norms, *Artificial Intelligence and Law*, Vol. 7, No. 1, pp. 69–79 (1999)
- [Dinkloh 04] Dinkloh, M. and Nimis, J.: A Tool for Integrated Design and Implementation of Conversations in Multiagent Systems, in Dastani, M., Dix, J., and Fallah-Seghrouchni, A. E. eds., *Programming Multi-Agent Systems, First International Workshop, PROMAS 2003, Melbourne, Australia, July 15, 2003, Selected Revised and Invited Papers*, pp. 187–200, Springer-Verlag (2004)
- [Doi 05] Doi, T., Tahara, Y., and Honiden, S.: IOM/T: An Interaction Description Language for MultiAgent Systems, in *Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 778–785 (2005)
- [Drogoul 03] Drogoul, A., Vanbergue, D., and Meurisse, T.: Multi-agent Based Simulation: Where Are the Agents?, in Sichman, J. S., Bousquet, F., and Davidsson, P. eds., *Multi-Agent-Based Simulation III*, pp. 1–15, Springer (2003)
- [Epstein 96] Epstein, J. and Axtell, R.: *Growing Artificial Societies: Social Science from the Bottom Up*, MIT Press (1996)
- [Esteva 04] Esteva, M., Rosell, M., Rodriguez-Aguilar, J., and Arcos, J.: AMELI: An agent-based middleware for electronic institutions, in *Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 236–243 (2004)
- [Finin 97] Finin, T., Labrou, Y., and Mayfield, J.: KQML as an Agent Communication Language, in Bradshaw, J. ed., *Software Agents*, MIT Press (1997)

- [Fornara 03] Fornara, N. and Colombetti, M.: Defining Interaction Protocols Using a Commitment-based Agent Communication Language, in *Proc. of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 520–527 (2003)
- [Hanks 93] Hanks, S., Pollack, M., and Cohen, P.: Benchmarks, Testbeds, Controlled Experimentation, and the Design of Agent Architectures, *AI Magazine*, Vol. 14, No. 4, pp. 17–42 (1993)
- [Hareesh 00] Hareesh, P., Shibano, N., Nakanishi, H., Kashiwagi, M., and Sawada, K.: Evacuation Simulation: Visualisation Using Virtual Humans in a Distributed Multi-User Immersive VR System, in *International Conference on Virtual Systems and Multi-Media (VSMM2000)*, pp. 283–289 (2000)
- [Holzmann 91] Holzmann, G. J.: *Design and validation of computer protocols*, Prentice Hall (1991)
- [Huget 03a] Huget, M.-P. and Koning, J.-L.: Interaction Protocol Engineering, in Huget, M.-P. ed., *Communications in Multiagent Systems*, pp. 179–193, Springer-Verlag (2003)
- [Huget 03b] Huget, M.-P. and Koning, J.-L.: Requirement Analysis for Interaction Protocols, in Marík, V., Müller, J., and Pechoucek, M. eds., *Multi-Agent Systems and Applications III, 3rd International Central and Eastern European Conference on Multi-Agent Systems*, pp. 404–412, Springer-Verlag (2003)
- [Ishida 02a] Ishida, T.: Digital City Kyoto: Social Information Infrastructure for Everyday Life, *Communication of the ACM*, Vol. 45, No. 7, pp. 76–81 (2002)
- [Ishida 02b] Ishida, T.: Q: A Scenario Description Language for Interactive Agents, *IEEE Computer*, Vol. 35, No. 11, pp. 54–59 (2002)

- [Ishida 04] Ishida, T.: Society-Centered Design for Socially Embedded Multiagent Systems., in Klusch, M., Ossowski, S., Kashyap, V., and Unland, R. eds., *Cooperative Information Agents VIII*, pp. 16–29, Springer (2004)
- [Johnson 00] Johnson, W., Rickel, J., and Lester, J.: Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments, *International Journal of Artificial Intelligence in Education*, Vol. 11, pp. 47–78 (2000)
- [Juan 02] Juan, T., Pearce, A. R., and Sterling, L.: ROADMAP: extending the gaia methodology for complex open systems, in *Proc. of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 3–10 (2002)
- [Kawamura 99] Kawamura, T., Hasegawa, T., Ohsuga, A., and Honiden, S.: Bee-gent: Bonding and Encapsulation Enhancement Agent Framework for Development of Distributed Systems., in *Proc. of the 6th Asia-Pacific Software Engineering Conference*, pp. 260–267 (1999)
- [Kendall 00] Kendall, E. A.: Agent Software Engineering with Role Modelling, in Ciancarini, P. and Wooldridge, M. eds., *Agent-Oriented Software Engineering*, pp. 163–169, Springer-Verlag (2000)
- [Kitano 97] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., and Osawa, E.: RoboCup: The Robot World Cup Initiative, in *Agents1997*, pp. 340–347, ACM Press (1997)
- [Kolb 95] Kolb, M.: A Cooperation Language., in *Proceedings of the First International Conference on Multiagent Systems*, pp. 233–238 (1995)
- [Kurumatani 03] Kurumatani, K.: Mass User Support by Social Coordination among Citizens in a Real Environment., in Kurumatani, K., Chen, S.-H., and Ohuchi, A. eds., *Multi-Agent for Mass User Support*, pp. 1–17 (2003)

- [Kuwabara 96] Kuwabara, K.: Meta-Level Control of Coordination Protocols, in *Proc. of the Second International Conference on Multi-Agent Systems*, pp. 165–173 (1996)
- [Labrou 99] Labrou, Y., Finin, T., and Peng, Y.: Agent Communication Languages: The Current Landscape, *Intelligent Systems*, Vol. 14, No. 2, pp. 45–52 (1999)
- [Lesser 99] Lesser, V., Atighetchi, M., Benyo, B., Horling, B., Raja, A., and Vincent, R.: The UMASS Intelligent Home Project, in *Proc. of the Third International Conference on Autonomous Agents*, pp. 291–298 (1999)
- [Lester 00] Lester, J., Towns, S., Callaway, C., Voerman, J., and FitzGerald, P.: Deictic and Emotive Communication in Animated Pedagogical Agents, in Cassell, J., Prevost, S., Sullivan, J., and Churchill, E. eds., *Embodied Conversational Agents*, pp. 123–154, MIT Press (2000)
- [López 02] López, y F. L., Luck, M., and d’Inverno, M.: Constraining autonomy through norms, in *Proc. of the First International Joint Conference on Autonomous Agents & Multiagent Systems*, pp. 674–681 (2002)
- [López 03] López, y F. L. and Luck, M.: Modeling Norms for Autonomous Agents, in *Proc. of the Fourth Mexican International Conference on Computer Science*, pp. 238–245 (2003)
- [Marsella 02] Marsella, S. and Gratch, J.: A Step Toward Irrationality: Using Emotion to Change Belief, in *Proc. of the First International Conference on Autonomous Agents and Multi-agent Systems*, pp. 334–341 (2002)
- [Mazouzi 02] Mazouzi, H., Fallah-Seghrouchni, A., and Haddad, S.: Open Protocol Design for Complex Interactions in Multi-agent Systems, in *Proc. of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 517–526 (2002)

- [Milgram 69] 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)
- [Murakami 03] Murakami, Y., Ishida, T., Kawasoe, T., and Hishiyama, R.: Scenario Description for Multi-Agent Simulation, in *Proc. of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 369–376 (2003)
- [Murakami 05] Murakami, Y., Sugimoto, Y., and Ishida, T.: Modeling Human Behavior for Virtual Training Systems, in *Proc. of the Twentieth National Conference on Artificial Intelligence*, pp. 127–132 (2005)
- [Nakanishi 99] Nakanishi, H., Nakazawa, S., Ishida, T., Takanashi, K., and Isbister, K.: FreeWalk: A 3D Virtual Space for Casual Meetings, *IEEE Multimedia*, Vol. 6, No. 2, pp. 20–28 (1999)
- [Nakanishi 04] Nakanishi, H.: FreeWalk: A Social Interaction Platform for Group Behaviour in a Virtual Space, *International Journal of Human Computer Studies*, Vol. 60, No. 4, pp. 421–454 (2004)
- [Nakashima 03] Nakashima, H.: Grounding to the Real World - Architecture for Ubiquitous Computing., in Zhong, N., Ras, Z. W., Tsumoto, S., and Suzuki, E. eds., *Foundations of Intelligent Systems*, pp. 7–11 (2003)
- [Odell 00] Odell, J., Parunak, H., and Bauer, B.: Representing Agent Interaction Protocols in UML, in Ciancarini, P. and Wooldridge, M. eds., *Agent-Oriented Software Engineering*, pp. 121–140, Springer-Verlag (2000)
- [Padgham 02] Padgham, L. and Winikoff, M.: Prometheus: A Methodology for Developing Intelligent Agents, in Giunchiglia, F., Odell, J., and Weiß, G. eds., *Agent-Oriented Software Engineering III*, pp. 174–185, Springer-Verlag (2002)

- [Paurobally 04] Paurobally, S., Cunningham, J., and Jennings, N. R.: Developing Agent Interaction Protocols Using Graphical and Logical Methodologies, in Dastani, M., Dix, J., and Fallah-Seghrouchni, A. E. eds., *Programming Multi-Agent Systems, First International Workshop, PROMAS 2003, Melbourne, Australia, July 15, 2003, Selected Revised and Invited Papers*, pp. 149–168, Springer-Verlag (2004)
- [Pelachaud 02] Pelachaud, C., Carofiglio, V., Carolis, B., Rosis, F., and Poggi, I.: Embodied Contextual Agent in Information Delivering Application, in *Proc. of the First International Conference on Autonomous Agents and Multi-agent Systems*, pp. 758–765 (2002)
- [Poole 87] Poole, D.: Theorist: A Logical Reasoning System for Defaults and Diagnosis, in Cercone, N. and McCalla, G. eds., *The Knowledge Frontier*, Springer-Verlag (1987)
- [Poole 94] Poole, D.: Who chooses the assumptions?, in O’Rorke, P. ed., *Abductive Reasoning*, MIT Press (1994)
- [Rickel 99] Rickel, J. and Johnson, W.: Virtual Humans for Team Training in Virtual Reality, in *Proc. of the Ninth International Conference on Artificial Intelligence in Education*, pp. 578–585 (1999)
- [Sandholm 93] Sandholm, T.: An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations, in *Proc. of the Eleventh National Conference on Artificial Intelligence*, pp. 256–262 (1993)
- [Sen 94] Sen, S. and Durfee, E. H.: The Role of Commitment in Cooperative Negotiation, *International Journal of Intelligent and Cooperative Information Systems*, Vol. 3, No. 1, pp. 67–81 (1994)
- [Shoham 92a] Shoham, Y. and Tennenholtz, M.: Emergent Conventions in Multi-Agent Systems: Initial Experimental Results and Observations (Preliminary Report), in *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pp. 225–231 (1992)

- [Shoham 92b] Shoham, Y. and Tennenholtz, M.: On the Synthesis of Useful Social Laws for Artificial Agent Societies (Preliminary Report), in *Proc. of the Tenth National Conference on Artificial Intelligence*, pp. 276–281 (1992)
- [Shoham 95] Shoham, Y. and Tennenholtz, M.: On Social Laws for Artificial Agent Societies: Off-Line Design, *Artificial Intelligence*, Vol. 73, No. 1-2, pp. 231–252 (1995)
- [Shoham 97] Shoham, Y. and Tennenholtz, M.: On the Emergence of Social Conventions: Modeling, Analysis, and Simulations, *Artificial Intelligence*, Vol. 94, No. 1-2, pp. 139–166 (1997)
- [Sugawara 04] Sugawara, T., Kurihara, S., Fukuda, K., Hirotsu, T., Aoyagi, S., and Takada, T.: Reusing Coordination and Negotiation Strategies in Multi-Agent Systems for Ubiquitous Network Environment., in *Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 496–503 (2004)
- [Sugiman 88] 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)
- [Sycara 03] Sycara, K., Paolucci, M., Velsen, M., and Giampapa, J.: The RETSINA MAS Infrastructure, *Autonomous Agents and Multi-Agent Systems*, Vol. 7, No. 1 and 2, pp. 29–48 (2003)
- [Tahara 01] Tahara, Y., Ohsuga, A., and Honiden, S.: Mobile agent security with the IPEditor development tool and the mobile UNITY language., in *Proc. of the Fifth International Conference on Autonomous Agents*, pp. 656–662 (2001)
- [Tambe 97] Tambe, M.: Towards Flexible Teamwork, *Journal of Artificial Intelligence Research*, Vol. 7, pp. 83–124 (1997)

- [Traum 02] Traum, D. and Rickel, J.: Embodied Agents for Multi-party Dialogue in Immersive Virtual Worlds, in *Proc. of the First International Conference on Autonomous Agents and Multi-agent Systems*, pp. 766–773 (2002)
- [Vincent 01] Vincent, R., Horling, B., and Lesser, V.: An Agent Infrastructure to Build and Evaluate Multi-Agent Systems: The Java Agent Framework and Multi-Agent System Simulator, in Wagner, T. and Rana, O. F. eds., *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems.*, Vol. 1887, Springer-Verlag (2001)
- [Walker 95] Walker, A. and Wooldridge, M.: Understanding the Emergence of Conventions in Multi-Agent Systems, in *Proc of the First International Conference on Multiagent Systems*, pp. 384–389 (1995)
- [Wook 00] Wook, M. F. and Deloach, S. A.: An Overview of the Multiagent Systems Engineering Methodology, in Ciancarini, P. and Wooldridge, M. eds., *Agent-Oriented Software Engineering*, pp. 207–221, Springer-Verlag (2000)
- [Yamashita 05] Yamashita, T., Izumi, K., Kurumatani, K., and Nakashima, H.: Smooth traffic flow with a cooperative car navigation system., in *Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 478–485 (2005)
- [Zambonelli 03] Zambonelli, F., Jennings, N., and Wooldridge, M.: Developing Multiagent Systems: The Gaia Methodology, *ACM Transactions on Software Engineering and Methodology*, Vol. 12, No. 3, pp. 317–370 (2003)

Publications

Major Publications

Journals

1. Yohei Murakami, Toru Ishida, Tomoyuki Kawasoe, and Reiko Hishiyama, “Multi-Agent Simulation Based on Interaction Design,” *Transactions of the JSAI*, Vol. 18, No. 5, pp. 278–285, 2003 (in Japanese).
2. Yohei Murakami, Yuki Sugimoto, and Toru Ishida “Constructing Agent Model for Virtual Training Systems,” *Transactions of the JSAI*, Vol. 21, No. 3, pp. 243–250, 2006 (to appear, in Japanese).

International Conference

1. Yohei Murakami, Toru Ishida, Tomoyuki Kawasoe, and Reiko Hishiyama, “Scenario Description for Multi-Agent Simulation,” in *Proc. of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, pp. 369-376, 2003.
2. Yohei Murakami, Yuki Sugimoto, and Toru Ishida, “Modeling Human Behavior for Virtual Training Systems,” in *Proc. of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pp. 127-132, 2005.

Other Publications

Awarded Papers

1. Kazuhisa Minami, Yohei Murakami, Tomoyuki Kawasoe, Toru Ishida, “Evacuation Simulation by using Multi-agent System,” *National Convention of JSAI*, 2002 (in Japanese). Best Paper Award.
2. Toru Ishida, Hideyuki Nakanishi, Yohei Murakami, Tomoyuki Kawasoe, and Reiko Hishiyama, “Multi-agent Simulation for Urban Crisis Management,” *The Third SICE System Integration Division Annual Conference*, 2002 (in Japanese). Best Session Award.

Workshops

1. Yohei Murakami, Kazuhisa Minami, Tomoyuki Kawasoe and Toru Ishida, “Multi-Agent Simulation for Crisis Management,” *IEEE International Workshop on Knowledge Media Networking (KMN-02)*, pp. 135–139, 2002.
2. Sachiyo Arai, Yohei Murakami, Yuki Sugimoto and Toru Ishida, “Semantic Web Service Architecture using Multi-agent Scenario Description,” *Intelligent Agents and Multi-Agent Systems (Pacific Rim International Workshop on Multi-Agents (PRIMA-03))*, Lecture Notes in Artificial Intelligence, Vol. 2891, pp. 98–109, Springer-Verlag, 2003.
3. Yohei Murakami, and Toru Ishida, “Protocol Design Using Multi-Agent Simulation,” *Joint Agent Workshop (JAWS2004)*, pp. 66–73, 2004 (in Japanese).
4. Yohei Murakami, Yuki Sugimoto, and Toru Ishida, “User Modeling by Participatory Simulation,” *Joint Agent Workshop (JAWS2005)*, pp. 85–92, 2005 (in Japanese).

5. Yohei Murakami, and Toru Ishida, “Participatory Simulation for Designing Evacuation Protocols,” *First International Workshop on Agent Technology for Disaster Management (ATDM-06)*, 2006 (to appear).

Conventions

1. Sachiyo Arai, Yohei Murakami, Yuki Sugimoto, and Toru Ishida, “Multi-agent Scenarios introducing Semantics for Web Services,” *The Sixth SANKEN(ISIR, Osaka University) International Symposium on New Trends in Knowledge Processing - Data mining, Semantic Web and Computational Science*, 2003.
2. Yuki Sugimoto, Yohei Murakami, Sachiyo Arai, and Toru Ishida “User-Oriented Web Service Composition with Semantics,” *National Convention of JSAI*, 2003 (in Japanese).
3. Sachiyo Arai, Yohei Murakami, Yuki Sugimoto, Masahiro Tanaka, “Evaluation Credibility of Reputation Using Boosting,” *JSAI SIG-SWO & ONT Notes*, 2003 (in Japanese).
4. Yuki Sugimoto, Yohei Murakami, and Toru Ishida, “Modeling Action Rules through Participatory Simulation in Virtual Space,” *IEICE SIG-ICS Notes*, 2005 (in Japanese).