NAIST-IS-DD0161017

Doctoral Dissertation

Mediation Architecture of Personal Robot Applications Based on a Communications Model

Akihiro Kobayashi

August 25, 2005

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Akihiro Kobayashi

Thesis Committee:
Professor Masatsugu Kidode (Supervisor)

Mediation Architecture of Personal Robot Applications Based on a Communications Model *

Akihiro Kobayashi

Abstract

This paper presents a middleware architecture for personal robots and applies such architecture to various environments. The architecture allows a robot to consistently integrate environment-oriented applications with its original and familiar characteristics for its user. Applications that generate familiar characteristics and environment-oriented applications tend to be developed independently. However, the two kinds of applications should share sensors and actuators to generate consistent actions. To this end, I have analyzed the relationship between robot actions and the mental effects of these actions on the user, and I have designed a middleware for personal robots, called a Personal Robots' Intermediating Mediator for Adaptation (PRIMA). Using videos I created an experiment to demonstrate the effect of mediation on PRIMA's outputs.

Keywords:

Personal Robot, Human-Robot Interaction, Open Robot Architecture

* Doctoral Dissertation, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0161017, August 25, 2005.

Contents

1. Introduction **1**

2. Model Based Mediation Framework **7**
2.1 Integration of Two Different Kinds of Applications 7
2.2 Mediation Framework 7
2.3 Communication Interference 10
2.4 Restrictions of PRIMA 13
2.5 Summary 17

3. Communication Model **19**
3.1 Information Unit 19
3.2 Communication Stream 20
3.3 Communication Channel 23

4. Implementation of Mediation Strategy **27**
4.1 Implementation of the Information Unit 27
4.2 Basic Rules Mediating an Information Unit 30
4.3 Implementation of the Communication Channel 36
4.4 Implementation of the Communication Stream 36
4.5 Summary of Programmers' Restriction 40

5. Experiment **41**
5.1 Evaluation of Mediated Motions Using Video 41
5.2 Experimental Results 48
5.3 Discussion 51

6. Conclusion **59**

Acknowledgements **60**

References **61**

List of Publications **66**

List of Figures

1.1 Personal Robots 1
1.2 An Example of Parallel Execution 2
1.3 Relational Techniques 3
2.1 Internal States of Applications 8
2.2 A Mediation Architecture for Personal Robots 9
2.3 Relations between Developers using PRIMA 14
2.4 Functions of Personal Robots 15
3.1 Communication Stream 21
3.2 Robot's Internal States Expected by A User 22
3.3 Communication Channel 24
4.1 Atomic Behavior Tree 31
4.2 Attributes of an Atomic Behavior Class 31
4.3 Working Memory of PRIMA 32
4.4 Feedback Behavior 33
4.5 Implementation of PRIMA 35
4.6 Exceptions of Communications Stream 38
5.1 Output Example of FA2 43
5.2 Output Example of EA3 44
5.3 Mediated Output (Answer Sheet) 45
5.4 Mediated Output of FA2 and EA4 (Video) 46
5.5 Requests from EA3 47
5.6 Requests from FA2 47
5.7 Output Example of FA5 49
5.8 Output Example of EA2 49
5.9 Mediated Output of FA5 and EA2 50
5.10 Output Example of FA4 53
5.11 Output Example of EA1 54
5.12 Mediated Output of FA4 and EA1 55

List of Tables

4.1 Command Attribute 27
4.2 Examples of Command Types 28
5.1 Application Example 41
5.2 Test Sample 42

1. Introduction

Recent research studies provide several points of view for the use of autonomous mobile robots, for example, a mobile and intelligent interface for information systems [8, 43, 26], or a familiar and amusing robot that acts and looks like a pet. Human-like expression in the pet robot's appearance and motion is a particularly important issue [15, 19]. Lately, a robot for home use has attracted special interest. Such a robot is called a *personal robot* or a *home robot* [1, 5, 18, 44] A personal robot is expected to take on the roles of a pet and an interface of home appliances. A personal robot has a familiar appearance as shown in Figure 1.1. This type of robot also emphasizes the importance of smooth communications to its user, rather than quick or powerful outputs.

(a) PaPeRo (b) ApriAlpha (c) QRIO

Figure 1.1: Personal Robots

In the near future, a personal robot should be an interface between its owner and an information service in the environments an owner visits, e.g., an office building, a museum, a library, and an amusement park. This ability to interface gives benefits to both the owner of the robot and a provider of an information service. From the owner's point of view, accessing the service through his or her

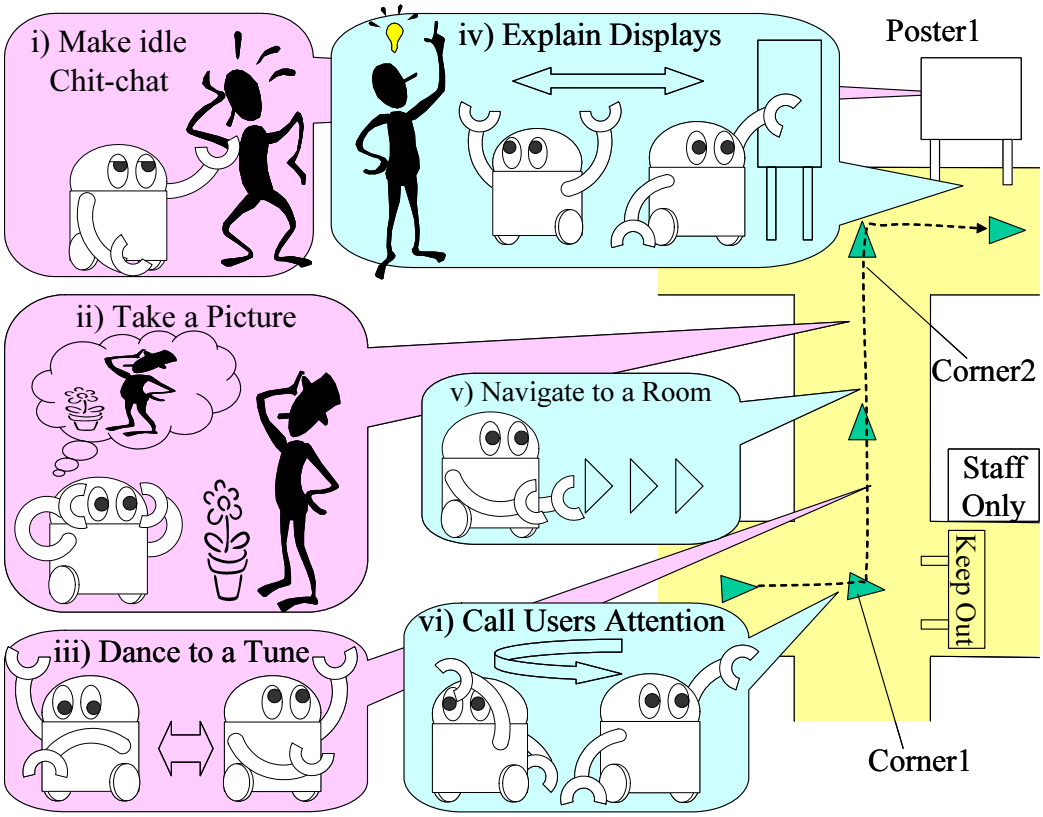

Figure 1.2: An Example of Parallel Execution

familiar robot is desired. A system of the information service can use the personal robot as its physical interface for a visitor, although a traditional service provider needs robots to provide these services. These services are composed of rules which stipulating relationships between inputs and outputs of a robot. In this paper, these rules are called an application. An application for an information service for a visiting environment is called an **Environment-oriented application** (*EA*). This research attempts to make a personal robot dynamically load an *EA* in its visiting environment in order to adapt to the environment [24, 23, 22].

On the other hand, a personal robot has its original rules to represent its personality. These rules are called a **Familiarity-oriented application** (*FA*). An owner feels the robot's personality from its familiar motions and conversations, in its *FA*. A personality of a personal robot might be an important factor in its

commercial value. A personal robot, while maintaining its personality, should accomplish a set of tasks given in a visiting environment. For example, a robot behaves as a pet interacting with the owner when the robot accompanies the owner. Figure 1.2 shows a personal robot accompanying its owner to an exposition. The robot has a *FA* to make idle chit-chat, to take a picture, and to dance to a tune. The robot loads an EA to explain displays, to navigate a room, and to call the users' attention to a corner. I attempt to integrate *FA* and *EA* , and attempt to make the robot take a picture or express emotions while guiding its owner.

In order for a robot to supply usages like above, I suggest, in the first chapter of this thesis, a middleware called a Personal Robots' Intermediating Mediator for Adaptation (PRIMA), which integrates a *FA* and an *EA*. Chapter 2 introduces concepts and problems of PRIMA. Figure 1.3 shows related works of the proposed concept.

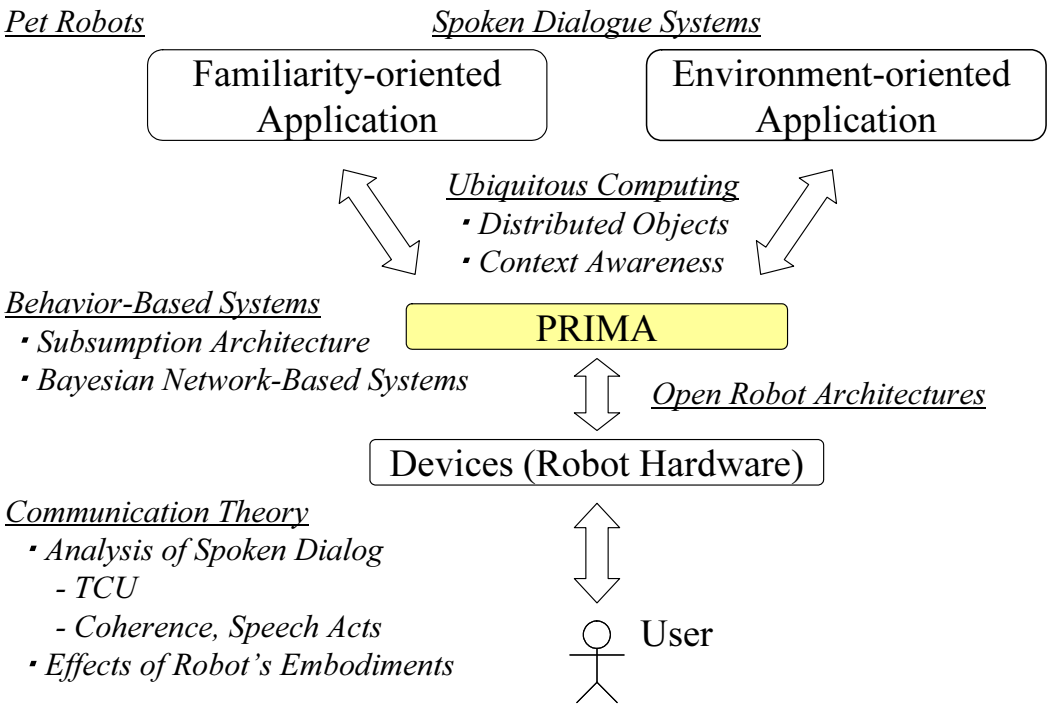

Figure 1.3: Relational Techniques

Section 2.1 introduces the *behavior-based architecture* often used by an autonomous mobile-robot, which has highly independent internal structures. PRIMA attempts to run an *EA* and *FA* , like as behavior-based architectures behave. In this case, an application interrupts communications between the user and other applications. In order to solve these interferences, this paper introduces a middleware which mediates the output of *EA* and *FA*.

The contents of an *EA* and a *FA* are rules of interactions like scenario documents in a spoken dialogue system. Section 2.2 describes why a model of behavior-based architectures is adopted, and compares this behavior-based architecture with *mixed-initiative spoken dialogue systems*. Integrating an *EA* and a *FA* into a document for an interpreter of generic spoken dialogue systems is difficult, because an *EA* and a *FA* are independently developed. Therefore, the middleware runs an *EA* and a *FA* in parallel and mediates between them, like as an operating system which schedules individual applications.

Unlike operating systems, interferences between an *EA* and a *FA* breaks continuity and response speeds, which are essential for human-robot communications. Section 2.3 explains the relationship between these problems and communication theories, by providing an *analysis of spoken dialogues* and *human-robot non-verbal communication*.

Section 2.4 clearly reveals problems that occur with PRIMA, when compared to *ubiquitous computing* and *open robot architectures*. For programmers, it is difficult to describe a behavior and its timing, because of a variety of situations, not least of which includes problems surrounding ubiquitous computing.In the case of PRIMA, an application's programmer cannot have the responsibility for the total results of a robot's output from a user's point of view.

Section 2.4 discusses how to divide the responsibility of middleware and that of a programmer. This paper refers to open robot architectures that try to solve problems for a variety of hardware. I focus on problems about communications as the most important problems, and have designed communication models for a user using two applications on a robot from communication theories.

Chapter 3 explains the basic ideas about these communication models. The communication models adopt three types of constructive communication units: the Information Unit, the Communication Stream, and the Communication Chan-

nel, based on communication theories about the analysis of spoken dialogues and human-robot non-verbal communications. Chapter 5 conducted an experiment to demonstrate the effect of mediation, by videos of PRIMA's outputs.

Chapter 4 discusses the implementation of the middleware, which has rules for mediation and internal expression based on the three communication models proposed in the previous Chapter. Chapter 4 also discusses restrictions on developing an application that complies with these models. Chapter 5 conducted an experiment to demonstrate the effect of mediation, by using videos of PRIMA's outputs. Finally, Chapter 6 concludes this work and provides suggestions for future research.

2. Model Based Mediation Framework

2.1 Integration of Two Different Kinds of Applications

A new mediation framework is required for a robot to execute multiple applications in parallel, and these applications are developed independently. Behavior based approaches [7, 13, 16] allow a robot to execute multi-behaviors in parallel. The *Subsumption Architecture*[7] model represents a process where parallel data flows from sensor inputs to actuator outputs. This model introduces layers of system modules according to the various levels of goals. Interventions from upper layers to lower layers are allowed. Designing under *Subsumption Architecture*, a programmer must understand the lower layers in detail before designing the upper layers.

In order to integrate multiple behaviors, *Situated Multi-Agent Architecture* [13] and *PEXIS* [16] adopt bayesian networks to express relations among modules running in parallel. These systems learn the relations which are nevertheless fixed under *Subsumption Architecture*. For robots to behave appropriately, these systems require learning time and predefined links between modules which are related to each other. PRIMA, compared with these architectures, aims at an instant use of a new *EA* loaded in its visiting environment without interference from a *FA*. Therefore, this thesis suggests a middleware with new models independent of applications logic and meaning, and which integrates *EA* and *FA* based on these new models.

2.2 Mediation Framework

In our system, contents of an *EA* and a *FA* are rules of interactions like documents of a spoken dialogue system such as, for example, VoiceXML [28], and XISL [21]. PRIMA should make appropriate output from these rules. Most spoken dialogue systems manage statements based on a FSM (Finite State Machine) as shown in Figure 2.1. In Figure 2.1, a node shows the state of a robot, and each node decides the output of a robot, for example, what a robot speaks. A links shows the next state when a robot gets input from environments such as a user's utterance. Handling of unexpected inputs is one of the most important problems for spoken

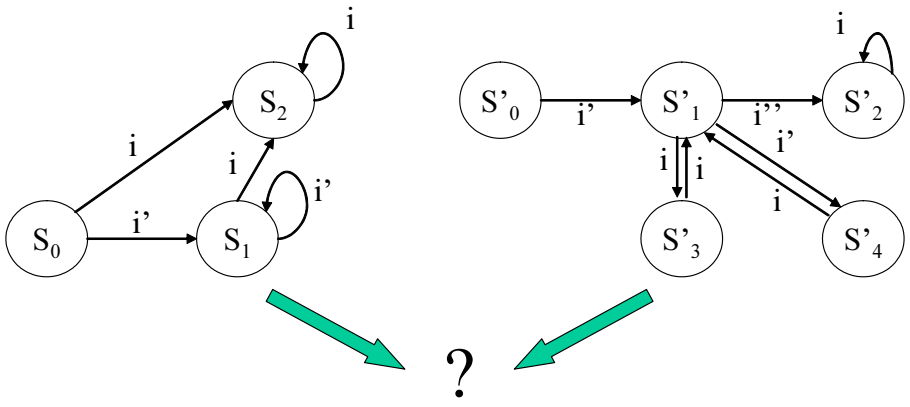

Figure 2.1: Internal States of Applications

dialogue systems.

Static and dynamic methods are used to integrate an *EA* and a *FA*. In the former methods, a robot integrates an *EA* and a *FA* into a consistent rule-set before running them. Static integration has the advantage of integrating behaviors predicting future behaviors. However, integrating an EA and a *FA* into a FSM which a generic spoken dialogue system can interpret, is difficult. The static methods require links between states of both the applications' FSMs as shown in Figure 2.1. There are three approaches to getting these links.

- Existing systems are given these links as predefined information or are given from a corpus in which a man takes both roles of both applications. For example, *caseframe* [33] reduces the costs involved in describing the structure of documents using mechanical learning from dialogue corpora. Okamoto improves a learning algorism for a Probabilistic Deterministic Finite-state Automaton, in order to learn a FSM for controlling a dialogue from corpora given from a wizard of oz method [32]. However PRIMA does not have these corpora and predefined links.

- A state explosion will occur, if PRIMA links a state to all states having the possibility to change from the former state.

- If PRIMA links only between specific states of an *EA* and a *FA*, PRIMA restricts situations changing a state into that of another application. The restriction reduces familiarity of the robot. For example, a robot links only between root documents of *EA* and *FA* in the case of VoiceXML. In this case, when the user of a personal robot want to watch the robot's *FA*'s reaction against touching the robot during executing *EA*, the user must input a signal to change the *EA*/*FA* before touching the robot. [1] This system also has a problem similar to the "Go Back" implementation in VoiceXML [4].

PRIMA, therefore, is designed based on behavior based architectures, which are composed of highly independent modules. In PRIMA, each application is an agent which runs as a process or thread, and has free-hands to express internal state. Furthermore, PRIMA schedules each access from applications to devices, just as a multi-task OS does.

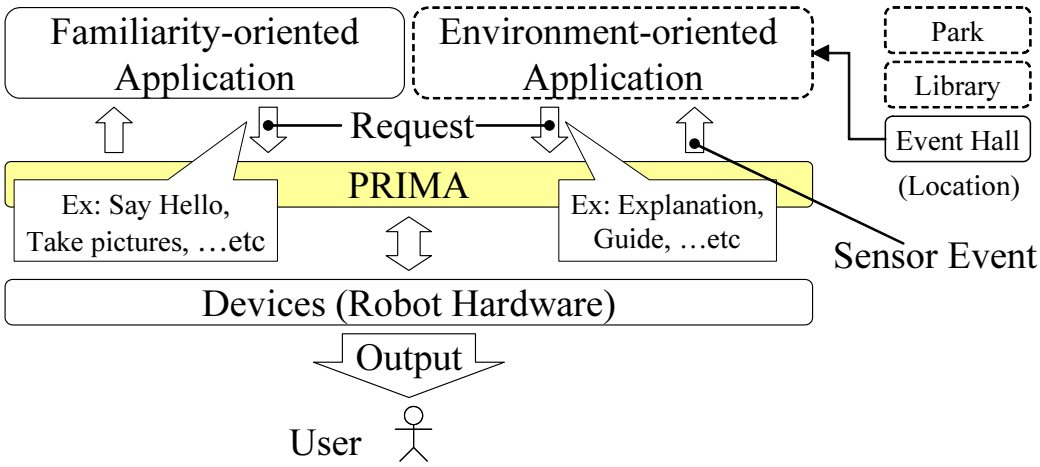

Figure 2.2: A Mediation Architecture for Personal Robots

Figure 2.2 shows the relations between PRIMA and applications. Each agent can access sensors to observe environments any time they like. In addition PRIMA sends the events and values of sensors to an application, which sets

[1] The signal to change applications may have a global scope in all documents

a condition to the event. *EA* and *FA* send requests about accessing each device to PRIMA. PRIMA dynamically schedules these requests and solves interferences in each device without applications' internal states, in order to make a robot show consistent motion to its owner.

2.3 Communication Interference

PRIMA focuses on designing a human-robot communication model for mediation. Usual operating systems can encapsulate interferences from users because these systems deal with interference among abstract resources. On the other hand, interference on PRIMA appears to a user as robot motions. Interference occurs and makes a robot unnatural, when an *EA* and a *FA* simultaneously access a single device. For example, when an *EA* requests to move to a goal away from a user, the *EA*'s request interferes with *FA*'s approaching the user in response to the user's touch. In such a case, the robot cannot reply quickly to the user because of interference. Interference also occurs when an application accesses a device which another application needs to remain motionless. For example, when a *FA* requests the robot to speak something, the request sometimes interferes with an *EA*'s request, e.g., speech recognition. PRIMA gives importance to continuity and response speed, which such interference reduces, because continuity and response speed are the essentials for natural communications between human and robot.

Continuity : These interferences cause interruption of the application's output although human-robot communication needs consistency in its sequence of interactions. PRIMA considers that human-robot communications requires continuity in three points, as minutely explained in the last half of this section.

i. Locally, an interruption in the robot's behavior causes recognition errors by the robot, and misleads the user.

ii. In multi-turns of interaction, an interruption breaks contexts of utterance and break relationships for embodied communications between a user and a robot.

iii. A Robot's body movements and direction of view lines often take important roles in starting and terminating human-robot communications. These motions of a robot often terminate not only a running application, but also an interrupted application. Therefore, a robot should take care of both applications, taking these motions into consideration.

Response Speed : A personal robot should respond to its user's inputs and environmental changes as quickly as possible. Improvement of response speed is one of the most important problem for an autonomous mobile robot. Subsumption architecture adopts a parallel layered architecture, to improve its response speed [7]. In particular, a personal robot needs response speed to realize cooperative embodied communications with its user. For example, QRIO dances with the user using an *entrainment ensemble model* (EEM) [39], and AIBO is trained like a dog using neural networks [42]. Nevertheless, a personal robot may delay its reactions to its user when such a *FA* and an *EA* interfere with each other.

There is a trade-off between continuity and response speed. If an extreme continuity of one application is kept in PRIMA, then the response speed of the other application will be reduced. This work attempts to design and to implement strategies mediating requests from an *EA* and a *FA*, so that PRIMA can balance continuity and response speed. In order to design these strategies, PRIMA assumes three communication models based on existing communication theories. The first and second models are based on studies about spoken dialogues. The third model is based on studies about non-verbal human-robot communications.

i. PRIMA's model includes constructive units and scheduling strategies for communications to analyze the necessity of continuity. A basic unit of talk, called a "Turn Constructional Unit (TCU)", is suggested in research of conversation analysis [38]. TCU reveals the timings of turn-taking for listeners. Intonation, clause, a slash, and inter-pause have relation to TCU, but there is no clear definition of a TCU [9]. PRIMA defines a TCU-like unit and assume that it has a great necessity of continuity.

ii. PRIMA focuses on relations of utterances. In general, a spoken dialog or the text of one person has coherence. Coherence for a human is affected by cohesion [12], such in referential expression. In PRIMA, cohesion between utterances may be broken. For example, when a robot is explaining a poster to an audience, and the robot says the following two sentences, interruption between the two utterances may confuse a user about what "it" refers to.

   (a) Robot: "Please look at a red line on this graph."

   (b) Robot: "It shows a case of success."

   Coherence is also affected by relational meanings between sentences. Internal structure [11], coherence relation [14], and rhetorical relation [25] are suggested to contribute to relational meaning. For example, when a robot says the following two sentences, continuity between these sentences is needed, so that a user can understand the latter sentence.

   (a) Robot: "I have just got a message from Mr. Kobayashi."

   (b) Robot: "There are some questions about your appointment."

   On the other hand, the output of a robot has relations with before-and-after input from a user. An utterance often includes illocutionary and perlocutionary acts [6], which bring a new utterance or new actions of an audience as follows. Studies of analyzing spoken dialogues often categorize an utterance into speech acts, e.g., "request", "propose", and "promise" [3]. These relationships need continuity between utterances of a robot and a user.

   (a) User: "Turn off the TV."

   (b) Robot: "Yes, sir." (The robot turns off the TV.)

iii. PRIMA should consider not only spoken dialogs, but also other modalities. Recent research reveals that a robot's gesture and a body movement and a direction of view lines are important roles for these relationships. Many robots try to apply these gestures in order to communicate with the user

effectively; for example ASKA [31], ROBITA [27], Robovie [35], and GestureMan [36]. Robovie uses its motions to help a user understand space by considering relationships between humans and robots [35]. ROBITA uses a robot's direction for turn-taking of multi-speakers [27]. GestureMan uses its position and direction for a part of the communication media for remote communication [36]. Article [30] discusses the effect of dancing robot gestures. PRIMA also focuses its attention on position and the viewing direction of a robot, and aims for continuity.

A goal of this work is that the communication models and their implements fill the following three requirements:

- PRIMA guarantees continuity and response speed to a user.
- PRIMA applies various personal robots.
- A Programmer can easily describe a *FA* or an *EA*.

2.4 Restrictions of PRIMA

PRIMA makes the following four assumptions on hardware and software. The communicationmodels proposed in Section 3 are designed based on these assumptions.

Developers using PRIMA :

Figure 2.3 shows the relations between 3 kinds of developers using PRIMA, i.e., the *EA Developer*, the *FA Developer* and the *Hardware Developer*. In this paper, a "user" represents a person who is the owner of a personal robot, and uses hardware and software, but doesn't engage in implementations such as programming. PRIMA satisfies common interfaces which access hardware for *EA Developers* who don't have a detailed knowledge of the hardware.

FA Developers also use common interfaces. Different from *EA Developers*, *FA Developers* can share information about hardware with the *Hardware*

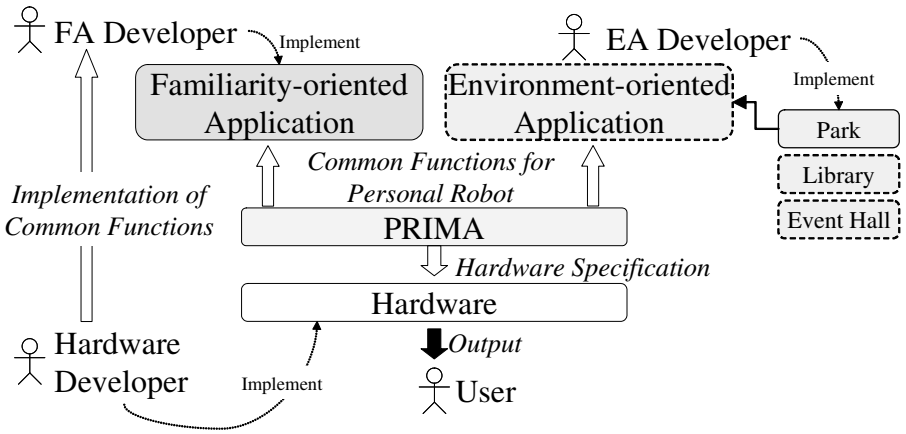

Figure 2.3: Relations between Developers using PRIMA

Developers, because the hardware of a personal robot and its original interactions are always designed by the same manufacturer. If *FA* uses privileged access to hardware without PRIMA, the output sequence including *EA* might be unnatural for the user. Therefore, PRIMA assumes that *FA* accesses hardware through PRIMA. However, common interfaces do not satisfy all the requirements of the original interactions in *FA*. PRIMA supports *Hardware Developers* who add options of common interfaces, such as velocity or acceleration of a common interface, i.e., "move 10 cm".

Hardware Specification :

One of the targets of PRIMA is a robot, which fulfills following conditions.

- A robot has a device which implies a head.
- A robot has functions of locomotion, conversation, human-sensing, and localization.

PRIMA assumes these hardware specifications in order for PRIMA to supply a common interface of middleware to an *EA*, which runs on various personal robots. The interface for digital home appliances and the entertainment of embodied interactions are mainly the tasks of personal robots

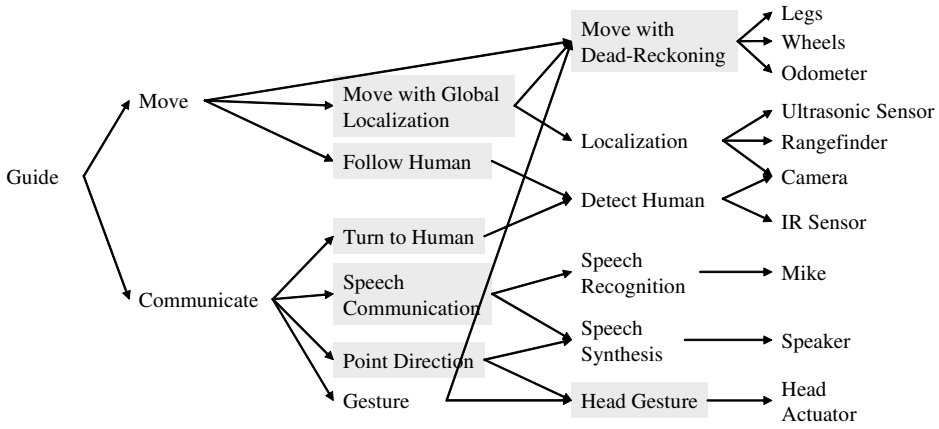

Figure 2.4: Functions of Personal Robots

[1, 5, 18, 44]. The functions of locomotion and conversation, are necessary for a personal robot which communicates with a user.

In this thesis, I normalize the functions and devices for personal robots as shown in Figure 2.4. I use a guide of a building as an example of *EA*s, and classify functions for the application. Figure 2.4 shows classified functions and the devices required by function. I define minimum units which developers can use without thorough knowledge of hardware, and define these minimum units as common interfaces of PRIMA. In order to satisfy requirements from these interfaces, I define hardware specifications. *Hardware Developers* implement these hardware specifications. Therefore PRIMA can guarantee the output of common interfaces to *EA* and *FA Developers*. For example, a Hardware Developer has a free hand to implement the common interface "move 10 cm" using legs or wheels.

These common interfaces are desined as a common protocol to communicate with devices, in order to support various devices. Open-robot-architectures that aim to unify these protocols have been focused on recent years. These architectures attempt to run a programs written by these common protocols on various robots. ORiN aims at a common protocol of industrial robots [29]. OpenHRP aims at humanoids [20], and OPEN-R at pet robots

[2]. ORCA attempts to communicate with home appliances [37]. Because PRIMA behaves as a wrapper for the functions defined in these architectures, application interfaces of PRIMA should consider these architectures. For example, PRIMA does not have an API which sets the velocity of a specific motor depending on robot's devices.

Loading Applications :

PRIMA assumes that each environment supplies only one *EA*, and PRIMA is responsible for *EA*'s consistency. PRIMA interchanges an *EA* in each environment, and simultaneously executes up to two applications, an *EA* and a *FA*. PRIMA makes these assumptions, because PRIMA focuses on the case where a developer cannot image any other applications.

Heterogeneity is one of the important issues of context-aware application in ubiquitous computing. In order for a programmer to easily define context deciding contents and timings, some describing methods of meta-contexts have been developed [10]. However, these methods assume that a programmer has responsibility to never interfere between outputs from a context. In PRIMA, both programmers of *EA* and *FA* can have this responsibility only about themselves. PRIMA assumes that a programmer of *EA* can have this responsibility in an area like an office-building. In addition, PRIMA assumes that a user lets his/her robot load an *EA*. The user gets instruction from the *EA* when he/she is at the entrance of a building, in order to simplify the problem of recognizing an area, and selecting an *EA*.

On uBlocks which is a middleware of ubiquitous computing, a user has this responsibility [17]. The system enables users to combine distributed objects flexibly on their own, because uBlocks is implemented based on an Independent Modeling Topology. The system is effective in a user's home, because a user knows the situations in which he or she will want services. However, in a new environment, a user may prefer entrusting mediation rather than combining unknown services and the user's robots.

Programming Styles of Applications :

An application does not require real-time control for long-term motion. In other words, 1) an application requires abstract behaviors such as position control, and allows the middleware to make a trajectory without restraint, or 2) an application requires a short term motion which needs strict reproducibility, and allows the middleware to decide the timing when starting the motion.

2.5 Summary

This thesis suggests that a middleware for dynamic mediation based on communication models is needed for flexible use of *EA* and *FA*. The communication models should balance continuity and response speed against a user. Response speed is required for a user's input and change of environments. Continuity is required for verbal and non-verbal communication. The communication models described in Chapter 3 assume that a personal robot has the basic functions of conversation and locomotion. Furthermore, each environment supplies only one *EA*, and both applications allow PRIMA to mediate these outputs' timing. Chapter 3 shows the details of the models.

3. Communication Model

PRIMA assumes that human-robot communication requires continuity of short-term motions, topics, and a start and end of communication. PRIMA defines three constructive units for communication, each with a different grain size. These units are called (in Section 3.1) an **Information Unit** (*IU*), (in Section 3.2) a **Communication Stream** (*CS*), and (in Section 3.3) a **Communication Channel** (*CC*). They have relationships as shown in Figure 3.3. These communication models allow PRIMA to mediate *EA* and *FA* without knowledge of applications, and don't differ *EA* from *FA*. Therefore, these communication model are based on analysis of generalized communications between human and robot. This chapter discusses the trade-off between response speed and each unit, and suggests a mediation strategy.

3.1 Information Unit

PRIMA defines primitive continuous sequences, as an **Information Unit** (*IU*). The size of *IU* should be sufficient for a robot or a user to understand the *IU*. For example, a turn of utterances, and a sequence of gestures are information units. If a robot recreates these motions discontinuously, an owner cannot understand them. A couple of question and answer pairs is also an *IU*. If a robot's outputs, such as in playing music or dancing, interrupts listing the user's utterance of other application, the recognition accuracy will be reduced.

It is difficult for PRIMA to divide an *IU* from outputs of a robot. For example, an utterance or a TCU doesn't have a clear definition in the research of analyzing spoken dialogues [9]. Therefore PRIMA uses programmer's definitions. Basically, PRIMA thinks a request from application is an *IU*. Such a semantic definition may be useful for a user, and easy for a programmer to describe.

PRIMA assumes that an *IU* has a strict restriction on its continuity. Therefore, PRIMA nevertheless applies a strategy to guarantee continuous execution of an *IU*, if response speed may be reduced. Section 4.1 explains how to decide the size of an *IU*, and how to guarantee its continuity, as well as the restrictions on *IU* for application programmers.

3.2 Communication Stream

In a usual robot's application, output of robot has coherence with before-and-after the robot's output, such as the discourse of human-beings as described in Section 2.3. If a robot doesn't keep continuity among utterances, relations break, and the user feels that the robot lacks coherence. In an extreme case, a user is confused by a robot's output if PRIMA alternates an *EA*'s *IU*, and a *FA*'s *IU* one by one, just as in the fair scheduling of a CPU. PRIMA defines a block of these semantic relations included in the *IU*s of an application as a **Communication Stream** (*CS*), and aims to keep continuity of a *CS* as usual.

PRIMA adopts a strategy giving more importance to response speed rather than continuity of a *CS*. PRIMA assumes that response speed is needed by reaction to a user's input and an environmental change. When a user's inputs are supported by an application which produces present streaming *CS*, a robot should replay an output of the application. In this case, both continuity and response speed are kept. However, when a user's inputs are unexpected by a running application, and are expected by the other, a robot should replay the latter. In this case, a *CS* is broken by the user's input or an environmental change, such as in a speech act. PRIMA assumes that a robot can keep coherence because the user can infer the cause of a reaction. Figure 3.1 shows internal states of PRIMA when PRIMA mediates requests of applications. Mediating *CS* is composed of three operations handling *IU*'s.

Interruption :

At $t = T_0$, PRIMA runs Stream0 of App0. At $t = T_1$, App1 creates Stream1 which is a reaction of the user's input. PRIMA waits starting Stream1, until a robot accomplishes an *IU* which was running when Stream1 was requested. At $t = T_1'$, PRIMA suspends Stream0, and starts Stream1, because PRIMA gives a preference over response speed.

Keeping Continuity : PRIMA accomplishes Stream1 at $t = T_2$ to keep the continuity of Stream1. In order to detect actual breakpoints of contexts in a robot's behaviors, PRIMA needs to understand the syntax and semantics of a robot's behaviors, or needs explicit application definitions about

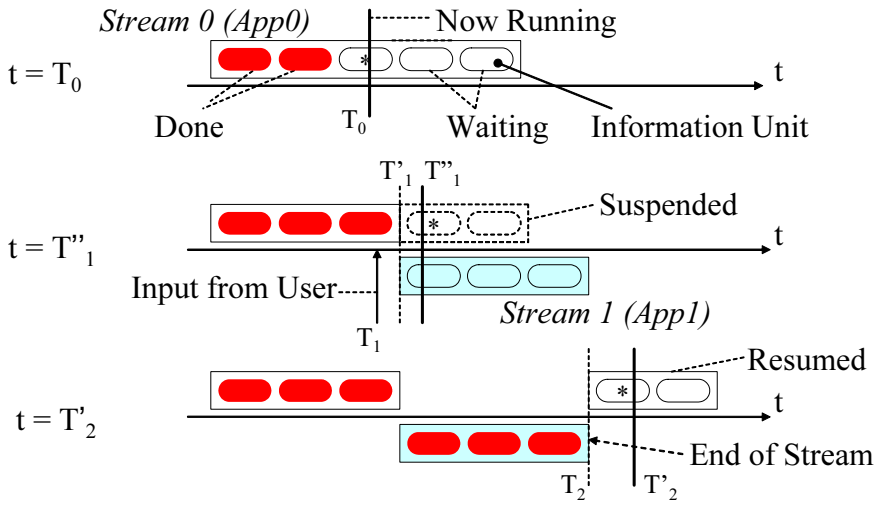

Figure 3.1: Communication Stream

the start and the end of a *CS*. In recent technologies, inferring these contexts from a robot's and a user's utterances is difficult without annotations about illocutionary and perlocutionary acts by an application programmer. Therefore, PRIMA aims at detecting an approximate end-point of a *CS* , even if a programmer need not worry about a *CS*.

PRIMA focuses on an elapsed time between an application's requests as the basic breakpoint of an application. In particular, PRIMA assumes that an application terminates a *CS* when requests of the application run out. In addition, PRIMA assumes that any of the relations described in Section 2.3 among *IU*'s in a CS of this definition exist. If PRIMA keeps a CS between application requests, which have a time gap, PRIMA should wait for a future request of the application, by suspending the other applications. In this case, the robot greatly lacks response speed.

This assumption of terminating the *CS* applies to many cases. However, there are some exceptions that need the continuity of CS, which the assumption doesn't cover. For example, an application cannot request a new behavior, while the application is waiting for a user's input, or waiting for

the end of output in order to observe environments at the time. PRIMA regards these cases as exceptions of the terminating rule. Notably, PRIMA supports methods for an application to express illocutionary acts bringing a user's reply, and detects the application's needs of continuity.

Resume :

At the end of Stream1, PRIMA resumes Stream0 from the suspended *IU* marked with an asterisk because PRIMA assumes the following user-model: A user may think that his/her robot has multiple tasks, and stacks a state of interrupt topics, as shown in Figure 3.2 (a). Some interactive systems, like as GALAXY [40] and Caseframe [33] can deal with multiple tasks by managing a history of dialogues. These dialogue-control methods have the advantage of decreasing turns to accomplish multiple tasks.

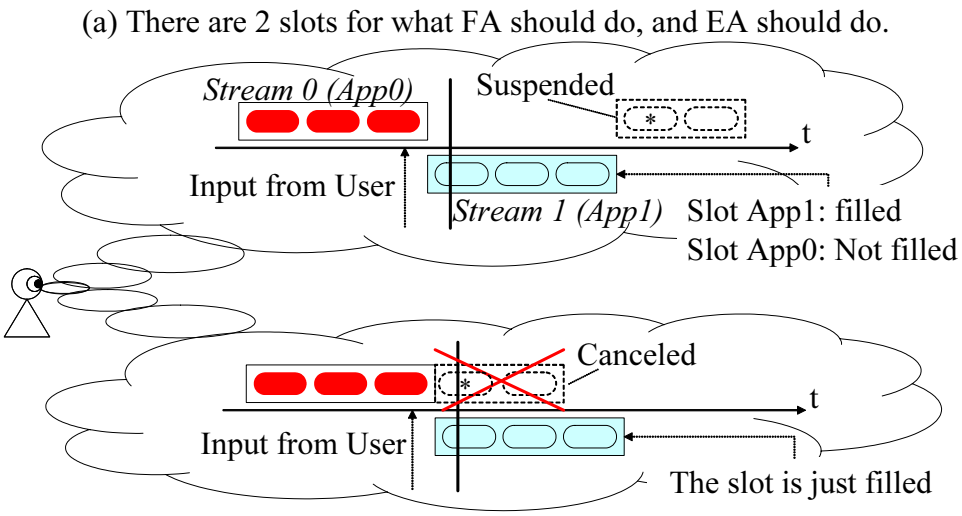

Figure 3.2: Robot's Internal States Expected by A User

Contrary to the above case 3.2 (a), 3.2 (b) shows how a user thinks a robot gives up an interrupted *CS*. In this assumption, the robot should either recover nothing or the whole *CS*. However, PRIMA does not adopt this assumption for the following reasons:

- A user knows and accepts running an *EA* loaded from each environment, as described in Section 2.4.
- An *EA* and a *FA* cannot completely unify the way of speaking and other interactions.

In consequence, a user thinks his/her robot has multi-tasks at one time as shown in Figure 3.2 (a). This strategy is effective for application programmers, because they don't have to describe all cases of interruption and recovering.

Section 4.4 shows the methods to decide the end of *CS*, as well as an *IU*, which needs response speed, and explains restrictions for application programmers.

3.3 Communication Channel

If a robot follows the strategy described in Section 3.2, the robot gives priority to a running a *CS* and an interruption caused by the user's input and environmental change. However, there are two patterns, PRIMA should not adopt the model of *CS*, because *IU*s and *CS*s may need different continuity and response speed, depending on their importance for a user.

- A robot should not interrupt Stream0 at $t = T_1$ in Figure 3.1, if Stream0 is much more important for a user than Stream1. For example, alarms should not interrupt important conversations.
- A robot should not continue Stream1 at $T_1 < t < T_2$ while suspending Stream0, if suspended Stream0 is much more important for a user than Stream1.

PRIMA attempts to define a priority to estimate important aspects of an *IU*. PRIMA focuses on a pose of the robot which shows the end of communications, so that PRIMA can adopt to a case of latter patterns. PRIMA may recognize easily these characteristics which apply to many users. PRIMA assumes that a personal robot should not move away from its user, thus suspending a *CS* which has not accomplished a task among *CS*'s interactions.

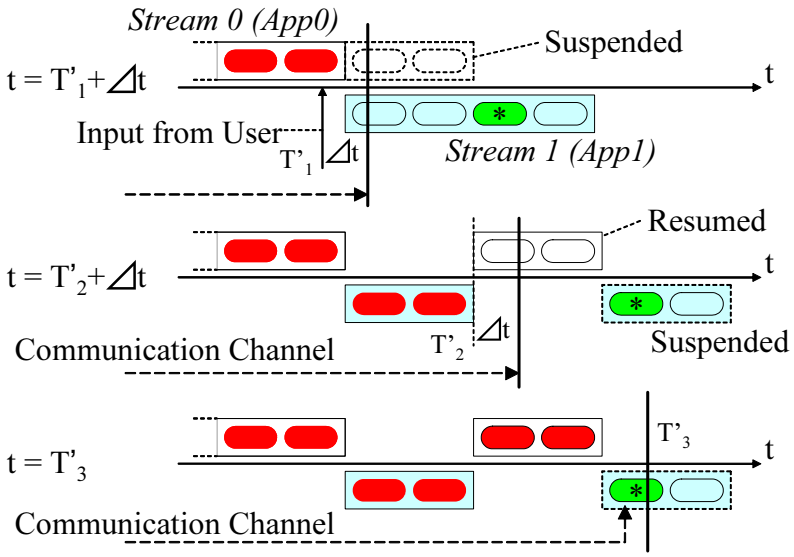

Figure 3.3: Communication Channel

Section 3.3 introduces a **Communication Channel** (*CC*) between a personal robot and its user to explain the above cases. In general, a user communicating with a robot, worries whether the robot gives its attention to the user. In resent research, communication robots express their attention using their gaze-direction and position, as described in Section 2.3. PRIMA defines that *CC* as established when a user decides that his/her robot can interact with him/her. PRIMA assumes that establishment of *CC* is related to the robot's position and pose, and defines a *CC* as follows:

- Face to face establishes *CC*.
- Leaving disconnects *CC*.
- Temporal distraction because of motions such as gestures does not disconnect *CC*.

If a user follows the user-model as described in Section 3.2, the user believes that his/her robot has multiple tasks. If the robot breaks the *CC* before achieving the goals of all the conversation tasks, the user may not feel that the robot is

coherent. PRIMA assumes that *IU*s disconnecting a *CC* are limited to specific predefined *IU*s. PRIMA gives these *IU*s less priority, in order to keep the *CC*. In particular, Figure 3.3 shows the case where Stream1 caused by a user's input interrupts Stream0, which maintains conversation. In this case, if *IU* (*) in Stream1 makes a robot go away from its user, *CC* will be disconnected before finishing Stream0. This leaving motion gives the impression that the robot breaks off its communication with its user. In order to keep the coherence of a robot, PRIMA mediates as in Figure 3.3.

Section 4.3 explains the classification of *IU*s, and restrictions of application programmers.

4. Implementation of Mediation Strategy

I implement the models described in Chapter 3 using an internal expression called an **Atomic Behavior** (*AB*), which is an implementation of *IU*. In principle, a block size of an *AB* complies with the definition of *IU* described in Section 3.1. PRIMA never simultaneously executes an *AB* which interferes with other *AB*s, in order to guarantee continuity of an *IU*. In addition, PRIMA schedules *AB*s based on a priority of each *AB*, in order to keep *CC*, as described in Section 3.3. If *AB*s have the same priority, PRIMA schedules the *AB*s based on the strategy as described in Section 3.2, so that a robot can balance *CS* and response speed.

4.1 Implementation of the Information Unit

On PRIMA, *FA* and *EA* call API's in libraries named **Command**, which implement concrete hardware actions. An application send an object which has attributes shown in Table 4.1 to PRIMA. A Command type shows function of a Command. Table 4.2 shows examples of command types. Motion arguments are parameters of motions required by each command type. For example, velocity or distance is assigned in motion arguments. Only *FA* uses additional arguments as optional arguments of commands because *FA Developer* can share information with *Hardware Developer*, and can understand implementation about each type of commands. Therefor, *FA* set parameters using additional arguments, in order to bring capabilities of hardwares. Before node shows the order of execution, and is explained in Section 4.2.

Table 4.1: Command Attribute

Command
•Command Type
•Motion Arguments
•Before Node
•Additional Arguments

Table 4.2: Examples of Command Types

	Command Type	FB	Occupying Device
	Active Sensing Human	○	Human Detection, Locomotion, Localization
	Face Human		Human Detection, Locomotion
	Q&A		Speaker, Speech Recognition
	Tracking Human	○	Human Detection, Locomotion
	Face Human and Speak		Human Detection, Locomotion, Speaker
(i)	Turn and Speak		Locomotion, Localization, Speaker
	Set Direction		Locomotion, Localization
	Turn		Locomotion
	Set Speed and Time		Locomotion
	Stop Moving		Locomotion
	Speak		Speaker
	Move Head (Pan, Tilt)		Neck
	Point by Hand		Arm
	Gesture		Arm Neck
	Topological Move	○	Locomotion, Localization
(ii)	Set Position	○	Locomotion, Localization
	Set Position and Direction	○	Locomotion, Localization

FB: Feed-backs

When I design Commands, we enumerate behaviors required for communication by a personal robot which complies with the specifications described in Section 2.4. PRIMA supports abstract Commands, and supports simple Commands which are the minimum unit of behaviors such as set locomotion speed and time, independent from hardware. PRIMA also supports abstract Commands which implement behaviors such as moving a goal position written by topological expression.

In principle, a Command brings an *AB*. We designed an interface of Commands, so that the size of an *AB* can complies with the definition of an *IU* , as described in Section 3.1. PRIMA guarantees continuities of any *AB*. In consequence, PRIMA guarantees that most Commands requested from application are never interrupted.

As described in Section 3.1, the size of an *AB* depends on arguments from Commands given by application programmers. Therefore, PRIMA recommends

that an application programmer should provide these arguments, which are not too big and have a meaningful size, such as in an utterance. For example, an application should be able to divide a long poster explanation into some *IU*s. If an application programmer describes a request of PRIMA that is divided into meaningful units, legibility of the application increases. This recommendation is practical, and most *AB*s comply with the definition of an *IU*. However, there are two exceptions:

(1) **A Command includes multiple *IU*s.**

PRIMA supports Commands with enhanced functions, such as position control, so that an EA can adapt to various robots. On the other hand, the size of an AB should be as small as the *AB*, so that the *AB* can comply with the definition of an *IU* because the size strongly affects response speed. PRIMA assumes that a Command, which has a feedback loop, applies to the case of (1). When a robot resumes an interrupted Command, which uses feedback from environments at the time of resuming, a user can understand the divided motions, and these motions are not against the definition of an *IU*. Therefore, PRIMA defines each loop in a Command with the same feedback as an *AB*, except for device-level feedbacks such as controlling servo. If an application uses a Command setting a velocity, PRIMA entrusts controlling velocities to device drivers, and creates only an *AB*.

These Commands are marked with a circle in Table 4.2. For example, the Commands include tracking a human and setting a position on a map of each environment. In conclusion, PRIMA has flexibility in the trajectories that move toward their goals regarding these Commands, and does not guarantee continuity of these Commands to application programmers. In addition to Table 4.2, PRIMA will support new Commands applying this case because feedback loops are often used to program robots.

(2) **An *IU* includes multiple Commands.**

Complex gestures and emotional dances are application-specific sequences which need continuity for an owner to understand them. Nevertheless, it is impossible for a middleware to support all the Commands of complex

gestures which the applications need. A programmer of an application implements these complex gestures combining simple Commands. In this case, PRIMA should consider multiple Commands as an *IU*. Therefore, PRIMA allows applications to group Commands into an *AB*. PRIMA supports two methods of grouping Commands.

- An application defines a list of multiple Commands, and sends the list to PRIMA.
- An application defines a block of an *AB*, requesting the start and end of it.

A programmer should consider effects on appearance of a robot's output, because an *AB* strongly restricts response speed. PRIMA recommends the former definition. In the case of latter definition, PRIMA guarantees concluding an *AB* started. Programming an application without *AB*'s conclusions requires a technology, such as garbage collection of JAVA which does not perfectly guarantee performance.

4.2 Basic Rules Mediating an Information Unit

(1) Atomic Behavior Tree

An *AB* is an implementation of an *IU*, as described in Section4.1. Sections 4.2-4.4 assume that an *AB* fills definition of an *IU*. This section explains basic rule for mediating *AB*s and interactions between an application and PRIMA. Controlling exclusive execution by keeping sequences included in robot behaviors is easy if the expressing relations among *AB*s are made into a tree, called as *AB* tree. Figure 4.1 shows *AB*s divided from the behavior of a robot as it guides its owner through a museum and explains certain museum displays as shown in Figure 1.2. Each node represents an *AB*, and each link represents a running sequence. Each node runs from a root to the leaves of a tree, and all nodes of a branch run in parallel.

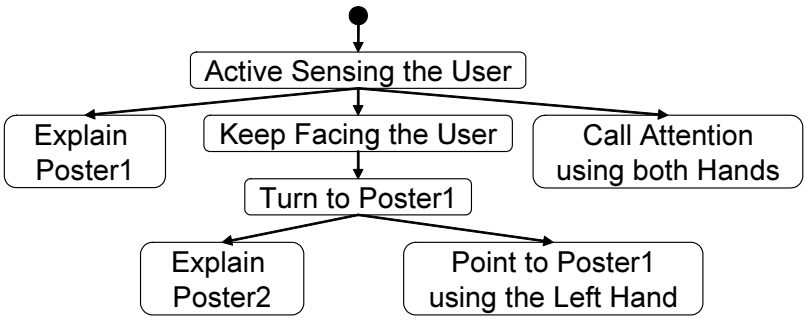

Figure 4.1: Atomic Behavior Tree

(2) **Attributes of an Atomic Behavior**

Both *EA* and *FA* request *AB* trees to PRIMA asynchronously. As in Figure 4.2, each *AB* has information about concrete actions, links, priority and occupying devices. PRIMA decides the values of *AB*'s attributes in the following way.

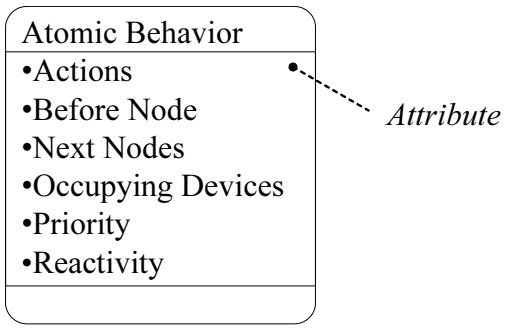

Figure 4.2: Attributes of an Atomic Behavior Class

- *Priority* and *occupying devices* are predefined on each Command.
- *Before node* means a parent node of this object in *AB* Tree, and *next node* means children of the object.

- *Actions* and *before node* are given as Command's arguments by an application.
- *Next nodes* are decided by PRIMA's exploring *before nodes* of leaf nodes.
- *Reactivity* is specially given depending on the application's description handling event, and is used for implementation of the *CS* model.

(3) **Scheduling AB trees**

PRIMA checks devices occupied with each *AB*, and exclusively executes *AB*s which occupy the same devices because simultaneous access to the same device causes interference between *FA* and *EA*. The priority of each *AB* decides the execution order of competitive *AB*s.

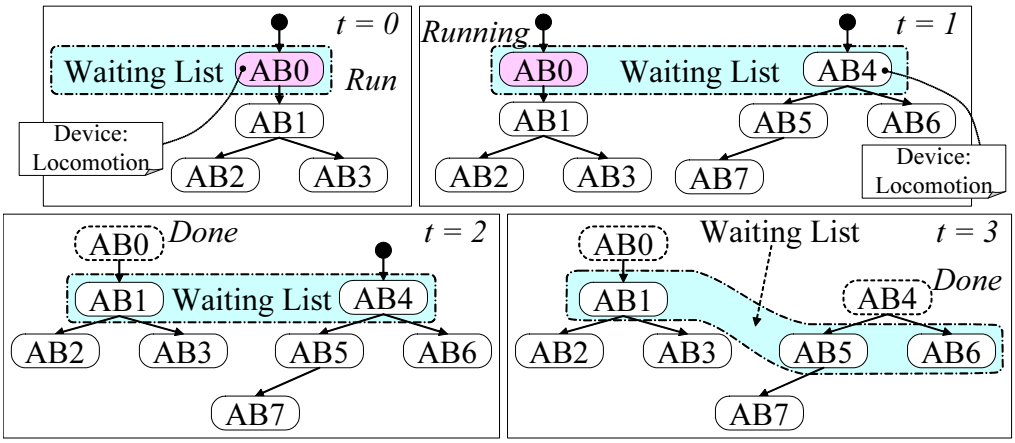

Figure 4.3: Working Memory of PRIMA

Figure 4.3 shows a snapshot of the working memory (*WM*) of PRIMA. In Figure 4.3, one application requests the *AB* tree (0-3) at t=0, and the other application requests the *AB* tree (4-7) at t=1. PRIMA updates *WM* when PRIMA accepts *AB* trees, and when each *AB* completes its actions. When PRIMA accepts an *AB* tree, PRIMA adds the root node of the tree into the waiting list in *WM*, and starts the *AB* of the root node as long as the *AB* doesn't interfere

with other running *AB*s. For example, PRIMA doesn't start AB4 at t=1, because AB0 and AB4 have a same occupying device. When each *AB* completes its action, PRIMA adds the child node of an accomplished *AB* into the waiting list, and selects starting *AB*s. If there is any interference among nodes in the waiting list, PRIMA tries to start the nodes which have a higher priority and does not interfere with any running AB. For example, at t=2, PRIMA adds AB1 into the waiting list, and checks interference between AB1 and AB4. In the case of t=2, AB4 is selected. After the end of AB4, PRIMA adds the children of AB4 into the waiting list and selects starting nodes.

(4) **Feedback Behavior**

As described in Section 4.1, some Commands have a feedback loop, and create one *AB* after another. PRIMA calls such an *AB*a **Feedback Behavior** (*FB*), and reconfigures an *AB* tree includes *FB*s.

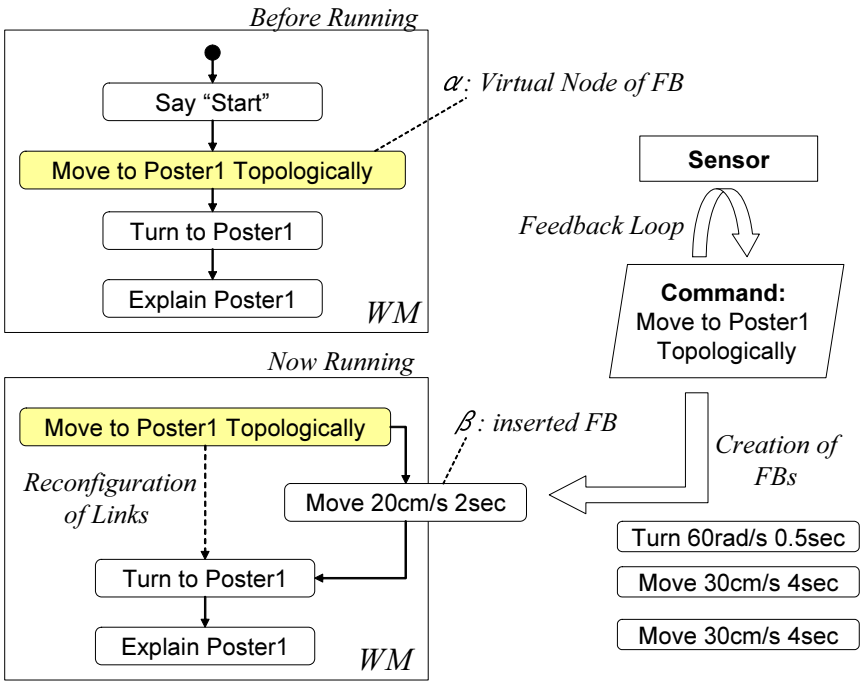

Figure 4.4: Feedback Behavior

For example, Figure 4.4 shows an *AB* tree including *FB*s in *WM*. The node α is a *FB* created by a Command of a topological move. The node α is a virtual node, and is linked before and after nodes, until α starts running.

After α starts running, the Command of α creates a *FB* depending on the state of the robots at each moment. PRIMA inserts a new *FB* between a *FB* just created, and an *AB* of the next Command. When a *FB* finishes running, a *FB* checks its own flag terminating feedbacks marked by a Command. When a Command fills its termination conditions, the Command marks a *FB* as a terminal *FB*, and PRIMA starts the next *AB*s created by other Commands. If the flag doesn't show termination, the *FB* waits to insert a new *FB*.

(5) **Descripting the order of Commands**

Figure 4.5 shows how PRIMA changes Commands requested by an Application into *AB* trees. An application creates a Command object from each Command Type, and sends it to PRIMA. The application gives a before node and parameters which show the amount of motion to the Command. PRIMA requires information about a before node of each Command, in order to understand running order required by an application, and to compose an *AB* tree. However, Setting a before node imposes complex description on *FA* or *EA* Developers. Therefore, PRIMA supports a simple method to express a before node. The running order expresses the order of Commands requested by the application. An application selects the following two options of a running order, and the argument is related to a before node of an *AB* brought by the Command.

Run this Command after the latest Command requested :

PRIMA selects this option as a default if an application sets no value for a running order. In this case, PRIMA considers that a before node of the Command is the latest Command requested, and these Commands are included in the same tree. In consequence, PRIMA runs Commands in order of the application's requests.

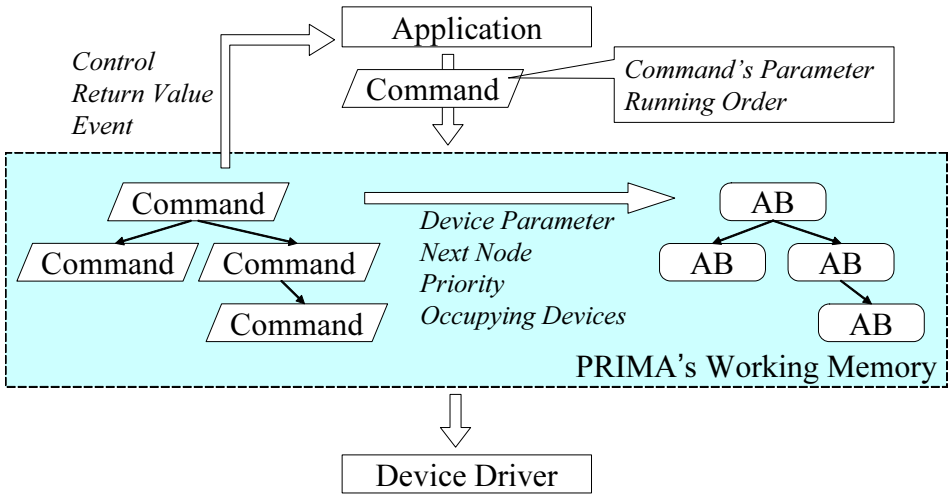

Figure 4.5: Implementation of PRIMA

Run this Command as soon as possible :

If an application selects this option, PRIMA starts the Command as soon as possible, even though some requested Commands are canceled. In this case, requests from an application may have some conflicts. For example, an application often requests "Stop moving", before a robot accomplishes a request of "Move for 30 seconds" from the same application. In this case, PRIMA believes a newer request. PRIMA cancels and deletes executing AB trees in cases which have the same occupying devices with later AB trees requested from the same application. As a matter of convenience, PRIMA assumes the canceled Command as a before node of new Command, and these Commands are included in the same tree, as described in Section 4.4.

A Command registers itself at a before Command of the Command, and makes a Command tree. Each Command creates an AB. The Command gives predefined priority and occupying devices, and brings parameters for the devices, as well as next nodes given by its children, to the AB. Each AB starts and sends parameters to devices, when PRIMA selects the AB . When these devices finish the motion, PRIMA deletes the AB and the Command, and returns values and events to the application.

4.3 Implementation of the Communication Channel

PRIMA aims to keep the *CC*, by way of delaying a Command which disconnects the *CC* as described in Section 3.3. As shown in Table 4.2, Commands are distinguished into two categories.

(i) A Command disconnects *CC*.

(ii) A Command doesn't disconnect *CC*.

We assume that a Command disconnects the *CC*, if the Command has locomotion without considering human position, and isn't used for gestures. For example, (i) includes moving to a topological goal, and moving to a map position.

A Command has a predefined priority, which is high or low. The Command in (i) has a low priority, and the Command in (ii) has a high priority. An *AB* succeeds a priority from a Command. PRIMA schedules *AB*s based on their priority as described in Section 4.2, in order to comply with the model in Section 3.3.

In consequence, when the request of an application includes the Command of (i), PRIMA does not guarantee continuity of the Command. Therefore, an application should not request these Commands when the application needs continuity.

4.4 Implementation of the Communication Stream

PRIMA aims at achieving the communication model as described in Section 3.2 under the rules of Section 4.2 and Section 4.3. PRIMA decides an end of a *CS* and the necessity of response speed. In addition, PRIMA guarantees continuity of a *CS*, as long as an application does not request an *AB* needing a response speed.

We assume an *AB* tree as a *CS*. When any node of an *AB* tree is running, PRIMA defines the *AB* tree as active. When a priority of an *AB* is equal to a priority of other *AB*, PRIMA selects and runs an *AB* included in an active *AB* tree, in order to keep the *CS*.

- The rules for keeping a *CS* is simple because PRIMA considers continuity only between nodes of an *AB* tree.

- An application programmer can easily understand that each tree is composed of Commands which have order or dependency relations with each other.

Guarantee of Response Speed :

We assume that PRIMA can predefine the conditions when a robot should react as soon as possible. PRIMA posts an event to both *FA* and *EA*when a robot fills these conditions, such as in the commands "lost human", or "hear a sound". When an application receives such an event, the application handles the event and requests Commands. An application sets a flag on the first Commands requested by an event handler. [2] The Command gives the *AB* the attribute "Reactivity", which shows the necessity of the response speed. PRIMA selects the latest *AB* marked reactivity in *WM*, when *AB*s in *WM* have the same priority, in order to guarantee response speed.

However, PRIMA cannot support all conditions, especially in the case where *FA* to need a response speed. Therefore, PRIMA supports an application to define an event, and to apply this priority. A *FA* has knowledge about the original devices of a robot, and can define a new event such as "A robot has just received an input from the touch sensor". In conclusion, PRIMA makes decisions in the following order, at t=2 in Figure 4.3.

1. PRIMA selects an *AB* with a higher priority, in order to keep *CC*.
2. If both of the *AB*s have the same priority, then PRIMA selects the latest *AB* whose reactivity flag is on, in order to guarantee response speed. For example, if the flag of AB4 is set, PRIMA selects AB4.
3. If there is no *AB* whose reactivity flag is on, PRIMA selects an *AB* whose parent has just finished, for example AB0.

Guarantee of Continuity :

As mentioned in Section 4.2, the option of running order affects the setting of before nodes. However, the option does not guarantee that PRIMA will link *AB*s. PRIMA defines a leaf node of an *AB* tree as an *AB* which has

[2] This basic function is supplied with an abstract class of applications.

no next nodes in *WM* when the *AB* finishes, as described in Section 3.2. In consequence, PRIMA creates a new *AB* tree and doesn't guarantee continuity between a new *AB* and its before node if an application requests the new *AB* after the before node finishes running. Nevertheless the application requests continuity.

However, an application sometimes needs continuity of *CS* although the application cannot request a Command in time for terminating *CS*. When an application waits for (1) the end of a Command or (2) the input of the user, there is a moment when no *AB* is in *WM* because the application requests a new behavior based on the state at the moment.

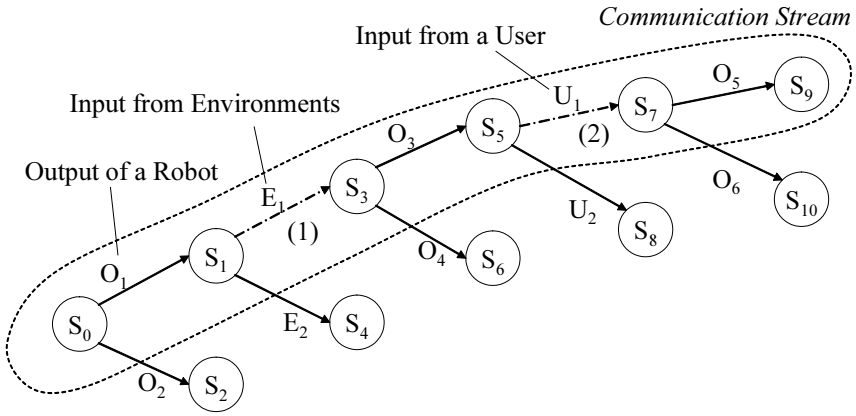

Figure 4.6: Exceptions of Communications Stream

As described in Section 2.2, many application manages their internal states based on a FSM. Figure 4.6 shows an example of FSMs includes above cases. An application changes its state when a robot gets input or finishes output. Although the application needs continuity while the sequence enclosed by a dashed line, a *CS* is terminated at (1) and (2) in Figure 4.6. PRIMA has exceptions, so that an application can express continuity in these cases.

(1) PRIMA uses the end event of a Command, in order to distinguish a state of an application waiting for the end of Command. A Command sends an end event, when it exits. PRIMA keeps the *CS* while an

application handles the event. [3] For example, if the application in Figure 4.6 requests O_3 from a handler of O_1's end event, PRIMA guarantees that O_1 and O_3 are included in the same *CS*. This exception doesn't allow E_1 to take many times, because of keeping response speed of other suspended applications. In particular, PRIMA terminates the *CS*, when the end-event's handler includes polling events.

(2) On the other hand, PRIMA supports a special Command which includes polling user's input, for example, "Q&A". For example, an application uses a "Q&A" at the state of S_3 in Figure 4.6. The "Q&A" creates a robot's utterance (O_3) and waits user's answer (U_1) for a predefined time. As described in Section 4.1, PRIMA defines O_3 and U_1 as an *AB*, because O_3 may include a speech act to require a user's answer. In addition, PRIMA assumes that a user's answer of a "Q&A" also includes such a speech act to require a robot's output. At least, a robot should output whether a speech recognition succeeds. Therefore, PRIMA defines that O_5 and O_3 are included in a same *CS*. At the time of S_3, if a user inputs U_3 which brings an event of other suspending applications, PRIMA interrupts the *CS* after timeout of U_1. [4]

PRIMA decides an end point of a *CS* without knowledge of an application's logic. However, the decision includes incorrectness depending on the contexts of an application. Therefore, PRIMA supports a method to describe links of Commands, and guarantees the links and their continuity. In this method, an application sends a list of Commands expressing an *AB* tree. For example, when O_1 in Figure 4.6 includes multiple Commands, an application sends the Commands all at once. PRIMA recommends this method to an application when an application can decide future behaviors based on a current state, for example by explaining displays along with a predefined course.

[3] In this case, PRIMA doesn't set the attribute of reactivity, because these events doesn't mean user's input or changing environment.

[4] An example of interruption is shown in Section 5.2.

4.5 Summary of Programmers' Restriction

One goal of PRIMA is to allow an application programmer to concentrate on an application's logic without the programmer having to worry about other applications. The expression of continuity is one difference between PRIMA and traditional styles of programming a robot. A block size of *AB* depends on the amount of motion given to a Command. In order to balance continuity and response speed, PRIMA recommends that an application give arguments divided at semantic breakpoints, such as at the slash of utterances. In addition, an application programmer pays attention to continuity between Commands.

PRIMA considers two kinds of continuity between Commands of an application. One is the continuity of the *IU*, and the other is the *CS*. An application should define a block of an *IU* when an application needs continuity between Commands. PRIMA overrides the definition of *AB*'s block by a Command into a definition based on start and end signals brought by an application. In this case, the programmer's permission is strong so that a programmer should be careful terminating AB. Allocating too many blocks reduces the response speed of other applications.

On the other hand, PRIMA does not allow an application to define a *CS* by barring some exceptions. Even if an application can request a block of *CS*, the continuity of CS is not always guaranteed because PRIMA prefers response speed to *CS*. It costs too much for an application to describe a block which needs continuity, but when continuity is not essential, such as in a chitchat. Therefore, PRIMA aims at detecting an approximate end-point of a *CS* without the programmer having to worry about a *CS*.

PRIMA allows an application to define a condition when PRIMA posts an event so that an application can express the conditions needed for response speed. Express important timing is important when a system should react, such as in the cases of JAVA [41] and CORBA [34]. Therefore, an application programmer does not require much cost in expressing response speed.

With regard to *CC*, PRIMA guarantees keeping *CC*, but not to connect *CC*. The motion of connecting *CC* may have a relation with the "personality" of each robot. Therefore, PRIMA entrusts an application to connect to the *CC*, but does not add unnecessary motions.

5. Experiment

5.1 Evaluation of Mediated Motions Using Video

This section has evaluated the effectiveness of models of subjects watching videos which have shown examples of mediations as described in Chapter 3. The test applications in Table 5.1 below have been designed. All lines categorized as *FA* correspond to functions included in the *FA* of a personal robot, e.g., FA1, FA2. Each line categorized as a *FA* corresponds to an *EA* loaded in each environment. This experiment includes 10 examples of mediated output, as shown in Table 5.2, by changing applications and user's inputs. This experiment includes videos of the examples shown in Figure 1.1(a) using the personal robot PaPeRo made by NEC.

Table 5.1: Application Example

FA	1	Express Emotions
	2	Walk Following User
	3	Dance to a Tune
	4	Take a Picture
	5	Set Video Timer
	6	Scheduler
EA	1	Guide User to Museum and Explain Displays
	2	Search for Books in a Library
	3	Navigate User to a Room & Tell Messages from Hosts

A subject watched videos, and compared an example mediating an *EA* and a *FA* to each example running only the *EA* or the *FA*. A subject read texts explaining the robot's outputs and the user's input of examples, and checked subjectively the order of the robot's outputs which the robot should change from the viewpoint of a user.

Figures 5.1-5.4 represent show illustrate materials checked by a subject. Figure 5.1 illustrates an example running only FA2, and Figure 5.1 illustrates an example of EA3. Figure 5.4 represents a part of a video illustrating an example mediating FA2 and EA3. Figure 5.3 is a text explaining the video. Each number

in Figure 5.4 corresponded to the behavior ID of Figure 5.3. The colored line in Figure 5.3 is an output requested from FA2. A robot's output is marked as "R", and a user's input behavior is marked as "U". In these videos, captions explained events about networks because a subject could not understand the timing of these events from the videos alone.

Table 5.2: Test Sample

EA	FA	Situations
1	2	A robot navigated a user to a host's room. A robot tracked a user because of the user's requests during navigation. The robot passed a corner, while tracking the user away from the room.
1	2	A robot navigated a user to a host's room. A robot tracked a user because of the user's requests during navigation. A robot got a message from the host, while tracking the user.
1	2	A robot navigated a user to a host's room. A robot tracked a user because of the user's requests during navigation. A robot got a message from the host, while detecting the user before
1	5	A robot navigated a user to a host's room. A robot started conversation for a video timer because of the user's requests during navigation. A robot got a message from the host while listening to a user's utterance.
2	5	A robot was requested to set a video timer while conversation of searching books.
2	6	It became the time to inform user's schedule while conversation of searching books.
3	2	A robot was requested to track a user while explaining a display.
3	1	A robot was touched a user while explaining a display.
3	3	An event for dancing arose while explaining a display.
3	4	A robot was requested to take a picture while explaining a display.

No.		Behaviors	Event
0	U	The user touched the sleeping robot.	○
1	R	"What's the matter with you?"	
2	U	"Come on, follow me."	
3	R	The robot repeated the special behavior Turn-and-Search, saying "Where are you?"	
4	R	The robot found the user.	○
5	R	The robot started following the user, saying "I found you, I'll start following you."	
6	R	The robot modified its position and pose to keep an appropriate distance from the user.	
7	U	The user touched the robot.	○
8	R	"May I stop following you?"	
9	U	"Yes."	
10	R	The robot said, “O.K. I’ll stop following you, and went into sleep mode.	

*** special behavior Turn-and-Search ***
The robot stays still, until the robot will recognize the human by its camera. If the robot doesn't find the human for ten seconds, the robot turns the angle of its camera's view, and retries the recognition.

Figure 5.1: Output Example of FA2

No.		Behaviors	Event
0	R	The robot entered the building, and started the EA3.	
1	R	"Where are you going?"	
2	U	"Prof. Kidode's room."	
3	R	"Come on, let's go"	
4	R	The robot started a topological move toward the goal.	
5	R	The robot received a socket message from Prof. Kidode while navigating the user to the goal.	○
6	R	The robot stopped there, and said "I've just got a message."	
7	R	"I'm Kidode, I'll be about 5 minutes late because of a meeting now."	
8	R	"Do you have a reply?"	
9	U	"Yes." "I see. I'll wait in your room."	
10	R	The robot sent the message to the server, saying "O.K. I'll send your message."	
11	R	The robot started a topological move.	
12	R	The robot reached a corner.	○
13	R	The robot pointed a hand and turned to the direction of the goal saying, "the room of Prof. Kidode is over there."	
14	R	The robot reached the goal.	○
15	R	The robot stopped, and said "I've just reached the goal."	

Figure 5.2: Output Example of EA3

	No.		Behavior	Event
	0	R	The robot entered the building, and started the EA3.	
┌	1	R	"Where are you going?"	
└	2	U	"Prof. Kidode's room."	
	3	R	"Come on, let's go."	
★	4	R	The robot moves topologically to the goal.	
	5	U	The user interested in a poster by the way to the goal.	
	6	U	The user touched the moving robot.	○
┌	7	R	The robot stopped and said "What's the matter with you?"	
└	8	U	"Come on, Follow me."	
★	9	R	The robot repeated the special behavior Turn-and-Search, saying "Where are you?"	
	10	R	The robot got a socket message from Prof. Kidode, while searching the user.	○
	11	R	The robot stopped there, and said "I've just got a message."	
	12	R	"I'm Kidode, I'll be about five minutes late for my appointment because I'm still in a meeting."	
┌	13	R	"Do you have a reply?"	
└	14	U	"Yes." "I see. I'll wait in your room."	
	15	R	The robot sent the message to the message server, saying "O.K. I'll send your message."	
★	16	R	The robot restarted the special behavior Turn-and-Search, saying "Where are you?"	
	17	R	The robot found the user.	○
	18	R	The robot said, "I found you, I'll start following you."	
	19	R	The robot started following the user.	
★	20	R	The robot repeated a modification of its positions and pose to keep an appropriate distance from the user.	
	21	U	The user returned to the poster, and watched it for while.	
★	22	R	The robot kept tracking the user.	
	23	U	The user finished reading the poster, and touched the robot.	○
┌	24	R	"May I stop following you?"	
└	25	U	"Yes."	
	26	R	"O.K. I'll finish following you."	
★	27	R	The robot restarted topological moving to the goal.	
	28	R	The robot reached the goal.	○
	29	R	The robot stopped, and said "I've just reached the goal."	

Figure 5.3: Mediated Output (Answer Sheet)

Figure 5.4: Mediated Output of FA2 and EA4 (Video)

No.	Root_Command(arg), Leaf_Command1(arg), ... ;	Event
0	Q&A("Where are you?");	
2	Speak("Come on, let's go."), Topological_Move(EA_DEF_POSITION_KIDODE_ROOM);	
10	Stop_Moving(), Speak("I've just got a message."), Spaek("I'm Kidode, I'll be about five minutes late for my appointment because I'm still in a meeting."), Q&A("Do you have a reply?");	○
14	Speak("O.K. I'll send your message."), Topological_Move(EA_DEF_POSITION_KIDODE_ROOM);	
28	Stop_Moving(), Speak("I've just reached the goal.");	○

Figure 5.5: Requests from EA3

No.	Root_Command(arg), Leaf_Command1(arg), ... ;	Event
6	Q&A("What's the matter with you?");	○
8	ActiveSensingHuman("Where are you?");	
17	Speak("I found you, I'll start following you."), Tracking_Human();	○
23	Stop_Moving(), Q&A("May I stop following you?");	○
25	Speak("O.K. I'll finish following you.");	

Figure 5.6: Requests from FA2

Figures 5.5-5.6 show Commands requested by each application in the example of Figure 5.3. The numbers in Figures 5.5-5.6 correspond to the behavior ID in Figure5.3. At a line including multiple Commands in Figures 5.5-5.6, an application requested a list of these Commands as described in Section 4.4. A mark in the Event column represents an application giving a flag of reactivity to a root Command.

In the example of Figure 5.3, strategies of *CS* and *CC* were helpful to the user. For example, at the 11th line of Figure 5.3, PRIMA selected a request from EA4 which was a reaction of a network event rather than keeping a *CS* of FA2, following the model described in Section 3.2. At the 6th line, PRIMA

also preferred the response speed. At the 15th line, because EA4 requested a command disconnecting a *CC* (Figure 5.5 No.14), PRIMA selected the request from FA2 (Figure 5.6 No.6), following the model described in Section 3.3. PRIMA kept a *CS* expect from these two lines.

This experiment evaluated the effects of these models based on texts such as in Figures 5.1-5.3. A subject interchanged lines by watching videos when he/she preferred to do so. A line corresponding to an AB, excluded the case in (2) and (3) below. In this experiment, a subject was given the following restrictions:

(1) A line marked by "⋆" included multiple *AB*s which allowed other lines to interrupt them.

(2) A line closed by "[" was composed of an *AB* that did not allow interruption.

(3) The subject was not allowed to change the timing of events, because PRIMA can not control the timing.

(4) The subject was not allowed to add or delete lines, because PRIMA can not add or delete *AB*s.

5.2 Experimental Results

In this experiment, 12 subjects checked each 10 examples, and 120 trials were carried out in total. A result of the experiment showed that 76% of 120 trials did not need interchange and felt natural to the subjects. However, two patterns were found in cases when many subjects required interchanges.

The first pattern was to terminate *CS*. 56% of 27 trials required to interchange robot's behaviors applied to this case. PRIMA decided a block size of *CS* without explicit descriptions of an application as shown in Section 4.4. PRIMA terminates *CS* when an application requests the new *AB* after the before node finishes running. Therefore, PRIMA does not guarantee continuity while an application waits for events to decide the next behaviors without outputs. When EA2 waited for a reply from a server of a library, and when FA5 waited for a reply from a server of a video, these applications were interrupted by other applications. Figure 5.12 shows a mediated example of FA5 and EA2.

No.		Behaviors	Event
0	U	The user pushed a touch sensor of robot.	
1	R	"What's the matter with you?"	
2	U	"Video Timer."	
7	R	"Please select a TV channel."	
8	U	"12."	
3	R	"Please tell me a time to start."	
4	U	"18."	
5	R	"Please tell me a time to end."	
6	U	"19."	
9	R	"May I set the video timer to record the 12th channel from 18 o'clock to 19 o'clock?"	
10	U	"Yes."	
11	R	The robot sent the request to a video, and said "Connecting the video server."	
12	R	The robot received a message about the succession of requests from the video.	○
13	R	"The video setting succeeds."	
14	R	The robot started sleeping.	

Figure 5.7: Output Example of FA5

No.		Behaviors	Event
0	R	The robot entered the library, and started the EA2.	
1	R	"Please tell me a title or an author of a book."	
2	U	"The C Primer."	
3	R	The robot sent the query to a server of the library, and said "Wait a minute."	
4	R	The robot got an answer of the query.	○
5	R	"Are you searching 'The C Primer' published by ASCII software science?"	
6	U	"Yes."	
7	R	"O.K. Come on."	
8	R	The robot moved topologically to a bookshelf which included the book.	
9	R	The robot reached the bookshelf.	○
10	R	"Here is the book you searched for."	

Figure 5.8: Output Example of EA2

	No.		Behaviors	Event
	0	R	The robot entered the library, and started the EA2.	
┌	1	R	"Please tell me a title or an author of a book."	
│	2	U	The user pushed a touch sensor of robot.	○
└	3	R	(The robot had no reaction for a few seconds.)	
┌	4	R	"What's the matter with you?"	
└	5	U	"Video Timer."	
┌	10	R	"Please select a TV channel."	
└	11	U	"12."	
┌	6	R	"Please tell me a time to start."	
└	7	U	"18."	
┌	8	R	"Please tell me a time to end."	
└	9	U	"19."	
┌	12	R	"May I set the video timer to record the 12th channel from 18 o'clock to 19 o'clock?"	
└	13	U	"Yes."	
	14	R	The robot sent the request to a video, and said "Connecting the video server."	
┌	15	R	"Please tell me a title or an author of a book."	
│	16	R	The robot received a message about the succession of requests from the video.	○
└	17	U	"The C Primer."	
	18	R	"The video setting succeeds."	
	19	R	The robot sent the query to a server of the library, and said "Wait a minute."	
	20	R	The robot got an answer of the query.	○
┌	21	R	"Are you searching 'The C Primer' published by ASCII software science?"	
└	22	U	"Yes."	
	23	R	"O.K. Come on."	
★	24	R	The robot moved topologically to a bookshelf which included the book.	
	25	R	The robot reached the bookshelf.	○
	26	R	"Here is the book you searched for."	

Figure 5.9: Mediated Output of FA5 and EA2

The second pattern occurred while starting communications. PRIMA guarantees to keep *CC*, but not to connect *CC*, as shown in Section 4.5. An application may skip face-to-face communication. For example, the robot started conversation as soon as the user touched at behavior No.1 in Figure 5.1. Therefore, in the mediated examples in Figures 5.3-5.4, a robot started conversation before turning to its user at behavior No.10. 25% subjects mentioned that the robot should start conversation of the 9th line before facing the user of the 19th line, in the case of this example.

5.3 Discussion

After each trial, a subject was interviewed. The subject was asked which behaviors of the robot's did he/she feel were unnatural without any restrictions. This interview showed that 58% of 120 trials stisfied with subjects. Based on these interviews and experimental results, this thesis summarizes the following functions which PRIMA should support in the future:

(1) Resumption of robot's state

(2) Interruption of an *AB*

(3) Description of the continuity of *CS*

(4) Guarantee connection to *CC*

(5) Concurrent execution of *AB*s which interfere with each other

(6) Relaxing restrictions

(1) A robot should fill various states in order to communicate with a user. Nevertheless, an interruption may break these states that were filled by an interrupted application. PRIMA guarantees the continuity of *IU*, *CS*, and *CC*, and contributes to keeping these states. In particularly, PRIMA attempts to keep states of relative position between a robot and a user. However PRIMA cannot keep all states an application requires in order to balance response speed and continuity. The size of an IU is restricted. A CS is interrupted by an AB marked as reactive. A CC doesn't restrict all locomotion of a robot, as shown in Section

4.3. As a result, an interruption has possibilities to break the states of relative positions.

In order to guarantee the states an application requires, PRIMA aims at resuming the robot's state rather than restricting interruption, such as stopping a motion out of limits based on sensors. Resuming states will fill the requirements of both applications by keeping response speed. Links in an *AB* tree include information of the state of a robot needed at each moment by an application. Exploring parent nodes will bring the preconditions of each node without the strict descriptions.

(2) PRIMA gives importance to keeping the continuity of an *IU*, and does not interrupt while an *IU* is running, as described in Section 3.1. The size of an *AB* depends on the programmer's descriptions, and there are few *AB*s which do not comply with the definition of an *IU*, as shown in Section 4.1. These *AB*s may cause a lack of response speed. In the interview after the experiment, some subjects said that they wanted to interrupt motions defined as an *AB*, such as an utterance and recognition of a turn of speech.

PRIMA will allow interruptions of an *AB* and will guarantee the restart of an *AB* interrupted from the beginning of the *AB*. PRIMA will assume that this rule will not mislead a user, and will not make a user uncomfortable. If the assumption is correct, PRIMA will be able to adopt this rule. In this thesis, I consider whether or not an *IU* should keep continuity even if PRIMA guarantees the restart of an *IU*.

Figure5.10-5.12 are examples used in the experiment of Section 5.2, and these examples show the necessity of interruptions. FA4 is an example in which a *FA* developer explicitly defines a block of the *AB*described in Section 4.1(2). A *FA* developer defines a 3-10 line in Figure5.10 as a a block of *AB* , on the condition that interruptions of *EA* do not make cameras depart from target. In the interviews, many subjects wanted the robot to reply as soon as possible when they touched the robot as in the 7R line of 5.12. If PRIMA allows the interruption, a robot should turn to the user and start Explanation-1-1 again. As described, the interruption of an *AB* needs a resumption of robot's state.

No.		Behaviors	Event
0	U	The user touched the sleeping robot.	○
1	R	"What's the matter with you?"	
2	U	"Take a picture."	
3	R	"Which direction should I turn?"	
4	U	"Turn to your right."	
5	R	The robot turns right.	
6	R	"May I take a picture?"	
7	U	"Yes."	
8	R	The robot captures an image with its camera, and sends the image to a mobile computer of the user.	○
9	R	"I just sent a picture."	
10	R	"May I take another picture?."	
11	U	"No."	
12	R	"O.K. Bye-bye."	
13	R	The robot goes to sleep.	

Figure 5.10: Output Example of FA4

No.		Behaviors	Event
0	R	The robot entered the exposition, and started the EA1.	
1	R	"I'm going to start the tour of the museum."	
2	R	The robot moves to Panel-1.	
3	R	The robot arrives at Panel-1	○
4	R	The robot repeated the special behavior Turn-and-Search.	
5	R	The robot found the user.	○
6	R	The robot turns to the user, and speaks Explanation-1-1. "This panel shows research points of our laboratry."	
7	R	The robot turns to Panel-1, and speaks Explanation-1-2."Take attention to this upper figure. Our laboratory has a project about	
8	R	The robot repeated the special behavior Turn-and-Search.	
9	R	The robot found the user.	○
10	R	The robot turns to the user, and speaks Explanation-1-3. "In addition, we are studing various techniques processing multi-media informations in the real world."	
11	R	"May I repeat the exaplantion?"	
12	U	"No."	
13	R	"May I go to the next panel?"	
14	U	"Yes."	
15	R	The robot says, "O.K.", and moves to Panel-2.	

*** special behavior Turn-and-Search ***
The robot stays still, until the robot will recognize the human by its camera. If the robot doesn't find the human for ten seconds, the robot turns the angle of its camera's view, and retries the recognition.

Figure 5.11: Output Example of EA1

	No.		Behaviors	Event
	0	R	The robot entered the exposition, and started the EA1.	
	1	R	"I'm going to start the tour of the museum."	
	2	R	The robot moves to Panel-1.	
	3	I	The robot arrives at Panel-1	○
	4	R	The robot repeated the special behavior Turn-and-Search.	
	5	I	The robot found the user.	○
┌	6	R	The robot turns to the user, and speaks Explanation-1-1. "This panel shows research points of our laboratry."	
│	7	I	The user touched the robot speeking Explanation-1-1.	○
└	8	R	The robot finishes Explanation-1-1.	
┌	9	R	"What's the matter with you?"	
└	10	U	"Take a picture."	
┌	11	R	"Which direction should I turn?"	
│	12	U	"Turn to your right."	
│	13	R	The robot turns right.	
│	14	R	"May I take a picture?"	
│	15	U	"Yes."	
│	16	R	The robot captures an image with its camera, and sends the image to a mobile computer of the user.	
│	17	R	"I just sent a picture."	
│	18	R	"May I take another picture?."	
│	19	U	"No."	
└	20	R	"O.K. Bye-bye."	
	21	R	The robot turns to Panel-1, and speaks Explanation-1-2."Take attention to this upper figure. Our laboratory has a project about wearable computers."	
★	22	R	The robot repeated the special behavior Turn-and-Search.	
	23	I	The robot found the user.	○
	24	R	The robot turns to the user, and speaks Explanation-1-3. "In addition, we are studing various techniques processing multi-media informations in the real world."	
┌	25	R	"May I repeat the exaplantion?"	
└	26	U	"No."	
┌	27	R	"May I go to the next panel?"	
└	28	U	"Yes."	
	29	R	The robot says, "O.K.", and moves to Panel-2.	

Figure 5.12: Mediated Output of FA4 and EA1

(3) PRIMA does not aim to describe termination of *CS* because of the costs to the programmers, as described in Section 4.5. As a result, PRIMA may not keep *CS*, which depends on the timing of requests from the application. Describing a block size of *AB* will adapt to this case. These definitions involve a risk of reducing response speed, as in Section 4.5. However, I expect that PRIMA will manage this risk, if PRIMA supports the functions of (1) and (2).

In addition, PRIMA will allow an application programmer to express a *CS* easily. In order to describe applications easily, PRIMA will adopt general languages of spoken dialogue systems such as VoiceXML. These languages of spoken dialogue systems have structures controlling dialogues, similar to a document's tree of VoiceXML. These structures often depend on contexts and topics of dialogues. Therefore, it may be possible for an application programmer to express easily a block of *CS*, by using these structures of documents.

(4) In general, the definition of *CC* is different depending on the situations and features of a robot's embodiment, as described in Section 4.5. A motion for connecting a *CC* brought by a middleware may break a posture imagined by an application. Therefore PRIMA dose not aim to connect *CC*s, except when these resuming from other applications breaking *CC*, as in (1). On the other hand, PRIMA can detect disconnecting *CC*s, and the strategy of the *CC*s as in Section 3.3 is effective.

(5) As described in Section 4.2, PRIMA allows simultaneously running *AB*s which doesn't interfere with. For example, a robot can say hello while navigating the user. In the future, PRIMA will also allow simultaneously running few kinds of *AB*s which interfere with each other. For example, a robot may dance while guiding the user if the robot doesn't go far away from the user. PRIMA will categorize robot's locomotion into two patterns. Locomotion for a dance should be loyal to trajectories required an application, but the global position of the dance isn't important. On the other hands, moving to a goal for navigations allows PRIMA to make free trajectories.

(6) Section 2.4 assumes three restrictions designing PRIMA; hardware specifications, loading applications, and programming styles of applications.

Hardware Specifications : In the future, PRIMA should increase kinds of devices which PRIMA can deal with. I'm planning to draw up specifications

of Commands so that a *FA* can add a new Command which handles an original device of each robot. PRIMA may allow variety of devices, guaranteeing an *EA* 's output, if a robot fills functions specified by PRIMA generalizing interface of PRIMA. For example, a robot uses a laser pointer instead of pointing by a finger.

Arguments are needed so that PRIMA can relax the restriction that a robot fills functions which PRIMA specifies. If a robot doesn't fill the restriction, PRIMA cannot accomplish an application's request. It is necessary to consider what PRIMA should guarantee to an application, and what PRIMA should mediate these requests.

Loading Applications : PRIMA should relax an assumption in order to load multiple *EA*'s which a user isn't aware of loading the *EA*'s. In Section 3.2, PRIMA designs rules of a *CS*, under the assumption that a user knows whether an *EA* is loaded. In addition, PRIMA may have multiple applications suspended at a time. PRIMA needs a new rule of resuming *CS*s.

Programming Styles of Applications : PRIMA should need more clear definition of programming styles, and usability for application programmers needs more arguments.

6. Conclusion

In this paper, I proposed a framework to give a personal robot the ability to supply local services in several environments while keeping its personality. I attempt to integrate an Environment-oriented Application which is an information service loaded in each environment, and a Familiarity-oriented application witch brings a personality of each robot. The integration of two different kinds of applications which have been independently developed causes a lack of continuity and a lack of response speed in human-robot communication. In order to solve the trade-off between continuity and response speed, I designed three communication models: the Information Unit, the Communication Channel, and the Communication Stream. I proposed a middleware named PRIMA, which mediates applications based on these models. I demonstrated the effect of PRIMA using videos which show examples of mediating applications. The outputs of PRIMA were given mostly positive evaluations. However, the experiment revealed some points to improve strategies of PRIMA. For future research, I will try to improve PRIMA by resuming robot's states, brushing up specifications of Commands, and adopting to a general language describing interactions, in order to relax restrictions of PRIMA.

Acknowledgements

First and foremost, I would like to show my deep appreciation to Prof. Masatsugu Kidode and Associate Prof. Yasuyuki Kono, for their support and patience over the course of my study at Nara Institute of Science and Technology (NAIST). I would like to thank Prof. Tsukasa Ogasawara in NASIT and Associate Prof. Michita Imai in Keio University for helpful comments. I would like to thank Dr. Atsushi Ueno in Osaka City University and Dr. Izuru Kume in NAIT for many of substantial arguments. I would like to thank NEC Personal Robot Research Center for leasing personal robots "PaPeRo". I would like to thank all members of Artificial Intelligence Laboratory in Nara Institute of Science and Technology for their participation in our experiment. Finally, I would like to thank two PaPeRos for suffering my harsh experiments.

References

[1] `http://www.incx.nec.co.jp/robot/`.

[2] `http://openr.aibo.com/`.

[3] Masahiro Arai, Toshihiko Itoh, Tomoko Kumagai, and Masato Ishikawa. "Proposal of a Standard Utterance-Unit Tagging Scheme". *JSAI Journal*, 14(2):251–260, 1999.

[4] Masahiro Araki, Akihiko Kaga, and Takuya Nishimoto. "Comparison of "Go back" implementations in VoiceXML". In *Proc. International Speech Communication Association (ISCA) workshop on ERROR HANDLING IN SPOKEN DIALOGUE SYSTEMS*, 2003.

[5] Minoru Asada. "Entertainment Robotics and Emotion/Intelligence". *JSAI Journal (in Japanese)*, 19(1):15–20, 2004.

[6] John L. Austin. *How to Do Things with Words*. London: Oxford Univ. Press, 1962.

[7] Rodney A. Brooks. "A Robust layered control system for a mobile robot". *IEEE Journal of Robotics and Automation, RA-2*, 2(1):14–23, March 1986.

[8] J. Buhmann, W. Burgard, A.B. Cremers, T. Hofmann D. Fox, F. Schneider, J. Strikos, and S. Thrun. "The Mobile Robot Rhino". *AI Magazin*, 16(1):31–38, 1995.

[9] Mika Enomoto, Masato Ishizaki, Hanae Koiso, Yasuharu Den, Etsuo Mizukami, and Hiroyuki Yano. "A Statistical Investigation of Basic Units for Spoken Interaction Analysis". In *Proc. Technical Report of IEICE SIG-HCS (in Japanese)*, volume 29, pages 45–50, 2004.

[10] Kaori Fujinami and Tatsuo Nakajima. "A Framework for Developing Context-aware Applications". *IPSJ Transactions on Advanced Computing Systems (ACS) (in Japanese)*, 44(SIG10 (ACS2)), 2003.

[11] B. Grosz and C. Sidner. "Attention, intention and the structure of discourse". *Computational Linguistics*, 12(3):175–204, 1986.

[12] M.A.K. Halliday and R. Hasan. *Cohesion in English*. Longman, 1976.

[13] Isao Hara and Yo-ichi Motomura. "Situated Multi-Agent Architecture for an Autonomous Robot". In *Proc. RSJ/SICE/JSME 5th Robotics Symposia (in Japanese)*, pages 86–91, 2000.

[14] J. Hobbs. *Literature and Cognition*, volume 21 of *CSLI Lecture Notes*. CSLI, 1990.

[15] Michita Imai, Takayuki Kanda, Testuo Ono, Hiroshi Ishiguro, and Kenji Mase. "Robot Mediated Round Table: Analysis of the Effect of Robot's Gaze". In *Proc. The 11th International Workshop on Robot and Human Communication (RO-MAN2002)*, pages 411–416, September 2002.

[16] Tetsunari Inamura, Masayuki Inaba, and Hirochika Inoue. "PEXIS : Probabilistic Experience Representation Based Adaptive Interaction System for Personal Robots". *Systems and Computers in Japan (in Japanese)*, 35(6), 2004.

[17] Masayuki Iwai, Jin Nakazawa, and Hideyuki Tokuda. "Composition of Distributed Application though Multiple and Multimodal User Interface". *JSSST Computer Software (in Japanese)*, 21(1):13–26, 2004.

[18] Koji Kageyama and Tatsuzo Ishida. "Entertainment Robot Business". *RSJ Journal (in Japanese)*, 20(7):668–671, 2002.

[19] Takayuki Kanda. *A Constructive Approach for Communication Robots*. PhD thesis, Kyoto University, 2003.

[20] Fumio Kanehiro, Kiyoshi Fujiwara, Shuuji Kajita, Kazuhito Yokoi, Kenji Kaneko, Hirohisa Hirukawa, Yoshihiko Nakamura, and Katsu Yamane. "Open Architecture Humanoid Robotics Platform: OpenHRP". *RSJ Journal (in Japanese)*, 21(7):785–793, 2003.

[21] Kouichi Katsurada, Yusaku Nakamura, Makoto Yamada, Hirobumi Yamada, Satoshi Kobayashi, and Tsuneo Nitta. "Proposal of MMI Description Language XISL". *IPSJ Journal (in Japanese)*, 44(11):2681–2689, 2003.

[22] Akihiro Kobayashi, Yasuyuki Kono, Atsushi Ueno, Izuru Kume, and Masatsugu Kidode. "Personalization of Dynamically Loaded Service Programs Keeping Human-Robot Communication Channel". In *Proc. The 13th IEEE International Workshop on Rbot and Human Interactive Communication (RO-MAN2004)*, pages OS–YN3, 2004.

[23] Akihiro Kobayashi, Izuru Kume, Atsushi Ueno, Yasuyuki Kono, and Masatsugu Kidode. "A Robot Programming Model for Mediating Between Familiarity-Oriented Behaviors and Environment-Oriented Behaviors". In *Proc. the 7th of the World Multi-Conference on Systemics, Cybernetics and Informatics (SCI2003)*, pages 295–302, 2003.

[24] Akihiro Kobayashi, Atsushi Ueno, Izuru Kume, Yasuyuki Kono, and Masatsugu Kidode. "Robot Middleware Architecture Mediating Familiarity-Oriented and Environment-Oriented Behaviors". In *Proc. The 5th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA2003)*, pages 544–551, 2003.

[25] W. Mann and S. Thompson. "Rhetorical structure theory: Toward a functional theory of text organization". *Text*, 8:243–281, 1998.

[26] Toshihiro Matsui, Hideki Asoh, John Fry, Youichi Motomura, Futoshi Asano, Takio Kurita, Isao Hara, and Nobuyuki Otsu. "Integrated Natural Spoken Dialoge System of Jijo-2 Mobile Robot for Office Services". In *Proc. of The 16th National Conference on Artificial Intelligence (AAAI-99)*, Florida, July 1999.

[27] Yosuke Matsusaka, Tsuyoshi Tojo, and Tetsunori Kobayashi. "Conversation Robot Participating in Group Conversation". *IEICE Transaction of Information and System (in Japanese)*, E86-D(1):26–36, 2003.

[28] Scott McGlashan et al. (Eds.). "Voice Extensible Markup Language (VoiceXML) Version 2.0". `http://www.w3.org/TR/voicexml20/`.

[29] Makoto Mizukawa, Hideo Matsuka, Toshihiko Koyama, Toshihiro Inukai, Akio Noda, Hirohisa Tezuka, Yasuhiko Noguchi, and Nobuyuki Otera.

"ORiN: Open Robot interface for the Network, The Standard Network Interface for Industrial Robots and its Applications". In *Proc. International Symposium on Robotics (ISR)*, 2002.

[30] Toru Nakata, Taketoshi Mori, and Tomomasa Sato. "Quantitative Analysis of Impression of Robot Bodily Expression based on Laban Movement Theory". *RSJ Journal (in Japanese)*, 19(2):252–259, 2001.

[31] Ryuichi Nisimura, Takashi Uchida, Akinobu Lee, Hiroshi Saruwatari, and Kiyohiro Shikano. "ASKA: Receptionist Robot with Speech Dialogue System". In *Proc. 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2002)*, pages 1314–1319, 2002.

[32] Masayuki Okamoto and Nobutoshi Yamanaka. "Wizard of Oz Method for Constructing Conversational Web Agents". *JSAI Journal (in Japanese)*, 17(3):293–300, 2002.

[33] Tomoki Oku, Takuya Nishimoto, Masahiro Araki, and Yasuhara Niimi. "A Task-Independent Control Method for Spoken Dialogs". *IEICE Journal (in Japanese)*, J86-DII(5):608–615, 2003.

[34] OMG. "CORBA". `http://www.corba.org`.

[35] Tetsuo Ono and Michita Imai. "Embodied Communications between Humans and Robots Emerging from Entrained Gestures". In *Proc. 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA2003)*, 2003.

[36] Sinya Oyama, Hideki Kuzuoka, Jyunichi Kosaka, and Keiich Yamazaki. "The Proposal of Requirements on Designing the System to Support Remote Communication on Embodied Space and the Development of the System". *IPSJ Journal (in Japanese)*, 45(1):178–187, 2004.

[37] F. Ozaki. "Open Robot Controller Architecture (ORCA)". In *Proc. AIM2003 Workshop on Middleware Technology for Open Robot Architecture.*

[38] H. Sacks, E. A. Schegloff, and G. Jefferson. "A simplest systematics for the organization of turn-taking for conversation". *Language*, 50(4):696–735, 1974.

[39] Yamada Seiji and Yamaguchi Tomohi. "Training AIBO like a Dog - Preliminary results -". In *Proc. The 13th International Workshop on Robot and Human Interactive Communication (RO-MAN2004)*, 2004.

[40] Stephanie Seneff, Ed Hurley, Raymond Lau, Christine Pao, Philipp Schmid, and Victor Zue. "GALAXY-II:A referance architecture for conversational system develpment". In *Proc. The 8th International Conference on Spoken Language Processing (ICSLP1998)*, 1998.

[41] Sun Microsystems. "Java(TM) 2 Platform Standard Edition 5.0 API Specification". `http://java.sun.com/j2se/1.5.0/docs/api/`.

[42] Fumihide Tanaka and Hirotaka Suzuki. "Dance Interaction with QRIO: A Case Study for Non-boring Interaction by using an Entertainment Ensemble Model". In *Proc. The 13th International Workshop on Robot and Human Interactive Communication (RO-MAN2004)*, 2004.

[43] Sebastian Thrun, Maren Bennewitz, Wolfram Burgard, Armin B. Cremers, Frank Dellaert, Dieter Fox, Dirk Hahnel, Charles Rosenberg, Nicholas Roy, Jamieson Schulte, and Dirk Schulz. "MINERVA: A Second-Generation Museum Tour-Guide Robot". In *Proc. 1999 IEEE International Conference on Robotics and Automation (ICRA1999)*, pages 1999–2005, 1999.

[44] Takashi Yoshimi, Nobuto Matsuhira, Kaoru Suzuki, Daisuke Yamamoto, Fumio Ozaki, Junko Hirokawa, and Hideki Ogawa. "Development of a Concept Model of a Robotic Information Home Appliance, ApriAlpha". In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2004)*, pages 205–211, 2004.

List of Publications

Journal (in Japanese)

[1] **Akihiro Kobayashi**, Izuru Kume, Atsushi Ueno, Yasuyuki Kono, and Masatsugu Kidode. "Mediation Architecture of Personal Robot's Applications Based on Communications Model". *IEICE Journal D-I*, (to Appear).

International Conferences

[2] **Akihiro Kobayashi**, Yasuyuki Kono, Atsushi Ueno, Izuru Kume, and Masatsugu Kidode. "Personalization of Dynamically Loaded Service Programs Keeping Human-Robot Communication Channel". In *Proc. The 13th IEEE International Workshop on Rbot and Human Interactive Communication (RO-MAN2004)*, OS–YN3, 2004.

[3] **Akihiro Kobayashi**, Izuru Kume, Atsushi Ueno, Yasuyuki Kono, and Masatsugu Kidode. "A Robot Programming Model for Mediating Between Familiarity-Oriented Behaviors and Environment-Oriented Behaviors". In *Proc. the 7th of the World Multi-Conference on Systemics, Cybernetics and Informatics (SCI2003)*, pages 295-302, 2003.

[4] **Akihiro Kobayashi**, Atsushi Ueno, Izuru Kume, Yasuyuki Kono, and Masatsugu Kidode. "Robot Middleware Architecture Mediating Familiarity-Oriented and Environment-Oriented Behaviors". In *Proc. The 5th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA2003)*, pages 544-551, 2003.

Domestic Conferences (in Japanese)

[5] Kenichirou Maeda, **Akihiro Kobayashi**, Izuru Kume, Atsushi Ueno, Yasuyuki Kono, and Masatsugu Kidode. "Robust Self-Localization Method Based on Geometric Features of the Light Sources on a Ceiling". In *Proc. The 66th National Convention of IPSJ*, 6ZB-9, 2004.

[6] **Akihiro Kobayashi**, Atsushi Ueno, Izuru Kume, Yasuyuki Kono, and Masatsugu Kidode. "A personal robot architecture with location-dependent attachable functions". In *Proc. The 3th SICE System Integration Division Annual Conference (SI2002)*, 1P2-32, 2002.

[7] **Akihiro Kobayashi**, Yasuyuki Kono, and Masatsugu Kidode. "A cooperative framework of heterogeneous for the analysis of interactive navigation with guests". In *Proc. The 16th Annual Conference of Japanese Society for Artificial Intelligence*, 1A1-03, 2002.

[8] **Akihiro Kobayashi**, Yasuyuki Kono, and Masatsugu Kidode. "A Multi-Robot Architecture Aiming for Cooperative Indoor Navigation". In *Proc. The 2th SICE System Integration Division Annual Conference (SI2001)*, 3P33-03, 2001.

[9] **Akihiro Kobayashi**, Yasuyuki Kono, and Masatsugu Kidode. "Graduate School of Information Science, Nara Institute of Science and Technology". In *Technical Report IPSJ SIG-HI*, 2001(38), pages 7-14, 2001.

[10] **Akihiro Kobayashi**, Yasuyuki Kono, and Masatsugu Kidode. "Cooperation of multi-robots in office environments". In *Proc. The 15th Annual Conference of Japanese Society for Artificial Intelligence*, 3C1-07, 2001.