A multi-expert model for dialogue and behavior control of conversational robots and aggents [⋆]

.

Mikio Nakano [a,∗] Yuji Hasegawa [a] Kotaro Funakoshi [a]
Johane Takeuchi [a] Toyotaka Torii [a,1] Kazuhiro Nakadai [a]
Naoyuki Kanda [b,2] Kazunori Komatani [b,3] Hiroshi G. Okuno [b]
Hiroshi Tsujino [a]

[a]*Honda Research Institute Japan Co., Ltd.*
8-1 Honcho, Wako, Saitama 351-0188, Japan

[b]*Graduate School of Informatics, Kyoto University*
Yoshida Honmachi, Sakyo-ku, Kyoto 606-8501, Japan

Abstract

This paper presents an intelligence model for conversational service robots. It employs modules called experts, each of which is specialized to execute certain kinds of tasks such as performing physical behaviors and engaging in dialogues. Some of the experts take charge in understanding human utterances and deciding robot utterances or actions. The model enables switching and canceling tasks based on recognized human intentions, as well as parallel execution of several tasks. This model specifies the interface that an expert must have, and any kind of expert can be employed if it conforms to the interface. This feature makes the model extensible.

Key words: conversational robot, conversational agent, robot intelligence, behavior and dialogue control, multi-expert model

---

[⋆] This paper is a considerably extended version of [1]. We thank the Association for Computational Linguistics for permission to reuse the material.
[∗] Mikio Nakano, Honda Research Institute Japan Co., Ltd.
8-1 Honcho, Wako, Saitama 351-0188, Japan. E-mail: nakano@jp.honda-ri.com
[1] Currently with Honda R&D Co., Ltd.
[2] Currently with Hitachi Ltd.
[3] Currently with Nagoya University

1 Introduction

As much attention is recently paid to autonomous robot and animation agent technology, spoken dialogue is expected to be a natural interface between humans and robots/agents. Our ambition is to establish a general model for the symbol-level *dialogue and behavior controller* of robots/agents that can engage in dialogue with humans to understand their requests and give useful information as well as perform desired behaviors. Although we are trying to build a model that can be used for both robots and animation agents, in this paper we only use the word robots for simplicity. In addition to avoid confusion with entertainment robots, we call robots that we try to build *conversational service robots*. They have conversation functions for understanding human requests and providing useful information, not just for chatting. Their dialogue and behavior controller receives output from sensor interpreters such as a speech recognizer, and is responsible for selecting utterances and physical actions to perform.

We focus on several features which we think are crucial for the usability of such robots. Since there are many kinds of tasks, they must allow various kinds of control strategies, such as hierarchical planning, reactive planning, and frame-based dialogue control. They also need to execute multiple tasks in parallel such as moving and engaging in dialogues when possible. Moreover, they must be able to handle human interruptions to the action and utterances the robots are executing. Although a number of models for conversational robots have been proposed so far, no model has all of these properties.

This paper presents a novel model for the behavior and dialogue controller in conversational service robots. It is called **RIME** (Robot Intelligence based on Multiple Experts). RIME utilizes components called *experts*, which are specialized for performing certain kinds of tasks by performing physical actions and engaging in dialogues in certain domains. The task that each expert can execute may be simple, but RIME-based robots can execute complicated tasks, which may include both physical behavior and spoken dialogues, by sequentially activating experts.

The rest of the paper is organized as follows. Section 2 describes requirements for the behavior and dialogue controller in conversational service robots. Then we mention previous work in Section 3. Next we describe our ideas in Section 4 and explain the details of RIME in Section 5. Then Sections 6 and 7 respectively describe the current implementation of RIME and present a process example of an application system. Section 8 discusses the similarities and differences between RIME and previous robot intelligence models before concluding this paper by mentioning future work in Section 9.

2 Requirements for Conversational Robot Intelligence Model

There are many features that need to be achieved for building conversational robots working for various tasks. Among them, this paper focuses on achieving the following features, which we believe are crucial for the better usability of conversational robots.

- *Integration of dialogues and physical actions*
  Robots need to execute tasks by integrating dialogues and physical actions. For example, to explain an object, a robot must be able to go close to the object and then verbally explain the object through dialogue.
- *Handling multiple task domains*
  Since robots are usually expected to perform multiple kinds of tasks, they need to work in multiple domains and switch domains according to user utterances. In addition, even while engaging in a dialogue in one domain, robots have to switch the domain when humans want to change to another task. If this is not possible, it will be stuck in one domain, and the dialogue cannot proceed.
- *Interruption handling*
  It is especially crucial for human-robot interaction to be able to handle users' interrupting utterances while speaking or performing tasks. This is because robots are expected to be human-friendly and the constraint that turn-taking must be orderly is not desirable.
- *Parallel task execution*
  Robots are expected to be able to execute multiple tasks in parallel when possible. For example, robots should be able to engage in a dialogue while moving.
- *Extensibility*
  Since robots can be used for a variety of tasks, various dialogue and task planning strategies should be able to be incorporated. For example, it should be possible to employ frame-based dialogue control for some kinds of tasks and plan-based dialogue control for other kinds of tasks.

Building an extensible intelligence model achieving all of the above-mentioned properties is not simple. For example, the robot must determine whether a user utterance is either an interruption to the robot's utterance or physical action, a new utterance in the current dialogue domain, a request to move to another dialogue domain, or an attempt to initiate a new dialogue while the agent is performing a physical action.

Note that we focus only on human utterances as inputs, and that we do not deal with other kinds of inputs such as gesture recognition results and emotion estimation results, although we consider that dealing with such inputs is also important.

3 Previous Work

Although a number of models for conversational robots, agents, and spoken dialogue systems have been proposed, and they satisfy some of the above requirements. However, no model has all of the above properties.

As for the integration of dialogues and physical actions, some work tried to incorporate spoken dialogue system technology and service robots [2–7]. They combined simple speech understanding and standard dialogue management functions with service robots, so their systems separately plan dialogues and physical behaviors, and it is not easy to execute tasks by integrating dialogues and physical actions.

Handling multiple task domains has been tried in the context of spoken dialogue systems. Lin et al. [8] proposed an architecture of multi-domain spoken dialogue system, which employs distributed modules called agents, each of which is responsible for engaging in dialogue in a certain domain. A user utterance is classified into one of the domains and it is handled by the agent in the domain. Hartikainen et al. also presents system architecture for handling multiple domains [9]. O'Neill et al. [10] proposed to incorporate object-oriented programming framework to facilitate to build such agents (They call them *experts*). Since these pieces of work are not intended to be incorporated into robots, they do not provide ways to integrate physical behavior controllers and handle interruptions.

The WITAS system [11] can manage dialogues in multiple domains by maintaining data called a dialogue move tree. Although it can dynamically switch dialogue domains based on human utterances, the dialogue state in each domain needs to be represented in the dialogue move trees, thus, it is not easy to incorporate a variety of tasks which may require very different control strategies and internal state representations.

A robot named *jijo-2* [12] can perform tasks that require physical actions such as delivery as well as engage in task-oriented dialogues in several task domains such as telling office members' current locations and route directions. It can switch dialogue domains and stop navigation based on human utterances. Although it achieves high functionality in robot conversation, it has limitations in that the dialogue management strategies are fixed and it is not easy to add various kinds of dialogue domains and tasks requiring physical behaviors.

4 Basic Ideas

This section describes the basic ideas to build a dialogue and behavior controller that satisfies the requirements listed in Section 2. Our approach is to combine systems dedicated to small tasks. To combine them, we extract the symbol-level dialogue and behavior controller in such systems as *experts*, and employ modules for coordinating them. Other modules, such as speech recognizers, speech synthesizers, and hardware controllers are shared by those experts. A small number of experts are activated at the same time and are responsible for understanding human utterances and selecting actions. Such experts are called *being in charge*. Each expert has knowledge for utterance understanding and action selection.

This approach makes it possible to achieve the requirements. Integration of dialogue and physical actions is possible by sequentially activating experts dedicated for dialogue tasks and physical behavior tasks. Engaging in multi-domain dialogues becomes possible by employing experts for dialogues in different domains. Dialogue domains can be dynamically switched by changing the dialogue expert being in charge according to the human utterance understanding result. Interruption utterances are handled by the expert that selected the action being executed. Each expert needs to have knowledge on what type of utterances are interruptions and how to handle interruptions. Parallel task execution is possible by allowing multiple experts to take charge at the same time. By checking the properties in the experts, the system can determine which two experts cannot take charge at the same time. For example, we can design experts so that if two experts have "physical" property, they do not take charge at the same time (This will be described in Section 5.3.2). Finally, it is extensible because any kind of expert can be employed if they have a small number of mandatory accessing methods, which will be explained in Section 5.3.3.

This approach shares the fundamental ideas with the multi-domain spoken dialogue system models (e.g., [10,8]), but it is extended in many aspects, for interruption handling and parallel task executions.

5 RIME: Dialogue and Behavior Control Utilizing Multiple Experts

This section describes the details of RIME, our model for conversational service robots.

5.1 Robot Intelligence Architecture

Conversational service robots are dedicated to performing service tasks in a specific environment such as in a house and in an office. For example, they are supposed to clean up rooms, collect garbage, and provide weather information. They must be able to engage in dialogues with humans to understand their requests to perform tasks and provide them with some necessary information.

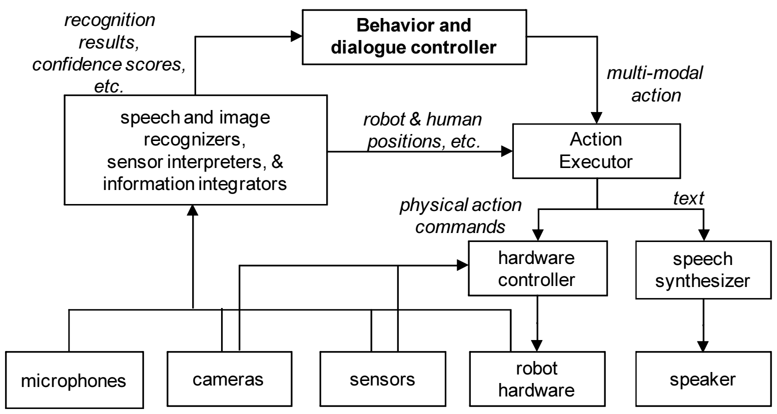
recognition results, confidence scores, etc.
Behavior and dialogue controller
multi-modal action
speech and image recognizers, sensor interpreters, & information integrators
robot & human positions, etc.
Action Executor
physical action commands
text
hardware controller
speech synthesizer
microphones
cameras
sensors
robot hardware
speaker

Fig. 1. Architecture for Conversational Service Robots

Fig. 1 depicts our underlying architecture for robot software. At the top level is the behavior and dialogue control subsystem, on which this paper focuses. It receives recognition results and their confidence scores from several kinds of recognizers, such as speech recognizers, image recognizers, multimodal recognizers, and other sensor interpreters. Multiple speech recognizers can be employed for recognizing the same speech with different acoustic and language models. At appropriate times, it outputs multimodal actions which may include texts to speak (e.g. "hello, I am a robot") and symbolic representations of physical actions (e.g. "`gesture hello`" and "`approach john`"). They can also be commands to stop a physical action being executed or utterance being spoken. They are sent to the action executor which controls modules such as the hardware controller and the speech synthesizer. The action executor can execute more than one action at one time if they do not conflict with each other. For example, while speaking, if it receives a new action containing only physical action, it can start executing the physical action. This enables parallel task execution described later. The action executor returns the execution report to the behavior and dialogue control subsystem when the execution of one action finishes.

5.2 Overview of RIME

RIME is a model for the dialogue and behavior controller. As mentioned earlier, RIME employs modules called experts which are specialized for performing certain kinds of tasks by performing physical actions and engaging in dialogues in certain domains. An expert is a kind of object in the object-oriented programming framework. Each expert has its own internal state, which includes the results of user intention understanding, the dialogue history, and status of task execution. It has methods to access its internal states, such as one for understanding user utterances and one for selecting robot actions. Here, an *action* is a multimodal robot command which includes a text to speak and/or a physical action command. Methods of experts can access a common database called *global context* to store and

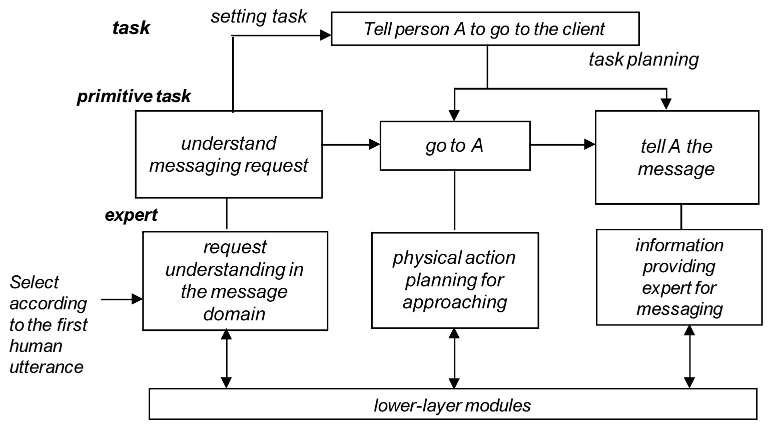
task
setting task
Tell person A to go to the client
task planning
primitive task
understand messaging request
go to A
tell A the message
expert
Select according to the first human utterance
request understanding in the message domain
physical action planning for approaching
information providing expert for messaging
lower-layer modules

Fig. 2. Change in Primitive Tasks and Experts in the Example

utilize information across domains.

In this paper, we call tasks performed by one expert *primitive tasks*. Experts should be prepared for each primitive task type. For example, if there is an expert for a primitive task type *telling someone's extension number*, *telling person A's extension number* is a primitive task. By performing a series of primitive tasks, a complicated task can be performed.

Let us consider an example that person B asks the robot to call A (Fig. 2). When B speaks to the robot, all the experts try to understand the utterance. Based on those results, an expert for understanding messaging becomes in charge. It controls the dialogue with the human unless he/she changes the topic. After some exchanges of utterances, it understands B's request which is set as the goal. Then it plans a primitive task sequence, approaching A, and telling A that he/she is being called. Then an expert for moving becomes in charge. It not only performs moving but also accepts human utterances. If the person who asked to call A tells the robot to cancel the call, it stops moving and goes back. When the robot reaches A, an expert for providing messages becomes in charge and it tells A that he/she is being called by B.

A primitive task type can be a different notion from a dialogue domain [10,8]. For example, understanding a request for telling an extension number and telling the requested information number can be different primitive task types, while they would be treated in the same domain in the previous multi-domain dialogue models. This is because request understanding and information providing require different dialogue control strategies [13].

There are other modules for coordinating these experts. The *understander* is connected to speech recognizers and dispatches the speech recognition results to each expert and selects the most appropriate expert to handle the speech recognition results. The *action selector* asks the selected expert to decide actions to perform, then it sends the actions to the action executor, from which it receives the execution re-

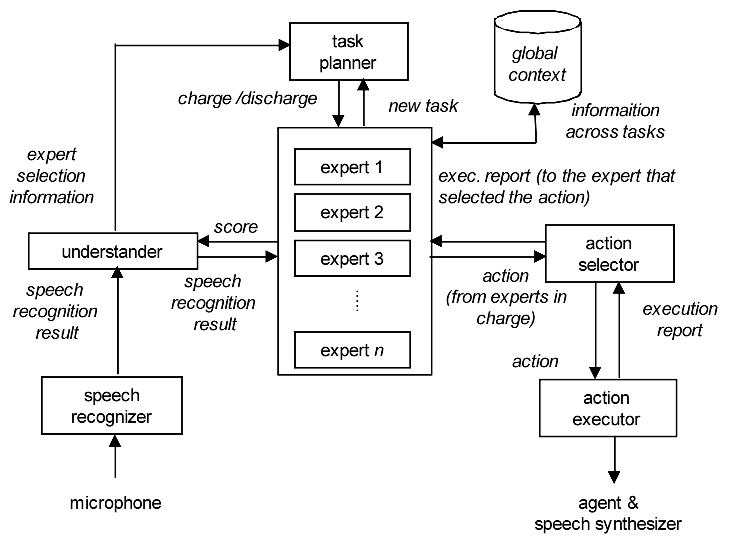
task planner
global context
charge /discharge
new task
informaition across tasks
expert selection information
expert 1
expert 2
expert 3
expert n
exec. report (to the expert that selected the action)
score
understander
action selector
speech recognition result
speech recognition result
action (from experts in charge)
execution report
action
speech recognizer
action executor
microphone
agent & speech synthesizer

Fig. 3. *Modules in RIME*

ports. The *task planner* receives tasks requested by users from experts, and decides which expert should take charge when the robot is performing a task. Fig. 3 depicts the module architecture of RIME.

Conversational robots can be developed based on RIME by creating experts and a task planner, then preparing speech recognizers and an action executor.

5.3 Experts

5.3.1 Internal State

Each expert holds information on the progress of the primitive task. They are classified into task-type-independent information and task-type-dependent information. Task-type-independent information includes information such as which action in this primitive task is being performed, and whether the previous robot action finished. The contents and the data structure for the task-type-dependent information, such as the user intention understanding results and the dialogue history, can be determined by the system developer and they can be changed depending on the type of primitive tasks.

5.3.2 Expert Classification

Experts are classified in two ways. First there are *system-initiative task experts* and *user-initiative task experts*. In this paper, *initiative* of a task means who can initiate the task. For example, the task "understanding a request for weather initiative" is a user-initiative task, and the task "providing weather information" is an system-initiative task.

Second, in RIME, executing multiple tasks in parallel becomes possible by making multiple experts take charge. Experts therefore need to have features for deciding which experts can be in charge simultaneously. For example, if there are two features *verbal* and *physical*, which indicate that the expert execute tasks respectively by dialogue and physical actions, two experts both having the *verbal* feature cannot take charge simultaneously.

5.3.3 Interface of Experts

Each expert must support several methods for accessing its internal state. Those methods are also classified into task-type-dependent ones and task-type-independent ones. The task-type-independent methods are for accessing task-type-independent information mentioned above.

Below we explain some of the task-type-dependent methods, which can be implemented by system developers.

- The *initialize* method is called when the expert is first created.
- The *understand* method updates the information state based on the user speech recognition results, using domain-dependent sentence patterns for utterance understanding. This method returns a score which indicates the plausibility the user utterance should be dealt with by the expert. Since this paper focuses on the system architecture, not the detailed design of each expert, how each expert computes the score is out of the scope of this paper. However, we think that previously proposed domain selection methods are useful and can be incorporated into experts in our framework. For example, methods that use the results and scores of domain-dependent speech understanding [14,15] and methods that incorporate contextual information [16,10,17,18] are considered to be effective.
- The *unselected-hook* method cancels the change in its internal state caused by the previous call of the *understand* method when the expert is not selected.
- The *select-action* method outputs one action based on the content of the internal state. The action may be an empty action. Actions can have several kinds of properties concerning control. For example, the “wait” property means that the system waits for a human utterance after the execution of the action, and the “finish” property means that the primitive task can be considered finished after the action is executed. In the case of experts for request understanding, the understood request is sent to the task planner as a new task at the same time. If the action to return is a prompt utterance to the user, it has the “wait” property. If the expert’s primitive task is found completed, the action has the “finish” property.
- The *action-success-hook/action-failure-hook* methods are called when it receives the execution report of the action selected by this expert. This enables experts to work depending on actions’ success and failure.
- The *detect-interruption* method determines if the previous user utterance is an interruption to the action being performed when this expert is being in charge,

and the *handle-interruption* method returns the action to be performed after an interruption is detected.

5.4 Processes of Coordinating Modules

Here we explain the algorithms for the understander, the action selector, and the task planner.

5.4.1 Understander

Each time the understander receives a user speech recognition result, it performs the following process.

First it dispatches the speech recognition result to the experts in charge and the user-initiative experts with their *understand* methods, which then return the scores mentioned above. The expert that returns the highest score is selected as the expert to take charge. Next, it calls the *unselected-hook* methods of the experts that are not selected. If the selected expert is not in charge, it tells the task planner that the expert is selected as the user-initiative expert to take charge. If the selected expert is in charge, it calls the *detect-interruption* method of the expert. If *true* is returned, it raises a flag to indicate an interruption utterance was detected, and otherwise, it raises a flag to indicate a non-interruption utterance was detected.

The understander classifies user utterances into three categories using experts' knowledge: interruptions to the current action, new utterances in the current dialogue domain, and requests to move to another dialogue domain. The classification result is used by the action selector.

5.4.2 Action Selector

The action selector repeats the following process for each expert being in charge in a short cycle.

(1) If the expert's flag to indicate that an interruption was detected is up, it calls the expert's *handle-interruption* method, and it then sends the returned action to the action executor.
(2) Else if the expert detects the execution of an action finished, it does the following:
    (a) If the previous action is successfully finished, it calls the expert's *action-success-hook* method, otherwise, it calls its *action-failure-hook* method.
    (b) (1) If the previous action has the "wait" property, it raises the expert's flag to wait for a user utterance. (2) Else if the previous action has the "finish"

property, it tells the task planner that the expert finished its primitive task. (3) Otherwise, it calls the expert's *select-action* method, and then sends the returned action to the action executor.

(3) Else if an action of this expert is being performed, it does nothing.
(4) Else if the expert's flag to wait for a user utterance is up, it does the following: (1) If the expert's flag to indicate that a non-interruption utterance was detected is up, it calls the expert's *select-action* method, and sends the returned action to the action executor. (2) Else it does nothing.
(5) Else, it calls the expert's *select-action* method, and then sends the returned action to the action executor.

If a non-interruption user utterance in the domain of the expert in charge is found while an action of that expert is being performed, the understanding result in the expert is changed but the robot does not execute any action. Thus it is possible to change the way to react to a user utterance by distinguishing interruptions and non-interruptions. For example, let us consider the case where an expert for understanding requests in the weather information domain is in charge and the robot is speaking "Would you like to know tomorrow's weather for Tokyo?". If the human user says "no, today" while the robot is speaking and this matches an interruption utterance pattern, this will be regarded as an interruption and the *handle-interruption* method will be invoked. On the other hand, if he/she says "yes" and this does not match any of the interruption utterance patterns, this will be handled as a non-interruption utterance.

5.4.3 Task Planner

The task planner is responsible for deciding which experts take charge and which experts do not. It sometimes make an expert take charge by setting a primitive task, and sometimes it discharges an expert to cancel the execution of its primitive task.

To make such decisions, it receives from the other modules the following pieces of information.

- It receives from the understander information on which expert is selected to understand a new utterance.
- It receives from an expert being in charge information on the finish of the primitive task. It comes with information on success or failure and the reason for failure when the primitive task failed.
- It receives new tasks from the experts that understand human requests.

In addition, it can consult the global context to access the information shared by the experts and the task planner.

We assume that the algorithm of task planning is given by system developers. Although some efforts on building task planning modules that achieve tasks in dynamically changing environments (e.g. [19]), it is still difficult to build a planner which can automatically decide an optimal strategy.

Currently we employ an algorithm that uses an agenda to contain tasks to execute. It repeats the following process in a short cycle. (1) If it finds an expert that successfully finished its primitive task and there are remaining primitive tasks for the task, it tries to make an expert take charge for executing its subsequent primitive task. (2) If a new user-initiative expert is selected by the understander, it makes the expert take charge. At the same time, if some of the experts in charge have a conflict with the new expert, those are discharged. Here, it checks conflicts between experts using their features mentioned in Section 5.3.2. In addition, one expert cannot take charge for two tasks simultaneously. This is also considered as a conflict. (3) If the agenda is not empty, it decomposes a task in the agenda into primitive tasks, and makes the corresponding experts take charge if there is no conflicting expert in charge. In the above process, parallel task execution is made possible because more than one expert can take charge simultaneously.

The task planner needs to be further extended by the system developers by designing the decision strategies such as how to decompose a task into a primitive task sequence and which expert should take charge when two experts have conflicts.

6 Implementation

We have implemented RIME as a toolkit called RIME-TK for building a dialogue and behavior control subsystem. RIME-TK provides a hierarchy of expert templates [10] so that system developers can easily create new experts. For example, a template for the frame-based request understanding dialogue experts is provided. Experts can be implemented using Java. The current implementation provides templates of experts so that it becomes easy to create new experts in specific domains. For example, the templates for the frame-based dialogue management [20,21] are prepared for request understanding. The *understand* method in some templates employs finite-state-transducer-based language understanding. Developers of experts can configure the language understanding component by preparing a set of utterance patterns and keyword lists as in spoken dialogue system toolkits [22]. Below are examples.

```
utterance patterns:
  action-type: question-weather
    Tell me the weather in *city* *day*
    I'd like to know *day*'s weather in *city*
  action-type: refer-city
    It's *city*
keywords:
```

```
class: *day*
  today, tomorrow
class: *city*
  tokyo, kyoto
```

For language generation, a template-based generation mechanism is prepared.

The understander is connected to a speech recognizer Julius/Julian [23], which can employ either statistical or network-grammar-based language models, although it is possible to employ other speech recognizers and sensor interpreters.

The action selector is connected to an action executor which can control a text-to-speech system and a 120cm tall humanoid robot. For moving in the room where we demonstrate the system, the current system utilizes ultrasonic tags to locate humans, the humanoid robot, and obstacles such as tables and chairs [24]. For speech input and output, currently we use microphones and speakers directly connected to computers. For a text-to-speech system we use NTT-IT Corp.'s FineVoice.

The current version of RIME-TK provides a simple task planner template. It always adds a new task on the top of the agenda. When possible, it decomposes the first task into primitive tasks, and tries to sequentially execute those primitive tasks.

The difference between RIME-TK and previously built toolkits for building spoken and multimodal dialogue systems such as VoiceXML browsers [25], XISL [26], CSLU Toolkit [27], and SpeechBuilder [22] is that the dialogue strategy is fixed in systems built using these toolkits, while our toolkit provides functions to integrate different kinds of dialogue strategies. By creating experts having the same functionalities with these toolkits, we can build a RIME-based system that subsumes those toolkits.

7 Process Examples

Here we explain how a RIME-based robotic system works through example interactions between a user and an example robotic system. The robot is supposed to work at a reception, and can perform several small tasks such as providing extension numbers and guide guests to several places near the reception such as a meeting room and a restroom. Some experts in the system are listed in Table 1.

When a user says to the robot "where is the meeting room?" and the speech recognizer correctly recognizes it, the understander calls the *understand* method of the user-initiative experts with its speech recognition results. Each user-initiative expert tries to understand with their own task-dependent knowledge. It computes the score that the *understand* method returns using the hand-crafted rules that employ speech recognition confidences, dialogue context, and information on whether lan-

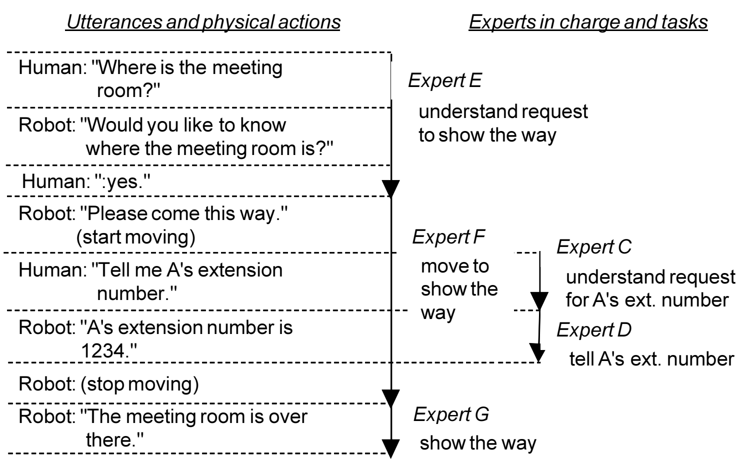
Utterances and physical actions
Experts in charge and tasks
Human: "Where is the meeting room?"
Robot: "Would you like to know where the meeting room is?"
Human: ":yes."
Robot: "Please come this way." (start moving)
Human: "Tell me A's extension number."
Robot: "A's extension number is 1234."
Robot: (stop moving)
Robot: "The meeting room is over there."
Expert E
understand request to show the way
Expert F
move to show the way
Expert C
understand request for A's ext. number
Expert D
tell A's ext. number
Expert G
show the way

Fig. 4. *Expert Selection in the Parallel Task Execution Example*

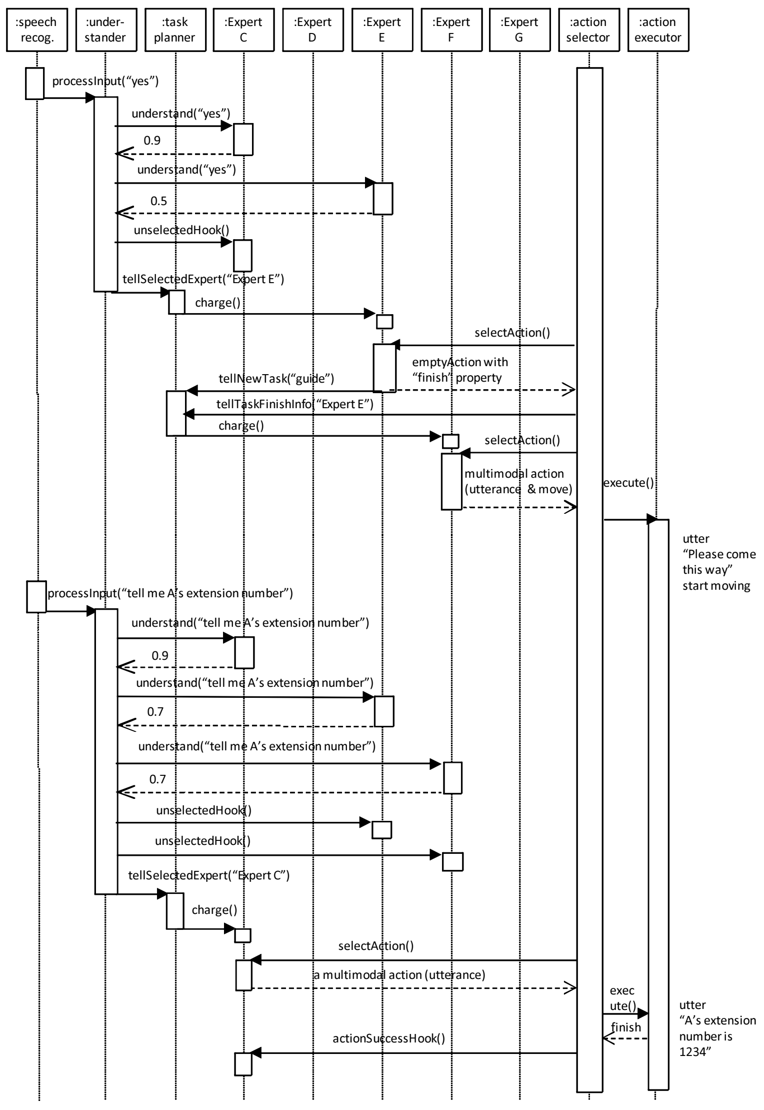
:speech recog.
:under-stander
:task planner
:Expert C
:Expert D
:Expert E
:Expert F
:Expert G
:action selector
:action executor
processInput("yes")
understand("yes")
0.9
understand("yes")
0.5
unselectedHook()
tellSelectedExpert("Expert E")
charge()
selectAction()
emptyAction with "finish" property
tellNewTask("guide")
tellTaskFinishInfo("Expert E")
charge()
selectAction()
multimodal action (utterance & move)
execute()
utter "Please come this way" start moving
processInput("tell me A's extension number")
understand("tell me A's extension number")
0.9
understand("tell me A's extension number")
0.7
understand("tell me A's extension number")
0.7
unselectedHook()
unselectedHook()
tellSelectedExpert("Expert C")
charge()
selectAction()
a multimodal action (utterance)
exec ute()
finish
utter "A's extension number is 1234"
actionSuccessHook()

Fig. 5. *Sequence Diagram for the Process Example*

Table 1
Experts in the Example Robotic System

ID	task type	initiative	feature
A	understanding weather information requests	user	verbal
B	providing weather information	system	verbal
C	understanding extension number requests	user	verbal
D	providing extension numbers	system	verbal
E	understanding requests for guiding to places	user	verbal
F	moving to show the way	system	physical
G	explaining places	system	verbal

guage understanding results are obtained. We also plan to incorporate a machine-learning-based domain estimation method [17,18]. Since this utterance matches one of the sentence patterns in *Expert E* in Table 1, it returns the highest score, and it becomes the expert to take charge. Then the task planner makes the expert take charge. The action selector calls *Expert E*'s *select-action* method. If the confidence of the recognition result is not high enough, it returns an action to say "would you like to know where the meeting room is?" with "wait" property to the action selector. The action selector sends this action to the action executor, and when it receives the report on the action's success, it raises *Expert E*'s flag to wait for a user utterance. Then the user says "yes" which is correctly recognized, the recognition result is sent to all user-initiative task experts and the expert in charge. This matches all experts' utterance patterns, but the expert in charge returns the highest score. Then the action selector calls the expert's *select-action* method. Since the understanding result is confirmed, the expert sends a new task to guide the user to the meeting room, and returns an empty action with "finish" property. The task planner decomposes the task into two primitive tasks, moving to show the way to the meeting room, and telling where the meeting room is, and makes *Expert F* take charge. Then the action selector calls its *select-action* method, which then returns a physical action to move to show the way to the meeting room. Thus, the robot can understand user requests and execute requested tasks.

While the robot is moving to show the way to the meeting room, if the user says "stop", this is sent to all user-initiative task experts and the experts in charge, that is, *Expert F*. This utterance matches one of the patterns in *Expert F*, and it matches one of its patterns of interruptions to the action being performed. Then the understander raises a flag to indicate that an interruption utterance has been detected. Then the action selector calls the expert's *handle-interruption* method. It tells the task planner to cancel the current task, and returns a "stop" action to the action selector. Then the robot moving is interrupted. Even if the action being performed is not a physical action but an utterance, the process for handling interruptions would be the same.

If the user utterance is not “stop” but “tell me person A’s extension number”, *Expert C* is selected to take charge. The task planner checks if this conflicts with the current expert, *Expert F*. Since there is no conflict, *Expert C* also takes charge. The action selector accesses both experts and executes two tasks in parallel. Fig. 4 illustrates this process. In addition, Fig. 5 is a sequence diagram for the portion of this process (from human utterance “yes” to robot utterance “A’s extension number is 1234.”). It shows the communications among the processes and experts.

The above examples show that RIME-based systems can handle interruptions and execute tasks in parallel. Through the development of several systems based on RIME, we confirmed that the algorithms for the modules in RIME work correctly. For example, we have built a robotic system that understands requests for doing a presentation and performs it using a projector screen [28] and we have a system that can engage in both task-oriented dialogues and non-task oriented dialogues [29]. We have also developed a RIME-based system that can learn location names by building an expert that can learn a new name and remember it together with the physical location information [30].

8 Discussion

RIME can be considered as a kind of multi-agent-based architecture for asynchronous control [31,11,32], since several modules running in parallel communicate with each other. Indeed, RIME can be implemented using a distributed software development platform such as the Open-Agent Architecture as [11]. However, RIME is different from previous architecture in that it employs distributed task-dependent experts that contain task execution information. This makes the control structure clear and extensible. Here we compare our model with the previous architecture for robot intelligence.

First, multi-agent architecture employs distributed agents working in parallel. Since RIME employs several modules such as the understanding module and the action selection module working in parallel, it can be considered as one of the multi-agent models. Experts can also be built as different process modules. However, previously built robot intelligence models based on the multi-agent model controlled physical actions and dialogues in different agents. Such systems need a lot of communication between agents when the task to execute needs both physical actions and dialogues, and it is not easy to design the knowledge for executing such tasks. Since experts in RIME can have all of the required knowledge for executing tasks, it is easy to design it.

It is also worth comparing RIME with hybrid reactive/deliberative architecture (e.g., [33,34]), which employs multiple levels of planning. As mentioned earlier, the dialogue and behavior control subsystem, which RIME models, is at the top level of the robot intelligence architecture. In addition, RIME has two levels, the task planning level and the action selection level. Therefore our robot intelligence architecture has several levels. However, RIME does not limit the planning strategies for each expert and task planning. Both deliberative planning and reactive planning as well as other types of planning can be employed. Rather, our focus is put on how to coordinate different types of planning.

It is possible to consider a model that has only task experts, which outputs an action according to their degrees of activation, and does not have any coordination modules such as the task planner and the action selector in RIME. It is not easy, however, to design such experts and the activation strategy so that the resulting agent achieve complicated tasks.

9 Concluding Remarks

This paper presented RIME, a model for conversational robot development. It facilitates building conversational robots that achieve various crucial features, such as interruption handling and parallel task execution. It is implemented as a toolkit called RIME-TK, with which developers can build robots that can engage in a variety of tasks by implementing experts utilizing various kinds of spoken language technologies that conform to the RIME interface.

Since this paper focuses on not technologies to improve the performances of conversational robots but their model, it does not provide performance evaluation results of experimental application systems. The performance of the application systems can be improved without basing them on RIME. Establishing simple and modular models leads to reducing the cost of application development and scaling up the complexity of the application system. The contribution of this paper is providing one of such models so that many application developers can refer to it.

Developing RIME-based robotic systems requires expertise in spoken dialogue technology, although several templates for experts and task planners are prepared. Future work includes improving RIME-TK so that non-experts can create experts and configure the task planner.

One issue we need to consider when we build a RIME-based system is the granularity of experts, that is, how we design a set of primitive task types. If we employ very small task types as primitive task types, task planning will be complicated, and if we employ large and complicated task types as primitive task types, it will be difficult to execute multiple tasks in parallel. Currently we do not have a clear criterion, and establishing it is a future work.

The algorithms for dialogue management and task planning used in the experts in our current implementation are simple. It would be effective to incorporate recently developed techniques such as reinforcement-learning-based dialogue management [35] and strategies for avoiding unnecessary confirmation requests [36]. We need, however, to make it clear how to incorporate them into conversational service robotic systems which are more complicated than single domain spoken dialogue systems. We as well need to explore a systematic way to construct rules for task domain selection and rules for out-of-domain utterance detection. Our future work also includes enabling a robot to schedule tasks when humans ask it multiple tasks.

Developing a dialogue and behavior controller of conversational robots and agents that work for any kind of task in real environments is still far from our current technology. However, RIME makes it possible to easily expand the variation of tasks by combining experts. We believe that this is a crucial step in improving conversational robot/agent technology.

References

[1] M. Nakano, K. Funakoshi, Y. Hasegawa, H. Tsujino, A framework for building conversational agents based on a multi-expert model, in: Proceedings of the 9th ACL/ISCA SIGdial Workshop on Discourse and Dialogue (SIGdial), 2008, pp. 88–91.

[2] Y. Makihara, M. Takizawa, K. Ninokata, Y. Shirai, J. Miura, N. Shimada, A service robot acting by occasional dialog – object recognition using dialog with user and sensor-based manipulation –, Journal of Robotics and Mechatronics 14 (2) (2002) 124–132.

[3] T. Yoshimi, N. Matsuhira, K. Suzuki, D. Yamamoto, F. Ozaki, J. Hirokawa, H. Ogawa, Development of a concept model of a robotic information home appliance, ApriAlpha, in: Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2004, pp. 205–211.

[4] M. Zobel, J. Denzler, B. Heigl, E. Noth, D. Paulus, J. Schmidt, G. Stemmer, MOBSY: integration of vision and dialogue in service robots, in: Proceedings of the Second International Workshop on Computer Vision Systems (ICVS), 2001, pp. 50–62.

[5] T. Konashi, M. Suzuki, A. Ito, S. Makino, A spoken dialog system based on automatic grammar generation and template-based weighting for autonomous mobile robots, in: Proceedings of the Eighth International Conference on Spoken Language Processing (Interspeech 2004 – ICSLP), 2004, pp. 189–192.

[6] E. Topp, D. Kragic, P. Jensfelt, H. Christensen, An interactive interface for service robots, in: Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA), 2004, pp. 3469–3474.

[7] L. S. L. Marcelo Quinderé, A. J. S. Teixeira, An information state based dialogue manager for a mobile robot, in: Proceedings of the 10th European Conference on Speech Communication and Technology (Interspeech 2007–Eurospeech), 2007, pp. 162–165.

[8] B. Lin, H. Wang, L. Lee, Consistent dialogue across concurrent topics based on an expert system model, in: Proceedings of the 6th European Conference on Speech Communication and Technology (Eurospeech-99), 1999, pp. 1427–1430.

[9] M. Hartikainen, M. Turunen, J. Hakulinen, E.-P. Salonen, J. A. Funk, Flexible dialogue management using distributed and dynamic dialogue control, in: Proceedings of the Eighth International Conference on Spoken Language Processing (Interspeech 2004 – ICSLP), 2004, pp. 197–200.

[10] I. O'Neill, P. Hanna, X. Liu, M. McTear, Cross domain dialogue modelling: an object-based approach, in: Proceedings of the Eighth International Conference on Spoken Language Processing (Interspeech 2004 – ICSLP), 2004, pp. 205–208.

[11] O. Lemon, A. Gruenstein, A. Battle, S. Peters, Multi-tasking and collaborative activities in dialogue systems, in: Proceedings of the Third ACL/ISCA SIGdial Workshop on Discourse and Dialogue (SIGdial), 2002, pp. 113–124.

[12] H. Asoh, Y. Motomura, F. Asano, I. Hara, S. Hayamizu, K. Itou, T. Kurita, T. Matsui, N. Vlassis, R. Bunschoten, B. Kroese, Jijo-2: An office robot that communicates and learns, IEEE Intelligent Systems 16 (5) (2001) 46–55.

[13] M. Nakano, K. Dohsaka, N. Miyazaki, J. Hirasawa, M. Tamoto, M. Kawamori, A. Sugiyama, T. Kawabata, Handling rich turn-taking in spoken dialogue systems, in: Proceedings of the 6th European Conference on Speech Communication and Technology (Eurospeech-99), 1999, pp. 1167–1170.

[14] T. Isobe, S. Hayakawa, H. Murao, T. Mizutani, K. Takeda, F. Itakura, A study on domain recognition of spoken dialogue systems, in: Proceedings of the 8th European Conference on Speech Communication and Technology (Interspeech 2003–Eurospeech), 2003, pp. 1889–1892.

[15] I. R. Lane, T. Kawahara, T. Matsui, S. Nakamura, Topic classification and verification modeling for out-of-domain utterance detection, in: Proceedings of the Eighth International Conference on Spoken Language Processing (Interspeech 2004 – ICSLP), 2004, pp. 2197–2200.

[16] B. Lin, H. Wang, L. Lee, A distributed agent architecture for intelligent multi-domain spoken dialogue systems, IEICE Trans. on Information and Systems E84-D (9) (2001) 1217–1230.

[17] K. Komatani, K. Tanaka, H. Kashima, T. Kawahara, Domain-independent spoken dialogue platform using key-phrase spotting on combined language model, in: Proceedings of the 7th European Conference on Speech Communication and Technology (Interspeech 2001–Eurospeech), 2001, pp. 1319–1322.

[18] S. Ikeda, K. Komatani, T. Ogata, H. G. Okuno, Integrating topic estimation and dialogue history for domain selection in multi-domain spoken dialogue systems, in:

Proceeding of the 21st International Conference on Industrial, Engineering and Other Applications of Applied Intelligence Systems (IEA/AIE-2008), LNAI5027,, 2008, pp. 294–304.

[19] M. Beetz, Structured reactive controllers, Autonomous Agents and Multi-Agent Systems 4 (2001) 25–55.

[20] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, S. Busayapongchai, A form-based dialogue manager for spoken language applications, in: Proceedings of the Forth International Conference on Spoken Language Processing (ICSLP-96), 1996, pp. 701–704.

[21] J. Chu-Carroll, MIMIC: An adaptive mixed initiative spoken dialogue system for information queries, in: Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP-00), 2000, pp. 97–104.

[22] J. Glass, E. Weinstein, S. Cyphers, J. Polifroni, G. Chung, M. Nakano, A framework for developing conversational user interfaces, in: Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces (CADUI-04), 2004, pp. 354–365.

[23] A. Lee, T. Kawahara, K. Shikano, Julius – an open source real-time large vocabulary recognition engine, in: Proceedings of the 7th European Conference on Speech Communication and Technology (Interspeech 2001–Eurospeech), 2001, pp. 1691–1694.

[24] Y. Nishida, H. Aizawa, T. Hori, N. Hoffman, T. Kanade, M. Kakikura, 3D ultrasonic tagging system for observing human activity, in: Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2003, pp. 785–791.

[25] W3C, Voice extensible markup language (voicexml) version 2.0, W3C Recommendation (2004).

[26] K. Katsurada, Y. Nakamura, H. Yamada, T. Nitta, XISL: A language for describing multimodal interaction scenarios, in: Proceedings of the Fifth International Conference on Multimodal Interfaces (ICMI-03), 2003, pp. 281–284.

[27] S. Sutton, R. A. Cole, J. de Villiers, J. Schalkwyk, P. Vermeulen, M. W. Macon, Y. Yan, E. Kaiser, B. Rundle, K. Shobaki, P. Hosom, A. Kain, J. Wouters, D. W. Massaro, M. Cohen, Universal speech tools: The CSLU toolkit, in: Proceedings of the Fifth International Conference on Spoken Language Processing (ICSLP-98), 1998, pp. 3221–3224.

[28] Y. Nishimura, S. Minotsu, H. Dohi, M. Ishizuka, M. Nakano, K. Funakoshi, J. Takeuchi, Y. Hasegawa, H. Tsujino, A markup language for describing interactive humanoid robot presentations, in: Proceedings of the 2007 International Conference on Intelligent User Interface (IUI-07), 2007, pp. 333–336.

[29] M. Nakano, A. Hoshino, J. Takeuchi, Y. Hasegawa, T. Torii, K. Nakadai, K. Kato, H. Tsujino, A robot that can engage in both task-oriented and non-task-oriented dialogues, in: Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots (Humanoids-2006), 2006, pp. 404–411.

[30] K. Funakoshi, M. Nakano, T. Torii, Y. Hasegawa, H. Tsujino, N. Kimura, N. Iwahashi, Robust acquisition and recognition of spoken location names by domestic robots, in: Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2007, pp. 1435–1440.

[31] J. Boye, B. A. Hockey, M. Rayner, Asynchronous dialogue management: Two case-studies, in: Proceedings of 4th Workshop on the semantics and pragmatics of dialogue (Götalog-2000), 2000.

[32] N. Blaylock, J. Allen, G. Ferguson, Synchronization in an asynchronous agent-based architecture for dialogue systems, in: Proceedings of the Third ACL/ISCA SIGdial Workshop on Discourse and Dialogue (SIGdial), 2002, pp. 1–10.

[33] E. Gat, Integrating planning and reaction in a heterogeneous asynchronous architecture for controlling mobile robots, in: Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92), 1992, pp. 809–815.

[34] T. Hasegawa, Y. I. Nakano, T. Kato, A collaborative dialogue model based on interaction between reactivity and deliberation, in: Proceedings of the First International Conference on Autonomous Agents (Agents-97), 1997, pp. 75–82.

[35] S. Singh, D. Litman, M. Kearns, M. Walker, Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system, Journal of Artifical Intelligence Research 16 (2002) 105–133.

[36] K. Dohsaka, N. Yasuda, K. Aikawa, Efficient spoken dialogue control depending on the speech recognition rate and system's database, in: Proceedings of the 8th European Conference on Speech Communication and Technology (Interspeech 2003–Eurospeech), 2003, pp. 657–660.