

System Architecture to Realize Widely Applicable and Interactive Behavior of the Robot

Yosuke MATSUSAKA, Tetsunori KOBAYASHI

Department of Electrical, Electronics and Computer Engineering, Waseda University

Abstract

We designed a system architecture and an environment that enables the development of interactive robots under the open development process. In the proposed architecture, an individual developer develops each module by his own decision and the total system is developed by the cooperation of these developers. To realize smooth collaboration of these developers, we provide an environment to support the proper activity of the developers. Here, inter-module cooperation architecture is required to avoid the confliction between the modules developed by each developer. Selection of the appropriate modules from the developed modules should be executed instantly, but at the same time the selection should be best situated to the tasks. To realize both interactive and reasonable behavior of the robot, we introduce a priority based cooperation mechanism for modules.

The priority for each module under the various situations is described in a Situated-Priority Description Script (SPDS). Flexible module selection is realized by the modification of the SPDS under the various situations and the tasks. We also evaluate the efficiency of proposed architecture through the development of the multi-modal conversation robot.

1. Introduction

The need for human-symbiotic robots has been recognized recently and several developmental projects for robots that assists in every-day tasks are currently in progress [1]. Such robot requires the ability to contend with the variety of environments and hence requires a significant number of interrelated functions. Thereby a methodology and an environment are required to make them highly integrated widely applicable system.

To realize a large-scale integrated robot system, experts of various different fields are required to develop modules that implement the functions of the robot. Therefore, it is almost impossible to develop the entire robot system by only one developer, and collaboration between engineers, designers and other specialists is essential for the development process.

Most conventional approaches for the development of robots are based on the top-down design flow model; the chief engineer analyzes the problem and designs the

global system framework, then other engineers develop the detailed design of modules to satisfy the specifications provided by the chief engineer. However, it is sometimes desirable that each component (or module) of the system is designed and developed in-dependently, but the behavior of the combined system is often too complicated for the chief engineer to anticipate all of the problems. In such cases, the design process should be performed in parallel to assure flexibility. The module developers must analyze the problems from the viewpoint of their own specialties. We call the former development model the cathedral-type development model, and the later the bazaar-type model, according to the controversy of the open source development models [2]. In the bazaar-like development of robot system, execution of each task by the robot is realized by combining numerous functional modules that developed by individual developers. Under the circumstance, designing process of the tasks by combining developed modules become important in addition to development of each functional module. To simplify the designing process of tasks, a software architecture that realizes the cooperation of developed modules is required.

The purpose of this paper is to develop a software architecture that realizes the cooperation mechanism of individually developed modules by introducing a priority to each module depending on various situations. This software architecture is based on our previously proposed architecture [8].

The efficiency and applicability of the proposed architecture is verified through the development of the multi-modal conversation robot.

2. Recently Proposed Methodologies for Robot Development

Brooks [3] has proposed the methodology to develop the interactive behavior of the robot. In the proposed architecture, locomotion of the 6-legged robot realized, by developing several layers. Each layer has input and output in each and the behavior are generated by local decisions of each layer. Local decisions of each layers has the priority to be selected to be a real behavior. The crucial behavior (e.g. "recover its balance") has the higher priority, while the additional behavior (e.g. "move toward destination") has the lower priority, thus can

successfully combine the reasonable behavior and the immediate behavior. But the system developed using this methodology has the difficulty to expand the versatility, therefore the methodology to develop an more widely applicable layered system is eagerly discussed [4][5].

Another approach to realize the widely applicable robot system is multi-agent cooperation approach. There have been many discussions about inter-module cooperation methods (e.g. [6]). Some effort to apply multi-agent cooperation approach to robotic system is done by Motomura [7]. Motomura named a module that designed to deal with particular situation as situated agent. In his method, the developer will prepare the various situated agents depending on various situations, and connect those agents by Bayesian network. The selection of each situated agent is determined by maximizing expected utility that is utility of each agent's behavior multiplied by probability of each situation. In the proposed method, the utility of each situated agent's behavior can be determined externally, but final selection of agent under particular situation is stochastically determined. One of the problems of these methodologies is that, there exist iteration to calculate the utility function, thus exists certain delay to determine the best-situated module.

To successively combine reasonable, interactive and applicable behavior of the robot, we propose an architecture that has two phases to determine the behavior of the robot. One phase is the layered selection of the modules and the other phase is the selection by estimation of the situation. In the proposed architecture, the server process estimates the proper situation from the situation description given by the developer, and applies the priority based behavior selection in each of the situation. We introduce a Situated-Priority Description Script (SPDS) and implement the servers to realize the above architecture. And also we provide an environment to realize an open development of each module. Discussion about the environment that realizes open development of integrated modules is described in section 3 and discussion about situated priority based behavior selection is described in section 4.

3. Architecture to Realize the Open Development Process of Integrated Modules

3.1. Basic Composition and Design Process

Before entering into detailed discussion, we should define terms used in this paper.

We define the word "Scope of Interest (SoI)" as the expertise of the module developer, and the word "Area of Profession (AoP)" as the fields in which a module developer possesses sufficient knowledge to provide new

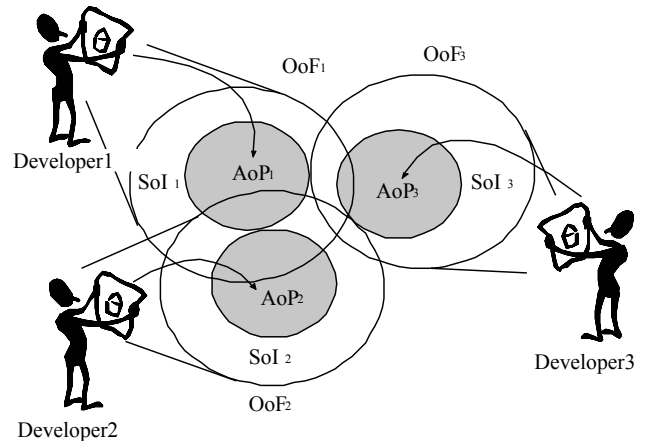


Fig. 1: Basic composition and design process

information for the development of module.

We will use the term "commitment" to refer to the act of disseminating information from one's AoP, and the term "collaboration" to refer to the commitment of information by a request from other developers.

A schematic of this process is presented in Fig. 1. The shared workspace called the blackboard is provided at the center of the system for free information sharing between developers.

The basic design process of module developers is as follows: A module developer obtains information from one's SoI field on blackboard, and implements the individual module that generates new information belonging to the one's AoP from the obtained information or from sensor data.

Then the developer prepares the environment to execute the module, and connect the module to the system to commit the new information generated by the module.

3.2. Requirements to Realize Open Development Process

Under the bazaar-type development model, developers from various field of expertise, such as speech recognition, acoustics processing, image processing, mechanical control and AI, implement functional modules from their AoP and SoI fields. The AoP and SoI fields for each developer are restricted, and it is almost impossible for the individual developer to understand the entire system precisely. Therefore, each module is developed by restricted information from AoP and SoI of the module developer.

To develop the total robot system by integrating such modules, the system is required to satisfy the requirements described as follows:

1. Easy integration of behavior of individually developed modules.
2. Realization of cooperative processing between modules according to tasks and various situation (or local environment).

In an interactive system like a multi-modal robot, new functional modules are frequently appended to the system by the module developers. Number of the tasks executed by the robot system increases continuously, and those tasks become more complicated. Therefore, the realization of inter-module cooperation mechanism under various situations will become an important issue of the development process.

4. Situated Priority Based Cooperation Mechanism

4.1. Priority-Based Module Selection

The system software should satisfy following conditions to realize inter-module cooperation mechanism.

1. The selection of module under particular situation or task is determined by the task designer.
2. The system is flexible and adaptable for the supplement of the new modules, tasks and situations.
3. Each module can acquire/abandon the execution authority by its autonomous decision.

We propose the software architecture that satisfies these conditions by introducing priority based cooperation mechanism for modules. Selection of modules under various situations is controlled according to the priority of each module, and only selected modules are provided execution authority.

The priority for each module is determined by the task designer according to the particular situations or tasks, and described in the Situated-Priority Description Script (SPDS). SPDS defines basic rules for cooperation of modules by describing priorities of modules under particular situations or tasks. When the selection of particular modules is required, the system software refers the priority of the modules from SPDS and selects the module of which the priority is highest.

To append new module to the robot system, task designer describes priority of the module and the situation that the module should be executed. Conflicts between modules are also resolved by selecting appropriate module under the situation by referring SPDS.

Also, autonomous acquisition/abandonment requests of execution authority from each module are available. For example, when a module whose priority is highest relinquishes the execution authority for some reason, the system software successively provides the execution authority for the module whose priority is the second

highest. This mechanism would enable the robot system to execute tasks continuously without delay.

4.2. Development of Modules and Situated Subtasks

The developer who designs the partial behaviors (or subtasks) of the robot system is denoted the task designer.

The task designer designs the subtasks of robot system by selecting and integrating the developed modules appropriately according to the particular situation.

For implementation of modules or design of tasks by utilizing inter-module cooperation mechanism, basic information of developed modules should be disclosed to all of the module developers and task designers.

For this aim, each developer is obligated to expose module information like below: name of module for identification, basic functionality of the module, an information (or data) required for the module-execution, an information (or data) that the module generates, another module(s) required for the module-execution.

Module developer implements new modules and task designer describes new SPDS by using the disclosed information.

5. System Architecture

The software architecture proposed in this paper consists of 3 parts: priority management service, situation observation service and data exhibition/message dispatch

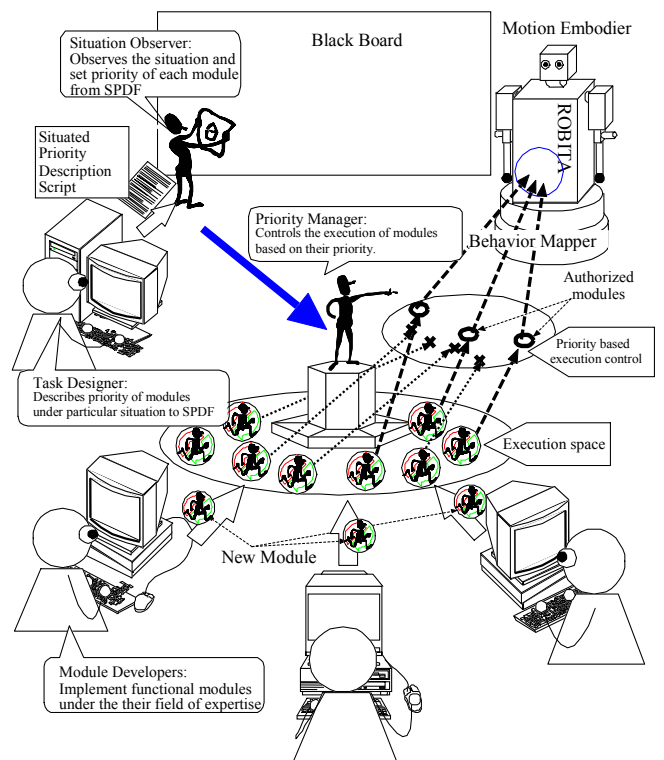


Fig. 2: Working process.

service.

5.1. Priority Management Service

Priority management service controls the selection and execution of modules according to the situation.

Execution control of each module is realized by introducing the mapping service between the data on the blackboard and the behavior of the robot. If the data is not mapped into the behavior, the module that provided the data can not directly exert any influence on the behavior of the robot. Therefore, mapping control between the data on the blackboard and the behavior of the robot system is substantially equivalent to the direct control of module execution.

Modules that requires the mapping to the robot system is registered with the priority management system. Selection among the registered modules is determined by integrating the module's priority and acquisition/abandon request of execution authority from each module.

Mapping request is checked from the module which priority is highest under the situation. If the module relinquishes the execution authority, the system successively checks the request of the module which priority is the second highest.

5.2. Situation Observation Service

The situation observation service observes the various data on blackboard, and infers the situation by combining value of those data. Data set to observe, correspondence of situation to value of data set, and priority of each module under each situation is described in SPDS.

The situation observation service refers the priority of modules under inferred situation from SPDS, and transmits the priority value to the priority management process.

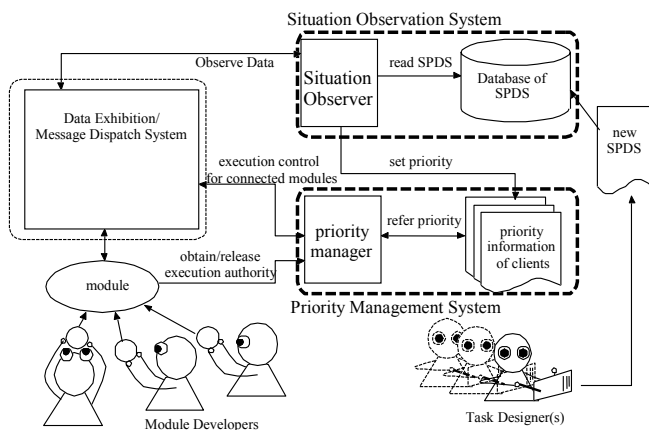


Fig. 3: Priority management service and situation observation service.

The inter-module cooperation is realized by the combination of the situation observation system and the priority management system (Fig. 3).

5.3. Data Exhibition/Message Dispatch Service

Information generated by any module must be open to all modules, including new ones. To realize open data-sharing framework, the blackboard model is adopted. The blackboard approach realizes equal access to data from any module in the system. Commonly, access to the specific area of blackboard is specified by addresses, but our system uses tags specify the area. In addition to this, a publish/subscribe service is introduced for the event notification framework [8]. Modules can view event by subscribing to the interested data in the publish/subscribe framework. The data exhibition system consists of two processes that provide data exhibition service based on the blackboard model and an event notification service based on the publish/subscribe model.

6. Implementation

6.1. System Composition

The robot has two cameras and two microphones on its head, two arms and omni-directional wheels on its foot. Seven client PCs and the data exhibition server were connected.

The inter-module cooperation service is executed on the same machine that the data exhibition server is executed. Module developers implement and execute each module on each PC as a client process.

6.2. Description Form of SPDS

The SPDS is described in XML format for convenience of describing by human and interpreting by computer. Description in XML format also enables easy modification and extension of data structure.

6.3. Basic Module Composition

Each module is a client that is connected to the data exhibition server via the BSD socket protocol. Each module gets necessary information from data exhibition server, and sets newly generated information to the server. Modules also send acquisition/abandon request of execution authority to priority management server by autonomous decision.

Class libraries were prepared for high-level programming languages to conceal the low-level system-calls. We currently have libraries for C++, Java and Perl. The APIs exhibit minimal language dependency.

7. Current Operation Status

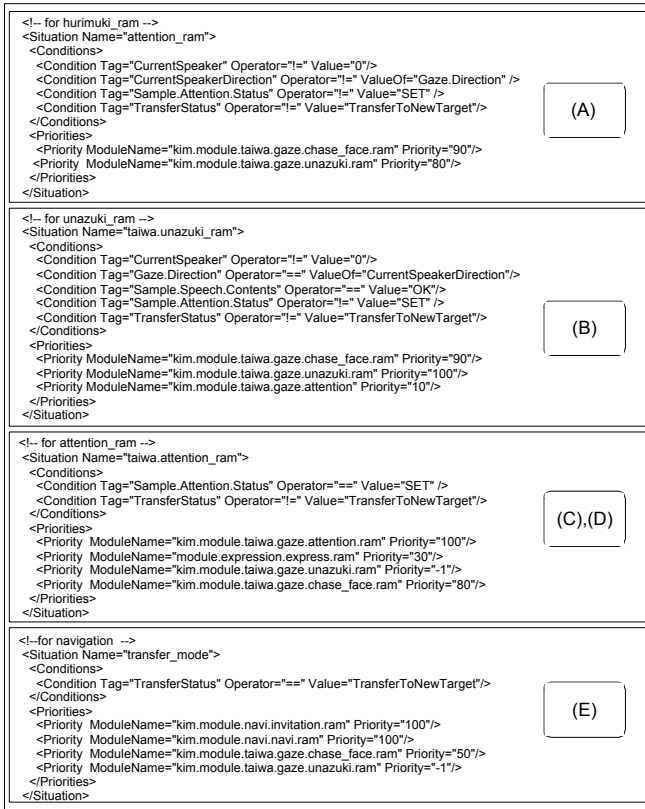


Fig. 4: Description of SPDS (main part).

We have developed a robot that participates in group conversations [9]. In this task, speech, auditory, image analysis and language processing functions are combined on the robot system. The developers were eight students with speech/image processing, auditory analysis, pattern recognition, and AI and motion control expertise.

For extension of this task, we implemented new function that enables the robot system to change its attention by external stimulus. This extension is implemented mainly another two developers. For this extension, the module that enables detection of two external stimuli, that is slapping sound and hand movement of human, is appended to the system. When an external stimulus is detected, the robot system discontinues the conversation task and begins new task such as transference. SPDS was described to shift the execution authority from modules of conversation task to modules for attention management when external event had occurred.

A main part of described SPDS is shown in Fig. 4. The part (A)-(E) shown in Fig. 4 corresponds to a part (A)-(E) shown in Fig. 5. Operation status of task and selection of modules under each situation is shown in Fig. 5. The robot chases the face of the human speaker (Fig. 5 (A)). If the content of the speech is acceptable, the robot nods

(Fig. 5 (B)) to the speaker. When the slapping sound has occurred, the robot gazes the direction of sound source with continuing conversation (Fig. 5 (C)). When the robot is called by another person, the robot turns its face toward the calling person (Fig. 5 (D)). And if the calling person invites with waving his hand, the robot discontinues the conversation and begins to transfer toward the person (Fig. 5 (E)).

As the result, appropriate selection of modules is realized by describing the SPDS and using proposed inter-module cooperation mechanism.

8. Discussion

One of the features of the proposed architecture is that it can select a reasonable behavior instantly. It can control a global behavior through SPDS and local behavior through priority layered system. The architecture is realized using blackboard for both situation estimation and information exchange between the modules. The local decision of the module can use any of the information written in the blackboard. The global decision can proceed to estimate the best situated script, while the current layered system is dealing with the changing environment. The global decision uses the information same as the local decision. We think the realization by blackboard is reasonable to make the system open and highly integrated.

Another feature of the proposed architecture is that it realizes an open development process. Generally, it is difficult for a developer to precisely analyze inside of a module that is developed by a developer of another expertise field. And it is also difficult to implement new modules to cooperate with those other modules under particular situation. The same might be said of the case of task designers. Modules used in the task were developed by different developers in different time. And each developer had not given consideration to other developed modules in implementation process. As a result of task development using proposed architecture, module developers could implement necessary modules without considering other modules, and task designers could realize cooperation between modules by describing SPDS, without preside analysis of each module. But still, our architecture has not implemented a self learning process to sophisticate its behavior. Those problems are left for future work.

9. Conclusion

We have proposed architecture for system software that realizes cooperative processing of functional modules on the robot system.

The efficiency of the system is verified through the development of a new task on a multi-modal conversation

robot.

References

[1] S.Hashimoto et al. "Humanoid Robots in Waseda University -Hadaly2 and WABIAN-," IEEE HUMANOIDS 2000, 2000.

[2] E.S Raymond, The Cathedral & the Bazaar, O,Rellly & Associates Inc., 1999

[3] R.Brooks, "A robust layered control system for a mobile robot," IEEE Journal of Robotics and Automation, vol.2, 1986.

[4] E.Gat, "On three-layer architectures," In Artificial Intelligence and Mobile Robots. MIT/AAAI Press, 1997.

[5] H.Nakashima, I.Noda, "Dynamic Subsumption Architecture for Programming Intelligent Agents," Proc. Int. Conf. on Multi-Agent Systems '98, pp.190-197, 1998.

[6] R. Smith, "The contract net protocol: High-level communication and control in distributed problem solver," IEEE Transactions on Computers, 29(12), pp.1104-1113, 1980.

[7] Y.Motomura, I.Hara, "Probabilistic Network based Multi Agent Model," Technical report of IPSJ SIG-MPS, 2000.MPS.28, pp.17-20, 2000.

[8] Y.Matsusaka, T.Kobayashi, "System Software for Collaborative Development of Interactive Robot," IEEE HUMANOIDS 2001, pp.271-277, 2001.

[9] Y.Matsusaka, T.Tojo, S.Kubota, K.Furukawa, D.Tamiya, K.Hayata, Y.Nakano, T.Kobayashi, "Multi-person Conversation via Multi-modal Interface .A Robot who Communicate with Multi-user-," ISCA Proc. EUROSPEECH,99, vol.4, pp.1723-1728, 1999.

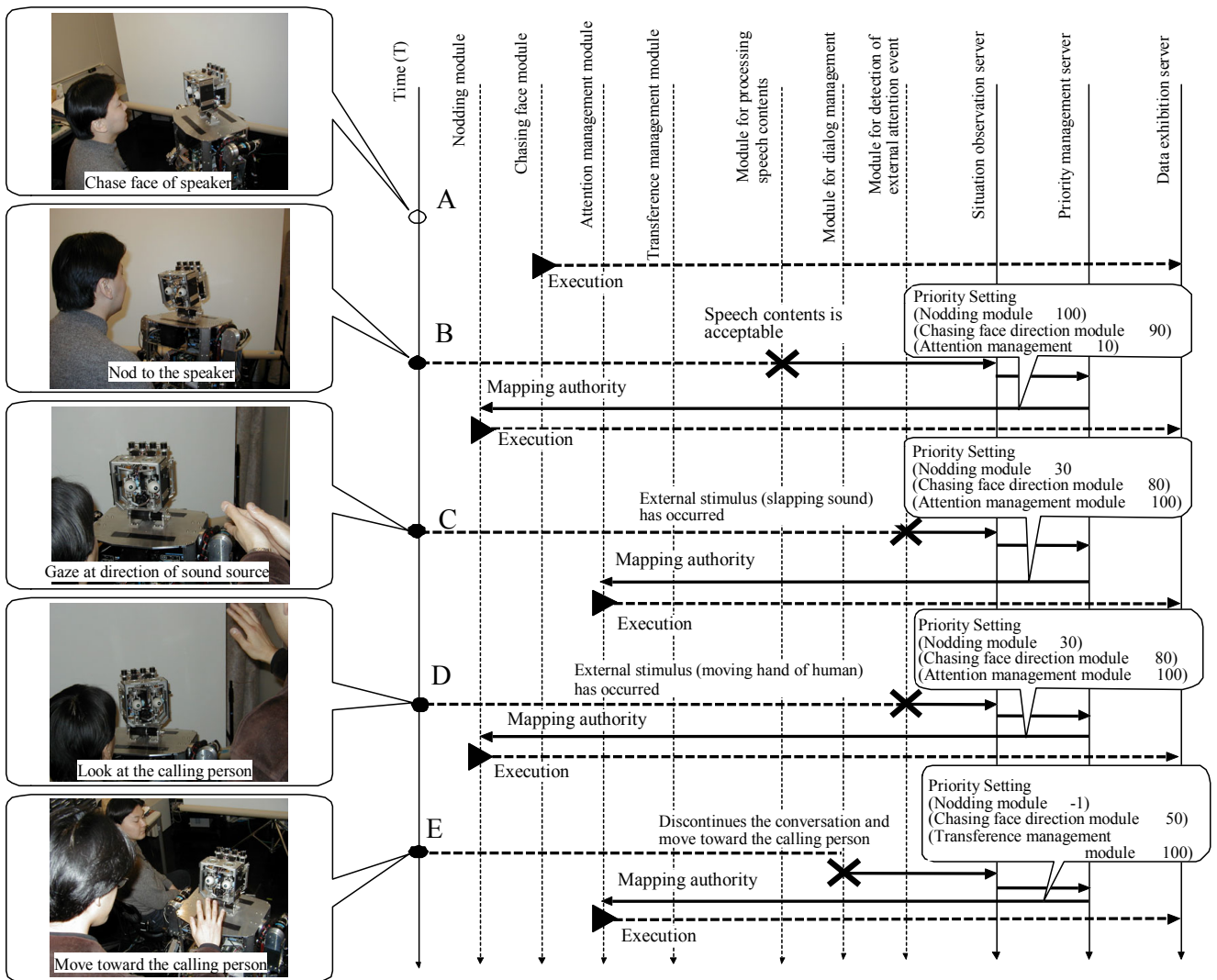


Fig. 5: Example behavior.