VARAMM: Mental Model of Remote Communication

Hideaki Shimada, Muneo Kitajima, and Osamu Morikawa

National Institute of Advanced Industrial Science and Technology (AIST)
1-1-1 Higashi Tsukuba Ibaraki 305-8566 Japan
{shimada.hideaki, kitajima.muneo, morikawa.osamu}@aist.go.jp

Abstract

When encountering a remote communication system, people transform the knowledge known to work in daily face-to-face communication into a workable form for the current situation and utilize it for generating appropriate actions. However, we argue that this transformation often fails, causing serious communication problems. In this paper, we regard this transformation of knowledge as a knowledge mapping process between *virtual actions* performed on a remote communication system and *real actions* conducted in the real world, and propose the Virtual-Action Real-Action Mapping Model (VARAMM), a new way of representing users' knowledge about mappings. VARAMM works on a hierarchically organized knowledge base, i.e., script–goal–action hierarchy, and explains users' behavior in remote communication systems observed in two independent studies. First, VARAMM explains the diversity of behavior people demonstrated when using a single remote communication system (Morikawa & Maesako, 1998) because of the diversity in the mappings users might have generated when initially using the system. Some behaviors were intended by the system designer, but others were not. Second, VARAMM explains users' behavioral changes when a remote communication system is reconfigured (Yamazaki et al., 1998) because of the reorganization of the script–goal structure triggered by the recognition of the reconfiguration that eventually enabled the intended functionality of the system. Based on these explanations, we claim that usability problems in remote communication systems should exist at least at two levels, action mapping and script–goal structure. These problems are very serious since they are not solved inherently by mere repetitive use of the remote communication systems, which only strengthens existing knowledge on mapping and the script–goal structure. We suggest that designing to users' knowledge will solve these usability problems. We also feel that representing users' knowledge in VARAMM provides useful insights for system developers to identify the knowledge, mapping or script–goal structure, that may block the intended system's functionalities and for discovering efficient ways to make them available to the users.

1 Introduction

A fundamental problem in current remote communication systems is that they are not usable. We believe that such systems do not appropriately consider how humans actually behave in order to perform their ordinary tasks such as conversation, greeting, and discussion when such systems are introduced. This is a design flaw. Our research aims at deriving a solution necessary for designing useful remote communication systems by focusing on users' knowledge employed when using such systems and identifying the portions that indicate discrepancies between users' expectations on how the system should respond and actual system response that would have caused trouble in using the systems. Well-designed systems should eliminate such discrepancies by providing appropriate displays of the participants and/or by appropriate introductory training.

This paper proposes the Virtual-Action Real-Action Mapping Model (VARAMM), a new way of representing a mental model of users in a remote communication system, as our initial attempt for accomplishing the ultimate goal mentioned above by gaining understanding of users' activities in remote communication systems and deriving suggestions for designing usable systems. In this paper, we start by describing VARAMM. We then show that VARAMM successfully explains users' behaviors in two remote communication systems in which detailed observational data are available: variations in handshake behavior in the HyperMirror system (Morikawa & Maesako, 1998) and changes caused by reconfiguration of a remote instruction system (Kato et al., 2001; Yamazaki et al., 1998). Finally, we discuss applications of VARAMM to develop a usable environment for remote communication.

2 Outline of VARAMM

When using a remote communication system, users must combine *knowledge in the real world* necessary for performing face-to-face conversation, such as handshakes, with *knowledge in the virtual world*, such as the system's functionality, in order to interpret what is happening through images generated by the remote communication system. This section describes VARAMM's definition of these kinds of knowledge.

2.1 Knowledge in the Real World

Remote communication systems realize face-to-face communication for people located in distant places. Thus we assume that users would recruit pre-existing knowledge concerning face-to-face communication and modify it by considering the nature of the functionalities provided by the systems. VARAM describes such users' knowledge in a script–goal–action hierarchy, after the script theory proposed by Schank and Abelson (1977).

The script theory, which is expanded to Memory Organization Packet (MOP) theory by Schank (1982, 1999), claims that people have knowledge on goal-oriented behavior as sequential actions in memory. For example, people who visit a restaurant retrieve the "restaurant script." The restaurant script has four scenes, which contain typical behaviors in a general restaurant. The following is a part of restaurant script described by Bower, Black, and Turner (1979).

- Scene 1: Entering
  Customer enters restaurant, Customer looks for table, Customer decides where to sit, ...
- Scene 2: Ordering
  Customer picks up menu, Customer looks at menu, Customer decides on food, ...
- Scene 3: Eating
  Cook gives food to waitress, Waitress brings food to customer, Customer eats food.
- Scene 4: Exiting
  Waitress writes bill, Waitress goes over to customer, Waitress gives bill to customer, ...

The theory claims that people utilize a coherent memory structure, i.e., the script, as a top-down memory process.

VARAMM represents people's knowledge necessary for using remote communication systems as a variant of Schank's script, a script–goal–action hierarchy. Let's take "handshake" behavior as an example to illustrate the VARAMM's script–goal–action hierarchy (see left side of Fig. 1). In VARAMM, a *script* represents a purposeful behavior such as shaking hands. It is expanded in three subscripts and a goal: "Keep distance appropriate," "Face each other," "Take partner's hand," and "Synchronize hands." The subscripts are further expanded into sub-subscripts. This expansion process terminates when a sub-…-subscript cannot be further expanded. We call the terminal sub-…-subscript the *goal*. For example, sub-subscript "Check present distance" is expanded in two goals, "Check my position" and "Check partner's position". VARAMM assumes that a *goal* is associated with a set of *actions*. An action represents a mutually distinguishable body movement in terms of the meaning it conveys. For example, the goal "Move my body" is associated with four actions that are mutually distinguished by the directions of the movement, "Move forward," "Move backward," "Move right," and "Move left." In VARAMM, a hierarchical cluster consisting of a script, its subordinate subscripts, sub-subscripts, …, sub-…-subscripts, goals, and actions associated with the goals jointly specifies how a specific purpose of behavior is accomplished by actions.

2.2 Knowledge in the Virtual World

A remote communication system converts actions in the real world into actions in the virtual world. The latter are the representations to be shared by parties in the system. VARAMM represents the relationships between actions in the real world and those in the virtual world as *action mapping*. The idea is derived from Moran's (1983) External-Internal Task Mapping analysis, which is called ETIT.

ETIT lists the concepts and operations both inside and outside the computer world, and creates rules to map the external task elements onto internal elements. The number and complexity of mapping rules can be used as a measure of the difficulty with which knowledge from one environment can be transferred to another. For example, External task space might include the following concepts.

Terms: Character, Word, Sentence, Line, Paragraph, Text

Tasks: Add, Remove, Transpose, Move, Copy, Split, Join
Execution: Copy Characters, Split Characters, Join Characters

Internal task space might include the following terms and tasks.

Terms: String
Tasks: Insert, Cut, Paste

Finally, the mapping rules could be represented as follows for these External and Internal tasks.

1) Split, Join Sentences ➔ Change String 2) Text ➔ String 3) Add ➔ Insert
4) Remove ➔ Cut 5) Transpose ➔ Move 6) Split ➔Insert 7) Join ➔ Cut
8) Change String ➔ Cut String + Insert String 9) Move String ➔ Cut String + Paste String
10) Copy String ➔ Cut String + Paste String + Paste String

According to Moran, ETIT may also be used to determine the difficulty of transferring knowledge from one application to another.

In remote communication, an external task corresponds to performing actions in the virtual world, and an internal task corresponds to performing actions in the real world. To give a simple example, the internal task of gazing at a remote partner in a video conference system must be mapped onto an external task of targeting one's gaze toward a camera mounted on the screen on which the partner is displayed.

VARAMM defines action mapping as a mapping from *real space* (i.e., virtual world) onto *virtual space* (i.e., virtual world). f and f^{-1} in Fig. 1 are examples of representation of such mapping for shaking hands in a HyperMirror system. VARAMM distinguishes two kinds of action mappings, *physical action mappings* and *mental action mappings*. These mappings dictate mappings from actions in the virtual world to those in the real world, and vice versa, but physical action mappings concern the system designer's conceptualizations of the system's functionalities whereas mental action mappings refer to the user's current understandings of the system's functionalities. This distinction is critical for understanding user's problems when a new remote communication system is introduced.

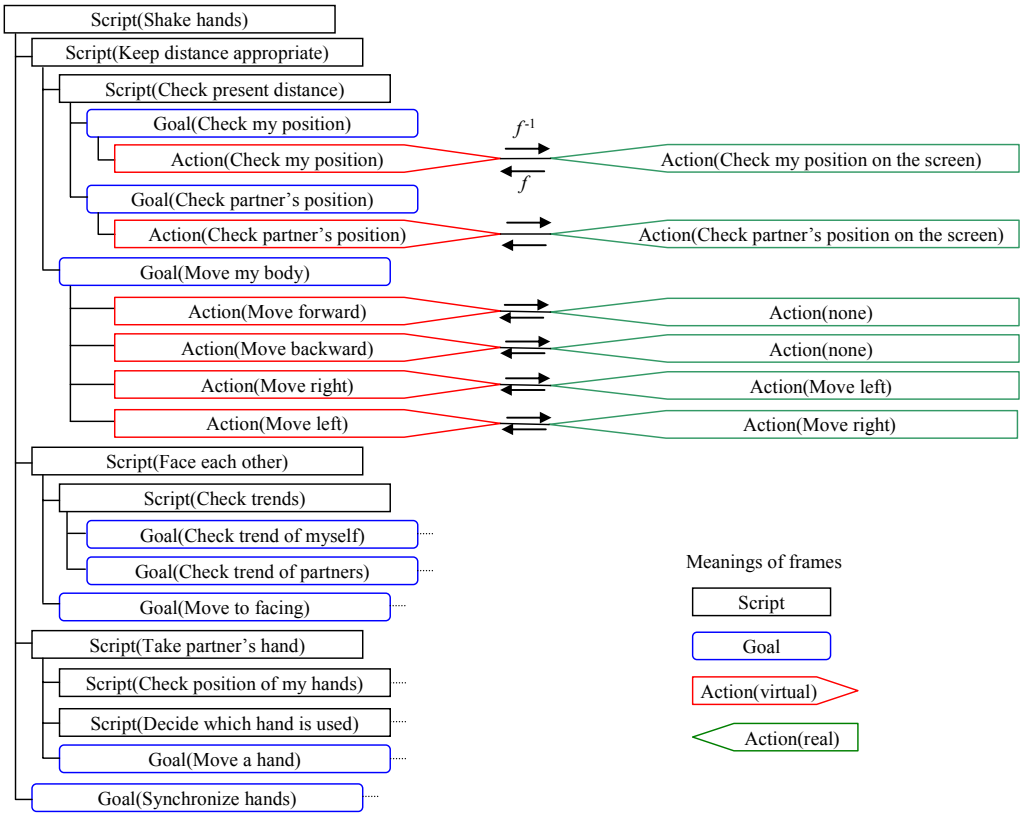

Figure 1: Representation of "shake hands" script and action mappings in HyperMirror. The left side shows script–goal–action hierarchy in the virtual space. The right side shows the actions in the real space. A virtual action is connected with a real action by a mapping rule f and f^{-1}.

When a user wants to carry out an action, he/she must transform a *virtual action* he/she wants to happen in the system to a *real action* he/she must execute in the real world by utilizing mental action mappings. However, this may fail, most likely when a new system is used, due to the lack of mapping knowledge or the discrepancy between mental and physical action mappings. However, some problems in mappings can be remedied as the user gains experience in the system by modifying the flawed mappings through feedback from the system. VARAMM explains that the adaptation is caused by acquiring mental action mappings consistent with physical action mappings.

3 Definitions of Action, Action Mapping, Goal, and Script

This section introduces VARAMM's definitions of action, action mapping, goal, and script as the components of the script–goal–action hierarchy; the next section introduces processing assumptions for VARAMM's knowledge structure.

3.1 Action

VARAMM defines an action as the smallest unit of behavior that corresponds to objectively discernible body movement, such as "Move left." VARAMM adopts a frame-like representation for actions. The action labeled "*a*" is represented as Action (Isa=*a*), where "Isa" and "*a*" denote the slot name and its value, respectively. In addition, VARAMM has a slot to indicate the space where the action is performed; Action(Isa=*a*, Space=real) and Action(Isa=*a*, Space=virtual) denote that the action *a* is performed in the real and the virtual space, respectively.

3.2 Action Mapping

VARAMM represents action mappings as follows.

f_p(Action(a_1, real))=Action(a_2, virtual); f_p^{-1}(Action(a_2, virtual))=Action(a_1, real)
f_m(Action(a_1, real))=Action(a_2, virtual); f_m^{-1}(Action(a_2, virtual))=Action(a_1, real)

Here, $f_.$'s denote mappings from real space to virtual space, $f_.^{-1}$'s, mappings from virtual space to real space, suffixes *p* and *m* stand for physical mapping and mental mapping, and a_1 and a_2 denote a real action and virtual action, respectively. VARAMM generally utilizes mappings from virtual space to real space, $f_.^{-1}$'s, because when performing an action users have to select a virtual action by expanding a script and then mapping it to a real action.

In some cases, no real action is associated with a virtual action. VARAMM represents these cases as follows.

f_p^{-1}(Action(a_3, virtual))=Action(none, real)
f_m^{-1}(Action(a_3, virtual))=Action(none, real)

The first case corresponds to the situation where the system's functionalities do not support creation of virtual action a_3 the user wants to perform. The second case describes the situation where users don't know how to associate virtual action a_3 with a real action.

We will give a simple example of action mappings. Suppose that a user is acting in front of a video mediated system with an avatar that represents the user herself. The system associates double-clicking the avatar's right hand, a real action, with the avatar's action of moving its right hand upward, a virtual action. VARAMM represents the physical action mapping involved in this operation as follows.

f_p^{-1}(Action(Move right hand upward, virtual))=Action(Double-click avatar's right hand, real)

The user may or may not know the physical action mapping is available in the system. Accordingly, mental action mapping for the virtual action "Move right hand upward" can have the following two forms.

f_m^{-1}(Action(Move right hand upward, virtual))=Action(Double-click avatar's right hand, real)
f_m^{-1}(Action(Move right hand upward, virtual))=Action(none, real)

3.3 Goal

VARAMM defines a goal as a set of actions and represents it as

Goal(g)={Action(a_1), Action(a_2), ..., Action(a_n)},

where g is a description of a goal, such as "Check my position," "Check partner's position," and "Move my body" in the handshake script depicted in Fig. 1. A goal can be generic, such as "Move my body," which may be accomplished by performing one or more actions sequentially to move the body in different directions. The constituent actions include "Move forward," "Move backward," "Move right," and "Move left"; each specifies a direction of movement.

3.4 Script

VARAMM defines a script as a set of goals and sub-scripts, and represents it as

Script(s)={Script(s_1), Script(s_2), ..., Script(s_n); Goal(g_1), Goal(g_2), ..., Goal(g_m)},

where s is a description of such behaviors as "Have a meal in a restaurant," "Shake hands," and "Say something in a classroom," all of which require a number of steps to accomplish.

For example, Script(Shake hands) in Fig. 1 contains three sub-scripts and a goal, and is represented as:

Script(Shake hands)={ Script(Keep distance appropriate),
Script(Face each other),
Script(Take partner's hand),
Goal(Synchronize hands) }.

A script contains sub-scripts and goals that are necessary to accomplish it. However, we assume that a user may perform some of the constituent sub-scripts and goals, which corresponds to the situation in which the script is partially accomplished. For example, a user may perform the "Shake hands" script without performing the "Face each other" sub-script. We regard the "Shake hands" script without "Face each other" as a version of "Shake hands."

4 Processing Assumptions

This section describes the cognitive processes that VARAMM carries out in order to use the knowledge defined by a script–goal–action hierarchy. The processes include those for selecting script, goal, or action, those for executing the selected one, those for evaluating their consequences, and those for modifying the existing knowledge structure.

4.1 Selection

VARAMM assumes that when users want to carry out a behavior they retrieve an appropriate script from long-term memory. The retrieved script is associated with a set of subscripts and/or goals. The users must thus select those that are perceived to be executable and to most likely lead to success given the circumstances that the remote communication system defines. VARAMM has a slot for storing the expected results of executing script, sub-script, goal, or action, called *expectation.* Its value, "success," "failure," or "don't know," is used for the basis of selection. The values are given based on the situation they are in. VARAMM represents Script(s) with expectation e_x as Script(s, Exp= e_x).

In Script(Shake hands) shown in Fig. 1, if the users expect that they can shake hands in their environment, they set expectation of Script(Shake hands) as Script(Shake hands, Exp=success). Next, they evaluate whether the three sub-scripts and the goal in Script(Shake hands) are executable. If users expect all sub-scripts and goals are executable, they give all four elements "Exp=success." In this case, they select all elements.

4.2 Execution

VARAMM assumes that users carry out the selected actions after completing expansion and selection for the script. In carrying out the actions, they have to utilize mental action mapping. For example, on selection of Action(Move right hand upward, virtual), they transform it into

f_m^{-1}(Action(Move right hand upward, virtual, Exp=success))
=Action(Double-click avatar's right hand, real, Exp=success)

and execute it in the real space.

4.3 Evaluation

After users execute the transformed actions in the real space, they evaluate whether the actions have been successful or have ended in failure. VARAMM represents the evaluation with a slot "Eval" as Action(Double-click avatar's right hand, real, Eval=success) if the action has succeeded, or as Action(Double-click avatar's right hand, real, Eval=failure) if the action has failed. The evaluations of actions propagate upward in the script–goal–action hierarchy to those of goals and ultimately to those of scripts.

The value of evaluation of a script or a goal is "success" if all evaluations of the selected elements of the script or the goal are "success." However, if "Eval=failure" is assigned to a script or a goal, a modification process described in the next section is carried out.

4.4 Modification

There are some cases in which the selected virtual actions can't be carried out because of the lack of mental action mappings (i.e., lack of knowledge of the system on the part of the user) or the lack of physical action mappings (i.e., lack of the necessary functions of the system). In these cases, VARAMM replaces in a systematic way the blocked actions with executable actions, even though they may not accomplish the purpose with complete satisfaction. VARAMM assumes that users would employ one of the following two strategies for modifying the existing script–goal–action hierarchy or action mappings in order to make the blocked actions executable in another way.

4.4.1 Modification at the Action Mapping Level

The initially blocked virtual action may become executable when the user selects the right mental action mapping and modifies the existing ones accordingly. Suppose that a user has the following mental action mapping for "Move right hand upward" action from the past experience of a remote communication system,

f_m^{-1}(Action(Move right hand upward, virtual))=Action(Double-click avatar's right hand, real),

but found that this won't work in the current remote communication system. Thus,

Action(Double-click avatar's right hand, real, Eval=failure)

is the current representation of the action. He or she would search for an appropriate action that causes the intended action "move right hand upward" and find the correct mapping,

f_m^{-1}(Action(Move right hand upward, virtual))= Action(Click at right hand and press up arrow key, real),

or fail to find it, in which case the user might consider the action impossible and give up carrying out the script.

4.4.2 Modification at the Script–Goal Level

Based on a series of expansions and selections in the script–goal–action hierarchy, a virtual action is proposed as the one to be mapped onto a real action. When the real action is found impossible to execute, the user might search for an alternative mapping by redoing the expansion and selection processes.

For example, a version of shake hands is generated by executing Script(Shake hands) in Fig. 1 with all of the four elements, but another version could be generated by not executing Script(Face each other), which may not be supported by a remote communication system because of physical constraints. In this case, Script(Shake hands) is expanded with {Script(Keep distance appropriate), Script(Take partner's hand), Goal(Synchronize hands)}. However, since this version does not perfectly mimic a real handshake, the evaluation would be moderate, Script(Shake hands, Eval=moderate success). Such fuzzy evaluations as "moderate success" are important for describing cognitive processes in remote communication because constraints of systems' functionalities or lack of system knowledge often prevent users from acting with perfect satisfaction as compared with face-to-face communication.

5 VARAMM's Explanations of Empirical Observations

This section presents VARAMM's explanations of users' behaviors in two remote communication systems.

5.1 Handshake Behaviors in HyperMirror

We now introduce the HyperMirror system (Morikawa & Maesako, 1998), a video mediated communication system, to show that VARAMM successfully explains observed variations in handshake behavior in the system.

HyperMirror is a remote communication system that mimics a physical glass mirror. Fig. 2 depicts a HyperMirror session in which a user extends her right hand toward the video projector screen where her mirror image and another user at a distant site are available. The most distinctive feature of the system is mirroring the images of the partner and the self, which enables the parties to feel as if they were next to each other and to perform collaborative behavior such as shaking hands as if they were in the same place.

We observed that users showed various behaviors when using the HyperMirror system. In this paper, we extract two episodes from our data archive and explain users' various behaviors at the knowledge level based on VARAMM.

5.1.1 Users who Shook Hands Smoothly

"Shake hands" is a common behavior for greeting. People can carry out a HyperMirror version of shaking hands as illustrated in Fig. 2. However, we have observed that some people can perform the HyperMirror handshake smoothly when they are first introduced to the system, but others cannot. VARAMM explains these behavioral differences in terms of their knowledge about script–goal–action hierarchy and mappings.

We first focus on users who perform a smooth HyperMirror handshake. Fig. 3 presents VARAMM's analysis of cognitive processes of such users starting from expanding Script(Shake hands) with its four subordinate subscripts, Script(Keep distance appropriate), Script(Face each other), Script(Take partner's hand), and Script(Synchronize hands), followed by further expansions of the first subscript Script(Keep distance appropriate) with its associated Goals and Actions, ending up with its successful execution, Script(Keep distance appropriate, Eval=success), and initiating the next subscript Script(Face each other). In Fig. 3, "<" means the start of processing a script, goal or action; ">" means its termination; and "+" means selection of scripts or goals.

5.1.2 Users who Held out their Left Hands

Some users hold out their left hands instead of holding out their right hands. In Fig. 4, the man on the left, an experimenter, held out his right hand, but the man on the right, a subject using the system for the first time, held out the wrong hand, his left hand, when greeting the experimenter while observing the experimenter extending his hand as shown on the HyperMirror screen. The subject spontaneously, but with a little hesitation, extended his left hand. However, the subject immediately recognized that he was wrong and extended the other hand to see a successful handshake greeting.

The move-hand action is executed by expanding the top-level Script(Shake hands) with its subscript Script(Take partner's hand) and its associated goal, Goal(Move a hand), in the virtual space. The script–goal–action hierarchy is

Figure 2: Shaking hands in HyperMirror. Note that image on the video projector screen is reverted.

fine, but VARAMM indicates that the problem is in the wrong action mapping when translating the virtual action into a real action. The incorrect mapping is described as follows. When attempting Action(Move my left hand right), the users should have utilized the action mapping,

f_m^{-1}(Action(Move my *left* hand right, virtual))=Action(Move my *left* hand left, real).

They should have thought that their left hands in the virtual space correspond to their left hands in the real space when performing action mapping.

As a result of utilizing the wrong action mapping, they failed to shake hands. However, they switched hands immediately after they recognized that they were responding incorrectly. In VARAMM, this change is described as the modification of action mapping as explained in section 4.4. The modified mapping replaces "left" in real space

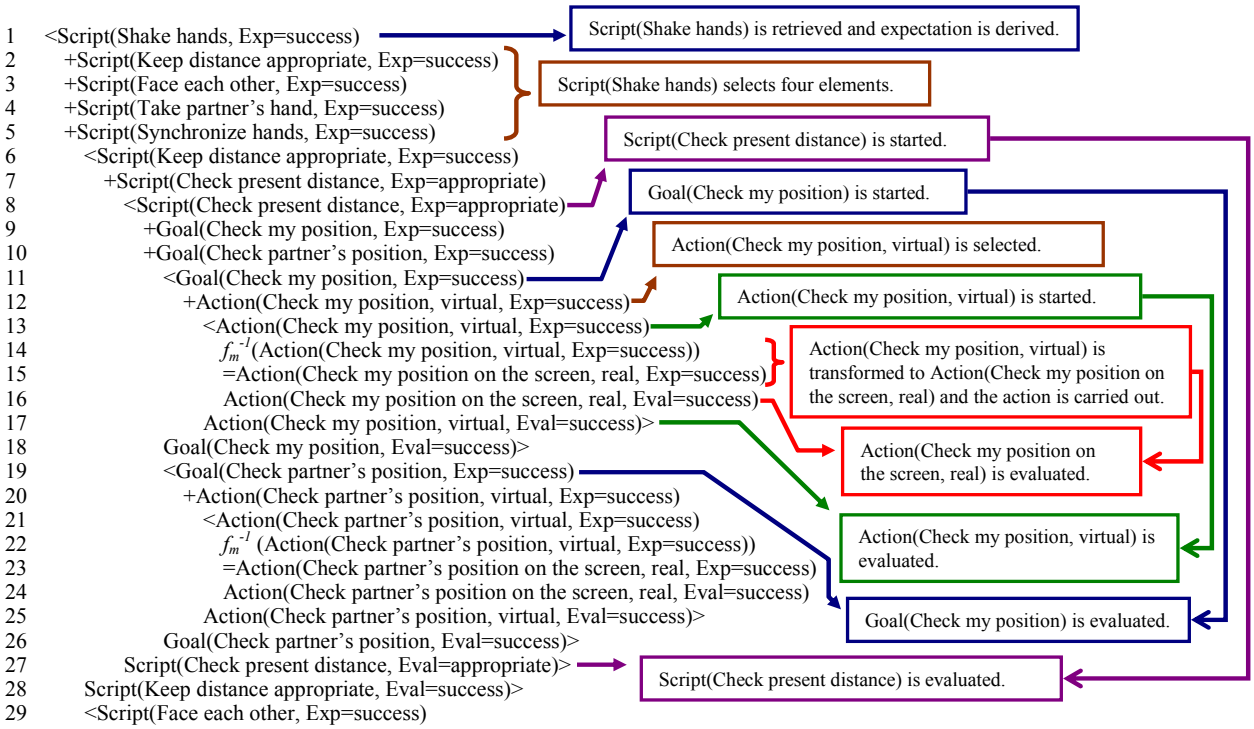

Figure 3: VARAMM's description of a smooth process of Script(Shake hands) from the start of Script(Shake hands) to the end of Script(Keep distance appropriate).

Figure 4: An example of unsuccessful "shake hands" greeting on the HyperMirror screen. The left person initiated a handshake greeting by extending his right hand in the real space but the right person extended the wrong hand, his left hand in the real space. VARAMM claims that the right person should have applied a wrong action mapping to this situation while he had a correct script–goal–action hierarchy for handshake.

with "right" as shown by the following rule.

f_m^{-1}(Action(Moving my *left* hand right, virtual))=Action(Moving my *right* hand left, real)

5.2 Behavioral Changes in Remote Instruction System

A user of a remote communication system needs to perform a virtual action that conveys his or her intention to be transmitted to the partner. However, this is not possible when the system does not have the functionalities to make the virtual actions executable. When using a system with poor functionalities, VARAMM predicts that users will have difficulties in carrying out the initial script–goal structure utilizing face-to-face communication. In this section, we draw an example from the study by Yamazaki et al. (1998) to illustrate the evidence of such reorganization.

Yamazaki et al. (1998) investigated the effect of the screen layout in a remote instruction system on the behavior of the participants. They proposed an improved screen layout that increased the system's utility. The system was designed for a teacher at a remote site to instruct students on the operation of a set of blocks. In Fig. 5, the diagram on the left shows the initial layout of the system; the diagram on the right is the improved one. The major difference is the location of the monitor B at the students' site. It displays the teacher pointing at the blocks on the teacher's site monitor D. This change enables the teacher to determine what the students paid attention to by use of the students' orientation monitor (monitor C); the students viewing monitor A indicated that the students paid attention to the teacher's face. The students viewing the monitor B indicated that they paid attention to the teacher's pointing. Otherwise the students were viewing the blocks.

VARAMM explains why this change leads to an improvement as follows. When the teacher forms the script Script(Index an object), he/she expands it with the two goals, Goal(Point to an object) and Goal(Check partners' attention). The latter goal, we believe, is an essential element in giving instructions because the instructions are wasted if the students aren't paying attention to them. The action associated with the Goal(Check partners' attention) is expanded to {Action(Check visually), Action(Check orally and acoustically)}. People usually utilize Action(Check visually) because the action is carried out more easily and at less cost than check orally. With the initial system, however, it was difficult to judge visually whether the students paid attention to the teacher's face or his/her pointing behavior because the monitor for displaying the teacher's face (monitor A) and the one for displaying the pointing behavior (monitor B) were next to each other; therefore, the teacher could not perform Action(Check visually) smoothly. Instead the teacher executed the more onerous action, Action(Check orally and acoustically) to perform Goal(Check partners' attention). VARAMM predicts that placing the two monitors at the opposite sides of the work desk, as shown on the right of Fig. 5, would enable Action(Check visually) and enable performing its super-ordinate goal Goal(Check partners' attention) more smoothly.

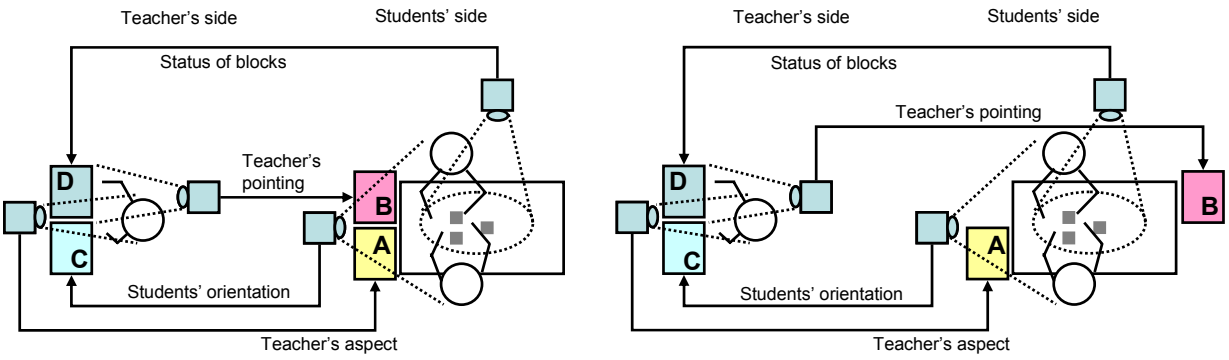

Figure 5: A remote instruction system studied by Yamazaki et al. (1998). With the left layout, the teacher had problem in grasping the students' attention. Relocating the monitor B made possible the teacher to distinguish between the students' behavior of viewing the monitor A (i.e., paying attention to teacher's face) and that of viewing the monitor B (i.e., paying attention to teacher's pointing behavior).

6 Discussion and Conclusion

We developed VARAMM as a mental model for the users of remote communication systems. We demonstrated that VARAMM could successfully explain users' behavior through two examples by assuming that users utilize knowledge in face-to-face communication, and by describing such knowledge as a script–goal–action hierarchy and action mappings.

In previous development of remote communication systems, it was believed that inconsistent correspondence between the virtual world and the real world is the major reason for usability problems, so much effort has been devoted to solve the problems of inconsistency by developing new technologies. However, VARAMM suggests another solution to the problem by defining a way of characterizing users of remote communication systems at the knowledge level. As described in this paper, VARAMM assumes that users perform their real actions by expanding their pre-existing knowledge concerning face-to-face communication, and VARAMM can predict virtual actions and their associated action mappings that the users would perform given the communication situation. These predictions should be very useful information for the system designers in evaluating whether the system's functionalities that should support the virtual actions are easily utilized by the target users. It should also help discover potential usability problems beforehand.

The above predictions are also useful for designing introductory instruction methods for new users of a remote communication system. The new users can learn the system's functionalities by repetitive use. However, this can happen only when they have the necessary knowledge for using the functionality; otherwise the users would never utilize the system as intended, though they may find the proper way to use the system through trial and error. Based on VARAMM's analysis, introductory instruction can focus on the portion in the script–goal–action hierarchy and action mappings where the new users would have difficulty in finding the correct actions due to the lack of knowledge. The analysis also provides appropriate information for the users to have a good knowledge structure. A good knowledge structure is the one that is consistent with the system's functionality or the one used by an expert user of the system. A well-designed instruction method would enable the users to create a good knowledge structure that is necessary to use the system effectively and efficiently.

Any new system that supports our daily activities requires the users to apply knowledge that works when performing the activities. VARAMM's knowledge level analysis provides a new insight for developing a usable and useful system. VARAMM suggests that having the user establish the correct knowledge is more important for improving usability than eliminating inconsistencies technologically between the system's response, i.e., virtual action for realizing the user's intention, and the user's real action.

References

Bower, G. H., Black, J. B., & Turner, T. J. (1979). Scripts in memory for text. *Cognitive Psychology*, 11, 177-220.

Kato, H., Yamazaki, K., Suzuki, H., Kuzuoka, H., Miki, H., & Yamazaki, A. (2001). Designing a video-mediated collaboration system based on a body metaphor. In T. Koschman, R. Hall, & N. Miyake (Eds.), *CSCL II Carrying forward the conversation* (pp. 409-424). London: Lawrence Erlbaum Associates.

Moran, T. P. (1983). Getting into a system: External-internal task mapping analysis. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 45-49.

Morikawa, O., & Maesako, T. (1998). HyperMirror: Toward pleasant-to-use video mediated communication system. *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, 149-158.

Schank, R. C. (1982). Dynamic memory: A theory of reminding and learning in computers and people. Cambridge University Press.

Schank, R. C. (1999). Dynamic memory revisited. Cambridge University Press.

Schank, R. C., & Abelson, R. P. (1977) Script, plans, goals and understanding: An inquiry into human knowledge structures. Hillsdale, NJ: Lawrence Erlbaum Associates.

Yamazaki, K., Miki, H., Yamazaki, A., Suzuki, H., Kato, H., & Kuzuoka, H. (1998). Instruction, artifacts, reflexivity – Designing tele-instruction system based on ethnomethodological perspective of human interaction. *Cognitive studies*, 5, 51-63. (in Japanese)