Modeling Human-Automation Interaction in a Unified Cognitive Architecture

Junya Morita
School of Knowledge Science, Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa, 920-2104 JAPAN
j-morita@jaist.ac.jp

Kazuhisa Miwa, Akihiro Maehigasi, Hitoshi Terai
Nagoya University, Graduate School of Informatics
Chikusa-ku Nagoya, JAPAN
miwa@is.nagoya-u.ac.jp, mhigashi@cog.human.nagoya-u.ac.jp,terai@is.nagoya-u.ac.jp

Kazuaki Kojima
Faculty of Human Sciences, Waseda University
Mikajima 2-579-15, Tokorozawa, SAITAMA, 359-1192 JAPAN
koj@aoni.waseda.jp

Frank E. Ritter
College of Information Sciences and Technology, The Pennsylvania State University
University Park, PA 16802
frank.ritter@psu.edu

Keywords:
Automation, Misuse, Disuse, Trust, ACT-R

ABSTRACT: *Automation is a necessity in modern society. People sometimes are inclined to trust automation too much. On the other hand, they sometimes tend to not be willing to use automation. To prevent these mistakes, this study explores factors of reaching an appropriate reliance on automation systems by using cognitive modeling. We have conducted psychological experiments on this problem using a simple line-tracing (driving) task where the participants had to track the line with a circle by pressing the arrow key on the keyboard (manual control) or rely on automation (auto control). They could switch between auto and manual control during the task. The success probabilities of each control mode were systematically varied. The ACT-R model that simulates these experiments was constructed by representing the reliance on the automation as utilities of rules. The model performs this task by firing rules that manage the perceptual/motor modules. The perceptual module finds and attends to the vehicle and the road on the screen, and the motor module press the keys depending on the current controlling modes or the current positional relation between the vehicle and the road. The utilities of these rules are updated based on the rewards in every screen update. This utility module is also compatible to a previous computational model of automation reliance. A preliminary run of this model simulated several qualitative features of the behavioral data. The ways it does not fit suggest that the model should be more sophisticated in its representation of space and process.*

1. Introduction

Automation systems are becoming increasingly necessary in almost all of human societies. Many people unconsciously use these in everyday life. Most email clients have a function that automatically sorts messages into folders by sender or topics. Some recent e-mail software also has automatic spam filters, which sometimes cause serious communication trouble. We can use Google maps or similar kinds of route-planning systems, which automate reading a map or checks transportation schedules. Many parts of driving behavior are also becoming automated. Drivers can now use an automatic transmission, automated cruise control, anti-lock braking systems, and even parking systems. These automation systems save time and help us lead more efficient lives.

Automation system, however, cannot replace human cognition in tasks completely. Bainbridge (1983) claimed that even highly automated systems need human operators to monitor system performance and to make decisions on system use. Some researchers have also pointed out that human decision-making on system use is not optimal. Parasuraman and Riley (1997) stated that there are two types of maladaptive use of automation: misuse, the overreliance of automation, and disuse, the underutilization of automation. Some studies indicated that human users have automation bias towards misuse of automation systems (Bahner,

Huper, & Manzey, 2008; Singh, Molloy, & Parasuraman, 1997; Skita, Moiser, & Burdick, 2000). On the other hand, other research indicated that human users have a manual bias, bias towards disuse of automation systems (Beck, Dzindolet, & Pierce, 2007; Beck, McKinney, Dzindolet, & Pierce, 2009; Dzindolet, Peterson, Pomranky, Pierce, & Beck, 2003; Dzindolet, Pierce, Beck, & Dawe, 2002).

de Vries, Midden, and Bouwhuis, (2003) experimentally revealed that the reliance of automation use is influenced by both the Capability of Manual control (Cm) and the Capability of Auto controls (Ca). To explain effects of Ca and Cm on reliance of automation, Gao and Lee (2006) proposed the Extended Decision Field Theory (EDFT model), which extends psychological decision-making theory (Busemeyer & Townsend, 1993), including the effect of previous decisions in the context of multiple sequential decision processes.

Figure 1 shows the basic process of the EDFT model. The model is given Ca and Cm values. The model constructs belief of Ca and Cm (Bca, Bcm) based on partially displayed Ca and Cm values. From the belief values, the model constructs trust (T) and self-confidence (SC). Preference of automation (P) is determined by subtracting T from SC. If P exceeds an upper threshold (θ), then the model turns the current control mode to auto. If P falls below a lower threshold (-θ), then the model turns the current control mode to manual. In every cycle, values of Bca, Bcm, T, and SC are updated by differential equations.

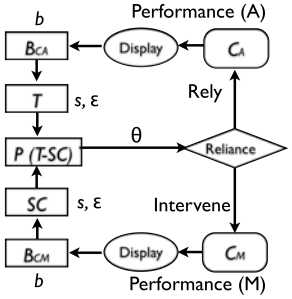

Figure 1: The process of Extended Decision Field Theory (Gao & Lee, 2006), modified and reproduced from the original figure.

The EDFT model is an abstract mathematical model that clearly explains the reliance on automation in dynamic situations. The strength of this model is its generality. It can apply to a wide range of situations involving automations. This model, however, does not have any knowledge about tasks. It cannot interact with a task environment, and it provides no prediction of human performance.

The present study constructs a cognitive process model that interacts with a specific task environment where sequential decision-making is made. To construct the model, we use ACT-R (Adaptive Control of Thought-Rational: Anderson, 2007), which is a unified theory of cognition combining several independent modules: a goal module, a production module, a declarative module, perceptual modules, and a motor module. A goal module holds the current task goal and other task related information. A production module and a declarative module hold procedural and declarative knowledge, respectively. Perceptual modules include a vision and an audio module, which take information from an external environment. A motor module manipulates devices like a keyboard or a mouse in an external environment. Modules other than the production module have buffers to hold information, called a chunk, temporary. A production module integrates the other modules by production rules, which consists of a condition / action pair that is used in sequence with other productions to perform a task. Conditions and actions in production rules are specified with buffer contents of each module.

ACT-R also includes sub-symbolic cognitive processes. If several rules match to buffer conditions, conflict resolution of production rules is made based on utility values assigned to production rules. The learning of utility is controlled by the following equation:

$$U_i(n) = U_i(n-1) + \alpha\left[R_i(n) - U_i(n-1)\right]$$

α is the learning rate set by a parameter; $Ri(n)$ is the reward value given to production i at time n; $Ui(0)$ is set by a parameter. The learning occurs when a reward is triggered, and all productions that have fired since the last reward are updated. This learning is essentially same as the process presented in the EDFT model using a basic reinforcement learning method.

So far, ACT-R has been used to model many fields of human-machine interaction including driving (Saluvucci, 2006), teleoperation (Ritter, Kukureja, & St. Amant, 2007), water flow control (Lebiere, Gonzalez, & Warwick, 2009), and airplane classification tasks (Taatgen, 2005). The utility update learning of this architecture also has applied to strategy selection (Lovett & Anderson, 1996). Therefore, we consider that this architecture is suitable to construct a model of human-automation interaction. Through this modeling, we try to find behavioral constraints in the model of automation use and include this topic in a unified theory of cognition.

2. The Task

To manipulate auto and manual performance in a dynamic situation, and to understand how users change

to use automation, we developed a simple tracking task, similar to driving. We call this task the *line-tracing task*. This environment was developed in Java. Figure 2 shows the screen shot of the task environment.

Figure 2: Screenshot of the line-tracing task. The green line and the blue dots are not visible in the experiments.

This task requires participants to control the horizontal position of the vehicle (red circle) to follow the black line that scrolls down at 24 pixels per second. The screen is updated every 40 ms. If the vehicle is not on the line, a warning is presented outside of the window. The line is drawn by randomly combining 48 pixels height line patterns of varied angles (30, 45, 90, 135, and 150 degrees).

The vehicle is controlled by commands of "left", "straight" or "right". For example, if the vehicle receives a left command, the vehicle moves 1 pixel left from the original position. The command is sampled at 48 Hz. Therefore, maximally, the vehicle can move 2 pixels per one pixel scroll of the line.

A participant can chose manual or auto controls to send commands. In the manual control, participants use left and right arrow keys to send commands. If a participant's finger is put on a right arrow key, the vehicle keeps receiving a right command at every 20 ms until this key is released[1]. In the auto control, participants monitor that the auto control moves the vehicle. The auto control tries to follow an optimal line presented as the green line in Figure 2. An optimal line is the shortest line to pass "goals" located on each corner. Figure 2 shows goals as blue dots. If the center of the vehicle is off the optimal line, the auto control system sends a command to correct the vehicle position. The command rate of the auto control is same as that of the manual control (50 Hz). In our experiment presented in the next section, the optimal line and goals are not visible to participants.

In both control modes, commands are not always successfully sent to the vehicle. Failures occur in a stochastically defined variable rate. In this study, we define Ca and Cm as such success rates. For example, when Ca or Cm is set at 50%, half of commands are dropped. Therefore, the vehicle controlled by the corresponding mode is lagged, and it becomes hard to follow the sharply angled line. To conduct the task successfully, participants need to select a suitable mode in each situation. The participants freely change two controls by pressing the spacebar during the task.

3. Experiments

Before describing the model, we summarize here two experiments that examined the use of automation in this task (further details are reported in Maehigashi et. al., in press).

In experiment 1, the baseline performance of the manual and the auto controls were examined. The participants (n = 65) were required to control the vehicle only with the manual control mode. There were five conditions where Cm values were manipulated from 30% to 70%. Every participant conducted the task in all of the five conditions. The order of conditions was randomized. Each condition lasted 40 seconds. The performance of the experiment was compared to the auto-mode performance measured in the corresponding Ca conditions. The results confirmed that the manual performance is lower than the auto performance in each condition.

Experiment 2 examined the ratio of automation use during the task. The participants (n = 35) conducted the task with the auto and manual control modes. They could freely change the mode during the task. Combining five levels of Ca and Cm values, 25 experimental conditions were prepared. Every participant conducted the task in all of 25 conditions. As in experiment 1, each condition lasted 40 seconds. The results of experiment 2 indicate that participants could adaptively select a suitable mode in a given situation.

4. Model

4.1. Simulated task environment

We used the ACT-R 6 architecture (Anderson, 2007) to develop a model simulating the above two experiments. Our model interacts with a simulated task environment developed in the ACT-R graphical user interface that is the standard device of the ACT-R 6. The simulated

[1] This command rate is not influenced by a key-repeat rate setting in OS. The environment monitors a key event. If a key-press event is detected, a flag of sending commands is on. This flag is off when a key-release event is detected.

environment is same as the original environment in the screen update rates (24 Hz), the line scrolling speed (24 pixels per sec), the vehicle size (24 pixels), the line width (5 pixels), and the screen size (480 pixels x 640 pixels). The auto control mode is also implemented with Common Lisp in the simulated task environment.

There are, however, some differences from the original environment. First, the simulated environment provides a perceptual cue to a lead optimal line. Visible goal positions are set at each corner to allow the model to directly perceive the optimal line. Second, the manual controls are slightly modified. The keys used in the manual control mode are changed from left and right arrow keys to "f" and "j" keys, which are located in a keyboard's home positions. In addition, the relation between command sending and key manipulation was changed. Unlike the original environment, the vehicle keeps receiving move-commands until other keys are pressed in the simulated environment. This change is because ACT-R 6 does not include a key-release function. Basic finger-movement is, however, the same as the original task.

4.2. Process of the model

In this study, the model is constructed by using the production, goal, vision, and the motor modules of ACT-R 6. The model has eleven production rules. These rules consist a basic perception-action cycle. Figure 3 indicates this cycle, presenting the rules in boxes. The cycle consists of a perceptual and motor process similar to previous driving models in ACT-R (Salvucci, 2006, Ritter, Kukreja, & St. Amant, 2007).

The top part of Figure 3 shows the perceptual process. In the perceptual process, the model picks visual information from a visual location buffer that holds location information of objects in the environment. *FindVehicle* finds the horizontal position of the vehicle in the visual location buffer, and places it into the goal buffer. *FindGoal* finds the horizontal position of the nearest goal position in the visual location buffer, and placed it into the goal buffer.

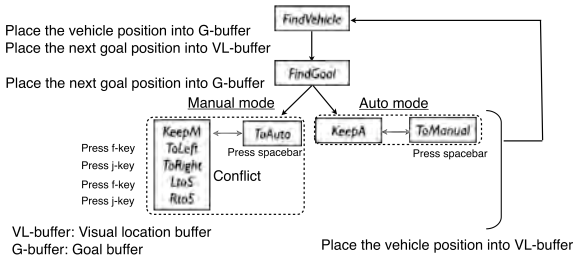

Figure 3: The basic cycle of the model.

The bottom part of Figure 3 shows the rules relating to the motor process. All of these rules clear the vehicle and the goal position in the goal buffer to begin the next cycle. The motor process is different between the manual control mode and auto control mode. In each mode, there is a rule to switch the current mode (*ToAuto* / *ToManual*). These mode-switching rules send a command to press a spacebar to the motor module, and compete with other rules in each condition. In the auto mode, *ToManual* conflicts with *KeepA* that just clears a goal buffer. In the manual mode, *ToAuto* competes with *KeepM*, *ToLeft*, *ToRight*, *LtoS*, and *RtoS*. These five rules have different conditions specifying the vehicle and the goal positions, and current move-commands (left, right, straight). The action clauses of *ToLeft*, *ToRight*, *LtoS*, *RtoS* send a command to press a key to a motor module. There are no action clauses relating motor module in *KeepM*. This rule just clears the goal buffer.

Figure 4 presents a time flow diagram showing the relations between the screen updates of the environment and the model cycles. The environment regularly updates the screen every 40 ms. Individual rule firings take 50 ms, but the cycle of the model is not regulated. There are delays in visual and motor processes. The process of the visual location module itself has no delay. However, to encode the location into the goal buffer, the perceptual process is required. As a result, encoding the locations delays 100 ms from the actual environment. The delay of the motor control is larger than that of the perceptual module. The ACT-R motor module needs preparation and execution time, which depends on the status of the motor module. As a result, these delays disadvantage manual control compared to automatic control.

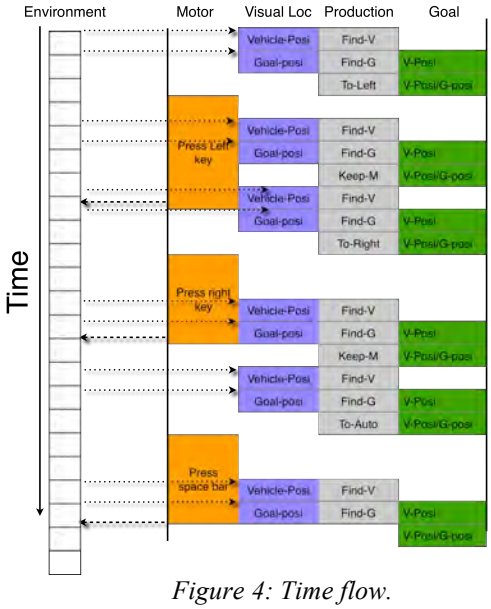

Figure 4: Time flow.

4.3. Learning and mode switching

The model adaptively learns to use a suitable mode in a given situation. We used the default reinforcement-learning algorithm in ACT-R 6 presented in section 1. The model receives rewards in every screen update. When the vehicle is off the line, rules used in the previous screen receive negative rewards (Ri (n) = 0). Otherwise, rules used in the previous screen receive positive rewards (Ri (n) = 10). This trigger corresponds to the warning in the actual task (Figure 2).

In the following simulations, the default learning rate is used (α = .2). The initial utility values of the mode switching rules (U_{ToAuto} and $U_{ToManual}$) are set to 5, and the initial utility values of other rules are set to 10. This initial setting corresponds the cost of mode switching. The initial utility values will not change unless the vehicle moves off the line because positive rewards and the initial values of utility are the same.

In ACT-R, strategy selection is often modeled by conflict resolutions based on utility values of rules (Lovett & Anderson, 1996). However, our task has differences from tasks used in the previous studies. As Figure 4 indicates the motor actions have delays. There are time-gaps between rule firing and manual control execution. These gaps make rewarding difficult because the next cycle begins before motor action completes. In addition, the structures of conflict are not the same between the manual and the auto control modes. *ToAuto* conflicts with five rules in the manual mode. On the other hand, *ToManual* conflicts with only *KeepM* in the auto mode.

Table 1: Production rule of ToAuto.

```
(p to.auto
  =goal>
    isa      move-vehicle
    - vehicle-loc nil
    - goal-loc nil
    - previous-rule press-space-m
    - previous-rule press-space-a
    current-mode  "M"
   !eval! (<= *self-conf* *trust*)
  ==>
  =goal>
    vehicle-loc nil
    goal-loc nil
    current-mode  nil
    previous-rule press-space-a
  +visual-location>
    isa      visual-location
    color    red
   +manual>
    isa      press-key
    key      space )
```

To solve this problem, we added meta-level decision making into the standard conflict resolution. In the usual conflict resolution, *ToAuto* can fire when its utility value exceeds a utility value of a competing rule.

In our model, *ToAuto* and *ToManual* have an !eval! condition that explicitly compares utility values of *KeepA* and *KeepM* (Table 1).[2] These values are stored in global variables referred as **trust** and **self-conf**. The Lisp function outside of the ACT-R model monitors the utility values of *KeepA* and *KeepM,* and sets these into the global variables in every screen update. ToManual fires only when *self-conf* exceeds *trust*, and the utility values of *ToManual* exceeds that of *KeepA*. Similary, *ToAuto* fires only when *trust* exceeds *self-conf*, and the utility values of *ToAutol* exceeds that of the competing production rule.

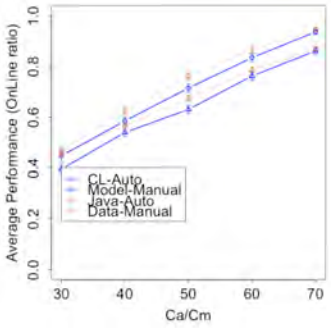

Figure 5. Performance of the model and the data in the baseline simulation. Error bars represent standard error of means (SEM).

5. Simulations

In this paper, we present two simulation experiments that aim to simulate the experiments presented in the section 3.

5.1. Baseline simulation

First, we conducted a simulation of experiment 1 to confirm the correspondence of base performance of the auto and manual modes.

5.1.1. Method

In experiment 1, the participants could not use the auto control mode (Data-Manual: n = 65). Similarly, we run the model with the initial control mode as the manual, and removed *ToAuto* from the model (Model-Manual: n = 100). We also compared baseline auto performance between the original environment (Java-Auto: n = 65) and the simulated environment (CL-Auto: n = 100).

5.1.2. Results

Figure 5 indicates the performances of the four conditions in each Ca/Cm level, showing the ratio of

[2] The !eval! condition is provided to allow the modeler to add any arbitrary conditions to a production rule.

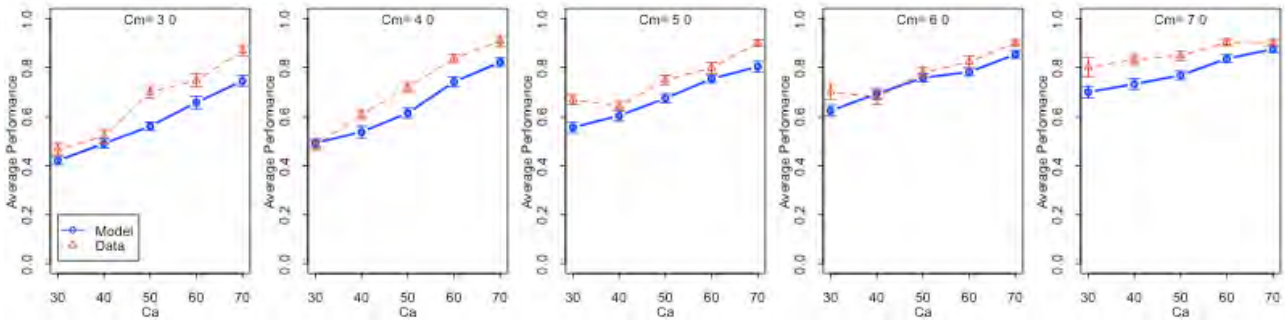

Figure 6. Performance of the simulation 2. Error bars represent standard error of means (SEM).

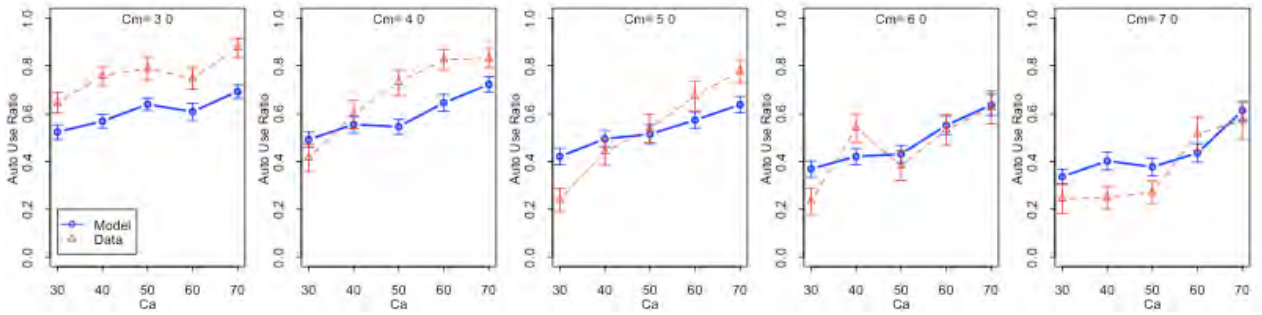

Figure 7. Auto use ratio of the simulation 2. Error bars represent standard error of means (SEM).

time that the vehicle is on the line. From this figure, it can be observed that the performance of the all four lines increases with higher Ca / Cm levels, consisting with the manipulations of capability. In addition, we can confirm that the auto controls are better than the manual controls in both the experiment data and the simulation. This result indicates the manual disadvantages in this task. Although the performance of model is relatively lower than that of the data, the correlations between the experiment and the simulation are high [Auto: r^2 = .994, p < .01. Manual: r^2 = .996, p < .01].

5.2. Simulation with two modes

This simulation is conducted to simulate experiment 2, which specify the automation use ratio.

5.2.1. Method

In experiment 2, the participant (n = 23) could use the auto control mode. They conducted the task in 25 conditions where Ca and Cm levels were manipulated (5 levels of Ca ranging from 30% to 70% vs. 5 levels of Cm ranging from 30% to 70%). Similarly, the model conducted the task choosing two modes of control in the 25 conditions (n = 50). In each condition, the model conducted the task for 40 seconds. The initial mode was randomly set in this simulation.

5.2.2. Results

Figure 6 presents the performance (on line ratio) of the model and the experimental data. Each of the five graphs indicates the performance in each Cm level, and the horizontal axis of the each graph indicates Ca levels. The figure indicates an increase of the performance of the model and the experiment data with higher Ca and Cm levels. The correlations of the model and the data are high [r^2 = .953, p < .01] although some differences between the model and the data can be observed. For example, the model is lower than the data in combination of the low Cm and the high Ca levels (e.g., Cm = 30 / Ca = 70). Similarly, the performance of the model fell below that of the data in combination of the high Cm and the low Ca level (e.g., Cm = 70 / Ca = 30). These differences suggest some difference in automation use between the model and the data.

Figure 7 indicates the auto use ratio in each Ca and Cm level, which represents how long the auto mode is used during the task. Comparison of the five graphs reveals decreases of auto use ratio with increases of the Cm level. We can also see an increase of the auto use ratio with increases of the Cm level from each graph. The model shares these tendencies with the data, and we obtained a significant correlation between the data and the model [r^2 = .718, p < .01]. These results suggest that the adaptive learning on the mode selection was made in both the experiment and the simulation. The model is, however, less adaptive compared to the data. The auto use ratio of the model is lower in the low Cm levels such as Cm = 30, and the model's lines is flatter than the data's line in Cm = 50.

6. Implications and Future work

The results of the simulations show overall correspondence with the experimental data, suggesting some validity of the assumptions made in our model. This study presents an ACT-R model that simulates human-automation interaction. We consider that this study is characterized as the connection of the cognitive process model (ACT-R) to an abstract model (EDFT). Utility update equation of ACT-R is almost same to Belief and Trust update equation used in the EDFT model. Unlike the previous model, our model has knowledge to execute a task and simulates performing the task.

Figure 8 summarizes the process of our model comparing the EDFT model presented in Figure 1. As the figure indicates, our model does not receive Ca/Cm directly. Randomized course conditions influence the performance (success / failure) of the task. Moreover, complex perceptual / motor factors are involved in the manual mode performance. As Bainbridge (1983) implied, to understand decision-making about the use of automation one needs to consider monitoring the auto and manual performance. This study is a first step to include performance factors into a modeling use of automation. We believe that our approach is useful to understand human-automation interaction because the performance and the auto use ratio usually interact with each other in many situations involving automation.

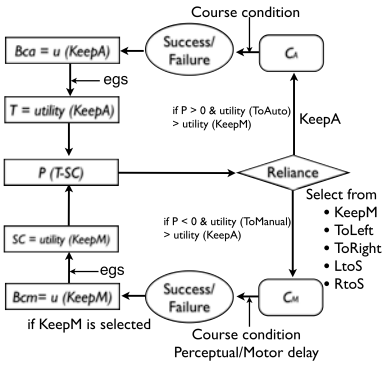

Figure 8: Correspondence with the EDFT model.

Our model is also different from previous models of strategy selection in ACT-R. Unlike the previous studies, our task requires and uses a complex perceptual / motor process. In such a situation, rewarding behavior is not easy problem. We introduced a function to monitor self-confidence and trust to solve this problem. We consider that the condition comparing utility values represents a type of meta-cognitive decision-making. Our model suggests the default sub-symbolic computation is not enough for explaining the use of this automation.

However, using an !eval! condition is not supported by ACT-R theory. ACT-R is designed as a unified cognitive theory that combines sub models from various fields (Anderson, 2007). It is difficult to combine sub models using !eval! conditions. Therefore, we need to consider other methods of modeling this mechanism, which can contribute the development of a unified theory of cognition, particularly meta-cognition.

In this study, there are several other limitations as a model of automation reliance. First, as shown in Figure 7, the human participants made better choice in each experimental condition compared to the model. This result indicates the need to explore more adaptive learning mechanisms to simulate and predict human-automation interaction. Second, this paper did not show detailed analysis concerning automation use. To confirm validity of the model, we need to examine matching of the model and the data in frequency and timing of mode switching during the task. Third, the simulation presented in this paper did not directly address the problem of misuse and disuse biases discussed in section 1. To address this problem, we need to develop a transformation technique of the auto use ratio to adjust a manual disadvantage of our task, and to obtain an optimal auto use ratio. Maehigashi et al. (in press) conducted a transformation of human auto use ratio using a non-linear regression technique. Our future study will apply this method to our model data. Finally, there is a limitation concerning a task setting. The task used in this study is relatively simple and artificial. Connecting to more complex tasks is required to extend our model to more realistic situaion. The factors involved in automation use are broad. In the future study, we explore other factors of automation use such as cognitive load, emotion, mental model, and individual differences.

7. Acknowledgements

This study is supported through the CREST program from the Japan Science and Technology Agency (JST), the Institutional Program for Young Researchers Overseas Visit from the Japan Society for the Promotion of Science (JSPS), and DTRA (HDTRA1-09-1-0054). The authors wish to acknowledge the helpful comments and suggestions made by Dan Bothell, William Kennedy, Christian Lebiere, Walt Mankowski, Ling Rothrock, Matthew Walsh, and John Yen.

8. References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Bahner, J. E., Huper, A. D., & Manzey, D. (2008). Misuse of automated decision aids: Complacency, automation bias and the impact of training experience. *International Journal of Human-Computer Studies, 66 (9)*, 688-699.

Bainbridge, L. (1983). Ironies of automation. *Automatica, 19(6)*, 775–779.

Beck, H. P., Dzindolet, M. T., & Pierce, L. G. (2007). Automation usage decisions: controlling intent and appraisal errors in a target detection task. *Human Factors, 49(3)*, 429–437.

Beck, H. P., McKinney, J. B., Dzindolet, M. T., & Pierce, L. G. (2009). Effects of human-machine competition on intent errors in a target detection task. *Human Factors, 51(4)*, 477-486.

Dzindolet, M. T., Peterson, S. A., Pomranky, R. A., Pierce, L. G., & Beck, H. P. (2003). The role of trust in automation reliance. *International Journal of Human-Computer Studies, 58(6)*, 697–718.

Dzindolet, M. T., Pierce, L. G., Beck, H. P., & Dawe, L. A. (2002). The perceived utility of human and automated aids in a visual detection task. *Human Factors, 44(1)*, 79–94.

Gao, J., & Lee, J. D. (2006). Extending the decision field theory to model operators' reliance on automation in supervisory control situations. *IEEE Transactions on Systems Man and Cybernetics, 36(5)*, 943–959.

Lee, J. D., & Moray, N. (1994). Trust, self-confidence, and operators adaptation to automation. *International Journal of Human-Computer Studies, 40(1)*, 153–184.

Lebiere, C., Gonzalez, C., & Warwick, W. (2009). Convergence and constraints revealed in a qualitative model comparison. *Journal of Cognitive Engineering and Decision Making, 3(2)*, 131-155.

Lovett, M. C., & Anderson, J. R. (1996). History of success and current context in problem solving: Combined influences on operator selection. *Cognitive Psychology, 31*(2), 168–217.

Maehigashi, A., Miwa, K., Terai, H., Kojima, K., Morita, J., & Hayashi, Y. (in press). Experimental investigation about misuse and disuse in human automation system interaction. *Proceedings of HCI International 2010*.

Parasuraman, R., & Riley, V. (1997). Humans and automation: Use, misuse, disuse, abuse. *Human Factors, 39(2)*, 230–253.

Ritter, F. E., Kukreja, U., & Amant, R. S. (2007). Including a model of visual processing with a cognitive architecture to model a simple teleoperation task. *Journal of Cognitive Engineering and Decision Making, 1(2)*, 121–147.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors, 48,* 362-380.

Sutton, R.S., & Barto, A.G. (1998). *Reinforcement learning: an introduction*. Cambridge, MA: MIT Press.

Singh, I. L., Molloy, R., & Parasuraman, R. (1997). Automation-induced monitoring inefficiency: role of display location. *International Journal of Human-Computer Studies, 46(1)*, 17–30.

Skita, L. J., Moiser, K., & Burdick, M. D. (2000). Accountability and automation bias. *International Journal of Human-Computer Studies, 52(4)*, 701–717.

Taatgen, N. A. (2005). Modeling parallelization and speed improvement in skill acquisition: from dual tasks to complex dynamic skills. *Cognitive Science, 29,* 421-455.

Vries, P. D., Midden, C., & Bouwhuis, D. (2003). The effects of errors on system trust, self-confidence, and the allocation of control in route planning. *International Journal of Human-Computer Studies, 58(6)*, 719–735.

Author Biographies

JUNYA MORITA is an assistant professor in Japan Advanced Institute of Science and Technology. His research interests include cognitive science, design cognition, cognitive modeling, and human factors.

KAZUHISA MIWA is a professor in Nagoya University. He is investigating human higher order cognition such as creativity based on the model based approach and the human experimentation method.

AKIHIRO MAEHIGASHI is a Ph.D. student in Graduate School of Information Science at Nagoya University. His research interests include cognitive science and human-computer interaction.

HITOSHI TERAI is an designated associate professor in Nagoya University / CREST-JST(Japan Science and Technology). His research topics focus on cognitive science studies including problem solving, information seeking behavior, and human system interaction.

KAZUAKI KOJIMA is an assistant professor in Waseda University. He is interested in human and computer creativity, and is engaged in research of intelligent supporting systems.

FRANK E. RITTER is a professor in the College of Information Sciences and Technology at the Pennsylvania State University.