Realization of the Concept of Responsiveness into a Computer System Using the Blackboard Architecture

- '흑판' 구조를 사용한 컴퓨터 시스템에의 '적절 반응' 개념 실현 -

Kwang Seok Lee*1)
이 광석
Byung Wook Lee*
이 병욱
Han Suk Ko*
고 한석
Byung Chool Won*
원 병출

요 지

본 논문은 상대방의 변화하는 필요사항을 충족시키기 위하여 사람들이 일반적으로 채택하고 있는 '적절 반응'(responsiveness) 과정을 컴퓨터 시스템에 적용하는 것에 관한 연구이다. '적절 반응' 과정은 일반적으로 관찰, 이해, 해석, 이행의 4단계로 구성된다. 본 연구에서는 이러한 '적절 반응' 과정을 '흑판'(blackboard) 구조를 이용하는 '반응 층'(responsive layer)을 통하여 MSLTRAIN이라고 하는 실제 시스템에 적용하였다. '반응 층'은 '적절 반응' 과정을 실질적으로 이행하는 '지식 소자'(knowledge source)와 정보의 상호 소통을 위한 '흑판'으로 구성된다. '적절 반응' 과정과 이를 이행하는 '반응 층'은 변화하는 사용자의 욕구를 충족시켜 나가는 지능을 제공한다.

1. Introduction

Many people talked about a computer system responsive to the user's needs [8, 10, 20, 21]. However, they didn't provide exact meaning of responsiveness and any prototype responsive system. This paper is about the concept of responsiveness in computer systems. This paper discusses the conceptual background of the concept and a prototype responsive system called MSLTRAIN.

The term responsiveness comes from the human world. Responsiveness is a valuable property human beings or groups of human beings have shown for their success. We can think of lots of examples: responsive secretaries to their bosses[4], responsive nurses to their patients[17], responsive salespeople to their customers[9], responsive governments to their people[27], responsive schools to the parents of their students[12], responsive teachers to their students[23], etc. For example, a responsive secretary is willing and able to listen to and understand the boss's needs and provides appropriate and timely services. Responsiveness implies a reasoning process occurring inside the human being. If we

* 한국원자력연구소(Korea Atomic Energy Research Institute)

define the process and transfer it to computer systems, we can have computer systems responsive to the user's needs.

When we talk about responsiveness, we assume there are two components: a responsive server(computer system) and a receiver to which the server is responsive(user). The goal of responsiveness is to satisfy the user's needs by providing appropriate responses. This paper explains the responsiveness process and presents a prototype responsive system into which the responsiveness process has been implemented.

2. The Responsiveness Process

Responsiveness is a property of the server toward the user. In computer systems, we can think of two kinds of situations in which the user's needs changes: 1)different users are using the system, and 2)the same user is using the system at different times. If the system's response doesn't evolve as the user's needs changes, then we need change the state of the system to match the changed needs of the user. Usually system designers do the change process, which limits the capability of the tool to respond to changes in the user's needs. If we transfer the role of the designer to the system, it will improve the system's capability to respond to the changes. Responsiveness implies the system's automatic change in the system's projection, by changing its state to match the change in the user's needs caused by the change in the user's state.

Responsive process is a process to infer the user's needs and provide responses appropriate for the user's needs. I divide the process into four stages: observing, understanding, interpreting and implementing. With these four stages, the responsiveness process is continuous and cyclic. This cyclic feature of responsiveness implies learning about the user. As the cycles go on, the server learns about the user and uses that knowledge in later reasoning. This learning aspect is vital for responsiveness.

2.1 Observing

Responsiveness starts from observation. By becoming a better observer, a responsive secretary can draw conclusions that will enable him or her to anticipate certain actions[4]. The responsive secretary observes the boss and the surrounding environment. From the observation, the secretary gets data about the boss. Observation is gathering data about the user with all perception devices. The issue here is what to observe and how to observe. The secretary knows what is important to the boss and therefore knows what to observe about the boss. The secretary has observing devices such as eyes, ears, feeling, etc. and has observing management processes such as a regular review of the boss' calendar.

In computer systems, the primary way of observing is monitoring the user's interaction with the system. Monitored data can be used: in diagnosing and providing the user appropriate aid, in assessing user performance and user satisfaction, etc.[20]. Other primitive forms of observing may be possible. For example, Starker & Bolt[26] use eye-tracking to monitor the user looking at a graphics screen in a gaze-responsive information display system. Computer vision and speech recognition will extend the meaning of observation in terms of both what to observe and how to observe in computer-based management tools.

2.2 Understanding

The second stage is understanding. Having observed data and prior knowledge about the user, the server reasons what the user really wants or needs. In this stage, the data about the user are transformed to information about the user. For example, the responsive secretary reasons that the boss needs information showing the trend of recent sales after reviewing the incoming letters and remembering the boss preferred information about the matter in the letters. In computer systems, it means to reason the goal of the user from the input of the user to the system and the context and then to reason the needs of the user from the goal of the user and the characteristics of the user. Here all reasoning occurs in terms of the user.

2.3 Interpreting

The third stage is interpreting. The server reasons what it can do to satisfy the user's needs, that is, the result of understanding. In this stage, a decision on what and how to respond is made based on information about the user. For example, the responsive secretary decides to obtain a specific line chart showing the trend of recent sales. In computer systems, this stage is the design of appropriate responses to the user's needs. Here all reasoning occurs in terms of the system.

2.4 Implementing

The fourth stage is implementing. The server provides the result of interpreting, that is, a specific response. The responsive secretary prepares a line chart and attaches it to the letters. This stage implies capabilities of the server actually doing something, such as functionality and user interface capability. From the management perspective, if I say the first three stages correspond to information gathering and decision making, then this last stage corresponds to taking action. Without action, the decision is meaningless[6].

3. MSLTRAIN: A Prototype Responsive System

I implemented the responsiveness process into a decision support system called MSLTRAIN. MSLTRAIN was developed to help managers schedule their training needs on computer packages. I used the Texas Instruments Explorer AI workstation with LISP. I added a responsive layer onto the original application program of MSLTRAIN. The responsive layer essentially overlays the application program and upgrades the system's responsiveness in both decision support and interaction support. The major feature of MSLTRAIN with the responsive layer is to provide information the user may need in the format (table or graph) and in the scope (whole or part) the user may prefer even without the user's request.

3.1 Overall Working Mechanism

MSLTRAIN works within two processes: the application process and the responsive layer process. The responsive layer process works as a layer on the application process. The responsive layer process continuously observes the application process and designs or modifies system parameters. If the user inputs something to the system, the application process accepts the input, which invokes the responsive layer process. Then the responsive layer process analyzes the input, reasons the user's goal, designs an appropriate response, and shows the response or sets the values of some system parameters the application process needs.

The connection between the two processes is realized using reporters and executors. The reporter resides in the application process as a demon and reports to the responsive layer process what happens in the application process such as the user's input, end of processing of some function in the application process, etc. The reporter has knowledge of what and when to report. The executor also resides in the application process as a demon and executes what the responsive layer process orders. Through the executor, the responsive layer process can change responses generated by the application process.

3.2 Architecture of the Responsive Layer

I applied the blackboard architecture to the design of the responsive layer. Several researchers support the blackboard architecture for modeling the user and designing responses to the user[1, 2, 5, 16, 25]. The blackboard architecture has been used in several areas such as speech recognition[7], errand planning[11], cooperative distributed systems [14], cockpit information management[3], self-improving instructional planner[15], etc. The blackboard architecture is a problem-solving framework based on the distributed problem-solving paradigm, which means the cooperative solution of problems by a collection of problem-solving experts when one expert doesn't have enough knowledge to solve the problem. The blackboard architecture usually consists of three major components: knowledge sources, blackboard data structure, and a control mechanism[18, 19].

3.2.1 Blackboard

A blackboard is a structured, globally accessible database containing intermediate or partial results of problem solving needed by and produced by knowledge sources. Typically, the blackboard is partitioned for representing hypotheses at different levels of abstraction and mediates the cooperative activities of multiple knowledge sources. Knowledge sources interact with each other only through the blackboard.

MSLTRAIN maintains a blackboard consisting of seven hierarchical levels: input, history, semantics, goal, user characteristics, needs, and design. The levels hold intermediate solution results of the responsiveness process as messages.

Input

This level keeps the primitive form of the user's input. Messages in this level contain the following information: type of input (command or continue), time when the user made the input, content of input, input device (keyboard or mouse), input window (menus, decision windows), input reference (the name of the item selected, number of step, etc.).

History

This level keeps the history of the interaction with the user. Messages in this level contain the following information: the name of a variable and its value. For example, total number of commands, steps the user went through, total use, total number of errors, total number of inputs made with the mouse, ratio of mouse use, error ratio, etc.

Semantics

This level keeps what the input means in terms of the application. Messages in this level contain the following information: type of action, object of the action, and reference of the action.

Goal

This level keeps the goal of the user. Messages in this level contain the following information: type of goal(decision or information access), focus of the goal (selection of items in the menu, decision on a specific package, or decision in general), and content of the goal(schedule, change schedule because of the organizational proficiency, or change schedule because of person-days in some month).

User Characteristics

This level keeps the characteristics of the user. Messages in this level contain the following information: the name of a variable and its value. For example, portrayal format preference, information scope preference, input device preference, decision pattern, etc.

Needs

This level keeps the user's needs. Messages in this level contain the following information: type of needs(information, cursor move, etc), required type of action, and specification of the needs.

Design

This level keeps the design of responses. Messages in this level contain the following information: type of design and values of elements of the design. Usually, at first, the elements don't have any values. Knowledge sources fill the message until all elements get values.

3.2.2 Knowledge Sources

Knowledge sources(KS's) are problem-solving experts working together to solve a problem. Each KS has necessary but not sufficient problem-solving knowledge. However, by working together, KS's exercise full knowledge of problem solving. The objective of each knowledge source is to contribute information that will lead to a solution to the problem (Nii, 1986). KS's are kept separate and independent. They communicate through a blackboard by posting and reading messages on it.

KS's are the key to the responsiveness process. They have the knowledge of observing, understanding, interpreting and implementing.

In MSLTRAIN, knowledge sources can take the following actions to the blackboard:

- POST-MSG : post a message on a level in the blackboard,
- UPDATE-MSG : update a message already posted on a level in the blackboard, and
- FILL-MSG : fill in an attribute value to a blank message on a level in the blackboard.

Each KS corresponds to a stage in the responsiveness process: observe, understand, interpret and implement. Observing knowledge sources are triggered by the reporter's report or events generated by the change in the "INPUT" level in the blackboard. These KS's post their messages to the "SEMANTICS" level or update messages in the "HISTORY" level in the blackboard. Understanding knowledge sources are triggered by events generated by the change in the "SEMANTICS," "HISTORY," or "GOAL" level in the blackboard. These KS's post their messages to the "GOAL," the "USER-CHAR," or "NEEDS" level in the blackboard. Interpreting knowledge sources are triggered by events generated by a change in the "NEEDS" or "DESIGN" level in the blackboard. These KS's post their messages to the "DESIGN" level or fill the messages in the "DESIGN" level in the blackboard. Implementing knowledge sources are triggered by events generated by the change in the "DESIGN" level in the blackboard. These KS's display information to the user or set the values of system parameters.

3.2.3 Control Mechanism

The control mechanism specifies a general problem-solving behavior. In the blackboard architecture, there is a set of control modules that monitors the changes on the blackboard and decides what actions to take next[19]. The control modules are responsible for determining which knowledge source should act at each point in the problem-solving process. The modules use various kinds of information globally available. Several kinds of control mechanisms exist in the blackboard architecture[7, 11].

In the responsive layer process, a scheduler controls the blackboard actions: triggering, choosing, and executing knowledge sources. The scheduler maintains the following information:

- Goal : the goal to accomplish, given according to the report from the application process;
- Events : a list of events generated by a change in the blackboard;
- Triggered KSAR's : a list of knowledge source activation records (KSAR) that were triggered but didn't satisfy the precondition;
- Invokable KSAR's : a list of knowledge source activation records that were triggered and satisfied the precondition ;
- Control Strategy : a strategy to choose among the invokable KSAR's.

The scheduler goes through the following steps iteratively.

1. Check whether the goal is satisfied. If so, stop. Otherwise, go to the next step.
2. Update the set of KSAR's. Inspect all knowledge sources to determine if they can be triggered by the event created. Add the knowledge sources triggered to the triggered-KSAR list.
3. Inspect all KSAR's in the triggered-KSAR list to determine if their precondition can be satisfied with the current solution state. If so, add them to the invokable-KSAR list and delete them from the triggered-KSAR list. If the invokable-KSAR list is empty, stop. Otherwise, go to the next step.
4. Select one KSAR from the invokable-KSAR list with the control strategy. The strategy is to choose one KSAR whose To-Level is the lowest level in the blackboard among the invokable KSAR's.
5. Execute the selected KSAR. Delete the KSAR from the invokable-KSAR list. Go to Step 1.

If the scheduler finishes the steps, the responsive layer process stops and waits until there is new report from the application process.

4. Conclusion

This paper presented the conceptual background of responsiveness in computer systems and a prototype responsive system implementing the concept. This paper provides a way for a computer system to respond to needs varying among different users(individual difference) and evolving over time(evolution of the user). I expect responsive systems to provide more effective support to users and result in higher user satisfaction, higher user performance, and ultimately, higher success rates and more sharing of computer systems. To a limited degree, a laboratory experiment supports that responsiveness in a computer system improves user performance and user satisfaction[13].

References

[1] Alty, J. L., & McKell, P., "Application Modeling in a User Interface Management System", In: Harrison, M. D., & Monk, A. F. (eds.), *People and Computers: Designing for Usability: Proceedings of the Second Conference of the British Computer Society Human Computer Interaction Specialist Group*, Cambridge: Cambridge Univ. Press, 1986.

[2] Balzert, Hemut, "A Blackboard Architecture for the Realization of Software-Ergonomic Demands", In: Bullinger, H. J, & Shackel, B. (eds.), *Human-Computer Interaction: INTERACT '87*, New York: North-Holland, 1987.

[3] Baum, L., Kaiser, K, Blevins, D., Miller, B., Jagannathan, V., & Anderson, M., *Adapting the Blackboard Model for Cockpit Information Management*, Boeing Technical Report, 1987.

[4] Belker, Loren B., *The Successful Secretary: You, Your Boss, and the Job*, New York: AMACOM., 1981.

[5] Belkin, N. J., Hennings, R. D., & Seeger, T., "Simulation of a Distributed Expert-based Information Provision Mechanism", *Information Technology*, 3(3), 122-141, 1984.

[6] Drucker, Peter F., *The Effective Executive*, Pan Books Ltd., 1967.

[7] Erman, L. D., Hayes-Roth, F., Lesser, V. R., & Reddy, D. R., "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty", *ACM Computing Surveys*, 12(12), 213-253, 1980.

[8] Fitter, Mike, & Sime, Max, "Creating Responsive Computers: Responsibility and Shared Decision-Making", In: Smith, H. T., & Green, T. R. G. (eds.), *Human Interaction with Computers*, New York: Academic Press., 1980.

[9] Grikscheit, Gary M., Cash, Harold C., & Crissy, W. J. E., *Handbook of Selling: Psychological, Managerial, and Marketing Bases*, New York: John Wiley & Sons, 1981.

[10] Hayes, Eugene B., & Reddy, Raj, "Breaking the Man-Machine Communication Barrier", *Computer*, March, 19-30, 1981.

[11] Hayes-Roth, Barbara, "A Blackboard Architecture for Control", *Artificial Intelligence*, 26, 251-321, 1985.

[12] Johnston, William F., *Responsiveness in American Schools Overseas: Discrepancies between Parental Expectations and School Performance*, Unpublished Ph.D. Dissertation, Virginia Tech, 1988.

[13] Lee, Kwang S., *Operationalizing and Implementing the Concept of Responsiveness in a Management Tool*, Unpublished Ph.D. Dissertation, Virginia Tech, 1991.

[14] Lesser, Victor R., & Corkill, Daniel D., "Functionally Accurate, Cooperative Distributed Systems", *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1), 81-96. 1981.

[15] Macmillan, Stuart A., & Sleeman, Derek H., "An Architecture for a Self-improving Instructional Planner for Intelligent Tutoring Systems", *Computational Intelligence*, 3, 17-27, 1987.

[16] McCalla, Gordon I., Bunt, Richard B., & Harms, Janelle J., "The Design of the SCENT Automated Advisor", *Computational Intelligence*, 2, 76-92, 1986.

[17] Murray, Malinda, *Fundamentals of Nursing*, Englewood Cliffs: Prentice-Hall, 1980.

[18] Nii, H. Penny, "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architecture", *AI Magazine*, 7(2), 38-53, 1986.

[19] Nii, H. Penny, "Blackboard Systems: Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective", *AI Magazine*, 7(3), 82-106, 1986.

[20] Penniman, W. D., & Dominick, W. D., "Monitoring and Evaluation of On-line Information System Usage", *Information Processing & Management*, 16, 17-35, 1980.

[21] Rich, Elaine, *Building and Exploiting User Models*, Ph.D. Dissertation, Carnegie-Mellon University, 1979.

[22] Rubin, Kenneth S., Mitchell, Christine M., & Jones, Patricia, "Using a Blackboard Architecture for Dynamic Intent Inferencing", *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 1150-1153, 1987.

[23] Sleeman, D. & Brown, J. S. (eds.), *Intelligent Tutoring Systems*, New York: Academic Press, 1982.

[24] Smith, J. Jerrame, "SUSI - A Smart User-System Interface", In: Johnson, Peter, & Cook, Stephen (eds.), *People and Computers: Designing the Interface - Proceedings of the Conference of the British Computer Society Human Computer Interaction Specialist Group*, Cambridge, Cambridge Univ. Press, 1985.

[25] Smith, Reid G., & Davis, Randall, "Frameworks for Cooperation in Distributed Problem Solving", *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1), 61-70, 1981.

[26] Starker, India, & Bolt, Richard A., "A Gaze-Responsive Self-Disclosing Display", *Human Factors in Computing Systems: Proceedings of CHI '90 Conference*, 3-9, 1990.

[27] Zeigler, L. Harmon, & Tucker, Harvey J., *The Quest for Responsive Government: An Introduction to State and Local Politics*, North Scituate: Duxbury Press, 1978.