Development of analytic rules and a support system for logical interface evaluation

Ho kyoung Ryu
IE Dept. KAIST, 373-1
Kusong-dong,Yusong-gu
Taejon , Korea
Tel: 82- 42- 869-3159
Email:hogg@cogsys.kaist.ac.kr

Wan Chul Yoon
IE Dept. KAIST,373-1
Kusong-dong, Yusong-gu
Taejon, Korea
Tel:82-42-869-3119
Email:wcyoon@sorak.kaist.ac.kr

Dongseok Lee
IE Dept. KAIST, 373-1
Kusong-dong,Yusong-gu
Taejon, Korea
Tel: 82- 42- 869-3159
Email: dslee@cogsys.kaist.ac.kr

ABSTRACT

Most of the previous user interface analyses have aimed at evaluating logical complexity and consistency on user interface. However, interface problems in real life are dealt with by users whose knowledge plays a key role in understanding and using interface. We need an analysis method that could explicitly take user¡¯s expectation or task knowledge into account. In this regard, Yoon proposed that features in an interface should be matched with corresponding task knowledge of users.
This study expands this viewpoint with an automatized evaluation and a support system for it. The results are an easy and objective analysis and evaluation on user interface.
The implemented system (Evaluation System for Task Interface Matching ;ESTIM) explicitly represents a lot of user¡¯s knowledge and logical procedures in the interface and then matches the two using a rule-based system. The ESTIM evaluates the logical characteristics that are defined within the interface such as operation images, procedural consistency, and matching with user¡s expectation.
The analytic method and ESTIM are demonstrated with a practical appliance, pager. The example shows that the implemented system has a good degree of explanatory power and could be effective in practical interface analysis process.

KEYWORDS: Consistency, Congruity, User interface analysis, and Task knowledge of users

INTRODUCTION

Our life is surrounded with intelligent products of more complex and diverse function. In real situation, users would use products in different ways that designers expect. In that situation, on committing errors, users experience frustrations and then resign their right to use high-cost function. So, product designers should try to reflect user¡s needs into implementing products. Up to present, in the studies on analysis and evaluation, researchers have mainly studied the logical consistency and complexity on interface in a product. However, users feel discomfort when a product has different behaviors with user¡s knowledge, so we need the cognitive evaluation method based on user¡s knowledge for getting an interface understood. Under this motivation, we have developed a support system and novel evaluative rules based on user¡s knowledge for analysis on logical interface.

In the following sections, we will explain and validate a framework and support system through an example of pager.

METHODOLOGY FOR LOGICAL INTERFACE ANALYSIS AND EVALUATION

The formal methods for analysis and evaluation on an interface, heuristic evaluation and craft approach, would take a lot of time and do not produce the consensus of result by analysts. Therefore, we need a system which supports an analysis of logical interface independent of analyst¡s ability. The methodology presented in this study seeks to overcome these difficulties.

Representation of logical interface

The first stage for analysis is representing of a logical interface. Multiple approaches to formal specification of interface have been suggested. Examples include state-transition diagram, BNF, Petri nets, production systems, logic programming, and temporal logic. In spite of this extensive research activity, almost no formal specification is currently used widely [3][4]. Therefore, we developed a new model for specifying the logic of interface centered on user¡s action. It is called OCD [1]. The OCD give the methods for specifying cognitive process and representing users¡ action in performing a task. The Figure 1 depicts the entities for expressing task procedure in OCD and Figure 2 exemplifies an OCD representation of task ¡Set date¡ in a pager.

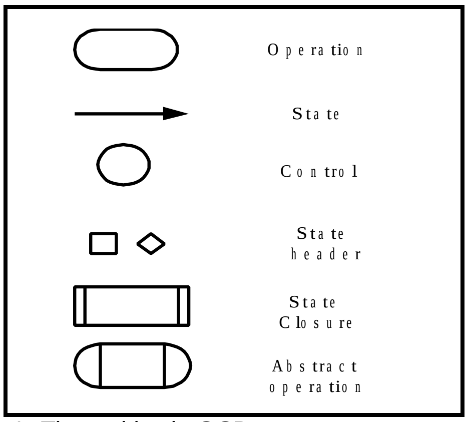


Figure1: The entities in OCD.

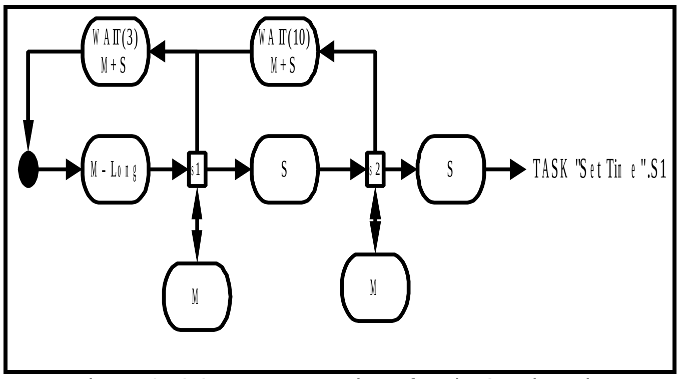


Figure 2: OCD representation of task ¡Set date¡ in pager.

An operation is an elementary action by the user that produces a system response or a state change. Some operations are useful only for changing states but not for getting system responses. Such navigational operations are called control. A state is a situation which is distinguished from another by the availability and behavior of operations in it. A state header indicates that the subsequent state(s) is altered from the preceding one(s) in some specified ways. Also, some state headers require information generated in the history of previous operations and called memory headers. There are two types of clusters in OCD. An abstract operation is a set of consecutive operations that is integrated by a common function or goal. Once learned, an abstract operation may seldom be dealt with in terms of its elements, which is a benefit of abstraction. A state closure is a set of contiguous states that allow the group of operations which are semantically related. While an abstract operation and its elementary operations typically have whole-part relationship, the operations or subtasks contained in a state closure tend to share semantic affinity [1].

Go into details of Figure 2, shown user¡s action sequences for accomplishing a task ¡Set Date¡. Firstly, Button ¡M¡ is pushed long, then menu icons appear in the LCD of pager. This state is ¡S1¡ of task ¡Set Date¡. At ¡S1¡, user can select a menu, ¡Date¡, with scrolling by button ¡M¡. Also, at ¡S1¡, user can go ready-state for recovering their error or other causes. For this behavior, user should wait for 3 seconds or push button ¡M¡ and ¡S¡ at the same time. In this way, OCD is designed to be visual and intuitive for one to identify a task ¡Set Date¡.

Also, Diagram as Figure 2 is converted automatically to a script-based fact list for rule-based system in a support system. The following box is a fact list corresponding to Figure 2.

ϒProcedural Facts	ϒ Function Facts	ϒResponse Facts
(Sleep ; M-Long ; SH(S1))	SHOW(MENU)	SHOW(MENU) & BLINK(MENU)
(SH(S1) ; Wait(3)||M+S ; Sleep)	CANCEL	SLEEP
(SH(S1); M ; SH(S1))	CHANGE(MENU)	CHANGE(MENU)
(SH(S1); S; SH(S2))	NEXT	!BLINK(MENU) & SHOW(DATE) & BLINK(DATE)
(SH(S2); Wait(10)||M+S ; Sleep)	CANCEL	SLEEP
(SH(S2); M; SH(S2))	CHANGE(DATE)	CHANGE(DATE)
(SH(S2); S ;Task ¡Set Time¡. S1)	NEXT	!SHOW(DATE) & !BLINK(DATE) & SHOW(HR) & BLINK(HR)

Representation of users¡ knowledge

We need to acquire user¡s prior-knowledge, metaphors or analogies about a product for cognitive evaluation. In this study, we divided the knowledge for users to employ into four-level categories based on cognitive process: Means-end structures of tasks, Organization of operations, User¡s procedural knowledge, and Familiar patterns of controls [1].

Means-ends structures of tasks(MES)

The primary structure of user¡s task knowledge is means-ends structure that can be drawn in the form of goal tree [1][4]. Users have many goals in process of performing a task and simultaneously a lot of methods to achieve them. Especially, users will perform a novel action based on difference between goal and present state. For example, the task ¡Viewing¡ in the VCR has two subgoals: playing, and searching. Searching is achieved by two methods: forward scanning and backward scanning. From this structure, user may expect that forward and backward searching have similar behaviors. Therefore, the logical interfaces should have reflection on MES which is one of expectation of users.

So, this viewpoint should be a measure for interface evaluation.

Organization of operations.

The primitive operations are understood by the user mainly in terms of two types of organization [1]: semantic affinity of tasks and whole-part relationship. Affinity between subtasks and operations make a user expect two aspects from the interface. First, the members with closer affinity should share more similar grammatical behavior. Second, the family may well be included into a cluster. If an interface is not designed for fitting this behaviors, then user may memorize each task procedure in a cluster with separation. Therefore, this task knowledge of user should be reflected in a logical interface.

Users have whole-part relationships among operations to be implemented in an interface. Users will often memorize the consecution of operations as semantic units based on their knowledge with implication. Accordingly, system responses which are distanced from user¡s expectation will prevent users from making relationship of operations.

User¡s procedural knowledge

Users know some natural orders of operations that are more or less generic to the task or got through interaction with different products or systems. There are three important procedural knowledge structures: Sequence, branch and loop. In Automatic Teller Machine example, when a user sends money, the user may have the sequence knowledge that after typing an account of receipt, key in an amount of money. Also, users may want to send money to several accounts without having to key in your account number each time. This is the loop knowledge of users.

Familiar patterns of controls

There is a few well-known control patterns in logical interface. Toggle button is one of them. Among others there are on and off switch, one-to-one correspondence between operations and responses, mutually exclusive radio button states, and resetting to a ready-state whenever the current state is unclear.

In summary, users do not memorize complicated procedures by rote. It would be not only hard to achieve but also prone to error. The user first tries to chunk the operations into clusters, abstract operations and state closures, then into the simplified procedure, applying some typical control patterns. If same pattern does not fit everything, sibling and relatives among operations are identified using the means-ends structure to account for the deviations.

Outline of analysis and evaluation for logical interface

Analysis and evaluation on logical interface can be performed from diverse viewpoints. For example, Nielsen argued that usability problems tend to fall into the following categories: Navigation, terminology, feedback, consistency, modality, redundancies, user control, and match with user tasks [4]. These diverse viewpoints may produce inconsistent analyses or take a long time to complete an evaluation on interface. Thus, Yoon proposed an effective and cost-efficient cognitive approach including the above viewpoints, which called Task-Interface Matching, based on OCD and task knowledge of users [1]. The following Figure 3 is shown the framework of TIM.

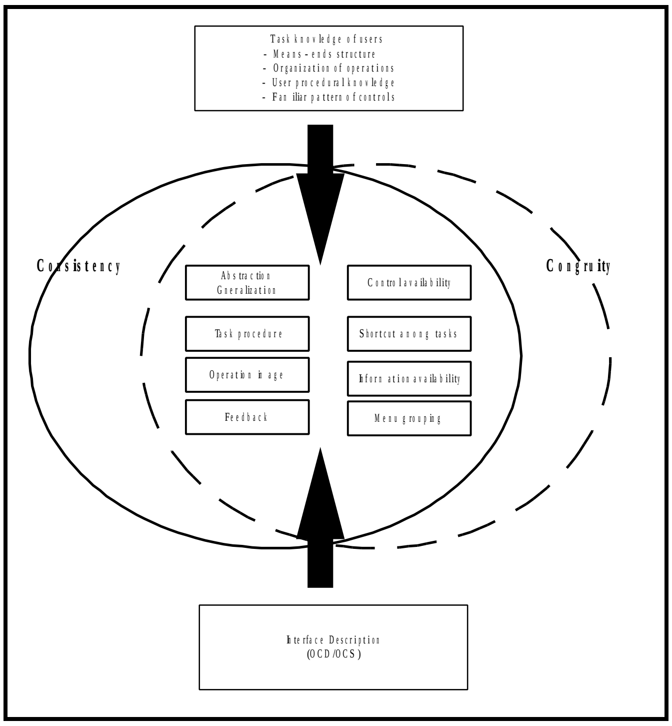


Figure 3:Framework of cognitive evaluation.

ESTIM: EVALUATION SYSTEM FOR TASK-INTERFACE MATCHING SYSTEM

Representation of logical interface in ESTIM

We developed the system for supporting the analysis framework in Figure 3, ESTIM. In ESTIM, interface representations are depicted in Figure 4. In Figure 4, OCD can be expressed through diagram format or form-filling editor. And, users tend to understand a procedure of tasks by operation level when they meet with an unfamiliar device. But, after they are accustomed to using the device, or originally they have the skill in that domain, they will usually memorize procedure by abstract level. Therefore we should consider the two levels of interface representation based on task knowledge of users. For example, in pager example, task ¡°Set Date¡± can represented in Figure 2 as primitive level and in Figure 5 as abstract level. In ESTIM, abstraction and generalization process is depicted in Figure 6. Users can select operations, states, and state headers for making a abstract operation or state closure. And then, the selected entities are endowed with a name and function(s).

Also, each operation produces a change of system response and has particular function. In ESTIM, figure 7 represents a form of expression on system response(s) and function(s).

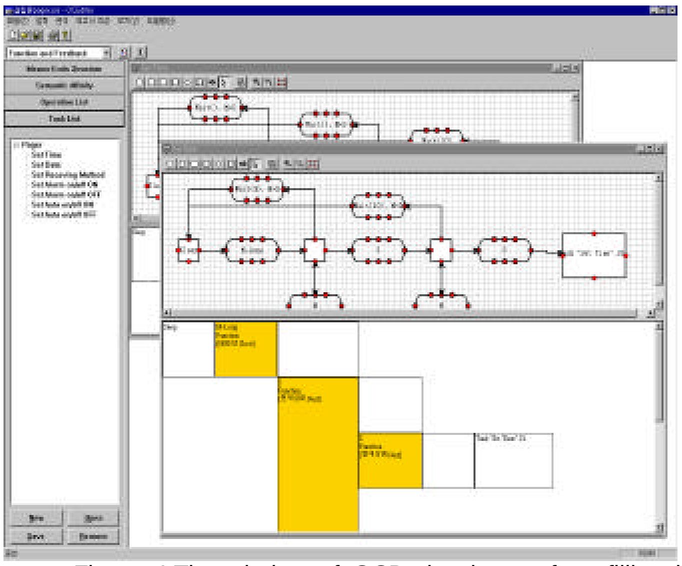


Figure 4:The window of OCD drawing or form-filling in ESTIM

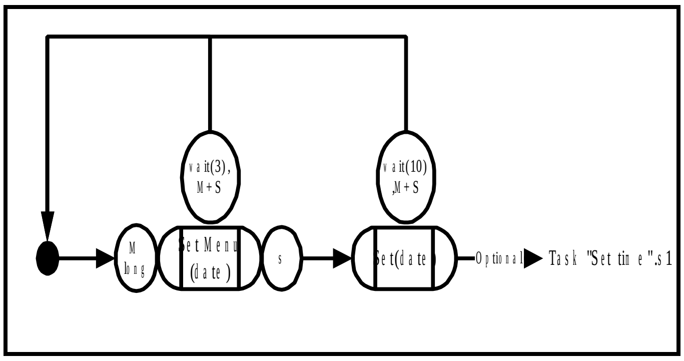


Figure 5: Abstract level of task ¡Set Date¡

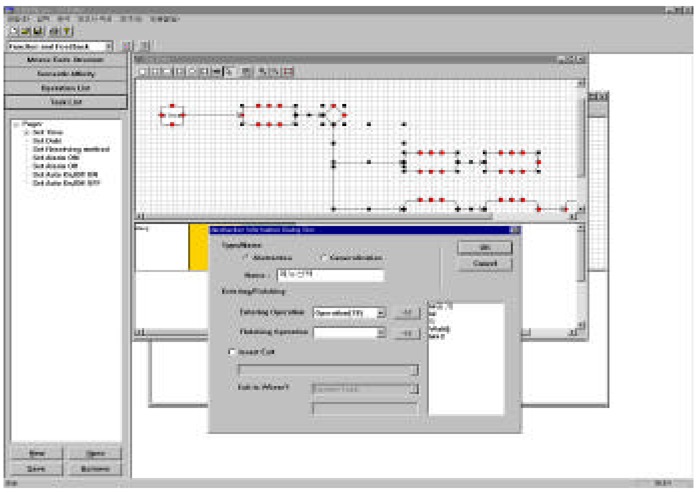


Figure 6: Abstraction and generalization process in ESTIM.

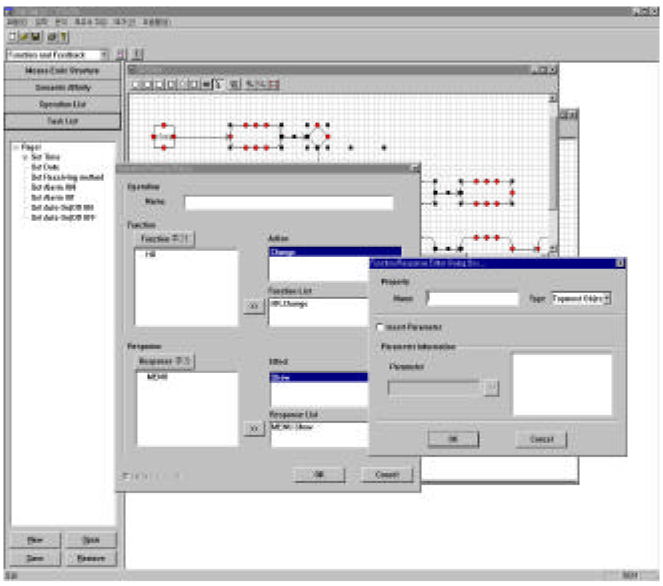


Figure 7: A form of expression about system response(s) and function(s)

Representation of users¡ knowledge in ESTIM

Users have a lot of knowledge for using intelligent products. We should use these knowledge to evaluate a user interface. These knowledge can be gathered by various methods and have different forms [4]. In ESTIM, there are various forms of expression of user¡s knowledge. Representation of MES is given in Figure 8. Especially, in USINE [6], Lecerof proposed that temporal relationship of tasks is classified into interleaving, synchronizing, enabling, enabling with information processing, choice, deactivation, and iteration. In this study, we proposed that temporal relationship of tasks which users think falls into 4 categories and 2 properties: unordered, parallel, sequential, alternative categories and optional, compulsory properties.

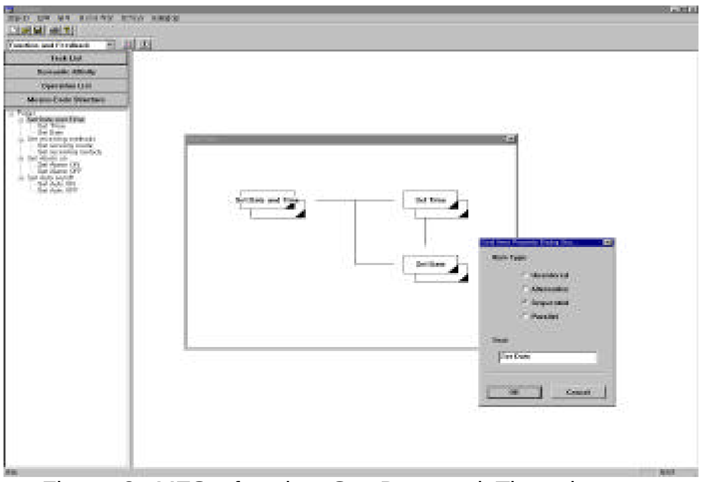
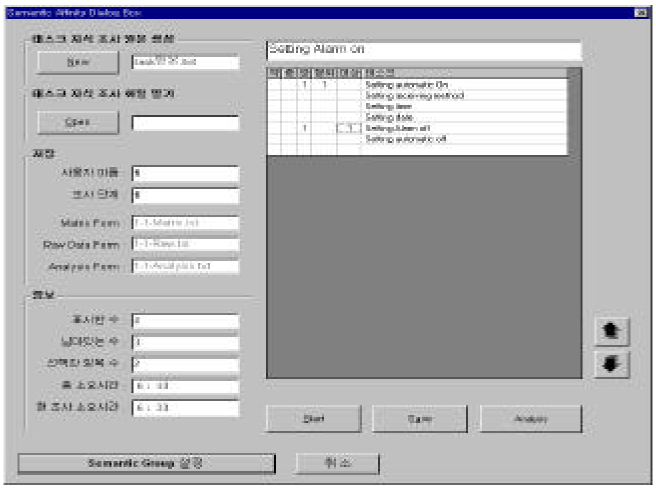


Figure 8: MES of task ¡ Set Date and Time¡ in pager example.

Also, in ESTIM, it is designed a modified clustering analysis for getting an affinity index between tasks in terms of data in Figure 9 [10][11]. These data is acquired by grading as 3-point scale(Low, Medium, High related) on each task pair. After getting data, through formulas we can calculate the semantic affinity index between tasks. For example, when we acquire data in Figure 10, with following formulas it will be calculated that semantic affinity index between task 1 and 2 is 0.6, task 1 and 3 is 0.73, task 2 and 3 is 0.55. Especially, this result is visualized by graphical type in Figure 11. In ESTIM, this semantic affinity index of task will be utilized for evaluating a congruity of menu structure and selection of tasks which should have similar grammatical behaviors.


Semantic Affinity Dialog Box

Setting Alarm on

		1	1		Setting automatic On
					Setting receiving method
					Setting time
					Setting date
		1		1	Setting Alarm off
					Setting automatic off

Figure 9: Comparison data among task in ESTIM

	Task 1	Task 2	Task 3
Task 1	3	1	1
Task 2	2	3	0
Task 3	2	1	3

Figure 10: Table of comparison among tasks. 3 represents high-related, 2 does intermediate-related, 1 does low-related. The formulas for calculating above data are following.

P_{ij} : Score of similarity in task i and j

S_i : Sum of score task i

$$Difference_{kl} = \underset{j}{\alpha}\, P_{kj} - P_{lj}, \text{ if } P_{kj} > P_{lj}$$

$$TD_{ij} = Difference_{ij} / S_i$$

$$TaskAffinity_{ij} = 1 - \frac{TD_{ij} + TD_{ji}}{2}$$

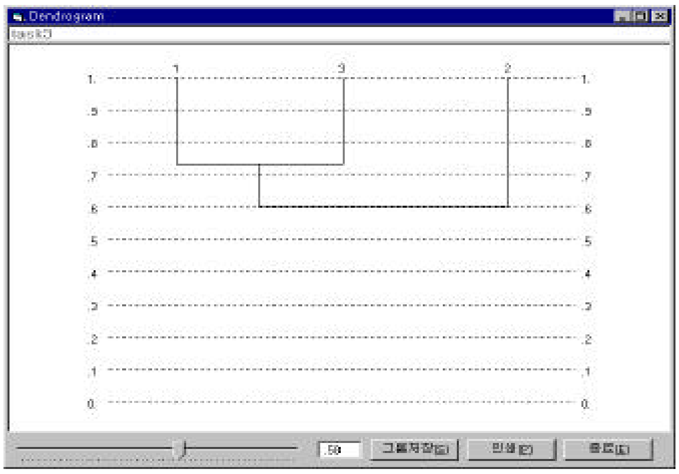


Figure 11: A graphical type presentation of semantic affinty index in ESTIM.

In ESTIM, we proposed a function diagram for representing the user¡s procedural knowledge. This is based on function analysis which describes the events that must occur for users to achieve their intended results. It doesn¡t describe physical actions that users will take or specify system actions. A function analysis obtains information on [15]:

- The relationship of the task to other tasks
- The events required in completing a task
- The functions that would need to be provided to support a task

Function diagram is made up of ¡Function¡, ¡Decision node¡, and ¡®T mflow¡. For example, in the pager, function diagram of task ¡Set ALARM ON¡ will be represented as following Figure 12.

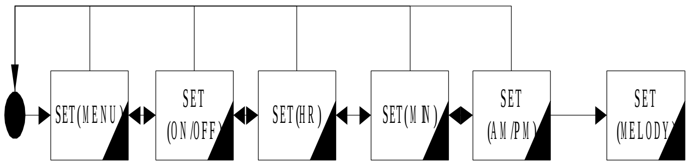


Figure 12: User¡s procedural knowledge of task ¡Set ALARM ON¡

In ESTIM there is a window for drawing a function diagram as Figure 13. However, we do not have useful manner to acquire a user¡s procedural knowledge. The formal methods are the interview technique or task analysis method. We think that ¡Users do not have the task knowledge perfectly in advance, but interacting with the interface, we form the knowledge more concretely¡. Therefore, we will give users the function-oriented item, AO, SC, and operation in ESTIM. And then, users manipulate adequately these items to represent their own procedural knowledge. After all, these user¡s procedural knowledge diagram may serve as reference materials to enhance the understanding of human-system interaction, or they can be used directly to identify training needs and contents [15].

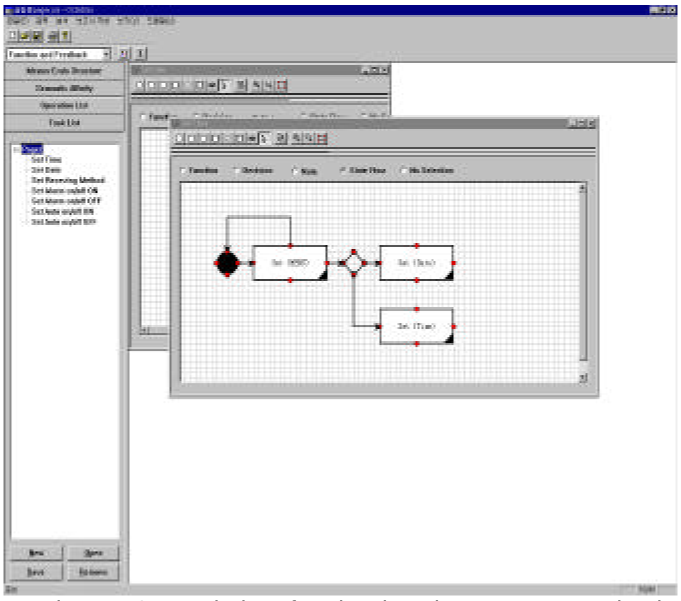


Figure 13: A window for drawing the user¡s procedural knowledge in ESTIM

There is a few well-known control patterns in an interface. We need to evaluate these in harmony with user¡s knowledge. Therefore, contents about pattern must be expressed. And then using this, check whether the operations are congruent with user¡s expression. This study uses a simple tabular format to represent the familiar pattern of controls.

Methodology for interface analysis and evaluation with ESTIM and rule-based system

Two perspectives of interface analysis are the consistency and congruity. Consistency is the most widely used measure for goodness of user interface. This is because that consistency greatly reduces complexity of required knowledge, and can be measured relatively objective. In case of TAG, it tries to measure the consistency of procedure, but it does not consider the system response[3]. Since users expect the same responses for the same operation at a similar situation, consistency test of tasks should include system response. Also, it is needed to consider the function-oriented consistency of operations. Because operation has peculiar function(s) as means to achieve the goal, each operation should have same function(s) in task space. Otherwise, users may be confused by interpreting a function in every action. And, similar tasks need to have a similar procedure, because user will understand the interface as simplified procedures.

The following analysis process is performed based on OCDs , corresponding scripts, and a lot of task knowledge with ESTIM and analytic rules.

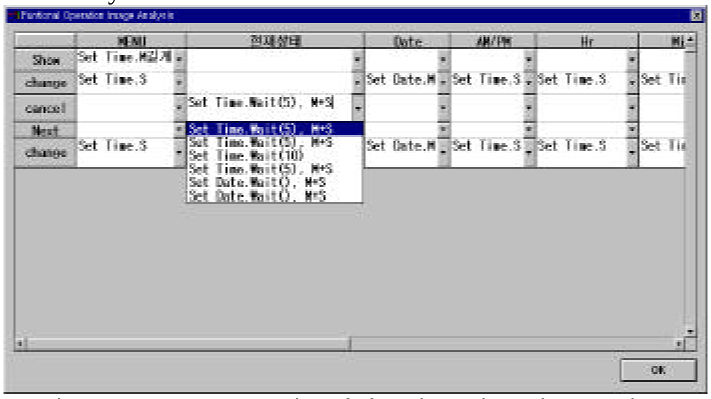
Operation image

Operations are classified into two categories. One is a primitive operation for reaching to a particular goal in performing task. The other is a navigational operation, so-called control, for reaching to a particular situation. Therefore, we have two types of operation image: primitive operation image, navigational operation image. Moreover, primitive operation image is divided into two categories from the orientation-viewpoint: function-oriented, state/response oriented. Function-oriented operation image is an analysis whether function and operation are consistent about following issues:

- Is a particular function corresponding with an operation?
- Is a way of performing a function consistent in entire task space?

We have implemented analytic rules and system to evaluate a function-oriented operation image. Figure 14 and 15 depict a function-oriented operation image in a pager by ESTIM and LISP, Figure 16 represents a control-operation image in pager by LISP.

In Figure 14,15, we can identify an inconsistency that press button ¡M¡ for changing date in task ¡Set Date¡, but button ¡S¡ for changing Hr, AM/PM, MENU in task ¡Set Time¡. That is, an action ¡changing¡ is performed by different operations in task space. We can identify an operation image in this way



	MENU	현재상태	Date	AM/PM	Hr	Mi
Show	Set Time.M길게					
change	Set Time.S		Set Date.M	Set Time.S	Set Time.S	Set Ti
cancel		Set Time.Wait(5), M+S				
Next		Set Time.Wait(5), M+S				
change	Set Time.S	Set Time.Wait(5), M+S Set Time.Wait(10) Set Time.Wait(5), M+S Set Date.Wait(), M+S Set Date.Wait(), M+S	Set Date.M	Set Time.S	Set Time.S	Set Ti

Figure 14: An example of function-oriented operation image analysis in ESTIM

```
--------------------------------------------------------------------------------
¡Set Date¡± Opeati oranalysis
--------------------------------------------------------------------------------
(M -> CHANGE(X)).
(M+S -> CANCEL(X)).
(WAIT(X) -> CANCEL(X)).
(M-Long -> SHOW(X)).
(S -> Set Time (X)OR NEXT   using --> Inconsistency).
--------------------------------------------------------------------------------
¡Set Time¡± Opeati oranalysis
--------------------------------------------------------------------------------
(M+S -> CANCEL(X)).
(WAIT(X) -> CANCEL(X)).
(M-Long -> SHOW(X)).
(S -> NEXTOR CHANGE(X)   using --> Inconsistency).
(M -> CHANGE(X)OR NEXT   using --> Inconsistency).
 (M -> CORRECT(X)OR CHANGE(X)   using --> Inconsistency).
```

Figure 15: An example of function-oriented operation image analysis in LISP

```
Abstract Operation: 'Set MIN'
Task      Entering    Finishing        Exiting              Exiting-State
AUTOON    NONE        NONE         (WAIT 10 OR M+S)      SLEEP
ALARM      NONE        NONE         (WAIT 10 OR M+S)      SLEEP
  =          =           =               M                  X1
TIME       NONE        NONE          (WAIT 5 OR M+S)       SLEEP

<<< Consistency Check in AO 'Set MIN' >>>
ENTERING :: CONSISTENCY
FINISHING :: CONSISTENCY
EXITING-STATE :: INCONSISTENCY
- Task 'TIME', 'ALARM', 'AUTOON'--> 'AO: Set MIN' 's EXITING-STATE is
'SLEEP'
- Task 'ALARM'--> 'AO: Set MIN' 's EXITING-STATE is 'X1'
Exiting INCONSISTENCY: Exiting operation's number is different in Task!
Exiting operation's parameter INCONSISTENCY: operation 'WAIT' 's
parameter is different!
```

Figure 16 : An example of control operation image in LISP.

Procedural consistency of similar tasks

Similarities between tasks make user expect the similar procedures or grammatical behaviors. Therefore, we need to evaluate whether the similar tasks have the similar behaviors or procedures or not. We have to focus on a following point:

- Are procedures of selected tasks alike? For example, in VCR, if users think task ¡Forward scanning¡ and ¡Backward scanning¡ is alike, then the procedures should have similarity. That is, in two tasks, we should evaluate whether the two tasks have similar operation orders and situations which are not analyzed in operation image, whether the entering and finishing operation are same, and so on.

In ESTIM, we use a cluster analysis for specifying the similar tasks, the result is depicted in left part of Figure 17. And supporting tools are shown for evaluating procedural consistency in Figure 18.

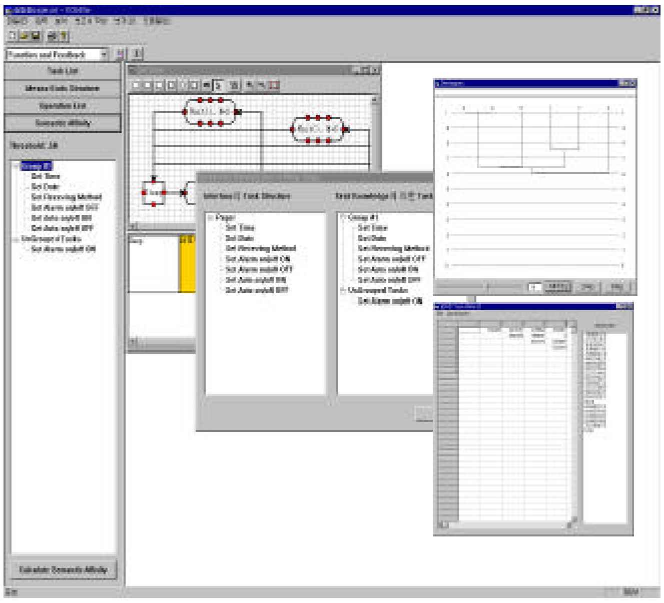


Figure 17: Selection of tasks to be similar by semantic affinity in pager

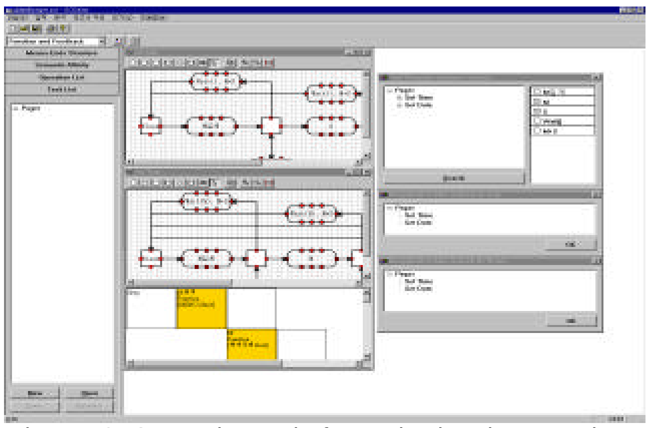


Figure 18: Supporting tools for evaluating the procedure consistency.

Users may have some procedures or goal tree or other prior knowledge about tasks that are more or less generic. We called this ¡task knowledge of users¡. This stage, in congruity analysis, shows that how well logical interface is in harmony with the task knowledge of users.

Matching a logical procedure with task knowledge of users

The first step in congruity analysis is to check whether procedural knowledge of users is in harmony with the procedure of logical interface or not. For example, in pager, we can acquire a result of procedural congruity with function diagrams and OCDs. The matching result appears in Figure 19.

```
TASK ¡SET ALARM ON¡
  <User¡s expectation>
(SLEEP -> (SET MENU (ALARMON)) -> ONOFF -> SET_TIME ->
SET_MIN -> SET_AMPM-> (TASK SET MELODY))
  <System behavior>
(SLEEP -> (SET MENU (ALARMON)) -> SET_TIME -> SET_MIN ->
SET_AMPM -> ONOFF)
  <User¡s expectation>        <System behavior>
-----------------------------------------------------------------
  (ONOFF-> SET_TIME)               (NO)
  (ONOFF-> SET_MIN)                (NO)
(ONOFF-> SET_AMPM)                  (NO)
```

Figure 19: An example of incongruent procedure with user¡s expectation in pager by LISP

There are three important matching structures in procedural congruity : sequence, branch, and loop matching [1]

- Sequence matching: In pager example, task ¡Set TIME¡ is composed of four function: Set(MENU), Set(HR). Set(MIN), Set(AM/PM). We assume that sequence of these functions are in turn Set(MENU), Set(AM/PM), Set(HR), Set(MIN), Set(AM/PM) in real system. However if user¡s procedural knowledge differs from a designed sequence, then it is easy to commit errors in that point which mismatches user¡s expectation. It is the reason why investigate sequence of user¡s expectation and system.
- Branch matching: In performing a task, users may

decide to select one of the actions for reaching the goal. They often plan ahead courses and care not to commit errors of selecting one of others. Therefore, if the options given to user at a branching point are different from user¡s expectation, the interface is not natural or intuitive to users. The branching options in an interface should be matched to the expected options of users. The order of branching points should also match the natural order of decisions as the user expects.

- Loop matching: For example, we can send money to several accounts without having to key in your account number each time. It would be exasperating experience to deal with the loops in an interface procedure that do not match those of user¡s task knowledge.

These three matching structures are evaluated by ESTIM and analytic rules with much procedural knowledge and OCDs.

Control availability

The mismatching between user¡s knowledge and logical interface will make error in performing tasks. Especially, when users expect a particular operation to accomplish the task in a specific situation, however not implemented in a real system, this may invoke problems. Therefore, availability of operations in the particular situation is an important issue in the error-prevention of interface. For example, in Figure 20, Users think that after ¡Selecting Menu: Alarm on ¡, ¡Selecting ON/OFF¡ is shown. But, in real system, ¡Selecting ON/OFF¡ is shown at last time. Therefore, it is natural that users are prone to error in that point of control unavailability.

```
TASK ¡SET ALARM ON¡
****************************
Control Availability Test
****************************
User expectation: ((SET MENU (ALARMON)) -> ONOFF)
System behavior: (((SET MENU (ALARMON)) -> SET_TIME -> SET_MIN
-> SET_AMPM ->   ONOFF))
Result: INCONSISTENCY

User expectation: (ONOFF -> (SET MENU (ALARMON)))
System behavior: ((ONOFF -> SLEEP -> (SET MENU (ALARMON))))
Result: INCONSISTENCY
User expectation: (ONOFF -> SET_TIME)
System behavior: ((ONOFF -> SLEEP -> (SET MENU (ALARMON)) -> SET_TIME))
Result: INCONSISTENCY

User expectation: (SET_TIME -> ONOFF)
System behavior: ((SET_TIME -> SET_MIN -> SET_AMPM -> ONOFF))
Result: INCONSISTENCY

User expectation: (SET_AMPM -> SET_MIN)
System behavior: ((SET_AMPM -> SLEEP -> (SET MENU (ALARMON)) ->
SET_TIME -> SET_MIN))
Result: INCONSISTENCY

User expectation: (SET_AMPM -> (TASK SET MELODY))
System behavior: NO PATH
Result: INCONSISTENCY
```

Figure 20: Control availability test of task ¡SET ALARM ON¡ in pager

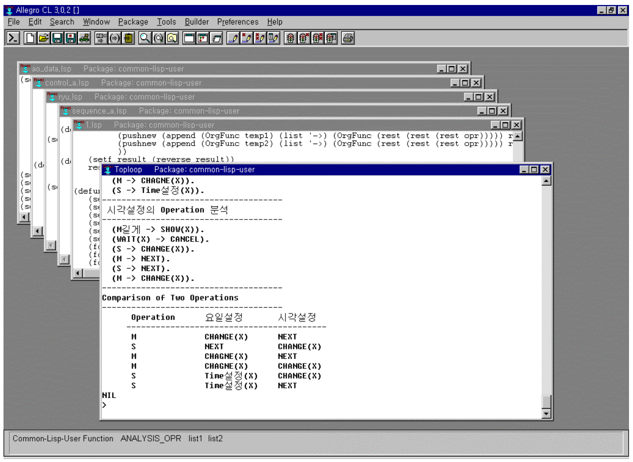
Shortcut among tasks

We can say that users may have an idea of necessity for tasks to be linked. In pager example, in case that users will set up receiving method as MELODY, users will usually want to decide MELODY TYPE. That is, users think that two tasks are closely connected by higher level goal. Therefore, we implemented an analytic rule so as to evaluate the differences with user¡s expectation in navigational availability. The result of this step is shown Figure 21.

```
TASK ¡SET ALARM ON¡
**************************
ShortCut TEST
**************************
User expectation: ShortCut(SET_AMPM-> (TASK SET MELODY))
System behavior: NO PATH
```

Figure 21: A navigational availability in pager

The above script-formed result is performed through ESTIM and LISP. OCDs and task knowledge of user in ESTIM is converted to fact lists which fit LISP syntax. Then, we load the fact lists to analyze in LISP. The main implemented form appears in Figure 22


Allegro CL 3.0.2 []
File Edit Search Window Package Tools Builder Preferences Help
ao_data.lsp Package: common-lisp-user
control_a.lsp Package: common-lisp-user
ryu.lsp Package: common-lisp-user
sequence_a.lsp Package: common-lisp-user
1.lsp Package: common-lisp-user
Toploop Package: common-lisp-user

```
        (pushnew (append (OrgFunc temp1) (list '->) (OrgFunc (rest (rest (rest opr))))) r
        (pushnew (append (OrgFunc temp2) (list '->) (OrgFunc (rest (rest (rest opr))))) r
        ))
   (setf result (reverse result))
```

```
 (M -> CHAGNE(X)).
 (S -> Time설정(X)).
-----------------------------------
시각설정의 Operation 분석
-----------------------------------
 (M길게 -> SHOW(X)).
 (WAIT(X) -> CANCEL).
 (S -> CHANGE(X)).
 (M -> NEXT).
 (S -> NEXT).
 (M -> CHANGE(X)).
-----------------------------------
Comparison of Two Operations
-----------------------------------
     Operation      요일설정      시각설정
    ----------------------------------------
     M              CHANGE(X)     NEXT
     S              NEXT          CHANGE(X)
     M              CHAGNE(X)     NEXT
     M              CHAGNE(X)     CHANGE(X)
     S              Time설정(X)   CHANGE(X)
     S              Time설정(X)   NEXT
NIL
>
```

```
Common-Lisp-User Function   ANALYSIS_OPR   list1  list2
```

Figure 22: Logical interface analysis on LISP

Menu grouping

Nowadays, the greater part of electronic or software system is a menu-based system. In these system, required behavior to complete a task is how to select menu item, how to recognize where the item is located. Thus, a sufficient condition of good interface is a menu structure to match the user¡s expectation. In this study, task affinity index is designed, then menu structure is evaluated through these indices. The result of this step is shown in Figure 23. From this Figure 23, we can identify that users expect that task ¡Set ALARM OFF¡ is located in a different group. However, it is included a menu group in interface.

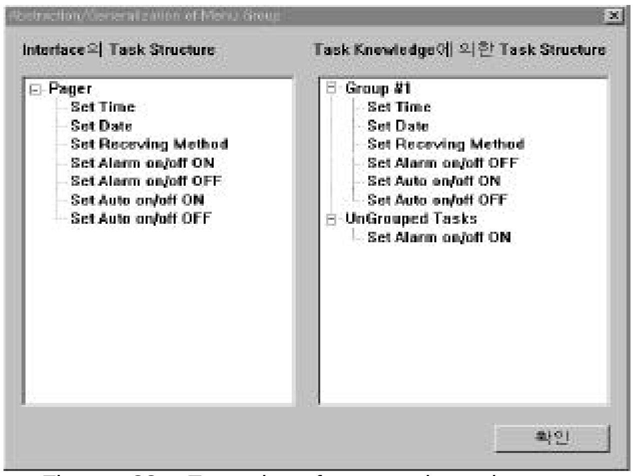


Figure 23: Example of comparison between menu structure of interface and user¡s knowledge

CONCLUSION AND FUTURE WORKS

In this study, the drawbacks of theory-based methods and experimental methods were identified. Then we have progressed in development of novel method for analysis for logical interface. Ultimately, the most important perspective on interface design is congruity with task knowledge of users as well as consistency. Thus we should have an expression for user knowledge and logical interface. Also, we have implemented a support system, called ESTIM, and analytic rules to support these process. We can validate that the results through a support system and rule-based system are equivalent to manual analysis under task-interface matching framework.

In future, through a novel acquisition and representation method about user knowledge and exquisiteness of inference engine, this study will be expanded to a total interface analysis tool. Also, embedded on interface design toolkits, ESTIM will be expanded as user interface management system(UIMS).

REFERENCES

1. Wan C. Yoon, Jisoo. Park, User interface design and evaluation based on task analysis, In *Proceedings of ICPR¡97* pp. 598-601.

2. Wan C. Yoon, Jisoo. Park, An interface model for evaluating Task-Interface congruity, In *proceeding of HCI international¡97*, 1997, 295-298

3. Harrison. M and Thimbleby. H, *Formal methods in Human-Computer Interaction*, Cambridge, 1988

4. Nielsen. J, *Usability Engineering*, American Press, 1993

5. Johnson. P, Task-related knowledge structure in *People and Computer IV*, Cambridge, 1988

6. Byrne. M.D, Wood. S.D, Foley.J.D, Automating interface evaluation, In *Proceedings of ACM/CHI ¡94* , pp232-237

7. Lecerof.A, Paterno.F, Automatic support for usability evaluation, *IEEE Tran. On Software*, 24, 10(Oct. 1988), 863-888

8. Kieras. D.E, Wood. S.D, Meyer. D.E, Predictive Engineering models based on EPIC architecture for multimodal high-performance human-computer interaction task, *ACM Tran. On Computer-Human Interaction*, 4, 9 (Sep. 1997), 230-275

9. Kieras. D.E, Wood. S.D, Abotel. K, Hornof. A, GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs, In *Proceeding of ACM/UIST ¡95* ,pp91-100

10. McDonald. J.E, Dayton. J.T, McDonald. D.R, *Adapting menu layout to tasks*, New Mexico State university, 1986

11. McDonald. J.E, Stone. J.D, Liebelt. L.S, Searching for items in menus : The effects of organization and type of target. In *proceedings of the 27th Annual Meeting of Human Factors Society*, 834-837

12. Pangoli. S, Paterno.F, Automatic generation of task-oriented help, In *proceeding ACM/UIST ¡95*, 181-187

13. Treu. S, *User interface design ? A structured approach*, Plenum press, 1994

14. Treu. S, *User interface evaluation ? A structured approach*, Plenum press, 1994

15. Williams E, Rideout. T, *Task analysis in the product design*, Hewlett Packard