A Multi-level Case-based Design Aid for User Interface Designers

Huhn Kim[1] and Wan Chul Yoon[2]
[1]S/W Development Dept., CDMA Handsets Laboratory, LG Electronics; huhnking@lge.com
[2]Department of Industrial Engineering, KAIST; wcyoon@mail.kaist.ac.kr

Designing modern user interfaces is a complex task that designers should optimize the interaction between the user and the interface considering user goals, preferences, capabilities, and available interface means. It has been recommended that the design procedure follows the task-based analytic process, which starts from eliciting the user's task needs and task knowledge, goes through a conceptual interaction design, and ends with a concrete, physical design of the interface. In practice, however, designers are seldom observed to undertake interface design in a top-down sequential process. Instead, they frequently manifest opportunistic search behavior in finding and choosing reusable design solutions. These strategies have advantages in utilizing the designer's previous work experience and existing interfaces and thus provides working designs very efficiently. In many practical cases, this efficiency advantage outweighs the prospect quality advantage promised by more rigorous design processes.
This paper proposes a design support system that aims to aid the abovementioned practical cognitive process and strategies of designers in designing user interfaces. The system provides the inventory of existing and socially well-accepted interface means. The task requirements can be identified and described as to be useful for finding and determining the most appropriate interface means. The task-interface matching follows a natural path and is explicitly described. The design aid system achieved these requirements by using a case base in which cases are related across multiple abstraction levels. The system provides tools and models so that the knowledge at each level can be explicitly described but be easily related with other levels through a consistent representation.

INTRODUCTION

Designing modern user interfaces (UIs) is becoming a complex task due to the increasing needs of functionality and convenience that challenge the constrained cognitive capability of users. The designer should optimize the interaction between the user and the interface considering user goals, preferences, capabilities, and available interface means. It has been recommended that the design procedure follows the task-based analytic process, which starts from eliciting the user's task needs and task knowledge, goes through a conceptual interaction design, and ends with a concrete, physical design of the interface (Wilson and Johnson, 1996). In practice, however, designers are seldom observed to undertake interface design in a top-down sequential process. Instead, they frequently manifest opportunistic search behavior in finding and choosing reusable design solutions (Guindon, 1992; Rasmussen, Pejtersen and Goodstein, 1994; Visser, 1996). These strategies have advantages in utilizing the designer's previous work experience and existing interfaces and thus provides working designs very efficiently.

This paper proposes a design support system that aims to aid the abovementioned practical cognitive process and strategies of designers in designing UIs. The need of support system is obvious for two reasons. First, the UI design is a cognitively intense work and requires vast knowledge in the domain. Second, for an organizational purpose, the rationales behind an UI design and the decision problems solved during the design process should be described in a well-ordered way as to become public and reusable knowledge.

To support the designer harmoniously and hence effectively, the cognitive process and difficulties that designers experience in design practice should be considered. The problem solving is not only restricted by the designer's limited knowledge and deficient memory retrieval but also by the cognitive complexity of the decision making itself. To aid, the system in this paper provides the inventory of existing and socially well-accepted interface means. The task requirements can be identified and described as to be useful for finding and determining the most appropriate interface means. The task-interface matching follows a natural path and is explicitly described. The design aid system achieved these requirements by using a case base in which cases are related across multiple abstraction levels: task, operation and physical interface. The three abstraction levels are adopted because designers generally view and analyze the domain or task from the three perspectives (Yoon, 2001; Benyon, 2002). Cases at the task level include structural and temporal relationships between user tasks. Cases at the operation level provide the knowledge about sequences and similarities between operations that are necessary for accomplishing each task defined at the task level. The physical interface level is concerned with interface means such as interface shapes and layouts required for the operations or tasks. In the design aid, the cases at each level are explicitly described and are easily related with other levels through a model called HOCD (Hierarchical Operation and Control Diagram).

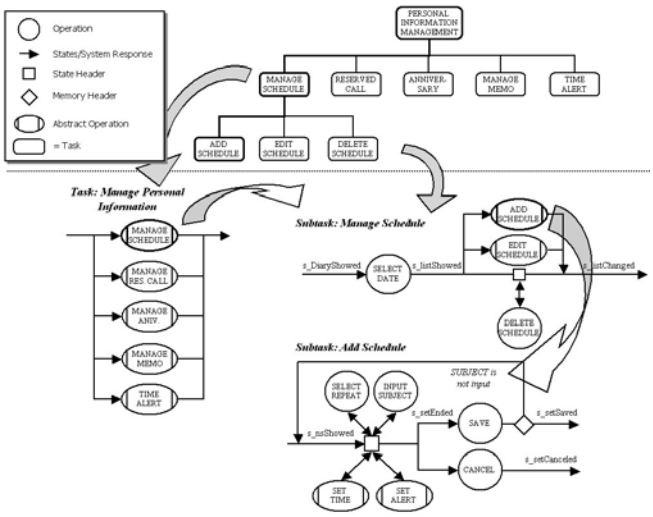

Figure 1. Notations and an example of HOCD

By employing represented knowledge at multiple levels, the design aid can provide proper answers to three types of questions, that can be raised at each level of abstraction, mentioned by Rasmussen, Pejtersen and Goodstein (1994): What tasks, operations or physical means are available? Why are these required? How can these be achieved? In other words, the aid can provide not only a case at one level (what question), but also related cases at other levels relevant to the why and how questions. Thus, designers can freely navigate between design cases at the three levels of abstraction. It enables the design process supported by the system to be compatible with the opportunistic search behavior of designers.

A CASE REPRESENTATION MODEL

To assist the designers, a model to design artifacts and to represent cases at the task and operation levels is necessary, while artifacts and cases at the physical interface level can be represented by pictures or storyboards. The OCD, developed by Yoon, Park and Lee (1996), is a diagrammatic model that represents the interaction between a user and an interface. The original OCD is suitable to represent cases at the operation level. However, it cannot represent hierarchical task trees. To represent all knowledge at the task and operation levels, the diagram is adapted by adding a simple tree and the concept of hierarchy. This model is called HOCD. Figure 1 shows the notations and an example of the HOCD in which the tasks and operations related to personal information management (PIM) in a mobile phone are shown. In addition, to communicate with computer, we developed OCS (Operation and Control Script) that can be transformed exactly into HOCD without loss of semantics.

METHODS FOR EFFECTIVE RETRIEVAL OF MULTI-LEVEL DESIGN CASES

The case-based design aid in this paper provides relevant cases at multiple levels of abstractions, similar to an HOCD drawn by designers as a query. For effective case retrieval, the followings are necessary. First, the indices, of the operations or tasks in the HOCD models for representing design cases at the task and operation levels, are required to measure similarity between two tasks or operations. Second, an efficient algorithm, which can measure similarities between HOCD models in the case base by using the indices and retrieve the most similar HOCD to a designers' query, is needed. Third, adaptation rules, that can be applied to modify a retrieved case to the context reflected in a designer's query, are required.

Similarity between Two Tasks/Operations

Each task or operation has inherent properties that affect relationships with other tasks or operations as well as the decisions of proper interface objects or means. Usually, the properties of a task or an operation can be defined as the combination of the following aspects: their names, types, target types, and target attributes. For example, "Select" is a type of operation and whether the selection should be unique or not is an attribute of the target. By this distinction, either a group of radio-type option buttons or a group of check boxes is preferred as the interface object.

By performing task analyses at each domain, we classified the types of tasks or operations that applied to the design domain of such things as Web applications, software and mobile phones: Input, Find/Scan, Get, Confirm/Verify, Process/Transmit, Store/Add/Create, Rearrange, Compare, Select, Determine, Revise, Delete, Move/Join, Cancel/Exit. The data types and attributes suggested by Vanderdonckt (1993) were adapted and then used as the targets and their attributes. In the aid, the similarity between two tasks or operations is measured by the degree of equality of the properties.

Similarity between two HOCDs

Operation sequences or task flows represented by the HOCD can be transformed into a graph form mapping operations to nodes and states to edges. At this time, the similarities among graph nodes can be measured by the method mentioned in previous section. Furthermore, the similarities between design cases and a designer's query can be calculated by well-known graph-matching algorithms. In this paper we employed Messmer (1996)'s algorithm that can efficiently search a case graph with minimal total cost of edit actions for being identical with a query graph.

The Rules for Adapting Prior Cases

When designers pose a query, the design aid with the graph-matching algorithm can provide similar prior cases to the query. The retrieved cases, however, cannot coincide with the designers' query but may have some differences with the

query. Thus, to reuse the retrieved cases, the designers should adapt the cases to their own query through an analysis of the differences and similarities between the two. This adaptation is assisted by the aid with several adaptation rules. The new case, that is the one derived from applying the adaptation rules, is then evaluated to determine if it is proper for the current context. Designers perform the evaluation and revise the adapted case to fit with their design context. Once the evaluation and revision ends, the resulting artifact is stored as a new case in the case base.

A PROTOTYPE OF THE AID AND DESIGN EXAMPLE

Through applying the abovementioned methods, we developed a prototype of the multi-level case-based design aid. The prototype is called MCBDS (Multi-level Case-Based Design Support). The MCBDS provides design cases similar to a designer's query represented by the OCS. Figure 2 shows a screen copy of the MCBDS developed using Allegro Common Lisp 5.0.1 for Windows. The prototype interacts with designers by the OCS of a grammatical form, instead of the HOCD of a diagrammatic form. However, it is still meaningful to verify the effectivness of the proposed aiding framework with the OCD interface since, except for the modest training requirement to use OCS, the cognitive process of designing user interfaces in cooperation with the support system remains basically the same.

A Design Example Using MCBDS

In this paper, we tried to solve various design problems in diverse domains to show the effectiveness of the MCBDS. More than one hundred design cases related to mobile phones, camcorders, and Web applications were stored in the case base of the MCBDS. The test results from various design queries showed that the algorithm could retrieve design cases that were similar to the requested ones, and the results of the case studies revealed that the proposed design aid was useful for interaction design of the UI.

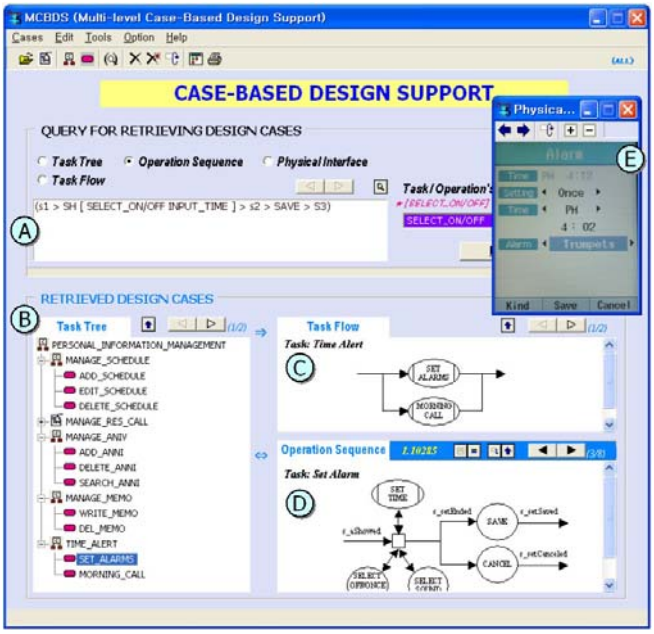

Figure 2. A screen copy of the CASES prototype

Cases at operation level. A simple example of the achieved case studies is as follows: the designer, who should design an interface for managing personal information in a mobile phone, requests a query for alarm design, as in the box A of Figure 2; the designer should also input the properties of each operation as explained in previous section. The designer's OCS query, shown in Figure 2, is transformed into the corresponding HOCD located at the center of Figure 3. The HOCD of the derived design case, shown in the box D of Figure 2, is equal to Case 3 in Figure 3. Figure 3 also shows four design cases that were retrieved with low cost when compared to the designer's query about operation sequences of an alarm in a mobile phone. In

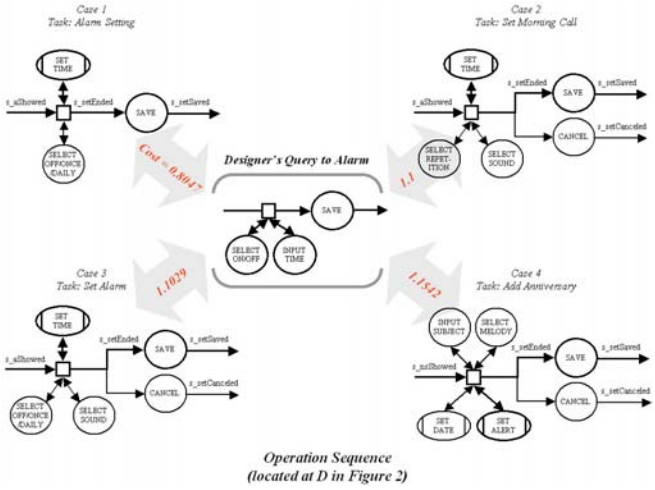

Figure 3. Design cases retrieved from the design query in Figure 2

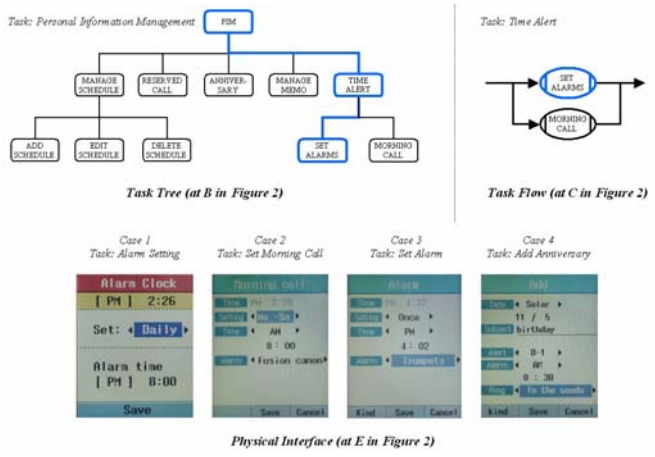

Figure 4. Design cases at the task and physical interface levels connected to HOCDs in Figure 3

Figure 3, Case 1 and Case 3 describe the operation sequences for setting an alarm, Case 2 is for setting a morning call and, lastly, Case 4 is for adding an anniversary.

Cases at task and physical interface levels. When the designer requests the query, the MCBDS supplies design cases at the same level of the query, as in Figure 3, together with the related cases of the task and physical interface levels, as in Figure 4. From the retrieved cases, the designer can directly refer to other design cases, such as “Manage Schedule”, “Reserved Call” and “Manage Anniversary” that are related to PIM. In addition, we can easily recognize that an alarm in a mobile phone must have a consistent structure with the alarm included in the interface for managing the morning call and anniversary, as in Figures 3 and 4.

Case Adaptations. Differences between the designer's query and the retrieved cases can be marked on the diagrams as in Figure 3 according to the adaptation rules, even if the marking function is not yet implemented in the prototype. By using the marks, the designer can easily identify differences between his or her own query and the retrieved cases, thereby readily determining which parts of those cases can or should be used.

Implications. The retrieved design cases in Figure 3 provide complete operation sequences that are more detailed than the query. In other words, the designer may give a rough OCS or describe only a part of the wanted interaction as the query to get the fuller and complete cases. In the example, the operations to set the number of repetitions of the alarm and select the alarm sound are included in Case 1 and 3, which may remind the designer of the necessity of such extra features. In this way, MCBDS helps designers to easily accomplish their work with more reliable quality. Moreover, when changing a design at the operation level into a specific physical interface, the designer can refer to several design cases at the physical interface level that correspond to each case at the operation level. Thus, the designer can easily embody their interface by combining or refining the cases.

CONCLUSIONS

During UI design, designers tend to behave in opportunistic fashions. Furthermore, their behavior is usually guided by their prior experience on proper matching between user tasks and interface means, rather than by an orderly analytic process. Designers, however, cannot accommodate all prior experiences due to constrained cognitive capabilities. Thus, they persistently adhere to their initial solution and neglect the opportunity to consider better alternatives (Visser, 1996).

In this paper we proposed the case-based design aid that makes it possible for designers to effectively reuse prior design cases. In the aid, design cases had three different levels of abstraction: task, operation and physical interface. Cases at the task and operation levels were represented by the HOCD. The HOCD is not only effective in representing and reusing a design case, but it is also effective as a model for task-based interaction design. With the HOCD, the aid can provide design cases at an abstraction level that corresponds to their focused design stage, based on case-based reasoning technique in terms of a graph-matching algorithm. At the same time, designers can acquire design cases at all the other levels that are related to cases at the focused level. Furthermore, designers can evaluate interim artifacts by comparing them with the design cases retrieved by the aid. In this manner, designers can easily move downward or upward along the levels of abstraction, and jump around the levels opportunistically.

Many case studies performed with the prototype system showed that the proposed aid was useful for assisting designers in UI design. The prototype, however, used the script language as a means of communicating with designers. Thus, to input task or operation sequences as a query was somewhat difficult. For the support to become a real application, we are now revising the prototype so that designers can represent, retrieve and store design cases by the HOCD. Moreover, if the case base does not have enough cases, designers may not be able to acquire a satisfactory design case. Thus, the aid should have the capability to be used as a design tool; the interim and final artifacts produced by utilizing the aid while designing an UI should be constantly reflected in the case base. This process may secure the effectiveness of the aid as time goes on.

REFERENCES

Benyon, D. 2002. Representations in human-computer systems development. *Cognition, Technology & Work*, 4, 180-196.

Guindon, R. 1992. Requirements and design of DesignVision, an object-oriented graphical interface to an intelligent software design assistant, In *Proceedings of CHI’92* (pp. 499-506). New York: ACM Press.

Messmer, B.T. 1996. *Efficient graph matching algorithms for preprocessed model graphs*. Unpublished doctoral dissertation, Institute of Computer Science and Applied Mathematics (IAM).

Rasmussen, J., Pejtersen, A.M., & Goodstein, L.P. 1994. *Cognitive systems engineering*. New York: Wiley.

Vanderdonckt, J. 1993. *A corpus of selection rules for choosing interaction objects*. Technical report 93/3, Institut d’Informatique, Namur.

Visser, W 1996. Use of episodic knowledge and information in design problem solving. In N. Cross, H. Christiaans, and K. Dorst (Ed.), *Analysing Design Activity* (pp.271-289). New York: Wiley.

Wilson, S. and Johnson, P. 1996. Bridging the generation gap: From work tasks to user interface design. In *Proceedings of the 2nd International Workshop on Computer-Aided Design of User Interfaces (CADUI ’96)* (pp. 77-94*)*. Belgium: Presses Universitaires de Namur.

Yoon, W.C., Park, J. and Lee, S.H. 1996. A diagrammatic model for representing user’s interface knowledge of task procedures. In H. Yoshikawa & E. Hollnagel (Ed.), *Proceedings of Cognitive Systems Engineering in Process Control (CSEPC’96)* (pp. 276-284). Kyoto, Japan.

Yoon, W.C. 2001. Identifying, organizing and exploring problem space for interaction design. In G. Johannsen (Ed.), *8th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems* (pp.81-86). Kassel, Germany.