Consistency in Human-Computer Interfaces for End-Users

Chang-tseh Hsieh
University of Southern Mississippi

Ming-te Lu
Metropolitan State University

Engming Lin
Eastern Kentucky University

End-users who employ different interactive systems on a regular basis must deal with dissimilar human-computer interfaces (HCIs). The varied HCIs tend to create confusion and cause unnecessary delays and errors to occur, thus hindering end-user productivity. Different syntactic knowledge required for different interfaces also impedes user learning and recall. Three types of inconsistencies existing among HCIs are discussed and their causes examined. A strategy is then recommended to the end user computing management to help reduce these inconsistencies. In addition to alleviating problems for end-users, improvement in HCI consistency will facilitate system development and maintenance.

The human-computer interface (HCI) consists of artifact-bound conditions (as defined by a combination of hardware and software constructs) which facilitate the interaction between a user/operator and the application system (Bodker, 1991, 36). Styles of HCI may include menus, command language, questions and answers, fill-in forms, natural language, object manipulation, etc. (Turban, 1993, 225-227) As HCIs are becoming more user-friendly and flexible, an increasing number of end-users have assumed a greater role in the operation and development of computer-based information systems (CBIS). Since different CBIS tend to use HCIs which are of different styles and/or syntactical conventions, end-users who are also operators often are confused with these differences; as a result, long training time and low productivity ensue. The ideal case of HCI consistency means that users/operators are allowed to interact with HCIs which have the same “look and feel” and the same syntactic conventions both within an application system and across different application systems. As a consequence, users/operators are able to transfer knowledge of interfaces in a well-learned system to a new system, thus reducing training time and avoiding much confusion and errors (Barnard, et al., 1981; Nielsen, 1989; Polson, Muncher, & Englebeck, 1986; Polson, 1988). It is claimed that user interfaces can account for as much as 50 percent of total system development life cycle costs for interactive systems (Myers 1989; Sutton & Sprague, 1978). Thus, this ideal case will also improve the system development process because HCIs developed previously may be reused, achieving the goal of reusable software. Developing HCIs for new systems independent of HCIs used previously falls into the re-inventing the wheel syndrome.

For end-user operators who primarily use only
one application system or software product, problems with inconsistent interfaces may be less significant. For example, a word processor may be the major system used by an employee. The HCI of the word processor may have to be revised from time to time when new versions are released. Since a software vendor tends to make an effort to retain the same “look and feel” of the old HCI, upgrading of the package generally causes only minor inconvenience to its users. With respect to in-house developed applications, however, different HCIs are often used when new systems are introduced.

Many end-user operators must utilize a number of application systems to generate the desired information. For them, switching from one application to another may involve two quite dissimilar HCIs. These dissimilar HCIs may confuse end-users and cause unnecessary delays and errors to occur, as a consequence, hindering end-user productivity. Different syntactic knowledge required for different interfaces also impedes their learning and recall. Problems of dissimilar HCIs become progressively more serious with infrequent uses and more applications.

Meanwhile, arguments have been made claiming HCI consistency may be unworkable. For example, interface consistency may not be easily achieved due to the following facts: (1) a system may require functionalities which may have specific features in HCIs that are not found in other systems; (2) it may not be possible to provide the same effective HCIs to different types of users/operators, such as novices and experts; and (3) independent software vendors employ different HCIs in their products for fear that close resemblance of their HCIs to a competitor’s will make them susceptible to legal action (Samuelson, 1989; Samuelson, 1992). In addition, Grudin (1989) rejected HCI consistency as the primary system development goal by claiming that interface consistency is a largely unworkable concept and that ease of learning can conflict with ease of use. Nevertheless, in a wide array of situations, consistency in HCI “look and feel” and syntactic conventions is quite attainable for the majority of end-users and many application systems. Users in these situations may be knowledgeable but intermittent users of computer systems (Shneiderman, 1987, 53-54), may use the same workstations (such as PCs), and may have similar work contexts and perform similar tasks. Thus, they have more or less the same level of computer and task semantic knowledge and are subject to similar physical constraints. In these situations, a well thought-out strategy is needed to achieve a higher level of HCI consistency to alleviate the pain and problems associated with inconsistent HCIs. This strategy will also promote system development productivity for CBIS developed by end-users as well as IS professionals. The recommended strategy needs to be implemented at the project level, application level, and departmental level in order to effectively unify the HCIs used in various CBIS.

This paper proposes a framework for classification of different HCI inconsistencies. This classification is then used as the basis for the recommended strategy to end-users and end-user computing management to reduce HCI inconsistencies and to promote system development effectiveness. After the introductory section, relevant theories associated with HCI design are reviewed. In section three, different types of HCI inconsistency are discussed and their causes examined. In section four, a strategy based on the proposed classification framework for enforcing HCI consistency is presented. Finally, concluding remarks are made in section five.

Human–Computer Interface

Norman (1986) proposed a seven-stage human–computer interaction from a user/operator’s perspective:

- Perception of the computer presentation
- Interpretation of the presentation
- Evaluation of the presentation
- Formulation of goals
- Formulation of intentions
- Specification of action sequences
- Execution of these sequences

These stages constitute a recognition-action cycle for the user/operator. Four cognitive processors are involved in this cycle; they are motor movements, perception, cognition, and memory (Olson & Olson, 1990). Except for long-term memory, these processors have limited capacity and thus constrain a user/operator’s behavior (Gerlach & Kuo, 1991). In order to satisfy a user/operator’s motor and perceptual needs, signals must be perceivable, and responses should be within the range of his/her motor skills; more importantly, HCI must empower the memory and cognitive capacity of its user/operator to learn and reason easily about the system’s behavior. Thus, inconsistent HCIs impede the user/operator’s ability to interact in the perception, interpretation, action specification, and execution stages. They also slow down the long-term memory write time.

Shneiderman (1987, 43-52) suggested that a semantic/syntactic model of user/operator knowledge