The Effects of Conceptual Consistency on the End User’s Mental Models of Multiple Applications

John W. Satzinger
Southwest Missouri State University

Consistent user interfaces across applications are thought to facilitate ease of learning and use because a user can draw on existing knowledge when learning any new computer application. Although empirical research has confirmed that transfer of learning occurs when procedural rules for multiple applications are consistent, no research has been reported that examines the effects of consistent versus inconsistent conceptual models across applications on the accuracy of the user’s mental models of the applications. Applications from a variety of sources available to an end user might conform to interface standards for their “look” and “feel;” however, consistent conceptual models across applications (conceptual consistency) are still not assured. This paper reports the results of a laboratory experiment that tested the effects of conceptual consistency across applications on user knowledge, performance, and satisfaction when two integrated cooperative work applications were learned and used by student participants. No differences in performance or satisfaction were found; however, participants learning the inconsistent applications formed more accurate mental models of the applications. Schema theory is used to explain the results, and some implications when users initially learn multiple applications are discussed.

Information system managers are increasingly adopting interface design standards for end-user applications and client-server organizational systems (e.g., Apple, 1987; Berry, 1988; Microsoft, 1992). The primary motivation for interface design standards is ease of learning and use resulting from consistent user interfaces (Shneiderman, 1987; Mayhew, 1992). Consistent user interfaces across applications are thought to facilitate ease of learning and use because a user can draw on existing knowledge when learning any new application (Kieras & Polson, 1985; Bennett, 1986; Polson, 1988). However, multiple applications of a variety of types and from a variety of sources are being integrated on the end user’s desktop, making interface consistency difficult if not impossible to achieve (Nielsen, 1989).

Despite the acceptance of interface consistency guidelines and standards, very little empirical research has been reported that examines the effects of consistent user interfaces across applications. Much of the reported findings have focused on transfer of learning (Kieras & Polson, 1985) from one application to another application of the same type, usually when command languages or procedural rules are inconsistent (Karat et al., 1986; Foltz et al., 1988; Polson, 1988). Only one study reported transfer of learning from one application to a different type of application, but interface consistency was not manipulated (Ziegler et al., 1986). Two studies have investigated the effects of interface consistency when a user interacts with multiple applications or modules (Kellogg, 1987; Satzinger & Olfman, 1998), but again the main interface consistency manipulations were procedural rules across applications.
There is a need for additional research on interface consistency because the user interface is much more than the procedural rules and devices used to issue commands. Moran (1981) defines the user interface as consisting of everything the user comes into contact with while using the application — physically, perceptually, and conceptually. Therefore, the user interface includes the underlying objects and relationships manipulated by the application, which Foley and van Dam (1982) call the conceptual model of the application. For example, a cooperative work application might include objects such as work groups, people, projects, schedules, messages, and documents. Learning this application requires understanding these objects, understanding how they are related, and understanding how they might be manipulated to complete work tasks regardless of the way the application is actually implemented on the computer. The conceptual model is therefore the “heart” of the user interface (Aksycyn, Yoder & McCracken, 1988), and the objects, relationships, and possible operations from the conceptual model are reflected throughout the implemented application.

This paper reports the results of a laboratory experiment that tests the effects of consistent versus inconsistent conceptual models across applications when a user learns and uses multiple applications. Two cooperative work applications were developed specifically for the experiment to manipulate the consistency of the conceptual models across applications while controlling the consistency of other interface attributes. The applications were integrated, learned together in one session, and used concurrently to complete the experimental tasks. A multiple application research model based on end-user training research was used to design and control the experiment (Bostrom, Olfman, & Sein, 1990; Satzinger, 1994).

The remaining sections of the paper are organized as follows. Section 2 discusses the user interface, interface consistency, and how the conceptual model of an application is reflected in the user interface. Section 3 draws on schema theory to describe how consistent conceptual models across applications might affect user knowledge, user performance, and user attitudes toward the applications. The hypotheses for the experiment are then presented. Section 4 describes the research method, and Section 5 presents the results. Section 6 discusses the results, including implications for managers and for researchers concerned with understanding the effects of consistency when a user learns and uses multiple applications.

The User Interface and Interface Consistency

The user interface and interface consistency are complex constructs (Moran, 1981; Grudin, 1989). Generally accepted models of the user interface are based on Foley and van Dam’s (1982) language model of human-computer interaction (c.f., Bennett, 1986; Gerlach & Kuo, 1991; Marcus & van Dam, 1991). In the language model, the interface consists of a conceptual model and two separate languages, an action language and a presentation language, used by the user and the computer to communicate. Human-computer interaction is therefore modeled as a dialog.

The conceptual model of an application includes the intrinsic objects from the task environment of the user, the relationships among the objects, and the operations possible to manipulate the objects. The conceptual model is a logical model which does not describe how the application is physically implemented, although it is sometimes confused with the designer’s conceptual model, which usually does include a brief description of the implementation (Moran, 1981). Figure 1 shows a written description of a conceptual model for an application used in the experiment reported in this paper. Objects, relationships, and possible operations are apparent, but no references are made to the way the user would actually interact with the computer.

The rest of the user interface involves the two separate languages that are used to communicate about the objects in the conceptual model: one used by the user to communicate to the computer (the action language) and one used by the computer to communicate to the user (the presentation language). The two separate languages are designed based on the communications requirements that follow from the conceptual model. The action language is used to tell the computer which operations to perform, and the presentation language is used to ask the user about the requested objects and operations and to provide the resulting information.

Figure 1: Description of a Conceptual Model

NUCLEUS is used to create work groups and the information they use. Work groups consist of members selected from lists of people in the system described by a work group name. An information structure is created to store various types of work group information. Any member can create an information structure and assign access to other work group members. Work group members then use the information structure by either browsing through it or typing in additional information. Any person can review work groups to see a summary of the work group. Work groups and information structures can be modified or deleted as needed.
Because of the language analogy, the interface can be described by a linguistic framework that includes semantic, syntactic, and lexical layers that make up the two languages. Figure 2 shows the top-down interface design method adapted from Foley and van Dam (1982). The user interface is designed by studying the task environment, describing a conceptual model for the application, and then creating the two languages that support communication about the objects in the conceptual model. The intrinsic objects, relationships, and operations become “words” in the two languages, so key aspects of the conceptual model are reflected throughout the resulting interface in menus, messages, labels, and dialog boxes. For this reason, Akscyn, Yoder, and McCracken (1988) describe the conceptual model as the “heart” of the user interface. Although it is widely recognized that to the user the interface is the application (Moran, 1981), the Foley and van Dam interface design method illustrates that to the designer the interface is the application as well.

Consistency has been loosely defined as doing similar things in similar ways (Reisner, 1993), and there are three types of interface consistency that are relevant to design: internal consistency, analogical consistency, and external consistency (Grudin, 1989). All three types can originate with the conceptual model. Internal consistency refers to the consistent use of terms and features within one application. Analogical consistency refers to the correspondence of terms and features in the interface to the world beyond computing. The “desktop” metaphor for the interface, for example, or references to “filing” a “document” in a “folder” and receiving “mail” in a “mailbox” are examples of analogical consistency with the user’s task environment. The degree of analogical consistency for an application becomes apparent early in the design process in the conceptual model, and analogical consistency may or may not result in internal consistency (the analogy may contain inconsistencies).

External consistency (addressed in this paper) refers to interface consistency across applications. Inconsistencies can result from differences in conceptual models and from differences in each linguistic layer of both the action languages and presentation languages. The conceptual models of two applications might be internally consistent, yet not be consistent with each other. For example, a description of a spreadsheet application might refer to “worksheets” “stored” in “folders” in a “file cabinet,” while a description of a word processor might refer to “text files” “saved” on a “mini drive” in the user’s “account allocation.” The applications based on these conceptual models would not be consistent, even if both used pull-down menus, dialog boxes, and selection with a mouse. This is what is meant by inconsistent conceptual models across applications. This study examined the effects of some external inconsistencies in the interfaces where each of the two interfaces are internally and analogically consistent.

**Conceptual Consistency and the Mental Model Development Process**

This section draws on user training research and schema theory to explore the effects of consistent versus inconsistent conceptual models on user knowledge of multiple applications. Five hypotheses tested by the laboratory experiment are then presented.

**Learning Computer Applications**

A user’s knowledge of a computer application is often described as a mental model (Gentner & Stevens, 1983;
Borgman, 1986; Sein & Bostrom, 1989; Wilson & Rutherford, 1989; Staggers & Forcio, 1993), although the terms user’s model and user’s conceptual model are often used interchangeably with mental model (Moran, 1981; Bennett, 1986; Gerlach & Ku, 1991). An accurate initial mental model of an application is formed when the user understands the objects, operations, and relationships in the conceptual model of an application (Sein & Bostrom, 1989).

How the user’s mental model is formed and expanded is a central concern of user training research. Consequently, research on the effects of conceptual consistency across applications can draw on Bostrom, Olfman, and Sein’s (1990) user training framework in which user learning is modeled as a multi-step process. Initially, characteristics of the system, characteristics of the user, and characteristics of the training environment interact to change the user’s knowledge of an application. Later, when using the application to complete tasks, changes in user knowledge continue.

Schema theory (Bobrow & Norman, 1975; Rumelhart & Norman, 1978; Rumelhart, 1980) can be used to explain the mental model development process when a user learns and uses computer applications. Schemata are knowledge structures that store concepts in human memory, including procedural knowledge of how to use the concepts. They are used when interpreting sensory data, retrieving information from memory, organizing actions, and determining goals and subgoals. Each schema is linked to a network of sub-schemata that collectively represent an individual’s knowledge about a domain. A schema is activated when incoming sensory information fits the pattern of concepts and relationships represented in the schema. Once activated, additional information that helps explain the external situation is inferred based on associated sub-schemata that are in turn activated. The set of schemata activated at one time makes up our mental model of the situation faced at that time (Rumelhart, 1980; Wilson & Rutherford, 1989).

A user’s mental model of a computer application can be thought of as a network of associated schemata and sub-schemata, and learning a new application involves assimilating new concepts and procedures. Inconsistent conceptual models across applications result in learning more concepts because fewer underlying concepts are shared by the applications. Multiple applications with inconsistent conceptual models should therefore require more effort to learn.

However, learning an application also involves activating and then establishing appropriate associations among existing schemata and sub-schemata. Associated schemata may include knowledge of the task the application supports, knowledge of general computer concepts, and knowledge of other computer applications. Once activated, changes in all associated schemata might occur when interacting with a computer application through accretion, tuning, and restructuring. Therefore, expanding on Bostrom, Olfman, and Sein (1990), a user’s knowledge of all computer applications might continue to change each time any one computer application is learned or used (Satzinger, 1994; Satzinger & Olfman, 1998). Inconsistent conceptual models across two applications could lead to two inaccurate mental models when concepts underlying one application are incorrectly associated with the other. Therefore, inconsistencies in the conceptual models across applications not only make it more difficult to learn each application, but more importantly, they could result in inaccurate mental models of both applications.

Hypotheses for the Effects of Conceptual Consistency

Since inconsistent conceptual models across applications might make each application more difficult to learn and could result in two inaccurate mental models, several detrimental effects were hypothesized. User training research based on Bostrom, Olfman, & Sein’s (1990) framework is concerned with two types of training outcomes: user knowledge of the application and user attitudes toward the application (c.f., Olfman & Bostrom, 1991; Davis & Bostrom, 1993). The mental model development process affects both outcomes, although some training research has focused only on the accuracy of the user’s mental model (Sein & Bostrom, 1989) and almost all interface consistency research has focused on user knowledge indicated by performance measures (Karaf et al., 1986; Foltz et al., 1988; Satzinger & Olfman, 1998). For this experiment, hypotheses addressed both user knowledge and user attitudes toward the applications.

Since consistent conceptual models should be easier to learn and result in more accurate associations of concepts in one application with the other application, more accurate mental models should be formed when both applications are learned and used. Additionally, more accurate mental models should result in performance benefits, including less time to complete tasks, more accurate work, and fewer user interactions caused by errors and slips (Norman, 1983; Staggers & Norcio, 1993). Finally, users should be more satisfied overall with the applications because they require less effort to learn and use (Davis, 1989). The hypotheses, stated in alternative form, were:

H1 Consistent conceptual models across applications will result in more accurate mental models of the applications when a user learns and uses both applications.

H2 Consistent conceptual models across applications will result in faster task completion when a user learns and uses both applications.

H3 Consistent conceptual models across applications will result in fewer user interactions when a user learns and uses both applications.

H4 Consistent conceptual models across applications will result in more accurate completed work when a user learns and uses both applications.
H5 Consistent conceptual models across applications will result in greater user satisfaction with the applications when a user learns and uses both applications.

Research Method

This section describes the research method. First, the multiple application research model used for the experiment is discussed. Then the participants, treatments, procedure, dependent variables, and controls are described.

Research Model

The multiple application research model used to design the experiment is shown in Figure 3. In this experiment, the degree of consistency of the interface across applications was manipulated by providing consistent versus inconsistent conceptual models for two applications while controlling the consistency of other interface attributes. The multiple application research model extends the user training framework of Bostrom, Olfman, and Sein (1990) to include degree of consistency, degree of integration, and degree of concurrent use as additional factors that affect user learning of multiple computer applications (Satzinger, 1994). Degree of integration is defined by the extent to which applications share data. Degree of concurrent use is defined by the extent to which multiple applications are used simultaneously in a work session. User characteristics and the training environment are additional factors retained from the training framework. All of these factors except interface consistency were controlled in this experiment.

Participants and Setting

The experiment included 58 undergraduate business students enrolled in an introductory information systems concepts course at a public university. They were offered extra course credit for participating in the two hour session where they would learn two new computer applications. The sessions were held in a computer training lab equipped with IBM PS/2 computers with color monitors and mice. The participants were chosen because they were novice computer users with limited knowledge of or experience with the type of applications used in the experiment. Most of the participants’ computer experience was in an introductory computing skills course previously taken at the university.

Treatments

Two cooperative work applications were designed specifically for the experiment. The first is called NUCLEUS, which supports creating work groups and structuring information used by them. The second application is called PIPELINE, which supports communication during and between meetings. These applications were designed to work together to support work group tasks, and they are integrated in that they provide access to the same objects in the database. Semantic, syntactic, and lexical design choices for both applications are consistent: they both use menus, dialog boxes, selection lists, prompts, error message boxes, data entry panels with buttons and check boxes, and selection with a mouse. The applications were developed using Layout (Matrix, 1988), which is a graphical application development environment for the IBM PC platform.

The independent variable was degree of consistency of the conceptual models across applications. Consistent versus inconsistent conceptual models was operationalized by developing two versions of PIPELINE: one with a conceptual model consistent with NUCLEUS and the other with a conceptual model inconsistent with NUCLEUS. Figure 4 shows
descriptions of the conceptual models. The description of NUCLEUS is the same as shown in Figure 2, and the first description of PIPELINE is consistent. The second description of PIPELINE, however, uses different terms to refer to the objects and operations. NUCLEUS refers to creating, modifying or deleting work groups and people, but PIPELINE refers to making, changing or erasing meetings for project teams and sending messages to system users. With NUCLEUS, objects are selected and text is typed, but with PIPELINE objects are chosen and text is entered. These differences appear in menus, prompts, messages, and labels throughout the applications. Figure 5 shows screen prints of the applications.

NUCLEUS was combined with the two different versions of PIPELINE to create the two treatments. Although the applications have consistent visual appearance and use consistent and easy to use interface features, the inconsistent conceptual model treatment requires learning more concepts and reconciling differences across applications. The applications in both treatments remained integrated. For example, for the inconsistent treatment, the “work group” “created” with NUCLEUS was used to “make” a meeting for the “project team” with PIPELINE.

**Procedure**

Participants signed up for one of six two-hour sessions. Upon arrival at the computer lab, they were randomly assigned to a computer which had the software for one of the two treatments preloaded. They then completed a short background questionnaire and took a timed test to measure their ability to manipulate mental images (VZ2), a cognitive ability found to be related to the ability to learn and use computer applications (Sein & Bostrom, 1989). Next, the participants worked individually through a tutorial manual to learn the basics of both applications. The first part of the manual briefly described the two applications and explained that the tutorial versions of the applications only simulated multi-user versions — a message sent to a person was not actually received, for example.

The second part of the manual included step-by-step instructions that led participants back and forth between the two applications. For example, they first created a work group using NUCLEUS, and then they sent a message to all work group members using PIPELINE. Next they added a new work group member using NUCLEUS, and then set up a meeting for the work group using PIPELINE. The tutorial manuals were tailored to versions of PIPELINE appropriate to

---

**Figure 4: Description of Partial Conceptual Models**

NUCLEUS Conceptual Model

NUCLEUS is used to create work groups and the information they use. Work groups consist of members selected from lists of people in the system described by a work group name. An information structure is created to store various types of work group information. Any member can create an information structure and assign access to other work group members. Work group members then use the information structure by either browsing through it or typing in additional information. Any person can review work groups to see a summary of the work group. Work groups and information structures can be modified or deleted as needed.

PIPELINE Conceptual Model (Consistent with NUCLEUS)

PIPELINE is used to support communication for work groups created by NUCLEUS. Members can send messages or memos to people or whole work groups. Any member can create a meeting for the work group by selecting from a list of work groups and meeting rooms and typing in the objective, date and time. If the meeting is in an electronic meeting room, work group members can use PIPELINE to participate in the meeting, which allows them to selectively send messages to other people at the meeting. Any person can review meetings to see a summary of the meeting. Memos and meetings can be modified or deleted as needed.

PIPELINE Conceptual Model (Inconsistent with NUCLEUS)

PIPELINE is used to support communication for project teams made by NUCLEUS. Project people can send new messages or memos to system users or whole project teams. Any project person can make a new meeting for the project team by choosing from a summary of project teams and meeting rooms and entering the objective, date and time. If the meeting is in an electronic meeting room, project people can use PIPELINE to join in the meeting, which allows them to selectively send messages to other system users at the meeting. Any system user can see old meetings to get an overview of the meeting. Memos and meetings can be changed or erased as needed.
Figure 5: Screen Prints of NUCLEUS and PIPELINE

NUCLEUS: In Both Treatments

PIPELINE: Consistent With NUCLEUS

PIPELINE: Inconsistent With NUCLEUS
the treatment. When all tutorial tasks were finished, participants completed the experimental tasks without the aid of training materials or online help. The experimental tasks were similar to the tutorial tasks, and the sequence of tasks also required moving back and forth between applications. Finally, participants completed an interface satisfaction instrument and took an objective test to measure their knowledge of the applications (described below). All materials and instruments were adapted from previous studies where they were extensively pretested.

### Dependent Variables

Five dependent variables were operationalized to measure the hypothesized effects. The first dependent variable was participant knowledge of both applications, measured by a 45 question objective test adapted from previous research. Each question made a statement about one of the applications, and the participant evaluated the statement as being either true or false. Participants then indicated their confidence in their answer on a scale from one to five. Confidence weights increase the reliability of objective tests, and they were used to calculate a weighted overall score by summing confidence weights for all correct answers (Ebel, 1965). Two versions of the objective test were created by substituting the appropriate terms used for objects and operations in PIPELINE for each treatment without changing the correctness of the statement about an application.

Three user performance variables were operationalized. First, the amount of time to complete the experimental tasks was automatically captured in a log file. The number of user interactions required to complete the experimental tasks was also captured in the log file, with more user interactions indicating more errors when interacting with the applications (Norman, 1983). The accuracy of the participant’s work was measured by inspecting the contents of the database and scoring the accuracy of the objects created, objects modified, and messages sent using a detailed scoring sheet. For example, for creating a work group one point was assigned for each of the following: it was created, it was created only once, the title was correct, the objective was correct, the leader was correct, and the members were correct. Therefore, higher scores indicated greater accuracy.

Finally, participant satisfaction was measured using a four question perceived ease of use instrument based on the technology acceptance model (Davis, 1989). A seven point Likert scale asked for an assessment of the likelihood that a statement about ease of learning and use would be true for the participant.

### Summary of Experimental Controls

All factors in the multiple application research model other than interface consistency were controlled. Degree of integration was controlled by designing integrated applications specifically for the experiment. Degree of concurrent use was controlled by the specification of the order of tasks completed. Participants switched back and forth between the two applications. The training environment was controlled by having participants learn and use both applications in one session. User characteristics were controlled by random assignment to one of two treatments and by using individual differences as covariates in statistical tests (VZ2, grade point average, access to a computer, length of use, and age). Responses on the background questionnaire and the VZ2 test were also used to test the effectiveness of random assignment, and no significant differences were found.

### Results

All participants completed the required work within the two hour session. Table 1 shows the summary statistics for the five dependent variables. The experimental tasks took on average just over 15 minutes, and the accuracy of the completed work was high, averaging 37 points out of a maximum of 44. Participants were very satisfied with the applications, as indicated by the perceived ease of use score of 6.41 on the 7-point scale. The raw scores on the objective test averaged 72 percent, and the average confidence per question was 3.63 on the five point scale. The adjusted scores based on the confidence weights averaged 134.4 (Cronbach’s alpha = .77).

Analysis of Covariance (ANCOVA) was used to test the effects of the treatments on the five dependent variables. The adjusted treatment means, standard deviations, significant covariates, $R^2$ for the model, power for the main effects, and significance of $F$ for the main effects are shown in Table 2. No significant differences were found in time for completing experimental tasks, number of user interactions, or accuracy of the work, although individual difference covariates explained part of the variance ($R^2=.23$ for time, $R^2=.18$ for user interactions, and $R^2=.27$ for accuracy). No significant difference was found for perceived ease of use, and individual differences did not explain any of the variance in scores.

There was a significant difference in accuracy of mental models as measured by the objective test ($F_{1,53}=7.45$, $p<.009$; power=.76). Three covariates in the model plus the treatment effect accounted for 37% of the variance on test scores ($R^2=.37$), and the treatment effect accounted for over 12% of the variance beyond the individual differences (partial $eta^2=.123$). However, the direction of the effect was different from that hypothesized. Scores of participants in the inconsistent conceptual models treatment were higher, indicating that inconsistent conceptual models across applications resulted in more accurate mental models of both applications.

Additional post hoc analyses were conducted to determine if a small set of questions on the objective test accounted for the higher scores in the inconsistent conceptual models group, which might indicate a treatment-instrument interaction. However, only two out of 45 questions had significantly higher scores in the inconsistent conceptual models group, and the greatest difference in means was only 1.58 points (3.90 vs. 2.32). Therefore, no one question or small group of questions
accounted for the difference in scores; rather, better performance on 31 of the 45 questions accounted for the difference.

### Discussion and Conclusions

The results of the experiment raise some interesting issues about the way mental models of computer applications are formed and used. No evidence was found for better performance or greater satisfaction when a user learns and uses multiple applications with externally consistent conceptual models. However, some evidence was found that indicates more accurate mental models might be developed when conceptual models are inconsistent. There are also practical implications to the findings.

### User Performance and Satisfaction

Performance benefits are thought to occur when the user develops more accurate mental models of applications. But in this study, time to complete tasks, the number of user interactions, and the accuracy of the work did not differ across treatments while accuracy of mental models did differ. Previous studies have found performance benefits when the procedural rules used were consistent across applications (Karat et al., 1986; Foltz et al., 1988; Polson, 1988; Kellogg, 1987; Satzinger & Olfman, 1998), but all applications in this experiment used consistent procedural rules. Perhaps the consistency of the underlying syntax of command sequences makes a difference in the accuracy of mental models, while the terms used to express “words” in the language are less important (As discussed above, there is much more to the conceptual model.
than the use of words). In terms of schema theory, rather than learning more concepts, users might have formed knowledge structures that contained multiple synonyms for each concept. With integrated applications that are learned together and used concurrently in a work session, the user might integrate the objects in the task environment into one overall structure and then readily accept multiple terms to describe the concepts.

The low power of the tests of these user performance variables makes these results inconclusive, however. Similar applications and performance measures have been used in previous studies (Satzinger & Olfman, 1998) where differences were detected, and individual difference variables in this experiment did account for a substantial amount of the variance in user performance scores. The measures might not be sensitive enough, however, to detect differences in performance because of inconsistent conceptual models. Procedural differences might more directly affect performance and be more measurable.

User satisfaction as measured by the perceived ease of use instrument also did not significantly differ between treatments. Participants in both treatments found these two applications very easy to learn and use, particularly when compared to the applications they had previously learned in their introductory computing skills course (WordPerfect, Lotus 1-2-3, and dBASE). NUCLEUS and PIPELINE were designed to be easy to learn and use, with menus, selection with a mouse, dialog boxes, and panels with check boxes. For this experiment, however, this may have resulted in a ceiling effect for the perceived ease of use measure.

Perhaps more importantly, the conceptual models (all versions) were designed to be analogically consistent with the task environment. As a result, the interfaces to the applications are intuitive when viewed in isolation. The differences in the two versions of PIPELINE resulted from their analogical consistency with different hypothetical task environments. As discussed above, participants did not seem to have difficulty forming an overall knowledge structure of the task environment and using synonyms to describe concepts that differed across applications. Therefore, analogical consistency with the task environment (even when inconsistently applied) might be more important for ease of learning and use than external consistency.

**Accuracy of Mental Models**

What about the result indicating more accurate mental models as measured by the objective test when conceptual models were inconsistent? It was expected that inconsistent terms used to describe the objects and operations across applications would require learning more concepts and then lead to incorrectly associating concepts in one application with the other application. The latter effect in particular, called restructuring (Rumelhart, 1980) or multiple mental model interference (Staggers & Norcio, 1993), was expected to appear in the results of the objective test. Users might confuse the terms used for objects and operations in one application with those in the other application when recalling facts about the applications.

However, the results indicate the opposite. Participants became less confused, although more research is required to understand why. One explanation is consistent conceptual models across applications might have led participants to form one global mental model of the cooperative work system rather than two separate mental models of the two applications. Since the applications were integrated and used concurrently to complete work tasks, participants in the consistent conceptual models treatment might have felt less need to distinguish between NUCLEUS and PIPELINE as separate applications. When the conceptual models were inconsistent, however, differences in the applications might have helped participants differentiate between applications, reinforcing that each application is a separate entity. As discussed above, multiple synonyms for concepts in the task environment were not difficult to learn, and their presence might have provided a cue to reinforce that separate applications were being learned. The result was that participants in the inconsistent conceptual models treatment were better able to differentiate between applications when recalling facts about the applications.

A distinctive visual appearance for the interface of an application has previously been found to facilitate the formation of separate mental models of multiple applications when they are initially learned (Satzinger & Olfman, 1998), so there is some support for this interpretation. The distinctive visual appearance effect, however, might provide additional benefits beyond initial learning when the user attends to multiple applications in multiple windows. Visual differences rather than visual similarities might help the user more readily locate the window that contains needed information when interacting with multiple applications.

**Conceptual distinctiveness** might also provide an initial benefit when learning multiple applications if it helps the user differentiate between applications, but unlike the distinctive visual appearance effect, it is difficult to recommend introducing inconsistencies deliberately. However, since differentiating between applications is an important training outcome, the training environment should help the user differentiate between applications when providing training for multiple applications.

Such differentiation is important because when confronted with a task, the user must first select the appropriate application that supports completing the task. Although the concept of separateness begins to break down with integrated applications, separate applications do provide a useful framework for organizing available functionality for both users and designers. As it becomes more difficult to differentiate between applications based on functions or manipulated objects, training materials can reinforce the differences to help reduce complexity for the user.

**Practical Implications**

There are many practical implications that follow from this research, but as with any carefully controlled experimen-
tual study, most of the practical implications should be considered speculative until they can be investigated further. Some issues discussed above obviously reinforce the importance of design guidelines other than just interface consistency. For example, an application that is carefully designed to be easy to learn and use in isolation will probably be easy to learn and use regardless of its consistency with other applications. The recent proliferation of Internet web pages demonstrates this point. Some are quite easy to use even though they vary considerably from each other in their design. It remains clear that designers should focus on the task environment of the user, selecting the best conceptual model even at the expense of consistency with other applications likely to be used by user.

Some additional practical implications concern decisions about adopting applications and training their users. Selecting applications to adopt that are not fully consistent might be appropriate if the applications are otherwise easy to learn and use. A “best of breed” decision rule is a viable alternative. Probably the most important practical application of this research concerns user training. As discussed above, training programs should help the user differentiate between applications so they can choose to use the correct application based on the task at hand. Training that focuses too much on the integration of applications rather than on their distinct functionality might be detrimental to new users. It is advisable to first help the user understand each application as a separate entity, particularly if all applications to be used are integrated.

Limitations and Needed Research

Several limitations of this study should be noted. First, user knowledge, performance, and satisfaction were only measured immediately after initial training. The results might be different if measured later. Second, the participants did not expect to have to use the applications after the session, so they were learning the applications without having to relate them to a real work environment. This may have affected the development of their mental models of the applications. Third, only terms used for concepts in the conceptual models were manipulated; otherwise, these applications were very consistent and easy to learn and use. The manipulation might have been too subtle to produce measurable performance effects, and user interviews might have helped demonstrate this. The careful manipulation of conceptual consistency, however, is also a strength of the approach used in this study.

This study was limited to participants interacting with integrated applications that were used concurrently and learned together in one training session. The possible explanations given for the results all require additional research, which can be guided by the multiple application research model shown in Figure 3. Based on the findings of this study, research should be conducted to examine the effects of conceptual consistency when applications are learned in separate sessions. Conceptual inconsistency might lead to detrimental restructuring or mental model interference when one application is thoroughly learned before the other. Additionally, since the only hypothesized benefit of conceptual distinctiveness across applications is the initial aid in differentiating between applications when they are learned in one session, research on approaches to training that help the user differentiate between integrated applications is certainly called for. Finally, more research should be conducted on the effects of additional types and degrees of interface consistency.

Designing the user interface is a complex task, and designing consistent user interfaces across applications remains problematic. In this study conceptual consistency was carefully manipulated, the concept of mental models and schema theory were used to predict the effects and explain the results of the manipulation, and a multiple application research model extended from user training research was used to guide and control the experiment. Additional research using this approach to the study of interface consistency and the mental model development process should lead to greater understanding of the way people learn and use the computer applications that support their work.

Acknowledgment

The author wishes to thank Lorne Olfman and Munir Mandviwalla for their support of this research.

References


John W. Satzinger is an associate professor in the CIS Department of Southwest Missouri State University. He holds a Ph.D. in MIS from the Claremont Graduate School and was on the faculty of the University of Georgia from 1991 to 1996. Dr. Satzinger has published articles on user interface design, group support systems, and end-user training in journals such as Journal of MIS; Database; Accounting, Management, and Information Technologies; and Information Systems Management. He is also co-author of The Object-Oriented Approach, a book on object-oriented information system development.
related content

Understanding the Nature of Task Analysis in Web Design
www.irma-international.org/chapter/understanding-nature-task-analysis-web/18260/

Electronic Mail in the University
www.irma-international.org/article/electronic-mail-university/55676/

Assessing User Experience via Biometric Sensor Affect Detection
www.irma-international.org/chapter/assessing-user-experience-via-biometric-sensor-affect-detection/183015/

Perceptions of End Users on the Requirements in Personal Firewall Software: An Exploratory Study
www.irma-international.org/chapter/perceptions-end-users-requirements-personal/7036/

Viewpoint Modelling with Emotions: A Case Study
www.irma-international.org/article/viewpoint-modelling-with-emotions-a-case-study/163933/