

The role of schemas in user-acceptance of software.

Name: Pavel Gokin

Course: HF 700, Fall 2004

Instructor: William Gribbons, Ph. D.

Date: 11/10/04

The way we perceive the word is determined by the interaction of bottom-up and top-down processing (Wickens, Lee, Liu, & Becker, 2004, p. 75). While bottom-up processing is governed by the way our sensory system is initially structured (or “hard-wired”), top-down processing is influenced by our past experiences, or prior knowledge, which lets us assign meaning to the things we perceive (Solso, MacLin, & MacLin, 2005, p. 75). This knowledge is stored in our long-term memory and organized in the form of schemas, mental models, and cognitive maps (Wickens, et al., 2004, p. 136). The focus of this paper is on schemas and their effect on our ability to transfer our knowledge to new, but similar situations.

Wickens et al. define schemas as knowledge structure about a particular topic (Wickens, et al., 2004, p. 136). We may have a schema for an interior of a hardware store that includes wrenches, cans of paint, garden hoses and so on (Matlin, 2005, p. 274). In a similar vein, our schema of a piece of word processing software may include the document view, toolbars, a spell checker, a “format” menu, a “print” command, etc. Schemas formed in one situation guide our recognition and understanding of similar situations, provide expectations of what should occur, and allow us to predict what will happen in these similar situations (Matlin, 2005, p. 275). For example, once we have learned—or created a schema of—one word processing program, we are likely to “apply” this schema to determine what functions, buttons, and displays we can expect to find in another word processor, what actions would be possible, and what results those actions are likely to have.

One common kind of schema is a script: a sequence of events that are associated with a highly familiar activity (Matlin, 2005, p. 275). In addition to event sequences, scripts contain information about the actors involved in an event and their goals (Markman, 1999, p. 195). To use the script example from Arthur Markman’s book “Knowledge Representation” a person whose goal it is to get drunk may choose to use a “Getting drunk at a bar script” in deciding how to accomplish this goal (Markman, 1999, p. 195). This script would contain the actors such as the person who’s actually getting drunk, a bartender and fellow drinkers, as well as the sequence of events required for successfully reaching the goal such as entering a bar, ordering a drink, consuming a drink, etc. To use an example from a software world,

accomplishing the goal of bolding a piece of text requires selecting the text, pointing to the “B” button on the toolbar with the mouse, and clicking the button. Since scripts are a subset of schemas (Matlin, 2005, p. 275), it follows that, like schemas, they can be used to allow us to predict what will happen in situations where the actors and goals are similar. For example, having learned the script for using the text highlighting function in Microsoft Word, a user may want to apply this script to the same function in a different word processor.

In the “getting drunk at bar” script example, Markman notes that this script not only contains a *sequence* of events for getting drunk, but also *causal* information: that drinking alcohol *causes* intoxication (Markman, 1999, p. 194). Causality comes into play when a user, wishing to accomplish the same overall goal, is faced with slightly different circumstances. To use our word processor example, a user may learn (through repeated use or by reading the manual) that clicking the print icon causes the document to be printed. Now, imagine the same person using a different word processor where the print icon is not on the toolbar. In this case the familiar script of clicking the print icon on the toolbar will not work. However, if the user’s schema of a word processor includes a print icon, and she finds it elsewhere in the UI, she is likely to activate it, expecting the same result as clicking the print icon on the toolbar. This example illustrates a situation when “many low-level actions cannot be carried out as planned, [but] the overall goals can be satisfied by substituting other actions that achieve the same end” (Markman, 1999, p. 195). In more general terms, this is the case of using prior knowledge of a familiar actor (the iconic button), action (clicking on the icon) and its effect (printing the document) to decide what to do in a novel situation.

The Case: StarOffice 7.0 Suite by Sun Microsystems.

In one of its press releases Sun touts StarOffice as “the right alternative to Microsoft Office,” calling it familiar and easy to use (Sun Microsystems, 2002). Considering Microsoft’s dominance in the productivity software arena with its Office suite, “familiarity” may be Sun’s best strategy for getting Microsoft’s customers to switch to StarOffice. This is because the users of MS Office will base their

generalized knowledge—or schema—of a word processor, a spreadsheet, etc. on their experience with the implementation of those programs in MS Office. Since people will “[use] information in one schema to deduce the properties of another” (Norman, 1988, p.116), they will try to apply what they know about word processors, spreadsheets, etc. to their experience with StarOffice. For example, once they learn that a word processor has a spell checker, they will expect to find that function in all other instances of a word processor they encounter. Therefore, unless the views, controls, and labels in StarOffice look and work similar to the ones in MS Office, users will have to learn new ways of accomplishing the same things—a process many of them may be unwilling to go through. Let’s take a detailed look at several aspects of StarOffice’s interface that differ significantly from their MS Office equivalents.



**New features.** This area is the least problematic from the standpoint of prior knowledge: the user encounters a feature, uses it, learns how it works. The schema is updated as a result. While the process requires learning, prior knowledge would play a minimal part here.

**Missing features.** Based on their schema of a word processor MS Word users will expect to find certain features in StarOffice like the ability to apply the format of a piece of text in one area to text in another area of the document (the format painter), set text in title case, or check the document for correct grammar. Unfortunately none of these features are found in StarOffice 7.0. If users are not aware that the features do not exist in the application, they may spend a considerable amount of time looking for them, becoming frustrated in the process.

**Same feature, different name.** MS Word “keeps” commands like “New,” “Open,” “Save,” “Print,” “Cut/Copy/Paste,” and “Undo/Redo” on a toolbar called “Standard.” StarOffice’s name for the toolbar with all these commands is “Function bar.” Users wishing to show/hide specific toolbars will have no way of knowing which toolbar they should check/uncheck. Since the names of most of the other toolbars are also different between the two office suites, the process of elimination can’t be used here. Therefore, the only way to determine the correct action here is through trial and error.

**Same feature, different location.** One example of this discrepancy is the location of the spell check button. In MS Word the button is on the “Standard” toolbar at the top of the document window,

whereas in StarOffice this button is on the “Main” toolbar placed on the left side of the document window. Another example is the location of the header and footer options: on the “View” menu in Word, but on the “Insert” menu in StarOffice. One consequence of this is that a user may conclude that a feature that does not “reside” in the familiar location does not exist. However, as users explore the application and these discrepancies become known, the users’ propensity to look elsewhere for features they can’t find in the familiar location is likely to increase. In this case, they may persistently search the application for a feature that does not actually exist.

**Features that look familiar but work differently.** In this case the familiar appearance of a feature may trigger a script that is not valid in the new situation. For example, to apply a color to a selection of text in MS Word one would select the text, click the little down arrow next to “A” (  ), and click the desired color. StarOffice has a similar-looking control that accomplishes the same thing. However, to use this tool one needs to click **anywhere** on the face of the button (  ) **and hold** the mouse button down for ½ seconds or more to make the color palette appear! A user familiar with MS Word’s implementation is likely to try to click the small green arrow next to “A.” Unfortunately this action will not produce the desired result. Similar appearance of the control would activate an invalid script to result in user error here.

All of these differences require learning; in this case, updating and creating schema and scripts to account for the behavior of the interface. This places an additional burden on the users’ cognitive capacity, making them less productive with their primary task: creating documents, spreadsheets, etc.

In “StarOffice 6.0 software and Microsoft Office XP Feature Comparison” Sun claims to have made ease of use advancements that let the users be more productive with StarOffice than with MS Office. These advancements may very well have placed StarOffice ahead of MS Office for users who have taken the time to learn it. Unfortunately, given the multitude of differences between the two products, not many users may be willing to commit to learning StarOffice to become that little bit more

productive. Jef Raskin describes a similar situation from his own consulting experience: “The client wants something that is significantly superior to the competition. But if it is to be superior, it must be different...Therefore it cannot be intuitive, that is, familiar.” He posits that in this case the only way to achieve an improvement while preserving familiarity is to remedy a major flaw with a minor fix (p. 152)—something that happens rarely.

In light of this, StarOffice designers must find a way to balance efficiency of use for existing users and ease of learning for new users, most of whom will be switching from MS Office. The overall recommendation for StarOffice designers, then, is to take a closer look at each of the features, and, absent a compelling reason to make it different, make it look and work as they do in MS Office. This would make it easier for MS Office users to transfer their knowledge to StarOffice. Only then will they embrace StarOffice in earnest.

Bibliography.

Markman, A. (1999). *Knowledge Representation*. Mahwah, NJ: Lawrence Erlbaum Associates.

Matlin, M. (2005). *Cognition, 6<sup>th</sup> Ed.* Hoboken, NJ: John Wiley & Sons.

Norman, D. (1988). *The Design of Everyday Things*. New York: Doubleday/Currency.

Raskin, J. (2000). *The Humane Interface: New Directions for Designing Interactive Systems*. ACM Press.

Solso R., MacLin, K., & MacLin, O. (2005). *Cognitive Psychology, 7<sup>th</sup> Ed.* Boston, MA: Pearson  
Prentice Hall.

Sun Microsystems, Inc. (October 17, 2002) Goodbye to licensing woes. Retrieved on 11/04/04 from  
<http://www.sun.com/executives/realitycheck/reality-101702.html>.

Sun Microsystems, Inc. StarOffice 6.0 software and Microsoft Office XP Feature Comparison. Retrieved  
on 11/04/04 from <http://www.sun.com/software/star/staroffice/6.0/compare/MSXPvsSO6.pdf>.

Wickens, C. D., Lee, J., Liu, Y., & Becker S. G. (2004). *An Introduction to Human Factors Engineering*.  
Upper Saddle River, NJ: Pearson Prentice Hall.