Published in *Interfaces*, No. 45 Winter 2000, 4-7.

Noddy's guide to consistency

Andrew Monk, University of York, UK., A.Monk@psych.york.ac.uk

Most people would feel that consistency in a web site or user interface is a GOOD THING, but are not terribly sure what consistency is. This article describes some basic kinds of consistency and why they are important in interface design.

Consistency not uniformity

Imagine a window where the menu tabs were labelled "Menu1", "Menu2", "Menu3". Each lead to the same number of items each of which was labelled "Item1, "Item2", "Item 3" and so on. This is an interface designed on the military principle "if it doesn't move paint it white". Everything looks uniformly the same.

The point is that uniformity does not necessarily lead to usability as it makes things difficult to discriminate. An examples of uniformity that recently irritated me was a standard front page for internal reports. The information that distinguished the reports from one another, title, authors, date etc., was in 8 point text and hidden in a mass of logos and titles that appeared on all the reports. Put together the collection of reports looked very neat and uniform but finding the one you wanted was a nightmare. The same principle applies to user interfaces. If all the icons are identical, except for the labels under them, why have icons at all?

So if consistency is not uniformity, what is it? Actually it is several things and to explain them I will need to develop a THEORY.

The bluffer's theory of human-computer interaction

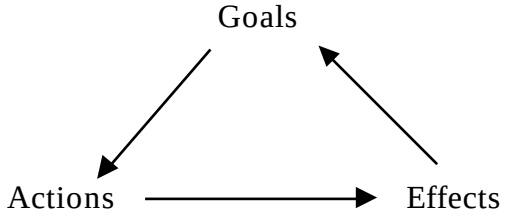

Figure 1. The Monk & Dix Triangle ([20])

Figure 1 is a caricature summarising several classic theories of what goes on when someone interacts with a computer (see [18] for a review of these models). The arrows indicate that it is cyclic. It works as follows.

The user has *goals*. Perhaps you are using some untrustworthy software (several examples come to mind) and you are getting nervous about the machine crashing. You decide it would be a good idea to save the work you have done so far. We can describe this state of mind as having the goal "save your work".

The goal "save your work" leads you to take some *action*. As this is a graphical user interface you scan the screen for something that might do this. Your eye lights on the "File" menu tab. You click on it. We can describe that as taking the action "click on menu tab File".

Clicking on the menu tab makes the display change, the menu drops down. This can be described as an "*effect*".

The visible effects of your action lead you to change your *goals*. Let us say that the goal "save your work" had lead you to generate the sub-goal "reveal save command". The effect "file menu drops down" could lead you to replace this sub-goal with another sub-goal "select save command".

The new goal set and new display state lead to a new *action* and the cycle continues.

OK, I suspect that many readers will be glazing over at this point. What is the point of all this? Well it leads to the two most important definitions of consistency: action-effect consistency, that is, consistency in the effects of actions; and task-action consistency, that is consistency in the way actions relate to task goals.

Action-effect consistency

Consistency is mainly about ease of learning. The hope is that if a user interface or web page is consistent then one will get what psychologists call "transfer of training". That is to say, learning to do one thing in one context will make it easier to learn how to do similar things in similar contexts. This will of course be easier if the same action always leads to the same effect.

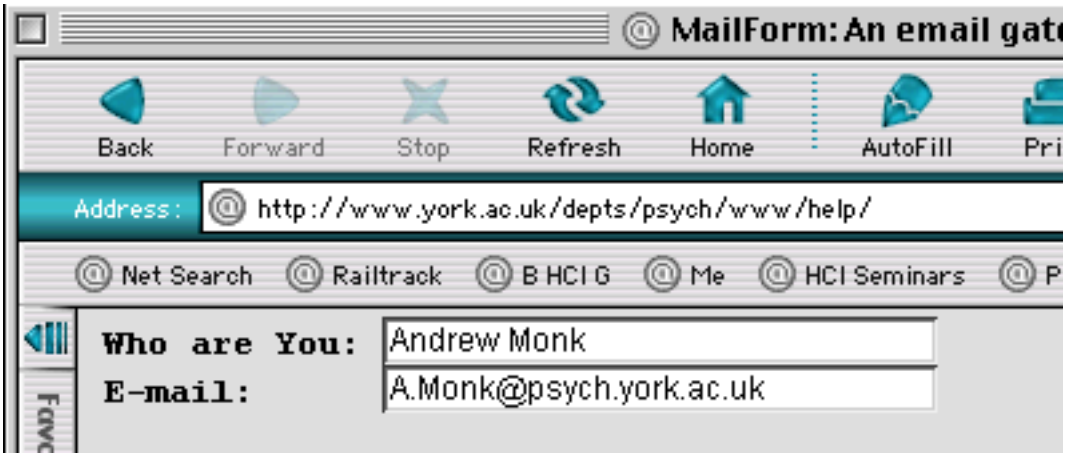

Figure 2. Action-effect consistency says that the same action will have the same effect irrespective of the context. What will happen if someone double clicks on the word "york" in the url ("Address:") and in my email address? My browser is action-effect consistent, the rules for what gets to be selected seems to be the same. Is yours?

Action-effect consistency then is the principle that if the user takes some action it should have the same effect whatever the context. Let us say you are working on a web page that contains a form as in Figure 2. Double clicking on a word should have the same effect whether one is editing a field in the form or editing the url. Try it on your own browser. Is your browser action-effect consistent?

Action-effect consistency has been the main contribution of the "style guide". This rather misleading term is taken to mean a set of guidelines describing how a graphical user interface should work. For example, they lay down what a dialogue box should look like, how it should behave when the user interacts with it and when it should be used rather than some other device such as a menu. Apple produced the first style guide in 1987 [1, 2]. Style guides encapsulate a great deal of empirical and analytic work carried out by HCI researchers to find out what actually was the best way of doing things. There are now style guides for all the commonly used graphical user interfaces, e.g.,[14]. A user interface designer will not generally have to consult a style guide because style guides are enforced by software tools. Thus a software developer using a programming tool such as Visual Basic will find it much easier to obey the style guide than to ignore it and develop idiosyncratic action-effect inconsistent interfaces.

So, action-effect consistency is enforced by style guides. Because of the Windows style guide, a user only needs to learn the effects of actions on a Windows component once. If you know how a dialogue box behaves in Excel then you also know how it behaves in Word. This was not always the case.

Another way of expressing this principle is to say that interfaces should be "mode free". Unnecessary modes (e.g., Excel-mode versus Word-mode) should be avoided but sometimes they are inevitable. In particular, small device like mobile phones have only a few buttons that are the only channel of communication from the user to the device. The enormous number of commands the user could issue to the phone have to be funnelled through this narrow channel. Inevitably the action of pressing a particular button will have different meanings depending on the mode the phone is in. For example, in normal mode

pressing a number key has the effect of putting a number on the phone's display. In letter-entry mode pressing a key adds a letter.

Modednesss (action-effect inconsistency) is less of a problem if the user is aware what mode they are in. The above example of modedness is workable because the mode is clearly signalled by the prompt in the display. Hidden modes should always be avoided. Actually there are two letter-entry modes on my mobile phone (see Figure 3). If you press "*" you can toggle between upper and lower case letters. Unlike the shift lock on a keyboard this changes the case of the last letter entered. This is rather clever. The shift lock on a keyboard is effectively a hidden mode. When you are looking at the screen, there is no way of telling what case the next letter will appear in. This regularly catches me out and I often type half a line before I realise it is all in capital letters. With the mobile phone the mode is signalled by the case of the last letter on the display. The only times that mode is not signalled in the display is when you have not yet entered any letters or the last character was not a letter. This was the only case of a hidden mode I have detected in my phone. It would appear that the people who devised the interface for Ericssen phones gave some thought to minimising the impact of action-effect inconsistency.

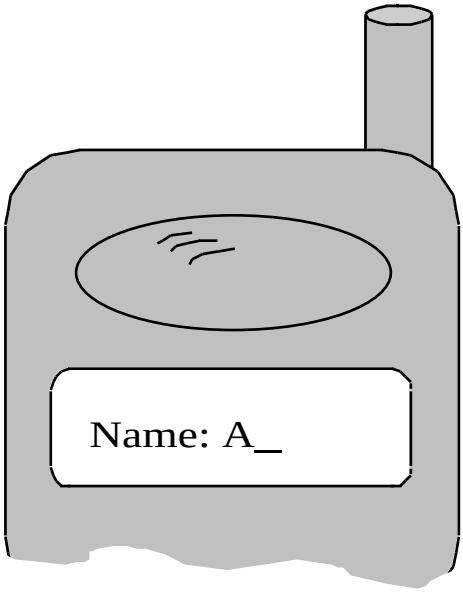

Figure 3. Hidden modes and the Ericssen PF768. Will the next letter be upper case or lower case? This mode is signalled by the last letter so we know that pressing the key for G will result in "G" not "g". Only when there is no last letter is this a hidden mode.

Task-action consistency

I use the terms "task" and "goal" interchangeably here. "Task-action consistency" is preferable to "goal-action consistency" as it links to work on Task Action

Grammars [23, 25]. "Goal" is preferable in the Monk & Dix triangle (figure 1) because it links to Card, Moran and Newell's model human processor [4].

Task-action consistency is intended to result in transfer of training when learning the set of actions needed to achieve similar goals. The idea is that similar goals should require similar sets of action to achieve them. The original work in this area took the example of a drawing package that allowed one to draw circles squares and so on, as well as to enter text. The set of actions required to enter text were quite different to those need to the other drawing commands. To draw a circle one selected the relevant tool and then drew the object. In contrast, there was no text tool, one simply clicked and typed. From the point of view of the designers this is eminently reasonable as, in terms of the software architecture, these are very different tasks. The problem was that the users saw these tasks as very similar. The users found it confusing to have to learn one set of rules for drawing objects and another to enter text. They would have found it easier to learn a single rule "select the tool for the thing you want to insert and then insert it".

```
rm [-f] [-i] file ...
ls [-RadLCxmlnogrtucpFbqisf1AM] [names]
mv [ -if ] file1 [file2 ...] target
cp - [-fip] source target
cat [-u] [-s] [-v [-t] [-e]] file . . .
lp [-c] [-ddest] [-nnumber] [-s]  file...
```

Figure 4. Unix commands with their syntax, an example of task-action consistency?

As another example of task-action consistency take the UNIX command set (see Figure 4). These have a consistent syntax for the arguments they take, also key command names may be generated by deleting vowels. So the task "move" (a file) is achieved by the command "mv", copy is achieved by "cp", list by "ls" and so on. This soon breaks down (cat, lp?) but was an admirable attempt at task-action consistency all the same.

Task-action consistency turns out to be more difficult to pin down than action-effect consistency. It is apparent in the examples given above that task-action consistency, like beauty, is in the eye of the beholder [8, 26]. Different people will see different tasks as similar and dissimilar. For this reason, style guides encourage task-action consistency by suggesting multiple task-action methods. Some people will see entering text in a drawing package as similar to entering text in a word processor, so, give them a method that is similar. Some people will see it as similar to drawing a circle, give them an additional tool that works that way. This accounts for the fact that there are many ways of achieving a given task in a large application like Microsoft Word, and that most people only ever use one of them.

Other sorts of consistency

So far we have two rules for consistency: (i) the same actions should lead to the same effects and (ii) similar tasks should require similar sets of actions. The former action-effect consistency rule has to be tempered by the knowledge that sometimes it will have to be broken and if this is the case one should avoid hidden modes. The latter task-action consistency rule is tempered by the knowledge that different people will see different tasks as similar and so there may need to be several ways of doing things. What other forms of consistency may lead to usability?

What about the third side of the Monk & Dix triangle? This would imply a need for consistency in the way effects on the display lead to changes in goals of the user. A designer can change the actions needed to achieve a goal or the effects of an action but has little control over what goes on in the head of a user. It is difficult to see what form these consistency rules might take.

Consistency of syntax can be seen as an example of task-action consistency at a very high level of generality. So, there is a general "noun-verb rule" stated in the style guides for most graphical user interfaces. This simply says that one should select an object before the action to be taken on it. This is task-action consistency at the level of all tasks.

Reversibility is another of these general principles expounded in all style guides. Being able to reverse the effect of any action encourages learning by exploration. Thus style guides prescribe a variety of devices for undoing the unwanted effects of actions taken by a user, e.g.: the "back" button in a web browser; the "cancel" button in a dialogue box or the "undo" function in a word processor.

Consistency *with* principles like reversibility is obviously a good thing but it is not really the same thing as consistency *in* the way a user interface works which is what this article is about. Consistency with style guides and international standards like ISO 9241 is also a good thing but not what this article is about.

Future research: consistency across heterogeneous user interfaces to the same data

I can access my bank account through an ATM, a call centre or a web browser. Soon I will probably be able to program my video recorder through my mobile phone or with a wireless keyboard and the TV screen. In each case the same data and functionality is accessed through a range of very different devices. The goals may remain the same but the actions and effects are very different. One could design for action-effect and task-action consistency within each of the separate interfaces, but how does one reason about consistency across interfaces?

A possible solution is to abstract. Rather than describing actions at a concrete level (e.g., click on X, type Y) one can use a more abstract level that applies whether one is using a keyboard, mouse, a stylus or speech (e.g., select X, enter Y). A type of consistency across heterogeneous user interfaces can then be achieved by ensuring that the same goals lead to the same abstract actions with each interface. This is different to the definition of task-action consistency provided above which is a requirement for simplicity in task-action mappings.

This is a requirement for the same abstract task-action mappings across interfaces.

It may also be possible to abstract the effects (e.g., "file menu drops down" -> "menu displayed"). One could then check that the same abstract action lead to the same abstract effect in all the interfaces. The problem with this is that different devices are capable of different effects. The small display on a mobile phone can display many fewer menu choices than a VDU screen. Menus implemented by speech will only work if they are limited to three or four items. Unless all devices are constrained to some lowest common denominator, the action-effect mappings will be necessarily different. It remains to be seen whether it will be possible to devise suitable abstractions to solve this problem and whether the kinds of consistency checking they allow will be enough to give effective transfer of training.

One problem that abstraction will not solve is how to provide a common product image. When I access my bank account I want it to appear familiar and to project the same brand image. I want the process of conducting transactions with it to be familiar too. Transactions need to have familiar landmarks marking the beginning and ends. How you ensure consistency of this kind across speech, very small and large displays is even less clear.

Conclusions

There has been much thought given to consistency over the years. The new problem of consistency across heterogeneous devices guarantees there will be even more thought given in the future. What I hope this article demonstrates is how even quite complex topics such as consistency can be tackled by systematic analysis. More generally, there is theory in HCI, but it is often hidden behind the recommendations that are passed on to engineers.

Bibliography

I have attempted to write this as an accessible introduction to the topic of consistency. To obtain a deeper understanding of the topic the following reading is recommended.

Theories of human-computer interaction involving goals [3, 4, 15, 18, 19, 21], theories of how we learn goal-to-action mappings [10-13] and a simple way of checking an interface that draws on these theories[24].

Early attempts to describe actions-effects mappings were made in the context of user interface management systems (UIMS) and "the separable user interface" [6, 7] where there was a concept of a "dialogue model" to serve just such a purpose.

More or less formal ways of describing how different actions can lead to different effects so that one can reason about the usability of a computer system [5, 9, 16, 17, 22].

Style guides [1, 2, 14].

ISO 9241, "Ergonomics requirements for office work with visual display terminals (VDTs)" is the International Standards Organisation standard for

different kinds of human-computer interaction. There is also a standard describing a user-centred process of design, ISO 13407, "Human-centred design processes for interactive systems".

References

1. Apple *Human Interface Guidelines: the Apple Desktop Interface.* Addison-Wesley: Reading, Massachusetts, 1987.

2. Apple *Macintosh Human Interface Guidelines.* Addison-Wesley: New York, 1993.

3. Blandford, A. and Duke, D.J. Integrating user and computer system concerns in the design of interactive systems. *International Journal of Human-computer Studies*, 46, (1997), pp. 653-679.

4. Card, S.K., Moran, T.P. and Newell, A. *The psychology of human-computer interaction*. Lawrence Erlbaum: Hillsdale, NJ, 1983.

5. Dix, A.J. *Formal methods for interactive systems*. Academic Press: London, 1991.

6. Edmonds, E. *The separable user interface.* Academic Press: London, 1992.

7. Green, M. Design notations and user interface management systems. In *Proceedings of the Seeheim workshop on user interface management systems*, Springer-Verlag: Berlin, 1985, pp. 89-107.

8. Grudin, J. The case against user interface consistency. *Communications of the ACM*, 32, 10 (1989), pp. 1164-1173.

9. Harrison, M.D. and Dix, A. A state model of direct manipulation in interactive systems. In *Formal methods in human-computer interaction,* Harrison, M.D. and Thimbleby, H., Ed., Cambridge University Press: Cambridge, UK, 1990.

10. Howes, A. A Model of the Acquisition of Menu Knowledge by Exploration. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems,* Plaisant, C., Ed., ACM Press: New York, 1994, pp. 232.

11. Howes, A. and Young, R.M. Learning consistent, interactive and meaningful device methods: a computational approach. *Cognitive Science,* 20, (1996), pp. 301-356.

12. Kieras, D.E. and Polson, P.G. An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies,* 22, (1985), pp. 365-394.

13. Kitajima, M. and Polson, P.G. A comprehension-based model of correct performance and errors in skilled, display-based, human-computer interaction. *International Journal of Human-Computer Studies*, 43, (1995), pp. 65-99.

14. Microsoft *The Windows interface guidelines for software design.* Microsoft Press: Redmond, 1995.

15. Miller, G.A., Gallanter, E. and Pribram, K.H. *Plans and the structure of behaviour*. Holt, Reinhart and Winston: London, 1960.

16. Monk, A.F. Mode errors: a user-centred analysis and some preventative measures using keying-contingent sound. *International Journal of Man-machine Studies*, 24, (1986), pp. 313-327.

17. Monk, A.F. Action-effect rules: a technique for evaluationg an informal specification against principles. *Behaviour Information and Technology*, 9, 2 (1990), pp. 147-155.

18. Monk, A.F. Cyclic interaction: a unitary approach to intention, action and the environment. *Cognition*, 68, (1998), pp. 95-110.

19. Monk, A.F. Modelling cyclic interaction. *Behaviour and Information Technology*, 18, (1999), pp. 127-139.

20. Monk, A.F. and Dix, A. Refining early design decisions with a black-box model. In *People and Computers 3*, Diaper, D. and Winder, R., Ed., Cambridge University Press: Cambridge, 1987, pp. 147-158.

21. Norman, D.A. Cognitive engineering. In *User centered system design: new perspectives on human-computer interaction*, Norman, D.A. and Draper, S., Ed., Lawrence Erlbaum: Hillsdale, NJ, 1986, pp. 31-61.

22. Olsen, D.R., Monk, A.F. and Curry, M.B. Algorithms for automatic dialogue analysis using propositional production systems. *Human-computer Interaction*, 10, (1995), pp. 39-78.

23. Payne, S.J. and Green, T.R.G. Task-action grammars: a model of mental representation of task languages. *Human-Computer Interaction*, 2, 2 (1986), pp. 93-133.

24. Polson, P.G., Lewis, C., Rieman, J. and Wharton, C. Cognitive walkthroughs: A method for theory-based evaluation of user interfaces. *International Journal of Man-machine Studies*, 36, (1992), pp. 741-773.

25. Reisner, P. Formal grammar and human factors design of an interactive graphics system. *IEEE Transactions on Software Engineering*, SE-7, 2 (1981), pp. 229-240.

26. Reisner, P. What is consistency. In *Interact'90*, (Cambridge, UK), (1990), North Holland, pp. 175-180.