# Icons at the interface: their usefulness

Yvonne Rogers

Iconic interfacing is now widespread. Increasing aspects of the system functionality — including objects, options, operations, states and messages — are being represented at the interface in this pictorial form. Against this zeitgeist, this paper sets out to discuss how useful icons really area and whether they live up to their expectations. A classification of the function and form of icons is outlined together with a proposal of the way in which a simple grammar of icon forms which maps onto the underlying system structure can be developed. Finally theoretical issues are discussed in the way in which information from icon-based displays is used when performing a task at the interface.

Keywords: user interface design, icons, pictorial representation, symbol systems, visual memory

Recently, the use of icons has been much heralded within system design as an effective form of displaying and interacting with information at the interface. To what extent this form of interfacing lives up to its expectations, however, has been a subject of debate within the HCI community. On the one hand there are those who believe that icons are an efficient means of communication while on the other hand there are those who consider them to be too vague and imprecise to be of any real value.

In support of their use, it has been argued that they can reduce the complexity of the system and hence make it easier to learn (e.g. Lodding, 1983). This is achieved by giving an immediate impression to the user that the system is easy to use and in doing so have a positive effect on the first time user. Underlying this is the assumption that since we live in a strongly visual and spatially organised environment it seems more compatible to learn interfaces that also use visual and spatial information. The way in which icons work at this level is as pictorial representations of various aspects of an interface metaphor. Hence as part of the desktop metaphor, in which the interface is designed to represent a desktop, objects typically associated with working at an office desk (e.g. files, calculators, folders) are depicted as icons.

Faculty of Mathematics, Computing Department, The Open University, Walton Hall, Milton Keynes MK7 6AA, UK. Tel: (0908) 655098

0953-5438/89/010105-13 $03.00  

Icons are also considered as a very powerful form of communication because they have the potential of being universally meaningful. For example, for those who know the code, an icon of a document looks like a document in English, French, Chinese and Arabic — even though it may be called by a different name in each of these languages.

Although we may find it easier to learn interfaces that correspond more with our way of viewing the world, it has been argued that icons may not actually fulfil this role. For example, many icons have been designed to represent system information whose meaning is often very ambiguous (see Figure 1). If, in these cases, users are forced to look up the meaning of icons in a manual to understand what they mean or just to get a more accurate description then clearly this type of communication is inadequate. Moreover, instead of being a form of representation whose meaning is supposed to be inherently obvious, the icon has become an arbitrary symbol. As such it may offer no real advantage over other coding methods (Jervell and Olsen, 1985).

One of the main problems with iconic interfacing is that while on some occasions it is relatively esy to interpret the intended meaning of an icon, for others a whole range of different meanings can be attributed to a single icon — each being as valid as the other. For example, an icon depicting a horizontal arrow by itself could be interpreted as 'turn the page', 'move the cursor to the end of the line', 'scroll the page to the right' and so on. The point is that the user has no way of knowing whether an initial interpretation is correct. Only by looking in a manual or activating the icon, which may have adverse effects, can the user find out whether the understanding matches the intended meaning. Unlike verbal language, in which there are a set of syntactic and semantic rules which provide us with a means of disambiguating the meaning of verbal language, pictorial language has, as yet, no equivalent set of rules underlying its comprehension. The paradox, therefore, is that while pictorial communication has the potential of being universally understood, it does not have the rules to guide this process.

The ambiguity of the meaning of an icon, however, can be narrowed down by the context in which it is displayed. Consider for a moment how we interpret pictorial symbols that we have not seen before, such as those that have been developed to represent public information and facilities. This typically happens

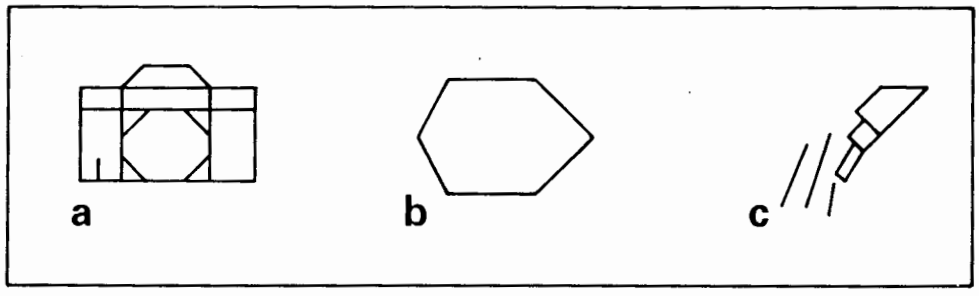


*Figure 1. Examples of potentially ambiguous icons based on those developed for UNICON*

when we are in a foreign country trying to locate something. One example is the signs that are used to indicate the location of toilet facilities in some Yugoslavian restaurants. Instead of displaying a male and a female figure on the respective doors, which are the symbols that are usually associated with these facilities, pictures of a typical man's shoe and a stiletto shoe are used. Even though these are not the symbols we would normally expect, they are none the less immediately recognisable. The reason for this is that when such symbols are used within the context of a restaurant our prior knowledge about where to find the toilet facilities provides us with the necessary information to interpret the intended message. In this instance, this would include the knowledge that there are generally separate toilets for each sex which are distinguished by either a gender-related label or symbol that is readily identifiable.

If the symbols of the shoes had been displayed in a supermarket, however, we might be more likely to attribute their meaning to refer to 'the different locations where men and women's shoes can be brought'. The significance of this is that it illustrates how the context in which a symbol is shown dictates the semantic features that are processed. To interpret this alternative meaning requires us accessing our stored knowledge about supermarkets rather than toilet facilities. It also depends on our expectations when looking at the sign. For instance, if we were looking for where to find shoes in the supermarket then the meaning of the shoe symbols would seem obvious. If we were looking where to find the toilets, on the other hand, it would be highly unlikely that we would associate the shoe symbol with this facility. The implications from this example is that different knowledge associations are triggered as a result of our expectations when looking at the signs and the environment in which they are displayed.

In relation to icons, the fact that they are being used at the interface and not, say, to represent public information already reduces their possible interpretation. In particular, the limited number of options, objects, etc. that are associated with a system's functionality can narrow down the range of possible meanings. For example, in Figure 1 the icons have all been designed to represent a specific class of operations concerned with the operating system activities of Unix such as creating a directory. However, for the context to provide the necessary disambiguating cues, the user must also know something about the system and the domain for which the icons have been designed. Similar to the public information symbol example, it cannot be expected for someone to interpret the meaning of an icon from just its pictorial content. In addition, a successful interpretation will depend on that person's expectations and knowledge of the task domain and their understanding of the underlying system operation.

For experienced users, who know how the system works and also how to carry out a task within that domain (e.g. designers who have been used to developing circuit boards by using a command-driven system) the link may be obvious, as they need only to infer the meaning of the icons from the pictorial code. It will be more difficult for those who are experienced in the task domain, but who have no knowledge of the system environment (e.g. designers who have not used a computer system before) as the users will also have to infer and

learn about the system functioning that the icons represent (e.g. students). The link will be the hardest to comprehend for those who have little or no knowledge of the domain or the system (e.g. students).

The usefulness of icons, however, does not appear to relate to the effort required by the user to comprehend the icons initially. Moreover, it seems the reverse is true in that many experienced users, who are used to command-based systems, are reluctant to use icons whereas new users often find icon-based interfaces easier to learn and use. The reason for this may have more to do with the trade-off between efficiency and the cognitive demands made on the user rather than the meaningfulness of icons per se. Experienced users, having learned how to interact with command-based interfaces, see no reason for converting to icon-based interaction. Furthermore, in many cases icon-based interaction is not as fast. This is because the user needs first to identify the icon on the screen and then select it whereas with command-based interfaces the user just has to press one or two function keys. To achieve the level of expertise required to use a command-driven interface efficiently, however, requires considerable learning. It is much easier for the new user just to recognise and then select an icon rather than have to learn the meaning of a set of command names and then recall which function keys to press.

The effectiveness of icons, however, does not have to be viewed as an 'all or nothing' situation. In particular, icons should not be seen just as a replacemen for command names to get new users 'started', as there are other ways in which they can be appreciated by all types of users. In particular, they can be used to represent information where verbal language and other forms of coding may not be as efficient. Examples include using icons as mnemonic tags for identifying files (Lansdale, 1988), as warning messages to alert our attention to important events, and as easily detectable visual structures for program debugging. The point is that there is potential for using icons at the interface for all types of users.

To determine the various ways in which users can benefit from iconic interfacing it is important, initially, to disentangle the different functions and forms icons can take. Such a classification can then provide us with a means of categorising icons with respect to their efficacy at communicating their intended meaning. It may then be possible to establish some form of primitive syntax and semantics which could be used as a basis to develop a set of icons for a given application.

## Classification of icons

The first commercially used icons were designed for the Xerox 'Star' office workstation (Smith *et al.*, 1982). As part of the design philosophy, Xerox decided to develop an interface based on the metaphor of an actual physical office. In so doing they created electronic counterparts to the physical objects of an office e.g. paper, folders, file cabinets, mailboxes etc. To represent these objects at the interface, concrete representations were designed in the form of icons. These were classified into two high level categories: *data* icons and *function* icons. Data icons were defined as representing objects on which actions are performed (e.g.

documents, folders and record files). Function icons represent objects that perform actions (e.g. file drawers, in- and out-trays). The two categories also had the following attributes:

- anything that can be done to one data icon can be done to all e.g. all data icons can be moved, copied, deleted etc.
- most function icons will accept any data icon e.g. you can move any data icon to a file drawer

Since this initial dichotomy, there have been a number of other classification schemes suggested which have attempted to distinguish further the different types of icons (e.g. Lodding, 1983; Jervell and Olsen, 1985; Gittins, 1986).

**Function**

Increasingly, icons are now being used for a range of functions. For example, as well as representing concrete system objects they have been used to represent tools such as those used for drawing packages (e.g. Macdraw) and architectural drafting systems (e.g. Icads; Cornell *et al.*, 1984); as messages for errors (e.g. exclamation marks and flashing objects); as indicators of system states (e.g. arrow for menu selection, curly lines for text insertion). In addition, recent developments in programming languages are beginning to incorporate various types of icons as a way of representing different features of trace histories (e.g. Eisenstadt and Brayshaw, 1988). The underlying idea behind this is that by representing the different parts of the program in a graphical form it can make it easier for the programmer to get an overall gestalt of the program while being able at the same time to detect rapidly specific features and patterns that can be attributed to anomalies in the program. Table 1 provides a summary of the various types.

**Form**

The form of an icon can relate to its function. This can consist of concrete objects, e.g. files, folders; abstract symbols, e.g. arrows, circles, dots, lines, mathematical formulae or a combination of both.

The form can be classified further into the way in which it represents the

**Table 1. Range of functions underlying interface ions**

| Function | Example |
|---|---|
| Labeling | Menu item |
| Indicating | System state |
| Warning | Error message |
| Identifying | File storage |
| Manipulating | Tool for zooming and shrinking |
| Container | Object for placing discarded objects |
| Gestalt pattern (detection) | Structure in programming language |

underlying concept i.e. the referent. This representation can take several forms: resemblance; being an exemplar; being symbolic; or by being arbitrary.

*Resemblance icons*
Resemblance icons depict the underlying referent through an analogous image. A good example is of the road sign for 'rocks falling' (see Figure 2(a)).

*Exemplar icons*
An exemplar icon serves as a typical example for a general class of objects. For example, the knife and fork used in the public information sign to represent 'restaurant services' (see Figure 2(b)). Within the context of public services, this depiction of a very simple image is very powerful as it shows the most salient attributes associated with what one does in a restaurant, i.e. eating.

*Symbolic icon*
Symbolic icons are used to convey an underlying referent that is at a higher level of abstraction than the image itself. For example, the picture of a wine glass with a fracture in it is intended to convey the concept of 'fragility' (see Figure 2(c)). When used in the appropriate context, such as on the side of a box containing fragile goods, the image can be quite effective for what is a highly abstract concept.

*Arbitrary icons*
An arbitrary icon bears no relationship to the referent and hence the association must be learned. The biohazard sign is an example (see Figure 2(d)). In this sense it is a completely arbitrary form of coding similar to characters that have been developed in verbal language where there is no physical or analogous correspondence between the characters used to form words and their intended meaning.

**Icon mapping**
The effectiveness of an icon in relation to its intended meaning also depends on the level of *mapping* between the phsyical form and function (c.f. *articulatory distance*; Hutchins *et al.*, 1986). This implies the extent to which the link between what is being depicted and the underlying referent can be inferred. By referent, the system information is meant (be it an object, operation or message)

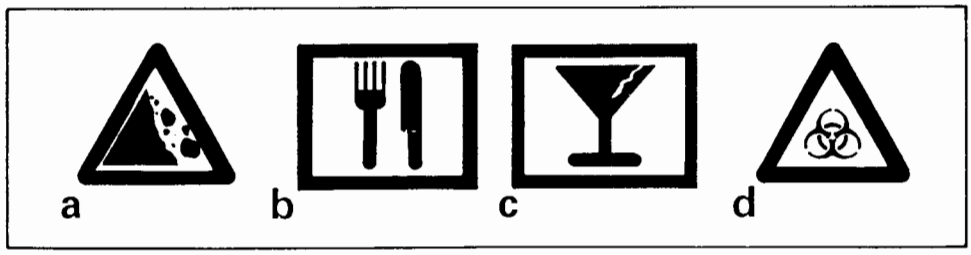


*Figure 2. Different forms that icons can take: (a) resemblance, (b) exemplar, (c) symbolic and (d) arbitrary*

as characterised by the interface metaphor in which it is embedded (e.g. objects and operations included in the office metaphor such as 'files', 'deleting', 'copying', etc.). Icons that are easiest to comprehend are considered to have the most direct mapping while those most difficult to understand are considered to have the least direct mapping (Hemenway, 1982). Direct mappings are characterised as having the most similar structures between form and function. An example is that of the depiction of a file to represent the data object of a file where clearly there is an isomorphic relationship between what is being represented and the form that is used (see Figure 3(a)). Less direct mappings, on the other hand represent their referents by drawing parallels between characteristics of the underlying referent and familiar structures in the icon. For example, the depiction of a pair of scissors to represent 'to cut' (see Figure 3(b)). Here the connection between the form and referent takes the form of the most typical tool used to perform the action of cutting.

Many of the functions underlying icons described in Table 1 are highly abstract. In this sense it is not possible to design icons with direct isomorphic mappings like the data file example. Alternatively, it is necessary to use less direct forms of representation. One method is to use completely arbitrary forms of representation in which there is no connection between the physical representation and the underlying representation. Examples include abstract symbols, geometric figures, nonsense shapes and various forms of lines. The argument behind this line of approach is that arbitrary forms are an effective form of coding because they have no prior associations. Hence, once a person has learned to associate the meaning between the physical form and the underlying referent of a symbol they should have no subsequent difficulty understanding the meaning of it. An obvious example of this type of philosophy is the abstract symbol designed in Figure 2 for biohazards. The problem with icons that are designed in this way, however, is that the arbitrary nature of the mapping between form and referent is not necessarily easy to learn or remember. While this form of representation can be effective for establishing a convention for one or two isolated symbols it is not effective for someone who has to learn a whole set of icons. For example, in a study investigating the meaningfulness of single icons and sets of icons, designed to represent command operations in a word processing environment, it was found that there was a marked contrast between the effectiveness of abstract icons in the two conditions (Rogers, 1988). Specifically, one or two of the abstract symbols, most

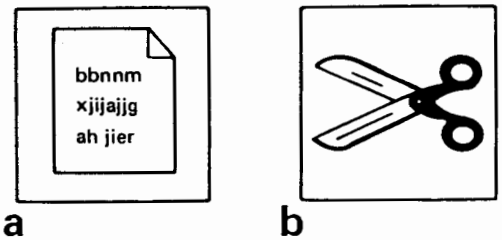


*Figure 3. Examples of different types of icon mappings: (a) file (isomorphic), and (b) to cut (physical analogy)*

notably a cross and an exclamation mark (representing 'to delete a block of text' and 'quit', respectively) were found to be highly meaningful as individual symbols, but when embedded within a set of other abstract symbols, the potency of these symbols considerably diminished. The problem, therefore, is that it is much more difficult to attribute meaning to a whole set of abstract symbols than to just one or two.

In contrast to using arbitrary forms of representation, however, it is possible to design icon mappings that have a certain level of directness. For example, in the same study as above, it was found that icon mappings depicting the operation through the use of concrete objects associated with the underlying action in combination with various abstract symbols were the most effective form of representation. The function of the latter was found to provide an indication of the state of the object. For example, the command operations 'go to the bottom of the text' and 'go to the top of the text' were most effectively conveyed through the depiction of a piece of paper with writing on together with an arrow outside pointing downwards and upwards, respectively (see Figure 4).

Another type of mapping that can be used is one based on analogy in which the similarities between the entities being represented and the iconic form are exploited. These can be either physical or semantic associations. The example of the use of a pair of scissors to represent 'to cut' is a physically based one where the object is used in the action. An example of a semantically based analogy with no physical connection is of a spinning top used in some CAD systems to represent the operation 'go to the top'. Here the connection is based on the metonymy of the word 'top'. In some instances this type of mnemonic chaining can be quite effective especially if the link is bizarre. However, if the link is rather tenuous and not made explicit to the user it can actually be counterproductive. For example, different icon mappings were compared, it was found that those based on concrete analogies were the most difficult to understand, learn and remember for a range of verbal and user performance tasks in a word processing environment (Rogers, 1988). Examples of the icons included a wheelbarrow with bricks in it for the operation 'to move' and a person with their arms in the air representing the operation 'to quit'. (These images were based on the findings of a previous study in which drawings were produced for a set of verbs in a context free situation and subsequently judged to be the most stereotypic, i.e. the most representative of their intended meaning.)

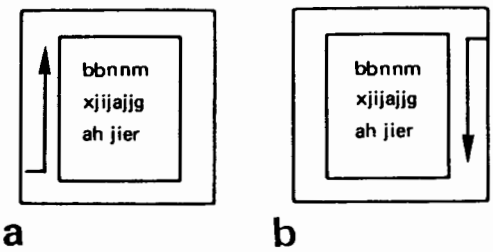


*Figure 4. Example of command operations with shared features: (a) go to top of text, and (b) go to bottom of text (concrete object with abstract symbol)*

## Shared features between icon and command sets

One of the main differences between the traditional use of pictorial symbols as a means of communicating public information, etc. and icons to be used at the interface is that pictorial symbols are often displayed as individual signs whereas icons are generally displayed as a set. Most commonly, icons are used to represent a set of menu items, be they system operations or objects, or as a palette of tools for a given application. As mentioned earlier there is a big difference between using icons in isolation and as a group insofar as an individual icon only needs to be able to represent the underlying referent efficiently whereas a set of icons needs also to discriminate effectively between the various referents.

Computer command sets are usually structured in the sense that they often have something in common with each other. Operations can be performed on similar objects (e.g. 'delete a line', 'insert a line') while the same actions can be operated on different objects (e.g. 'delete a character', 'delete a line'). In terms of command language it has been suggested that appropriate names can be selected that reflect this structure (e.g. Rosenberg, 1983). A 'good' set is considered to be one in which there is a high degree of similarity between the names and the underlying operations. Furthermore, an optimal command name set will *maximise* the similarity between an operation and its name while *minimising* the similarity between different names within the set. Hence a name whose underlying meaning maps closely to the features of the command operation, but which is also distinctive from the other names used within the set is considered to be the easiest to learn and remember.

This type of mapping works well when there are a small set of commands consisting of opposite pairs of commands that are distinct from the rest of the set, e.g. 'create' for open a new file and 'destroy' for deleting a file. The problem arises, however, when there is a lot of overlapping between elements and operations within a larger set of commands. Consider the command 'to delete' in a word processing environment. There are a number of objects for which it can be used for (e.g. character, line, paragraph, page). The question is, given that verbal language is linear, how do we go about selecting names that are semantically similar to the underlying referent while being distinctive from the other names? While there are several synonyms that can be used to mean delete (e.g. destroy, remove, erase) semantic confusion can arise if they are all used within a set. For instance, if we label the operation 'to delete a file' with the name 'delete'; to delete a block with the name 'erase'; 'to delete a word' with the label 'remove' and so on for the rest of the delete operations how can one then remember whether the name 'delete' was associated with file and not, say, block. A problem of semantic confusion, therefore, can occur similar to that experienced with large sets of abstract symbols. Namely, there are no obvious associations between the various forms of 'destroy' and the different elements to be destroyed. In this sense command names *per se* are a very poor means of representation since they are unable to discriminate effectively between the various combinations of same operations with different elements and *vice versa*.

One of the advantages of pictorial communication over verbal language, on

the other hand, is that it is multidimensional, i.e. it has a large number of graphic attributes such as form, shape and size. If used effectively these can be combined in such a way so as to visually map onto the structure of the command set. Hence, for the delete command 'crosses' and concrete representations of the various text objects can be used together to depict the common action of deleting while conveying the distinct types of objects. Figure 5 shows the nature of this relationship.

For graphically-oriented domains (e.g. CAD applications), which often have a large number of objects and operations with shared and distinct attributes, the need to differentiate between them may be critical. For example, when constructing a plan for a building or a circuit board, the designer needs to create objects and alter different parts of the drawing using various types of tools. While it may be possible to construct a limited number of meaningful names for this purpose, it is far more difficult to design a whole set of items that are both meaningful and discriminable. It is much easier to develop a set of meaningful icons that can distinguish between the similarities and differences of the objects, operations and tools by using common and distinct pictorial elements.

The potential of pictorial representation, therefore, is its ability to discriminate between commands that have common features. By combining the different pictorial elements to map onto the underlying command structure a simple form of grammar is essentially being established. The advantage of this form of graphical syntax and semantics is that it can provide powerful visual cues as to the way the domain is structured. The effect of this is that it has the potential of facilitating learning and subsequent memory of the commands (Gittins, 1986). In addition it has been suggested that it can enable the user to develop an enriched mental model of the system (e.g. Rohr and Keppel, 1984).

## Visual memory

One of the most obvious reasons why iconic-based interfaces may be easier to learn and remember is that users do not have to recall command labels to execute an operation. Instead they simply need to recognise and select the appropriate icon from a set (be it a menu, palette or other form of display). The effect of this is to minimise the cognitive load on memory as compared with having to recall a name or abbreviated code. It could equally be argued, however, that the same is true for labels or names that are visually displayed on the screen. This means, therefore, that any advantage arising from the use of icons compared with other forms of display-based interfacing must be a result of the differences between the content of the displayed information. An import-

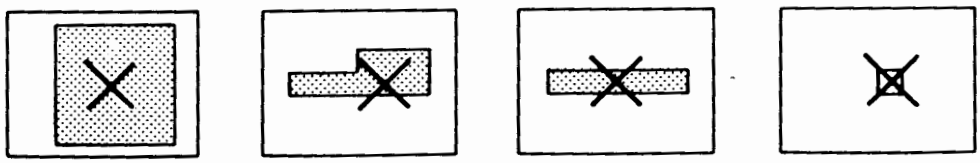

*Figure 5. 'Delete' icons for word, line, paragraph and page*

ant question, therefore, is whether there is any difference between the amount of information the user picks up and remembers from the different types of information displays and, moreover, how this effects the way in which a task is carried out.

Interestingly, in a study comparing user performance for icon-based with name-based menus designed to represent a set of 20 word processing and system operating commands it was found that there were a number of differences between the various performance measures that were taken (Rogers, 1988). The study was designed to evaluate the learnability and subsequent memorability of a number of icon sets with a command name set. Four icon-mappings varying in directness were compared. These were depictions of concrete objects and abstract symbols, concrete objects alone, analogical representations, and abstract symbols alone. Using a between-subjects design, five groups of naive subjects (in the sense that they had no previous experience with a word processor but had some experience with using a typewriter) were first given a training session in which they were taught how to use the word processor. They were then required to perform a number of operations including creating and deleting text, saving and printing a file. On completion of the tasks subjects were then given a memory task in which they had to provide descriptions for each of the icons or command names. The same procedure was then repeated a week later to investigate how much they could remember over time.

Overall, it was found that the most effective form of icon-mapping was that considered to be the most direct, which was the set depicting concrete objects and abstract symbols. This superiority was reflected by the smallest number of total help calls and accesses to the help facility and the highest number of icons correctly defined in the subsequent memory task. A similar pattern of results was found for the name set, but with the exception that significantly more errors were made in the memory task. Moreover, whereas memory for the meaning of all the icon-mappings improved over time it remained the same for the name set. This finding suggests, therefore, that subjects were generally able to perform the tasks equally as well for the icon set as with the command name set. When asked to recall what the names meant subsequent to the task, however, subjects found it very difficult to access this information from memory. On the other hand, subjects in the icon condition had very little difficulty in accessing this information.

This finding suggests, therefore, that there is a difference between the way in which the information from the two forms of representation is being used and stored in memory. In particular, it appears that once subjects had learnt the meaning of the icons they stored this information in a form which they could then readily recall away from the interface. Subjects in the command name set, despite having used the commands successfully to perform various tasks, found it much harder to recall the meanings for the names away from the task. At a theoretical level this finding can be explained in terms of the type of information which was coded. In terms of Paivio's dual coding theory (1981), the reason why the meanings of the icons are likely to be remembered better is that, being pictorial information, they are more likely to be stored in both the visual

and verbal store whereas the command names, being verbal information, are likely to be stored only in the verbal store.

In relation to this, the results can also be viewed in terms of *encoding specificity* (Tulving, 1974), in which the extent to which information can be recalled depends on the availability of the retrieval cues present at learning also being present at recall. Therefore, it may be that another factor that prevented the names from being recalled away from the interface was the way they were coded insofar as they were highly specific to the task. Conversely, the pictorial content of the icons was sufficient for them to enable the subjects to recall their associated meaning away from the context in which they were learnt.

The way in which information may be stored in memory and then accessed also has direct implications for how well tasks can be performed at the interface. As yet, however, it seems very little is known about how information flows between the external display of the interface and the user's internal memory. A recent study by Mayes *et al.* (1988) suggests that, regardless of their level of experience, people using display-based systems have in fact a very poor memory of the content of a screen at different stages of a task. In particular, they found that users were able to recall little of the menu contents when asked to recall what was displayed on a Macwrite screen at certain points of a task operation. To explain this they suggest that what is actually happening when we interact with computers is that our behaviour is guided from moment to moment by what we see on the screen. Hence the necessary information that we require to perform an operation at any given stage of a task is taken from the screen, used and then forgotten. In other words we do not actually learn the content of visual displays in the sense that we might attempt to learn a set of command names.

Visually displaying information that can be manipulated directly at the interface is an effective form of dialogue because it does not require the user to learn information and then access it from memory. Alternatively, the user simply has to recognise the part of the visual display (be it a menu item, scroll bar and so on) required to fulfil that part of the task and then select it. Once that action has been successfully performed, the user can then move onto the next stage of the task and again simply locate another object on the display that matches their next subgoal.

To what extent this form of ongoing 'pickup' of information depends on memory of spatial coding (i.e. where objects are positioned on the screen) or the type of information provided in the display is yet to be answered. The results from the study investigating the usability and memorability of icon-based menus versus name-based menus suggests that the visual content of the information displayed will play an important role.

## Summary

The use of icons at the interface is clearly here to stay. The problem that confronts designers is knowing when best to use them and what form they should take. To ensure that the potential of iconic interfacing is fulfilled, it is suggested that a taxonomy be developed initially to categorise the different

functions that need to be represented at the interface. In so doing it is important to devise a simple form of grammar that reflects the underlying operation structure of the system. Sets of icons can then be designed through the use of common and distinct pictorial elements which reflect the relationships between the various commands and system objects. At a theoretical level it is useful to view the efficacy of iconic interfacing in terms of the cognitive demands made on memory. In particular, it is fruitful to consider the nature of the interplay between the knowledge a user has to learn and subsequently access from internal memory and the external visual information that can be picked up from the various screen displays at the interface to perform a task successfully.

## References


**Cornell, D., Sambura, A. and Gero, J.** (October 1984) 'Icon-driven interfaces for drafting systems' in *Proc. Int. Conf. Computer Graphics '84*, Online Publications, Pinner, UK

**Eisenstadt, M. and Brayshaw, M.** (1988) 'The transparent Prolog machine' *Human Cognition Research Laboratory Technical Report No. 21a* Open University, Milton, Keynes, UK

**Gittins, D.** (1986) 'Icon-based human–computer interaction' *Int. J. Man–Mach. Stud.* **24**, 519–543

**Jervell, H.R. and Olsen, K.A.** (1985) 'Icons in man–machine communications' *Behav. Inf. Tech.* **4**, 249–254

**Hemenway, K.** (1982) 'Psychological issues in the use of icons in command menus' in *Proceedings on Human Factors in Computer Systems* ACM, Gaithersburg, MD, USA

**Hutchins, E.L., Hollan, J.D. and Norman, D.L.** (1986) 'Direct manipulation interfaces', in **Norman, D.A. and Draper, S. (eds)** *User-centered system design* Lawrence Earlbaum Associates, Hillsdale, NJ, USA

**Lansdale, M.** (1988) 'On the memorability of icons in an information retrieval task' *Behav. Inf. Tech.* **7**, 131–151

**Lodding, K.N.** (1983) 'Iconic interfacing' *IEEE Comput. Graph. Appl.* **3**, 11–20

**Mayes, J.T., Draper, S.W., McGregor, A.M. and Oatley, K.** (1988) 'Information flow in a user interface: the effect of experience and context on the recall of MacWrite screens' in *People and Computers IV: Proc. 4th BCS HCISIG Conf.* **Jones, D. and Winder, R. (eds)** Cambridge University Press, UK

**Paivio, A.** (1971) *Imagery and verbal processes* Holt, Reinhart and Winston Inc, NY, USA

**Rogers, Y.** (1988) 'Pictorial representations of abstract concepts in relation to human–computer interaction' *PhD Thesis* University of Wales

**Rohr, G. and Keppel, E.** (1984) 'Iconic Interfaces: where to use and how to construct?' in **Hendrick, H.W. and Brown, O. (eds)** *Human factors in organizational design and management* Elsevier Science Publishers, Amsterdam

**Rosenberg, J.** (1983) 'A feature approach to command names' in **Janda, A. (ed)** *Human factors in computing systems* ACM, Boston, MA, USA

**Smith, D.C., Irby, C., Kimball, R. and Verplank, B.** (1982) 'Designing the STAR user interface' *Byte* **7**, 242–282

**Tulving, E.** (1974) 'Cue-dependent forgetting' *American Scientist* **62**, 74–82