Investment of Attention as an Analytic Approach to Cognitive Dimensions

A.F. Blackwell

Computer Laboratory
University of Cambridge
Alan.Blackwell@cl.cam.ac.uk

T.R.G. Green

Computer Based Learning Unit
University of Leeds
thomas.green@ndirect.co.uk

Introduction

Green's Cognitive Dimensions of Notations framework has attracted sustained attention from researchers investigating programming (Yang et el., 1998; Roast & Siddiqi, 1996; Williams and Buehler, 1998) and other design activities in which the design notation is an essential element of the design process or product (Tweedie, 1995; Buckingham Shum and Hammond, 1994). More recently, the Cognitive Dimensions (CDs) have been applied in investigating the usability of other types of software notations, user interfaces, and "information artefacts" (Green and Blackwell, 1998; Britton and Jones, 1998). The increasing generality of CDs is both encouraging and challenging. It is encouraging in that it affirms the value of psychology of programming as a discipline that provides novel theoretical contributions to more general problems in cognitive ergonomics. It is challenging in that it draws our attention to issues that were less important in earlier formulations of CDs, but are required of general approaches to cognitive ergonomics.

This working paper addresses these two issues. Our main aim is to propose a new level of description of CDs, and this description appears to be useful in both contexts. Earlier formulations of the CDs have concentrated on providing a descriptive vocabulary for usability characteristics at a level recognisable to software designers and language designers. Those formulations have intentionally avoided any reliance on cognitive theories, and might even be described as a reaction against HCI theories motivated by cognitive science. On the other hand, a major problem with CDs has always been that it is difficult to tell whether the dimensions are genuinely orthogonal or complete. With no theoretical standard for measuring their relationship to each other, the only comparisons that have been possible are the identification of the trade-offs that structure the design space for notational systems. We developed these ideas after teaching a short course on the use of CDs as design tools, which made it clear to us that design methodologies required more precise definitions of some of the underlying concepts structuring the dimensions. We hope that this work will therefore broaden the transfer of psychology of programming research results into mainstream HCI.

The approach that we are taking here is to describe some characteristics of notation use that might underlie all the CDs. At the time of writing, the characteristics we have chosen are rather conjectural, but they seem to be reasonable contenders. The first section of the paper introduces them with the aid of some information artefacts that appear rather unlike programming: books and telephones. The second section extends these characteristics to typical questions in the psychology of programming and applied HCI. The third section describes the investigation that we are currently carrying out in order to test these ideas. We hope that some results will be available in time for presentation at the PPIG meeting. The final section includes some preliminary proposals for extending our new model to describe some familiar cognitive dimensions.

Tolstoy and the telephone: Locus of attention in information artefacts

After the first 50 pages or so of War and Peace, most readers start to meet characters whose names are vaguely familiar. Are they new characters, or the same people with different names (patronymics, familiar names, married names)? Perhaps they are are newly appeared, but relatives of people we have already met? After another 50 pages, it becomes clear that it will be hard to keep track of all the characters, not to mention who is married to whom, betrothed to whom, is a child/aunt/grandparent/commanding officer of another character and so on. At this stage, the diligent reader might take a piece of paper and start to draw a family tree, which he or she can keep in the front of the book, extending and consulting as necessary.

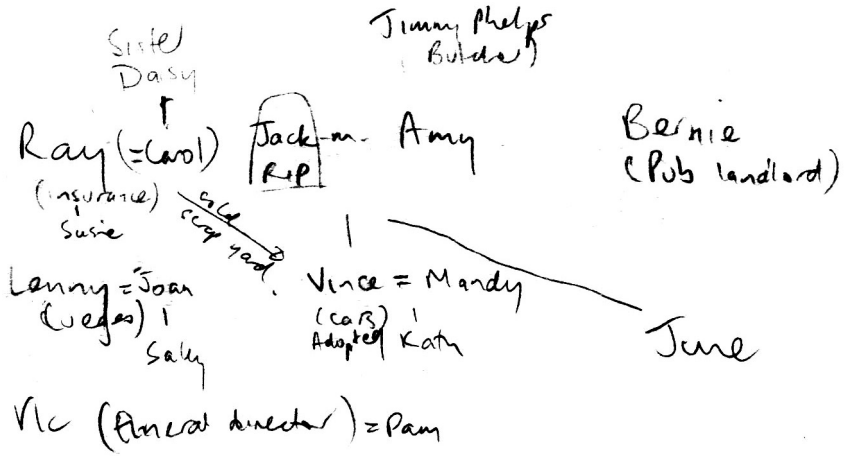

Figure 1 - Family tree found drawn in a novel
(in this case, the novel is Last Orders by Graham Swift).

Another thing. One of us has a very poor memory for phone numbers, and is always worried that some important phone number (the number belonging to the other one, for example) may be inaccessible due to a lapse of memory. When calling his collaborator on a new telephone, he starts dialling, then stops, thinking “Perhaps this telephone can be programmed with quick dial codes - will I often be calling the same number from here in future?” If the answer is yes, he postpones his call, learns how to program the quick-dial codes, stores the required number, then continues working on writing this paper.

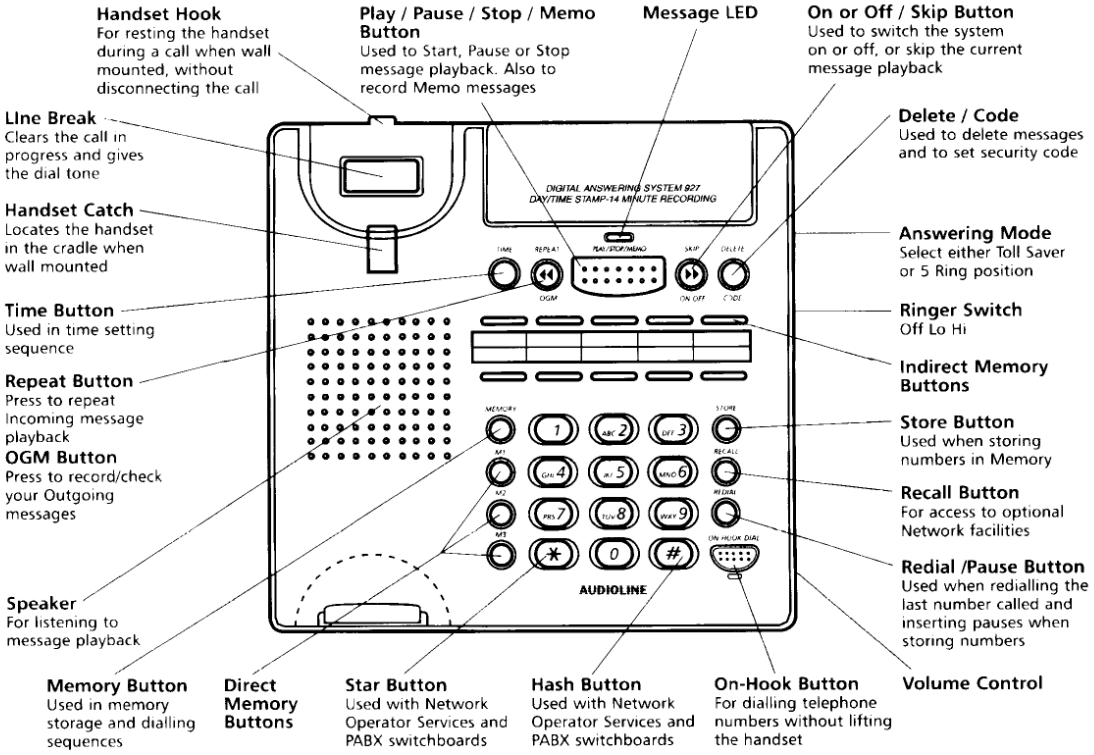

Figure 2 - Instructions that might assist in defining a quick dial code.

Each of these activities can be described in terms of the Cognitive Dimensions of notations. The telephone is *abstraction tolerant* - it provides a means of defining the quick dial abstractions via some abstraction manager (the abstraction manager has its own cognitive dimensions). The telephone also has *hidden dependencies* and *premature commitment* - it is not possible to tell which quick dial codes I have programmed without actually dialling them. The book supports *secondary notation*: I can draw my tree inside its cover. Alternatively, if I slip a piece of paper with the family tree between the pages of the book, it can support *juxtaposability* - I can pull the paper out and place it alongside the page I am reading.

Many programming tasks resemble the Tolstoy and telephone scenarios. A programmer must constantly evaluate the number of times he or she is likely to repeat some action in the future, as well as make reasonable estimates of his or her own mental capacity when dealing with large and complex sets of information. At almost any time, the programmer is faced with many strategic decisions: do I stop now and define a new abstraction, retrieve some hidden information, write something on a piece of paper, reorganise this hierarchy? The actual decision often depends on how severe the distraction will be - some of these actions might require starting a new program, or even moving to a different computer system. Good programming environments provide lots of facilities to allow these little adjustments without leaving the current context of work. Even severe disruption will be tolerated, however, if the future consequences are sufficiently extreme.

Many of the cognitive dimensions can be defined in terms of these two elements: the degree of disruption that they cause to the task and the anticipated benefits of the activities that cause the disruption. We describe disruption in terms of the “locus of attention” of a programmer. What is locus of attention? It is generally possible to describe some attentional focus during programming work (a place on the screen that I am looking at, a key I am pressing, a location of the mouse pointer, a page in my desk diary). This is not inconsistent with cognitive models of attention (Hardcastle 1998). There is some effort or cost entailed in attending to a sequence of foci, and it increases with the length of the sequence. *Repetition viscosity* is annoying because it results in a long sequence of attentional foci. Discontinuities in the attentional focus also have a cost. A flowchart too large to fit on a screen offers poor *visibility*, recognisable in the discontinuities of attentional focus as the user

scrolls the view window to a different part of the chart (locate the scrollbar on the screen, estimate how far to move, decide whether to click on the arrow or drag the bar, use the wheel on the mouse, or the page-down key, or move the cursor down until the screen scrolls automatically).

Invoking an *abstraction manager* to define a new abstraction (defining a macro, programming a quick-dial code) often brings a rather extreme discontinuity in the locus of attention. It's justified, though, because we expect significant advantages in the future. In fact, we wouldn't write programs at all if we didn't think they were going to be useful sometime in the future - the whole thing is just a distraction from the locus of attention of real life (drinking beer, folk dancing and playing music).

We have in the past described the "ironies of abstraction" (Green and Blackwell 1996) - the fact that abstraction is a costly tool, often bringing added labour in the pursuit of labour saved. Abstraction is a very good example of this irony, but the abstraction-related cognitive dimensions are not the only example. *Secondary notation*, for example, is an investment in the future requiring a short-term dislocation of attention (the degree of dislocation depending on how well the environment supports secondary notation). If we further consider the "locus of attention" traversing not only the world but the information behind the eyes of the programmer, as suggested by the results of Carlson, Wenger and Sullivan (1993), we see that cognitive dimensions such as *hard mental operations* also force a dislocation of attention as we use external crutches for working memory limitations.

Our current project is to describe the cognitive dimensions in terms of their associated dislocation in locus of attention, and in terms of the payoff that initiates that dislocation. The rest of this paper analyses a number of situations in which attention is critical to programming and other contexts of interacting with information artefacts.

Investment of attention in programming contexts

Let's consider some further situations that occur during programming in which it is necessary to make an investment of attention and accept some risk with regard to that investment.

- When invoking a Windows API call, I am not sure that I remember the correct order of the parameters. I could go to my bookshelf and look up a reference manual, or I can proceed on the basis of my vague memory, and wait for a compiler error to tell me I was wrong.
- I notice that two variables in my program have very similar names. This will probably result in confusion some time in the future, but changing one of them will distract my attention. I decide to leave them as they are.
- I am about to start writing a function that I vaguely remember having done before. I could engage in some "software reuse" by looking for my previous code, but I can't remember where I put it. Perhaps it will be safer to just start over again - I know it will only take 10 minutes, whereas if I spend 10 minutes searching, it is possible that I still won't have any code by the end.
- I notice that the specification for the Java package I am implementing includes two classes that might be better implemented as a single class. Rather than continue coding, I send an e-mail to the rest of the design team, telling them that I intend to make a change to the specification.
- There is a severe usability problem in the new product release I am testing - in half of the dialogues, the "abort" button is on the left hand side, while the other half have it on the right. I extend the release schedule by two weeks, so that the team can spend a further 10 days making all dialogues consistent. No copies of the product are ever sold.

As we can see, some investment decisions can entail a large amount of attention, expended by a sizeable number of people. Even large investments can be risky, though. Where larger groups of people are involved, the payback on the investment will often be received by someone other than the original programmer. This also happens in areas of the software industry that are less obviously programming tasks. What about the design of a web site? A large team of people can invest a lot of attention in creating a useless web site that nobody ever looks at. Would it be any better if people did

look at it? If 1000 people read through a web site that has no useful information, it is the viewers who have made a risky investment of attention. Perhaps in this situation, it would be better for the programmers (builders) of the site to make it obvious that it is not useful – doing so would save attention overall.

Attention economies

This takes our argument into familiar territory. It is well known that attention is valuable in the commercial world – companies pay to get the attention of web-surfers just as they pay for the attention of viewers watching television commercials. Portante and Tarro (1997) report Richard Lanham's proposal of an attention economy, which recognises that as information is not a particularly scarce resource, we can hardly be said to have an information economy. Lanham suggests that attention is the scarce resource which must be allocated in the developing global economy. This is true in many different fields – the value of attention is simply more explicit on the Web – personal home pages are adorned with web counters, and banner advertisements are priced according to the number of "click-throughs".

From this perspective, it is now clear that software vendors have a valuable resource in their ability to capture attention. The value of a permanent icon on the Windows desktop, or on the navigation bar of a popular Web browser is so great that the assignment of these resources is reported in newspaper business pages. Software users are increasingly accustomed to the idea that they are having information thrust at them, and that their priority must be to accomplish their current task without having their attention distracted. In this context, users even resent features that are part of the application. While preparing this paper, we found ourselves referring to the help system in Microsoft Word, and immediately found our attention distracted by the chap seen in figure 3:

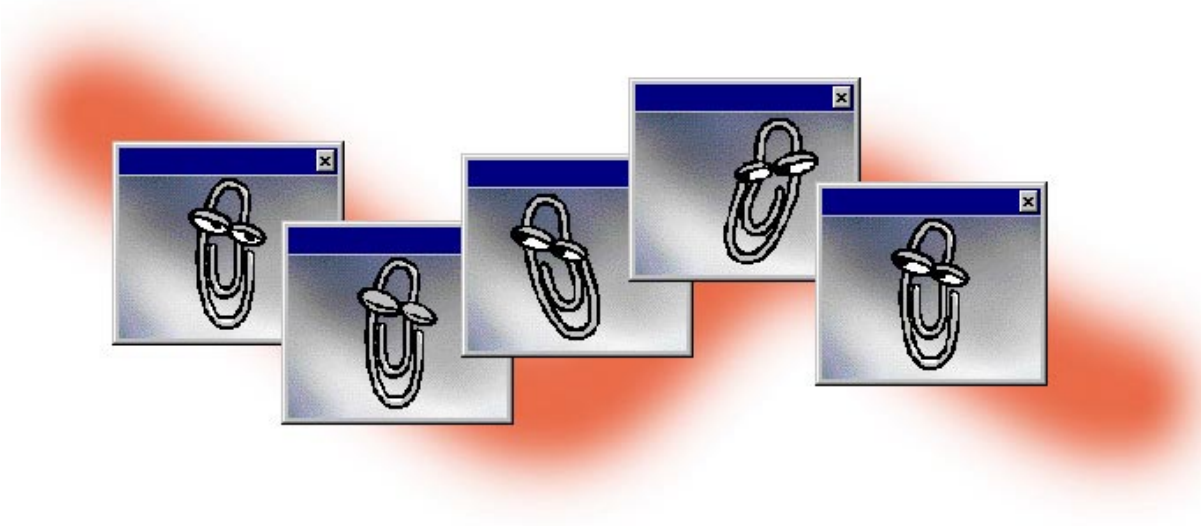

Figure 3 - Attention distractor in Microsoft Word.

The investment of attention caused by this smirking, jiggling character is usually restricted to clicking his "close" button. Someday, however, I will have to invest substantially more attention in finding out how to make him go away for good. In the meantime, this distraction has persuaded me that I can't waste my time investigating the help files of my application – another attention decision. Reading a help facility is an investment of attention very much like going to the shelf to reach down some API documentation. Most likely, I will proceed by trial and error instead.

This behaviour seems to be quite typical of modern software users, in fact. Many users are aware that the application they are using might have a feature to automate the task they are currently engaged in. If they were to invest some attention in finding it, they could save the effort of completing the task. On the other hand, that decision would be a very risky one. Just as when a

programmer chooses not to look for reusable software, it might be better to spend the time doing the task itself rather than looking for an existing solution that might not exist.

Perhaps this is the real cause of the oft-lamented plague of "software bloat". It is usually attributed to the competitive nature of the software industry, in which companies add more and more features to their products so they will appear superior to their competitors. The real situation is that those features do useful things, but that they do not justify the attention required to use them. It is not even necessary to invoke the problem of diverting attention to the on-line help facility. Even if the user knows perfectly well that the feature exists, he or she may choose not to use it because the feature itself demands too much attention. An example in terms of the Cognitive Dimensions is that many users do not create paragraph styles when using Word, because to do so requires the investment of attention in an *abstraction management* sub-device. It is far less risky to simply select each paragraph and change the formatting as required.

Investigating locus of attention through simulation

It seems that many contexts in HCI require these types of attention investment decisions, just as programming requires constant assessment of attention trade-offs. The rest of this paper describes two approaches that we are taking to investigate these phenomena. The first is a simulation of the processes that are involved in choosing how to invest attention when there are varying degrees of risk. The second (in the section after this one) revisits the Cognitive Dimensions in the light of attention investment.

This simulation currently models a specific situation when using a word processor. There are a number of occurrences of the same misspelling in a document, and the user must decide whether to correct the misspellings by reading through the document, correcting each mistake as it is found, or by invoking the search and replace facility. Of course this is a decision that is just as likely to occur in a programming context as in a word processing context.

The simulation models locus of attention in the user interface of a simple word processor (actually the Windows "WordPad"). It reads along marked lines of text, looking for misspellings and correcting them. At any time it can make the decision to stop this activity in order to invoke the search and replace dialogue instead. The aim of the simulation is to model the investment of attention and risk assessment involved in this simple activity. Figure 4 shows an annotated screen shot, giving an impression of the output of the simulation over time.

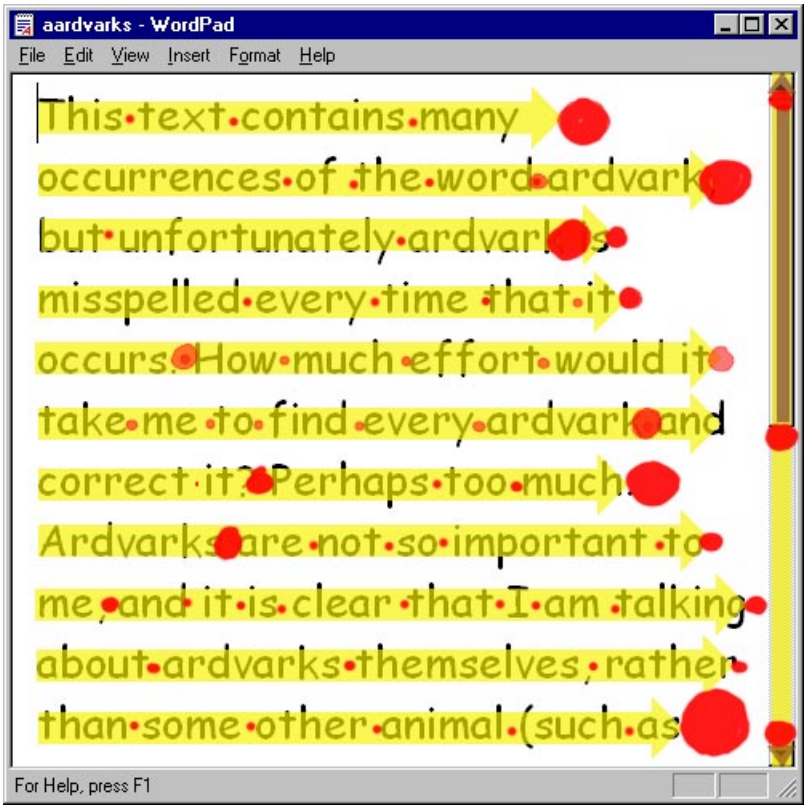

Figure 4 - Simulated attention paths when fixing spelling mistakes

The simulation works with a repertoire of potential actions - interactions with the user interface that involve some locus of attention. An example of a potential action is “read-the-text-on-this-page”. Although it can be described as a single locus of attention, this example can also be decomposed into simpler loci of attention: “read-a-line-of-text”, “find-the-next-line-of-text”, and these can be decomposed in turn (e.g. “read-a-word”, “find-the-next-word”) and so on.

The hierarchical decomposition of attentional loci might appear like a goal/sub-goal tree, but the order of selection and execution of the actions is rather more complex. There are always many possible decompositions of complex attentional sequences. Some of these decompositions might be regarded as distractions (e.g. {read-line, make-cup-of-tea, read-next-line}) while others have some clear value within the overall task context (e.g. {read-line, check-how-many-lines-to-go, read-next-line}).

In order to model the way that an attentional locus is constructed, each potential action has associated with it:

- an estimated potential value
- an estimated cost
- an estimated risk

For each action, there is a set of associated actions that can improve these estimates, as well as a set of associated actions that would be invoked as components of the stated task. At any time, the selection of the next action (and hence the locus of attention) can therefore be influenced by:

1. investing attention by making progress along the current locus
2. reducing the future attentional cost by finding an alternative locus
3. reducing the risk that the investment will be lost

The reduction of cost and risk are seen by the simulation as being valuable in themselves. If a

potential action can greatly reduce risk, it might therefore attract attention even though it is not directly related to the current locus of attention (e.g. save-working-file).

The behaviour of the simulation is expressed in terms of simple control rules: first identify potential progress along the locus of attention associated with the set of potential actions, and associate some value function with that progress. Second, identify actions that could be carried out to improve estimates of cost, and associate added value with that improvement. Third, identify actions that can reduce risk, and associate added value with that reduction.

This assessment process is costly in itself, so there are only a small number of candidate actions that are normally considered in each execution cycle. Consideration of potential actions also consumes some of the available attention, so there is an attention budget allocated to action consideration. If any of this budget remains after the best candidates have been considered, a few other possible actions might be considered at random.

After consideration of the candidates, a certain amount of attentional resource will be allocated to the winning candidate. There is a cost associated with changing the current locus of attention, so once a candidate locus becomes active, it is quite likely to remain active until it reaches some discontinuity (e.g. reading to the end of a line). The winning candidate may however allocate some of its budget to further consideration of risks and potential actions, as well as to following the current attentional locus. Small discontinuities may therefore result in large changes in attentional locus, if some risk estimate has been accumulating while following the locus (e.g. it's getting darker, and I'm not going to be able to read any more of this page - I have to get my glasses).

The assessments of cost, progress and risk are all carried out on the basis of maintaining estimates that have associated confidence factors. This means that the overall behaviour of the simulation is not deterministic. The probability functions used in comparing different estimates reflect the biases described in Kahneman and Tversky's (1979) Prospect Theory: there is a bias against risks with severe consequences, even when those risks are relatively small. There is a similar (less extreme) bias in favour of large gains.

Describing notational activities in attentional terms

The best known publications describing the CDs framework (notably Green & Petre 1996) have emphasised that CDs are a vocabulary that can be used by designers when discussing the needs of users. Green and Blackwell (1998) took an alternative approach by introducing their tutorial with a discussion of the activities that users carry out when using information artefacts: "Designing a garden is not like designing a bridge. Garden design is very aesthetic, and part of the job is exploring the options, changing one's mind, feeling one's way to a new vision. Bridge design is less inspirational and very much more safety-critical. These are different types of user activity, and they are likely to be supported by different kinds of software or drafting tools." (Green and Blackwell 1998, p. 9)

This observation was used to clarify the diverse criteria that might be relevant in designing an information artefact with the guidance of CDs: "It would be nonsense to claim that all information artefacts should satisfy some single set of criteria. We shall distinguish four main classes of activity, and as we explain the framework we shall develop a preferred profile for each type of activity. The four types we distinguish are: incrementation (adding a new card to a cardfile; adding a formula to a spreadsheet), transcription (copying book details to an index card; converting a formula into spreadsheet terms), modification (changing the index terms in a library catalogue; changing the layout of a spreadsheet; modifying the spreadsheet to compute a different problem) and exploratory design (typographic design; sketching; programming on the fly or 'hacking'). Each activity is best supported by its own profile of CDs which will be gathered together as the discussion proceeds."

In the context of the argument presented in this paper, we observe that these different types of activity represent very different loci of attention, as described in the following sections. These loci

integrate two information contexts - there is generally a source of information to which the user must attend, as well as a destination or target that is being modified.

Transcription

When transcribing information from one context to another, the attention of a user must alternate between the source and target, with the period of alternation determined by working memory capacity.

There is an attentional cost associated with each context shift, so the user will tend to increase the period as far as possible.

As the period increases, the perceived risk of error increases. This may lead to a trade-off decision (proof-read afterwards, or use a sub-device (e.g. multiple totals) to check accuracy), or to a satisficing solution - accept a certain level of error.

Incrementation

The activity of incrementing the content of a notation is different from transcription in that:

- The amount of material to be added to the target may all fit in working memory, so no attentional shift back to the source is necessary.
- There is a significant attentional demand in finding the location in the target where actions are required.

There is a risk that the incrementation will lead to further modification. This risk, and the attentional cost required in assessing it, are influenced by hidden dependencies and visibility. If the modification is substantial, this becomes a modification activity, and some prior assessment of risk is required before starting on it.

Multiple strategies are available for finding the location for incrementation - these are discussed under “search” below.

Modification

Modifying an information structure is different from incrementation in that:

- New material may not necessarily be added at all.
- Neither the existing nor the additional information need be explicitly represented.

Sub-devices are more likely to be involved in structure modification. Attentional requirements of the sub-device (especially learning to use the sub-device) must therefore be assessed against the risk of not making the modification at all, or attempting an alternative approach not using the sub-device (e.g. attempting to change an interface name in multiple modules of a large system without learning to use the cross-reference tool).

Exploratory Design

Exploring a design space through experimentation with a notational system is different from modification in that:

- The source material is not available for inspection.
- The attentional cycle alternates between *parsing* and *gnisrap* (Green, Bellamy & Parker 1987) (or sketching and inspecting (Fish & Scrivener 1990), each of which has its own attentional demands and risks associated.

Sub-devices may be available for explicit representation of design goals. The decision to use them depends on the perceived risk of forgetting important goals or dependencies.

New abstractions are often discovered. Risk assessment must establish the generality and expected

utility of the abstractions, and compare those to the attentional cost of abstraction management.

A further activity: Search

Each of the above activities can turn into the next one in the series, if a certain threshold of risks versus attentional costs is crossed. Similarly, many of these activities can degenerate into the previous one in the series if cost/risk is too high. Exploratory design may well be the final step in the series, but there may be an earlier activity that has not been described in our earlier publication, but should be placed before transcription.

Search for information or for context within a notation entails a similar combination of attentional costs and risks. Depending on what the purpose of the search is, it may quickly become a transcription activity, or a component of a transcription activity, but search can occur without transcription. (For example, checking to see whether a certain piece of information is present - when someone sends me a letter, and the envelope has a return address on it, I check to make sure that the address is in my address book before discarding the envelope).

A more familiar example

The discussion at this point has strayed quite a long way from the usual descriptions of Cognitive Dimensions. Are we still talking about the same kind of things? We believe that consideration of attention locus and attention investment has clarified the way in which different activities interact with different CDs. One of the products has been the discovery of activities that we should have described earlier (such as search). We hope that a further product might be the discovery of new CDs, or of relationships between those that have already been described. Rather than speculate too much further along those lines, however, we think that it will be useful to conclude with an example that bears more resemblance to other discussions of CDs.

Consider the case of Sue, a programmer who is writing code in Java, and is expected to document her design using a popular object-oriented design notation. The editor that she uses to draw her diagrams only produces pictures - it does not generate code. She has an almost complete design, and starts using it as a reference while coding. This is pretty much a transcription activity - copying information from the diagram into her program, with attentional requirements as described above. It only becomes difficult when there is some aspect of the design that has no obvious equivalent in Java, in which case she must attend to a textbook, a previous program that addressed the same problem, her knowledge of other solutions and so on. The cognitive dimension that is implicit in these intentional costs is closeness of mapping.

It turns out that there are two elements used in the diagrams that appear very similar to each other at first glance: some arrows can be drawn in red, which has a special meaning. It’s not so bad on the screen, but Sue is working from a hardcopy in which all the lines look the same. It’s a clear case of the cognitive dimension of error-proneness. Sue is well aware of this, and it results in additional attentional cost. Rather than risk a mistake that she often makes, whenever she sees one of these arrows she has to bring up the diagram editor on her screen to check what colour it really was. (It would have reduced the attentional penalty if the diagram was always visible on the screen, right next to the code - the cognitive dimension of juxtaposability).

Sue occasionally works on pieces of code where the arrows span multiple pages in the hardcopy she is working from. Rather than waste attention flicking backwards and forwards through the document (and risking losing her place), she leaves the line of code half finished while working down to the end of the page. The cognitive dimension that caused the problem is visibility. Leaving the line half-finished allows for a future incrementation task. This has a further risk, of course - if she forgets the pending change until it is picked up as a compiler error, it will take a really long time to find the place in the design. The reason it takes so much attention is that there are no references from the code back to the diagrams - the cognitive dimension of hidden dependencies.

Of course, Sue could anticipate that it will be necessary to relate the diagrams to the code, and invest for the future by inserting comments in the code saying which page in the document this code is from - exploiting the opportunities for investment that are supported by the cognitive dimension of secondary notation. If she does that, though, and then the number of pages of diagrams changes, it would take a huge amount of effort to update all these comments. The additional risk, and potential attentional cost, are predicted by the cognitive dimension of knock-on viscosity.

This speculative example could continue at length - we are still working on a more comprehensive list of the potential relationships between cognitive dimensions and activities that can be described in attentional terms. It seems clear, however, that investment of attention does provide a means by which it is possible to compare the trade-offs between dimensions that are central to the CDs framework. The users of information artefacts are constantly acting according to the perceived costs and risks of investing their attention, and it is this familiarity that makes CDs so easily recognisable to students of programming.

References

Britton, C. and Jones, S. (1998). *The untrained eye: how languages for software specification support understanding in readers who are new to them*. Unpub MS, Dept of Computer Science, University of Hertfordshire, Hatfeild, Herts AL10 9AB, UK

Buckingham Shum, S. and Hammond, N. (1994). Argumentation-based design rationale: what use at what cost? *International Journal of Human-Computer Studies* 40 (4), 603-652. http://kmi.open.ac.uk/~simonb/DR.html

Carlson, R.A., Wenger, J.L. & Sullivan, M.A. (1993). Coordinating information from perception and working memory. *Journal of Experimental Psychology: Human Perception and Performance*, 19(3), 531-548.

Fish, J. & Scrivener, S. (1990). Amplifying the mind's eye: sketching and visual cognition. *Leonardo*, 23(1), 117-126.

Green, T.R.G. & Blackwell, A.F. (1996). Ironies of Abstraction. In *Proceedings 3rd International Conference on Thinking*. British Psychological Society.

Green, T.R.G. & Blackwell, A. F. (1998). *Cognitive Dimensions of information artefacts: a tutorial.* Version 1.2, October 1998. (An earlier version was presented as a tutorial at HCI'98 under the title *Cognitive Dimensions of notations and other Information Artefacts*.) This document can be downloaded via links at http://www.ndirect.co.uk/~thomas.green/workStuff/Papers/

Green, T.R.G. & Petre, M. (1996). Usability analysis of visual programming environments: a "cognitive dimensions" approach. *Journal of Visual Languages and Computing*, 7, pp. 131-174. gzip (180K): ftp://ftp.mrc-apu.cam.ac.uk/pub/personal/thomas.green/VPEusability.ps.gz uncompressed (2.2 Mb): ftp://ftp.mrc-apu.cam.ac.uk/pub/personal/tg/VPEusability.ps

Green, T.R.G., Bellamy, R.K.E. & Parker, J.M. (1987). Parsing and gnisrap: a model of device use. In G.M. Olson, S. Sheppard & E. Soloway (Eds.), *Empirical Studies of Programmers: Second Workshop*. Norwood, NJ: Ablex, pp. 132-146.

Hardcastle, V.G. (1988). The puzzle of attention, the importance of metaphors. *Philosophical Psychology*, 11(3), 331-351.

Kahneman, D. & Tversky, A. (1979). Prospect theory: an analysis of decision under risk. *Econometrica* 47(2), 263-291.

Portante, T. & Tarro, R. (1997). Paying attention. *Wired* 5.09, 114-116.

Roast, C. R. and Siddiqi, J. I. (1996). The formal examination of cognitive dimensions. In A. Blandford and H. Thimbleby, (Eds.), *HCI96 Industry Day & Adjunct Proceedings*, pages 150-156, 1996.

Tweedie, L. (1995). Interactive visualisation artefacts: how can abstractions inform design? In Kirby, M. A. R., Dix, A. J. and Finlay, J. E. (Eds.), P*eople and Computers X*, Proc. HCI 95 Conference. Cambridge University Press.

Williams, M. G. and Buehler, J. N. (1998). Comparison of textual and visual languages via task modeling. Unpublished MS, Computer Science Department, University of Massachusetts Lowell, Lowell, MA 01854, USA; *Int. J. Human-Computer Studies* (forthcoming)

Yang, S., Burnett, M. M., DeKoven, E. and Zloof, M. (1998). Representation design benchmarks: a design-time aid for VPL navigable static representations. *Journal of Visual Languages and Computing*, 8 (5/6), 563-599.