A pattern language for the design of diagrams

Contribution to graduate textbook edited by Clive Richards, provisional title *Elements of Diagramming*.

Alan F Blackwell

Abstract

This chapter presents a critical framework with which diagram designers can anticipate the kinds of experiences that users might have with a new type of diagram. This framework is formulated and presented as a *pattern language*, building on an analogy to the use of that term in architectural design. The structure of the framework is derived from Green's Cognitive Dimensions of Notations, although broadened to give more direct consideration to the social and creative contexts in which different types of diagrammatic representation might be found. The intended application is for use as a set of heuristics or discussion tools, drawing attention to the opportunities and trade-offs that are always involved in user experience design.

Introduction: living with diagrams

It is easy to think of professions that are closely identified with the diagrams they use. Electronic circuit designers are associated with schematics; economists and chief executives with charts plotting growth or profit; structural engineers with blueprints of girders and assemblies; and orchestra conductors with musical scores. Many people spend their daily lives contemplating and manipulating a particular kind of diagram, and their expertise or career advancement may be determined by their fluency in reading or constructing that kind of diagram. The question posed by this chapter is how we take into account the needs of diagram users, from the perspective of what it is like to 'live' in these diagrammatic worlds. Even for those not wholly devoted to a particular kind of diagram, many will visit those worlds – not always comfortably – the world of the flowchart, furniture assembly instruction sheet, or spread sheet. We can think of convenience, comfort or usability in terms of an experience: the experience of visiting, or living in, these different diagrammatic places.

It has been more conventional to describe a species of diagram as a 'language', rather than a place, and indeed that approach is adopted elsewhere in this book. But taking this alternative 'place' perspective allows us to draw on some important intuitions from architecture and town planning. Every visitor to a building or city, and every resident, has his or her own individual experience. The architect who designs such places does not predetermine what that experience will be, but enables or encourages different kinds of experience. When we as diagrammatic 'architects' design a new kind of diagram, we can think of our work as analogous to designing a city. Depending on the design decisions that we make, certain kinds of experience will be more likely, and others less likely.

The pattern language concept

The rather anthropological notion that architects should work from analysis of local experiences in a building or city is associated with Christopher Alexander, an architectural philosopher who proposed the idea of an architectural *pattern language* (Alexander 1977). His books developing this concept work by identifying specific kinds of experience that people have in buildings, or in cities – for example, the experience of having light on two sides of a room, or seeing a landmark at the end of an avenue. Not all of these can be achieved in every design – and indeed, some may not always be desirable. But an expert architect will be aware of the many patterns that might arise in a particular design, and will create new master plans or adjustments in the light of these. Alexander's work suggested that younger architects might also be trained to recognise patterns, and that even clients or planners might consider and discuss them when commissioning architectural work.

So the analogy to diagrams is that the designer of a new kind of diagram is like an architect, creating a structure within which people will have different experiences. Those experiences will differ, depending on whether the diagram user is simply reading a diagram produced by someone else, modifying it, or using the language defined for a new kind of diagram to create one themselves. But their experiences can be anticipated in terms of a pattern language of diagram usage. New kinds of diagram could be more or less fit for purpose, depending on how well the diagram designer has understood and applied patterns that are relevant to the activities of the intended users.

As an example, consider the pattern 'You can change your mind easily'. Some kinds of diagram support this quite naturally, while others rather prevent it. To see how this affects users, consider the early days of the flow chart. It is easy enough to read a paper flow chart – just follow the steps! But it's not so easy to create one, especially if you change your mind. If you discover that you have to insert a new step, then the boxes next to it must be moved around to make space, and arrows must be erased and reconnected to pass through the new step. When flowcharts were drawn on paper with templates, a change was terribly time consuming[1]. Computer tools make it much easier, of course – perhaps as simple as selecting and dragging all the boxes below the new one – copy and paste, snapping the arrowheads onto new connection points.

As we see from this example, in the digital era, it is not only the visual specification for a new species of diagram that determines the user experience, but also the tools available to manipulate and view it. These tools may be called 'browsers', 'diagram editors', 'visual languages', or just 'graphical user interfaces'. The last of these – the GUI – is familiar to all software professionals. However, very few software professionals are familiar with the principles of diagrammatics that determine whether a GUI will be usable. Doing so requires designers who understand the patterns of user experience that are inherent in these diagrammatic worlds. And the designer of digital diagrams needs to understand not only the visual principles that result in patterns of reader experience, but also the interaction principles underlying the design of tools. 'You can change your mind easily' is one such

[1] Improvements are possible, of course – see (Richards & Johnson 1980).

pattern, characteristic only of particular kinds of diagrams used with particular kinds of tool. The rest of this chapter explores this and other patterns in more detail.

Applying patterns to information structure

The key principle underlying these patterns is that the diagram represents some kind of *information structure*, constructed from various kinds of components and relationships between them. In a flowchart, the components are the individual steps, and the relationships define the order in which those steps must be carried out (including any choices or repetition). The flowchart diagram represents this underlying structure, with boxes for the steps, arrows for the order, and diamonds for the choices. Software designers and computer scientists are familiar with the kinds of algorithm that interpret such structures inside a computer. But the task of the diagram designer is to specify the correspondence between the *abstract* structure, and the *visual* attributes of the diagram that the user sees and interacts with.

The necessary correspondence between abstract structure and visual appearance can be established using shapes, colours, regions of the plane, directions and so on – as discussed by Richards and Engelhardt in Chapter 3 of this book. However, the experience of *using* the diagram can be greatly changed if the tools are aware, not only of the colours, shapes and positions, but also the ways that they are related to each other in accordance with the underlying information structure. In the case of the flowchart, a flowchart editing application should know that the connectivity of the arrows is more important than their precise position on the page or screen. As a result, the software can be designed so that if a box is moved, the arrows will stay connected and follow it. If this did not happen, they would have to be corrected individually and laboriously – resulting in a system where you *cannot* change your mind so easily.

A flowchart drawn on paper is laborious to create, but straightforward to read. To the *reader*, it does not matter whether or not 'you can change your mind easily', because using a flowchart only involves following instructions, not changing the plan. On the other hand, to the *author* of a flowchart, the experience of being able to change your mind easily makes a very significant difference to the process of creation. It is for this reason that digital tools for flowchart editing are so popular with diagram authors.

The general principle is that different patterns of user experience may be relevant to diagram readers as contrasted with diagram authors. The same kind of general principle applies in architecture, where certain kinds of experience are relevant to certain kinds of usage. A wide aisle through the middle of an open space is appropriate to a warehouse, but less appropriate to an elegant restaurant. The design of a diagrammatic system – the visual notation and the tools to manipulate it – involves trade-offs between the needs of different users. These kinds of user can be more precisely defined in terms of the activities they engage in. There are several degrees of complexity in diagram authorship, and several degrees of complexity in readership. Complexity of activity is related, of course, to the complexity of the information structure that the diagram represents.

The concepts underlying this introduction, and many aspects of the pattern language description to follow, are derived from the work of Thomas Green and his collaborators in developing the Cognitive Dimensions of Notations framework. Further details of the original framework, including citations and attribution of specific aspects, are provided in an appendix to the chapter.

The pattern language

The remainder of this chapter builds on these general principles, to describe a pattern language of user experiences when interacting with diagrams. As is conventional with pattern languages, each pattern is described separately, with a reference number, and a description of its characteristics. The example used in the introduction, 'you can change your mind easily', appears below as number SE2.

The patterns are grouped into sections, although this grouping is not fundamental to their use. For a given design purpose, any combination of patterns may be relevant. They are, however, related to each other in a variety of ways – these relationships are explained where they occur.

The first three groups (Interpretation activities, Construction activities and Social activities) describe patterns in the way that different people want to *use* diagrams. Each of these has a different profile with regard to the experiences that are either unlikely to occur, or may be especially valued, for users carrying out that kind of activity.

Perhaps the most important thing to remember is that just as in architecture, a single design cannot be all things to all men. Although each of these types of experience sounds as though it would be a good thing, there are trade-offs between them, and no single design should attempt to support all possible kinds of good experience. These patterns of experience should never be regarded as a requirements checklist or universal evaluation criteria. The diagram designer must choose which kinds of user activity are most important for his or her design, and prioritise the different patterns of user experience accordingly.

Interpretation activities: reading information structures

This first group of patterns describes the activities of diagram *readers*. Many forms of diagram are never modified by their users – especially if they are printed on a static medium such as paper. It may still be necessary to consider how such diagrams are originally created (for the sake of the diagram authors), but the main design priority is to ensure that they will meet the need of readers.

IA1: Search

For example: Use a train schedule to find the departure time of my train

The user does not need to absorb all the information in the diagram, or even understand it. They might only need a single piece of information, in which case the remainder of the visual structure simply guides them toward finding it.

Relevant experience patterns include VE1, VE4, SE3, TE4.

IA2: Comparison

For example: Does the train leave later than my bus arrives at the train station?

The user needs to compare different parts or aspects of the information structure in order to make a decision or combine aspects of what they see. Carrying out the comparison is also likely to involve searching for relevant parts, so aspects of pattern IA1 will also be relevant.

Relevant experience patterns include VE5, SE4, ME4, TE2

IA3: Sense-making

For example: What is the best route, and time of day, to make a new journey?

The user is trying to learn about a new situation, or integrate data of a kind they haven't seen before. This involves understanding the overall structure, how parts are related to each other, and which are most important. Comparing different parts and aspects of the structure will be an important aspect of sense-making, so aspects of pattern 1A2 will also be relevant.

Relevant experience patterns include VE2, VE3, SE1, ME1, ME3, TE3, TE5

Construction activities: building information structure

This group of patterns describes different ways of *manipulating* diagrams – users who create or modify a particular kind of diagram using the defined tools and conventions. Many of the most powerful species of diagram are those that allow their users to create new information structures – computer programs, timetables, musical scores, engineering designs and others. Often these benefit from specialised computer software – the editors and GUIs described in the introduction to this chapter. In designing species of diagram to support this kind of user, we are really providing them with new design notations, in which to do their *own* design work more effectively.

CA1: Incrementation

For example: Write the length of today's journey in my diary

The information structure is already in place, and the user is adding another piece of information to it. This may involve finding where the new piece is to be added, in which case aspects of pattern IA1 will be relevant.

Relevant experience patterns include IE1, PE6

CA2: Transcription

For example: Plot the journey times from my diary as a line graph ordered by date

A new diagram is being created, but it follows the same structure as another one that already exists. The user doesn't need to think hard about how to create the new structure, but simply transfers information across into a new form. There will be some similarity to pattern CA1, but repeated actions are likely to reduce the element of search.

Relevant experience patterns include ME2, IE2, IE3, IE5, PE2, PE5

CA3: Modification

For example: Redraw the data as a pie chart comparing how often my journeys lasted, 1, 2 or 3 hours

The structure is being changed, which involves reorganising the information. Often diagram authors need to make relatively small changes to the structure, rather than creating a whole new diagram, as in the earlier examples of adding a new step to a flowchart. Modifications may well be mentally challenging, if they involve thinking about a problem in a new way – this could even involve extending the diagrammatic language itself.

Relevant experience patterns include SE2, ME5, IE4, TE1, PE1, CE1

CA4: Exploratory design

For example: Create a visual guide for an information leaflet advising other people when and how to travel

A completely new structure is being created, and the user does not know in advance what that structure is going to look like. The user (in the travel guide example, the leaflet designer) has a design problem that is theirs and is drawing the diagram in the process of solving that problem. All the requirements of pattern CA3 will also be relevant here (because the structure is modified during exploration), but the user is more likely to carry a lot of information (requirements and alternatives) in their head, and also to be more open to alternative interpretations.

Relevant experience patterns include TE5, PE3, PE4, CE2, CE3, CE4

Social activities: sharing information structure

This group of patterns describes the activities of people who use diagrams in collaborative contexts. Diagrams are often thought of as a personal thinking aid – something that users read like an instruction manual, reference chart or textbook, or perhaps draw in private when doing intellectual work. But there are also social situations where we share information with others using diagrams. These include data for business meetings and political debates, but can also include creative and playful situations. Of course, inventing a new kind of diagram need not involve computers – sometimes a whiteboard, an illustrated book or an oil painting can help people think together in new ways. Note that the set of activities listed below may not be complete!

SA1: Illustrate a story

For example: Tell someone about a journey you made, following the stops on a route map

The diagram helps the user to communicate with an audience, giving an external aid to help follow the story. To satisfy this purpose, visibility and clarity are high priorities, but it is also necessary that the user should be able to make reference to specific parts where appropriate. In artistic performance contexts, ambiguity and richness may be valued as much as clarity.

Relevant experience patterns include VE2, VE4, IE6, TE1, CE3

SA2: Organise a discussion

For example: Decide with your friends what time to meet up and travel to a football game

Collaboration around a shared representation is very different to following a narrative. There are multiple users, whose work needs to be co-ordinated, and who will develop a shared understanding of the information structure. The requirements of pattern SA1 are still relevant, but it is even more important to allow alternatives to be tried out as sketches.

Relevant experience patterns include ME5, IE2, TE2, PE3, PE4, CE4

SA3: Persuade an audience

For example: Create a new version of the train timetable that highlights the time wasted by business travellers waiting for connections

Persuasion involves the creation of a visual rhetoric, which invites the audience to develop their own understanding, but also emphasises particular characteristics of the information structure that might influence their deliberations. This may involve integrating aspects of patterns IA3 and SA1.

Relevant experience patterns include VE3, SE4, ME2, ME6, IE5, TE3, TE5

Patterns of experience in use

The three groups of patterns presented so far describe the different kinds of activities that users are engaged in when they use diagrams. Each of these kinds of activity results in a different profile of experiences that users are likely to value. The remainder of this chapter describes the patterns of experience that make up those profiles. They are described in a way that invites you (the reader) to imagine what it is like to have these experiences.

Experiences of visibility

This group of patterns is not so much about how you interpret or manipulate the information structure in a diagram, but at a rather more basic level, whether you can see it at all.

VE1: The information you need is visible

Complex information structures often have to be chopped up into chunks that fit on the page of a book, on a screen, or even a whole wall. Once this is done, the rest of the structure that didn't fit is invisible. Sometimes, important aspects of the user's problem don't get included in the diagram at all. There is a close link between this pattern and VE4 – if the layout is concise, then more of it will fit onto the page.

VE2: The overall story is clear

People often say they prefer diagrams to text because they get a kind of 'gestalt' view of the whole information structure – you can stand back and look at the overall configuration, and get a good idea of the whole story. Of course, it needs to be visible for this to work (patterns VE1 and SE1), but sometimes it is possible to leave out some of the detail in order to improve this overall understanding (pattern VE5).

VE3: Important parts draw your attention

This seems like a good idea – we've all seen diagrams with far too much information, with the result that you can't figure out what you are supposed to be looking at. The problem is, of course, that different parts are important to different people. What's more, different parts will be important in different kinds of activity. This pattern is therefore linked to ME2, IE6 and CE2.

VE4: The visual layout is concise

Nobody wants to be verbose, and it seems a shame to use up a lot of valuable 'real estate' (whether screen space, paper, or in the case of architecture, actual real estate). As already noted, if you have a concise layout, more of the information structure will be visible at any one time. However, concision has drawbacks. It might be achieved by showing fewer relationships (SE1), and smaller parts mean less user freedom (ME5, IE3) and more chance of accidents (IE4). Finally, concise notations are often abstract, with less chance that users will find them familiar (ME1, TE4).

VE5: You can see detail in context

Dealing with the constraints of visibility and concision often leads to a user interface where users choose a subset of detailed information that must somehow be understood in relation to the rest of the structure (VE1, VE2, VE3 and VE4). Many interactive diagrams allow the user to select which details they want to see – for example by panning, zooming, or operating scroll, selection and focus controls. In all these cases, the user can work more easily when the diagram provides clues on how the current view relates to everything else. Unfortunately, despite evolving standards for sets of gestures and widgets, it is not always easy to work out how to combine this pattern with IE1, IE2 and IE3.

Experiences of Structure

Things get much trickier once we deal with the relations between parts in the information structure. Many of those relations are in the mind of the user, or are derived from facts in the outside world, with only partial connection to the graphical elements of the diagram. The relations become particularly salient when things are changing. There will probably be some logic influencing multiple aspects that all seem to change at the same time (perhaps a new government policy, or whether the user happens to be in a bad mood). But this logic might be completely different to the logic of the diagram syntax, or the operation of the editor tools. As a result, the experience patterns in this group can be rather independent from the intentions of the original diagram designer.

SE1: You can see relationships between parts

The most obvious kinds of visual relation are elements located within the same bounded region, or with lines drawn between them. There are many other ways of establishing relations, of course. Are there multiple marks with the same colour, orientation, or size? Are there many verbal or numeric labels, some of which happen to be the same? In these cases, the user might have to search in order to find relationships (IA1 and IA2). If the layout is not concise (VE4) then there may be a simple problem of visibility (VE1, VE5). But it isn't possible to include every possible relationship, because the overall message would be lost (VE2, VE3). And every visible relationship introduces another element that may have to be changed if you change your mind (SE2).

SE2: You can change your mind easily

One of the biggest drawbacks of diagrams is the difficulty of changing them – that is why this pattern was used as an example in the introduction. If you need to move a lot of the boxes around in a flowchart in order to make one simple change, the experience can feel like wading through treacle. Thomas Green described this as "Viscosity – a sticky problem for HCI", when he initially proposed the *Cognitive Dimensions of Notations* framework that underlies this chapter (see the appendix for more details). In comparison, making changes to a piece of text is quite easy – just cut and paste the words where you want them. The reason it is so much easier for text is that unlike a flowchart, there are no visible connections following the words around. But as a result, it's also much harder to see the relationship between the ideas in a text at a glance (SE1). The only way to understand a paragraph is to read it. There aren't any lines or boxes showing which words are important in the sentence structure, or how they are related to each other.

SE3: There are routes from a thing you know to something you don't

'Navigation' is an important concern for designers of digital information systems. The same is true of complex diagrams. When the user is looking for something, they will have some kind of starting idea, that might help lead them there. This could be a matter of following the relationships between the parts (SE1), but it could also involve some kind of index, search function, or visual pop-out such as highlighting, animation or visual similarity (ME3). Easy navigation can reduce the need for the user to rely on working memory (TE1). And users often return to look at the same part of a diagram time after time. In this case, it helps if the part hasn't moved (IE3), or if you can add some kind of mark as a cue to yourself showing where to look (ME5).

SE4: You can compare or contrast different parts

Comparing different parts of the information structure is easier if they are both visible (VE1, VE5), but it is also necessary to see the respects in which they are different (ME3, ME4). Sometimes, close similarities in local appearance might mean that the distinctions jump out in some way (TE2).

Experiences of Meaning

This group of patterns introduces aspects of meaning that Richards and Engelhardt have dealt with far more thoroughly in Chapter 3 of this book. The words 'look' and 'tell' in the following patterns can be regarded as informal references to what Richards and Engelhardt describe more rigorously as *modes of correspondence*.

ME1: It looks like what it describes

Where a diagram refers to some situation in the real world, it is often helpful if that reference is instantly recognisable via a visual similarity to a real world object, or common real world convention. There are likely to be trade-offs here. Visual similarities to features of the user's domain may make the diagram less concise (VE4), or make the parts relatively hard to distinguish (ME4). The diagram will probably be less versatile (CE2, CE3), but fewer alternative interpretations will reduce the mental effort required from the user (TE4).

ME2: The purpose of each part is clear

Diagrams may be visually complex, as the information structures they correspond to are complex too. When we read or interact with them, those many parts and relationships will each have a variety of purposes with respect to our own intentions. This pattern can build on ME1 by assuming that knowledge of the real world will make it clear how these relate to each other, but an alternative approach is to make the diagram correspond more closely to the way a specific kind of user conceives their goals (PE2), or to direct users toward something they need to see (VE3).

ME3: Similar things look similar

(And conversely, things that look similar do have similar meanings). Consistency is a regular and important consideration in any kind of interaction design. Maintaining consistency between the visual elements and conventions used in a diagram is an essential enabler of SE4, and also contributes to VE2, VE3 and ME2. However, there is a trade-off relationship with ME4, meaning that it might be necessary to avoid too much visual similarity between different parts of a diagram, even where consistency would otherwise be desirable.

ME4: You can tell the difference between things

This is a fundamental property of any language – users must be able to tell the words, letters or symbols of the language apart from each other. In cases where this falls apart (for example, the letter O and the numeral 0 are hard to distinguish in many typefaces), it can lead to accidents (IE4). However, the perceptual properties that separate symbols aren't universal. For people who don't read Arabic or Chinese, many symbols in those languages look very similar. In a diagram, it will be more important to distinguish some things than others (VE3, SE4). But there are also occasions when ambiguity can be helpful (CE3), or where it might be useful to make the user to stop and think (TE3).

ME5: You can add comments

People often find it helpful to add informal comments or mark-up on top of a diagram they are using – for example, adding underlining, highlighter marks, or writing pencil notes to themselves and others. It's hard to guess in advance what these purposes might be, because every person's life has its own idiosyncrasies. The whole point of this pattern is that it doesn't follow the rules anticipated by the diagram designer – it provides an *escape from formality*. Being able to superimpose your own marks offers an everyday kind of creativity (CE1, CE4), as well as prompts or reminders (IE3, TE1). Designers of editors and other diagrammatic tools can easily become too focused on their own conception of the task at hand, forgetting that the user might think differently, so this pattern is often neglected. (Why aren't you allowed to draw directly on the Windows desktop, or the home screen of an iPad?)

ME6: The visual connotations are appropriate

The very word 'diagram' has unwelcome connotations for a lot of people. My research volunteers used to tell me that they associated it with self-assembly furniture, or programming a VCR – notoriously unwelcome household tasks. The standard visual conventions of a technical diagram (black and white, constant line widths, sans serif fonts) can be reassuring in a technical context, but inappropriate or even frightening in a setting requiring social inclusion or creative exploration. Sometimes, simply designing the diagram to fit the application is sufficient (VE2, ME1), but pictorial backgrounds and decorative elements (ME5, TE5, CE4) can be used to engage audiences, tell stories or motivate collaborators.

Experiences of Interaction

This group of patterns is concerned with the relatively mundane 'user interface' of diagram editors and tools. Standard textbooks on interaction design or human-computer interaction offer a lot more advice on how to create usable interaction elements. Nevertheless, these few patterns are sufficiently universal that they encompass a lot of diagram users' experiences.

IE1: Interaction opportunities are evident

Most electronic displays include two types of element: those that simply output information (e.g. text, pictures or video), and *control* elements that offer interactive functionality to the user (e.g. menus, buttons and links). The visual vocabulary used to differentiate the two, and show the users what they can interact with, has constantly evolved through generations of terminals, bitmapped displays, web interfaces and touch screens. These visual cues are often called 'affordances' drawing on the kind of physical analogy that imitates real world buttons or levers, where the shape suggests places to push or pull. Diagrams can incorporate such elements if appropriate (ME1, ME2, ME6), and if there is a genuine physical correspondence in the diagram meaning, this can be very helpful (TE4, PE2). However, more abstract or poetic diagrams (CE2, CE3, ME6) may not offer such obvious physical analogies. In these cases, it's worth thinking how the user will recognise the opportunity to interact, especially if one objective is encouraging them to explore (TE5).

IE2: Actions are fluid, not awkward

This should be obvious, but it's also hard to get it right. Many diagram editors carry out a lot of processing behind the scenes, to update the information structure in response to user actions. This may involve slow network accesses, large memory transfers, complex calculations, or iterative algorithms. As a result, the software may be slow to recognise user actions, and even slower to update the display. This makes changes frustrating (SE2) and increases unpredictability and mistakes (IE3, IE4). Ideally, system response and feedback should be immediate (less than a video frame of 20ms, or at worst under 100ms). In practice, many processes take longer than this to complete. One approach is to give the user fast visual feedback while waiting for the action to take effect. Fast feedback can ‘explain’ the delay using physical animations such as gravity, floating, bouncing or stretching when the diagram elements move. Such cues can give the user the idea that a process may take a little while to complete, while retaining the impression of fluidity.

IE3: Things stay where you put them

Most diagram editors help the user to keep the layout clear, by aligning elements, avoiding line-crossings and so on. Visual optimisation techniques such as force-directed layout can improve many visual aspects (VE1, VE2, VE4, SE1) while also allowing user freedom (SE2, SE4), revealing new relationships (TE2) and encouraging intended user actions (IE5). But all these automated corrections have the side-effect that parts of the diagram move by themselves. This can make it harder for the user to remember where things are (TE1) or recognise what they are (ME2). If parts move by themselves, this can result in errors (IE4). And users may have their own reasons for wanting a particular thing to be in a particular place (ME5).

IE4: Accidental mistakes are unlikely

In some situations of diagram usage, accidents are not a problem. They may even be an opportunity for serendipitous discovery (CE3). However, safety-critical technical and business settings often rely on diagrams to enforce predictable processes and policies. In these cases, the diagram designer may try to minimise the need for human judgement and interpretation (TE1, TE2, TE4). Visual cues can be used to enforce such policies (VE2, VE3, SE3, ME2, ME3, ME4). However, strict enforcement is likely to introduce trade-offs, so that the diagram is not as easy to manipulate and explore (SE2, ME5, IE2, IE3, PE3, PE4, CE2, CE4). A possible compromise is to encourage users to behave appropriately by making it easy for them to do so (IE5, PE1, PE2).

IE5: Easier actions steer what you do

Where an information structure has any complexity, there will be many options for the user to navigate, modify or construct diagrams. Some of these actions will be more readily available (IE1, SE3) or easy to achieve (IE2). Diagram designers may intend such distinctions with a particular narrative (SA1) or rhetorical purpose (SA3), or to support a particular kind of task (VE3, PE1). However, making particular kinds of action easier can reduce the generality of the diagram (CE2), and make it more difficult for users to achieve effects that the designer had not anticipated (SE2, ME5). Finally, easier is not always better (TE3).

IE6: It is easy to refer to specific parts

The visual elements of a diagram are not always easy to name – they might include unconventional graphic devices, or symbols (such as Greek letters) that not everybody can pronounce. This can make it difficult to discuss the diagram with other people (SA2), or even to describe it to one’s self (SE4, TE1). If there is an obvious visual resemblance, the name of a domain element might identify a particular diagram part that refers to it (ME1, ME4), but otherwise it will be necessary for users to point to the display. That can be awkward if the screen is too small (VE4) or too far away. It’s also irritating if this needs to be done over and over again (PE5), in which case the transient reference should be recorded (IE3) or perhaps encoded in a user annotation (ME5).

Experiences of Thinking

One view of diagrams is that they support *distributed cognition*: if an information structure is too large or complex to hold in our heads, then we move part of it to the outside world by writing it down. We then use the resources of the physical world (including parts of our own body such as eyes and fingers) to relate the external and internal representations to each other. Unfortunately this theoretical view does not yet provide much specific guidance for design, because brain scientists still don't know very much about what actually happens inside the head, so most of the theories are based on analogies to how a computer (having not much memory, with robot fingers and camera eyes) would do the same task.

TE1: You don't need to think too hard

There is a lot of cognitive psychology literature exploring the limitations of the human eyes and brain – which things can be perceived and distinguished, how much we can remember with our eyes closed, or by silently repeating it to ourselves and so on. The number of independent elements in short-term memory is pretty small: around half a dozen words, and a couple of pictures. Design tricks to help the user include ensuring that everything they need is visible (VE1, VE4, SE1, SE3, SE4), that actions correspond to what the user is already planning (PE1, PE2), and that they can recognise things by looking at the diagram, rather than having to remember them (VE3, ME1, ME2). Users also benefit from being able to focus their attention by referring to a particular part (IE6) or region (VE5), or making notes to themselves where they know it will be necessary to return to something in future (ME5). However, note that this usually desirable pattern may be directly contradicted (for some purposes) by TE3!

TE2: You can read-off new information

One of the nicest things about diagrams is the way they provide the user with *free rides* – logical inferences that don't need mental effort, because you can just see them. For example, if circle B is inside circle A, and you put a dot in circle B, it's immediately obvious that the dot is also inside A. This is true even if dozens of circles are nested, which would take a very long time to explain in words. Boundaries and connections, as well as gestalt properties, all provide this kind of facility. This pattern supports many of the benefits that come from the 'physical' behaviour of the human eye and visual system – edges and alignment, contrast and visual pop-out. Using these properties effectively in diagrams means anticipating the ways that visual cues will be used to encode structure (VE3, VE5, SE1, SE3, IE5) but may also mean sacrificing features that obscure such perceptual cues (ME1, ME6, CE3).

TE3: It makes you stop and think

Although interaction designers generally focus on ease of use (IE2), there are many situations where a little awkwardness can actually be beneficial. Studies of educational and problem-solving contexts show that making the diagram user work a little harder to interpret what they see makes it more memorable, and can even mean that problems get solved faster, because the user stops to think. These benefits are obviously in tension with TE1 and TE5. However, clarity of meaning is still likely to be beneficial (SE1, ME2, TE2) if it means that the user is drawing correct conclusions rather than getting confused.

TE4: Elements mean only one thing

Extremely abstract notations such as algebra can be hard to interpret (unless you're a mathematician), because the parts might mean anything at all. This is why popular references to algebra refer to 'x' as the 'unknown quantity'. Using such abstract symbols can mean that people are less able to do everyday reasoning (demonstrated in psychological experiments like the 'Wason task' (Wason 1968)). As a result, if diagram elements refer directly to the user's problem (ME1, PE2) this can make the diagram easier to use for simple tasks. On the other hand, increasing the *specificity* of meaning in this way reduces the expressive power (CE1, CE2, CE3). This is a rather constant trade-off in the design of highly generic technical diagrams such as visual programming languages.

TE5: You are drawn in to play around

Users are more likely to explore if they feel safe – if they can do things easily (IE2), are in control of the effects (IE4), can try things out (PE3, PE4) and can reverse their actions if they change their mind (SE2). It's also helpful to be able to return to previous transient states (IE3), although this history then becomes part of the information structure. Experimentation and tinkering is valued by users carrying out creative tasks, so may be associated with CE3 and CE4.

Experiences of Process

Some kinds of diagram carry within them a specification of the process that the user should follow – for example a flowchart. However, other diagrams refer to information structures that may be used or interpreted in a wide variety of ways. In these cases, the structure of the user's particular tasks may interact with the information structure, and with the tools available for editing or viewing the diagram.

PE1: The order of tasks is natural

The challenge in implementing this pattern is that every user may have a different view of what is 'natural'. For example, many people don't think of their tasks in an abstract way until they have already repeated them a number of times (IE2, IE5, TE5), or noticed similarities between different parts of an information structure (SE4). At this point, they may see an opportunity to deal with a larger class of problems in the same way (CE1). But diagram tools often require users to choose their approach before entering any information, or specify a relationship before creating the parts. This is not necessarily natural.

PE2: The steps you take match your goals

When people are using a diagram to achieve a particular end, or solve a problem, they will need to break down their overall problem into individual aspects that correspond to the features of the diagram. It helps if the complexity and descriptive power of individual diagram parts correspond reasonably closely to the units in which the user is thinking about their work. This will help to achieve pattern ME2, and depends to some extent on VE5, but means that the diagram is more likely to be specific to that particular application (ME5, TE4) rather than widely useful for general purposes (CE2).

PE3: You can try out a partial product

In a complex diagram, there may be lots of syntactic details that must be specified before the diagram is finished – connecting lines, completing boundaries, adding labels and so on. However, if a computer tool is being used, then it can also be useful to try out work-in-progress, to check if the work is proceeding according to plan before filling in the details. Unfortunately, many diagram editors respond to such trials by simply complaining about a syntax error or missing element, rather than acting on the parts that are complete. The ability to get feedback on the meaning of the diagram helps users to explore (IE2, TE5, PE4). It may also be useful to let users temporarily express some of the information in an informal way (ME5) in order to check the formal correctness of the rest.

PE4: You can be non-committal

Users are often unsure about what they are doing – either they don't understand the diagram, they don't understand the tool, or they don't understand what they are trying to achieve. It's really helpful in these situations to be able to try things out, without committing one's self. However, in some kinds of media, everything looks like a final product. It can be useful if the provisional working *looks* provisional – like a pencil sketch, or a drawing on a whiteboard – however, this appearance is because such things really are ephemeral (PE6). It might be possible to create cues for other readers to remind them of this (ME5), but the system also needs to know this.

PE5: Repetition can be automated

When a species of diagram is very general purpose, users often find it necessary to use the same combination of elements over and over again as a kind of idiom or cliché, to refer to some routine aspect of their own work that hasn't been anticipated in the language (PE2). If the system doesn't recognise that you are always doing the same thing, this can become repetitious and tedious. Repetition is also one of the reasons for viscosity (SE2) when a single change needs to be applied in many different places. In these circumstances, it's useful to be able to program a system to do the repetition for you, or even define a new language element that specifies where the same combination should be applied (CE1). The danger of doing this is that the original relationships become invisible, hidden by the new language element (SE1), and also that the more powerful new feature may result in larger accidents (IE4).

PE6: The content can be preserved

Some kinds of diagram are more permanent than others. Writing on a whiteboard is transient, so people are in the habit of photographing them. We usually imagine that everything on a computer can be saved, but what about the position of the cursor, or a particular set of selected objects? Many diagrams, and aspects of diagrams, are more or less transient. Even where they are permanently recorded, information in previous versions may be lost. It's not always useful to keep everything – previous versions themselves become part of the information structure, and you need ways to find them (VE3, SE3). It's not always easy to see what has changed (SE4), because the visual appearance hasn't emphasised the significant changes (ME4). Asking users to add information explaining the history (ME5) adds extra managerial overhead for every revision (SE2). Finally, people doing exploratory design don't always like the idea that possible mistakes or misjudgements will be preserved – transience can be an advantage in collaborative or creative work (PE4).

Experiences of Creativity

Diagrams, as a technical genre, often give the impression of rigidity or constraint (ME6). But at the same time, many creative people find the idea of expressing their ideas in visual form to be very appealing. It is perfectly possible to create diagrammatic forms that offer creative power by drawing on principles of sketching or visual programming to generate new ideas.

CE1: You can extend the language

Programming languages, and some kinds of diagram, offer users the ability to create new abstractions – new types of element that can be incorporated into the diagram. These might provide a succinct (VE4) alternative to a laborious combination of elements (PE5), or create elements that correspond directly to concepts in your own work (ME2, PE2). However, these powers come at a price. The 'meta-level' thinking necessary to make good abstractions is familiar to mathematicians, computer scientists and philosophers, but is not an everyday skill for most people (TE1). Doing it badly can even make the diagram harder to use (SE1, ME2, TE4). For this reason, most systems don't ask users to start every task by thinking about what abstractions they need (PE1), but provide some kind of meta-tool or meta-diagram that can be used to specify new elements once they are clearly necessary. These meta-diagrams can provide creative insights when you are immersed in them (TE5), but they have their own sets of patterns and experience profiles – for example, complex structures of abstraction can make it hard to modify the meta-diagram later (SE2).

CE2: You can redefine how it is interpreted

Everybody who reads a diagram sees it in their own way. But sometimes the diagram author wants to suggest a particular reading (SA1), or even change the way the diagram would otherwise be interpreted (SA3). It is possible to suggest other interpretations to human readers by adding informal notes or guidance (ME5), but if the 'readers' of the diagram include computer programs, it is necessary to specify how the new interpretation should be processed. This could be as simple as modifying the visual appearance of elements or wording of labels (ME1, ME2, ME6), but it could take the form of defining one element in terms of another. As soon as such facilities become generically powerful, they start to gain the advantages – and also disadvantages – of abstract languages (CE1).

CE3: You can see different things when you look again

In situations where a diagram is being created to explore ideas, or discover something new, the visual attributes can often support this by allowing parts and relationships to be blurry or ambiguous. Many creative designers intentionally work with soft pencils, broad brushes or rapid actions to encourage the serendipity of seeing something new or different in what they have drawn. Not only are sketches informal and provisional (PE4) – they are also resources for thinking (TE2, TE3). Of course, the informality of sketches makes them quite inappropriate for many other more rigorous kinds of diagram application (VE2, SE1, SE3, ME2, ME3, ME4, TE4).

CE4: Anything not forbidden is allowed

Users can always invent new ways of working with the tools you give them. In order to support the widest possible range of use, it may be helpful to allow all possible ways of reading and creating diagrams, not just adding comments (ME5). Once people become familiar with tools, they often prefer shortcuts, or leave out recommended steps. When they are not so expert, they may do things the 'wrong' way. Their approach might not be optimal, but if it works, preventing it would only annoy them. Flexibility like this can support patterns TE5, PE1 and PE4. Of course, flexibility does have disadvantages. If parts have multiple meanings, their purpose might not be so clear (ME2), and it might be harder to work out what to do with them (TE4). If you don't know what people are going to do, it is harder to give a clear story (VE2), or guide them to the most appropriate actions (IE5, PE2).

Conclusion

Diagrams provide a means of interacting with information structures. When users read them, manipulate them or share them, these activities can be made easier or harder by designing the diagram and tools to support particular patterns of user experience. When designing a new species of diagram, and the editors or user interface to interact with it, it is worth spending time in advance deciding what specific kinds of activity (from IA1 to SA3) will be the highest priorities. Identifying the patterns that support those priorities, and taking account of trade-offs to de-emphasise other less important patterns, can guide the designer to create interactive diagrams that meet users' needs.

Acknowledgements

The ideas presented in this chapter are developed from the Cognitive Dimensions of Notations (CDs) framework originally proposed by Thomas Green. The individual patterns, and the ways that they have been reorganised and structured into a pattern language, draw on the research of many collaborators working with Thomas and the present author, including Marian Petre, Steven Clarke, Darren Edge, Sabrina Bresciani, Luke Church and Sally Fincher. Thomas Green, Sally Fincher, Darren Edge and Luke Church have given valuable feedback during the preparation of this chapter.

Appendix

Many of these patterns have been described and analysed in previous research literature, but usually expressed in more technical terms, often drawn from Cognitive Dimensions of Notations (CDs). This appendix provides links from the individual patterns described in this chapter to relevant earlier research in the CDs literature and elsewhere, including the technical terms and authors responsible for introducing them. In the following table, the 'classic' set of CDs most often cited were first systematised in Green & Petre (1996), and rephrased for more general use by Blackwell & Green (2000, 2003).

Although this chapter has tried to indicate the CDs profiles of different activities, and the trade-offs between different CDs, the CDs literature also includes more detailed analyses, for example considering the consequences of different types of media (especially those that are transient rather than permanent), the role of abstraction in programming, the use of notational sub-devices, and the relationship between multiple layers that describe the same information structure in different notational forms.

This chapter does provide a more extensive collection of activities and dimensions than the previous literature. This is mainly amalgamated from earlier sources, but also introduces a new class of activities (SA1–SA3). The description of CDs in terms of a pattern language was originally proposed by Sally Fincher (2002), and developed by Blackwell & Fincher (2010).

The information you need is visible	Visibility (CDs classic)
The overall story is clear	Clarity Is the visualization easily understandable with low cognitive effort? (Bresciani et al 2008) Synopsie (was –'grokkiness') – understanding the whole when you stand back and look (Whitley & Blackwell 2001)
Important parts draw your attention	Directed Focus Does the visualization direct the attention to the main item(s) of a discussion? (Bresciani et al 2008) Penetrability – How does an API facilitate exploration, analysis, and understanding of its components? (Clarke & Becker 2003)
The visual layout is concise	Diffuseness (CDs classic)
You can see detail in context	Detail in context – It is possible to see how elements relate to others within the same notational layer (rather than to elements in other layers, which is role expressiveness), and it is possible to move between them with sensible transitions, such as Fisheye views (Milic-Frayling et al 2004)

You can see relationships between parts	Hidden Dependencies (CDs classic)
You can change your mind easily	Viscosity (CDs classic)
There are routes from a thing you know to something you don't	Indexing – The notation includes elements to help the user find specific parts. (Larkin & Simon 1987)
You can compare or contrast different parts	Juxtaposability (CDs classic)

It looks like what it describes	Closeness of mapping (CDs classic)
The purpose of each part is clear	Role Expressiveness (CDs classic)
Similar things look similar	Consistency (CDs classic)
You can tell the difference between things	Discriminability (Green, personal communication)
You can add comments	Secondary Notation (CDs classic)
The visual connotations are appropriate	Visual Impact – How attractive is the visualization? (Bresciani et al 2008/Crilly et al 2012)

Interaction opportunities are evident	Hidden Augmentations The physical objects are computationally augmented in a non-obvious manner (Edge & Blackwell 2006) Purposeful Affordances Interactions possible with the objects have a clear and meaningful purpose (Edge & Blackwell 2006)
Actions are fluid, not awkward	Unwieldy Operations Difficulties due to the size, shape, structure or weight of objects and the actions required (Edge & Blackwell 2006)
Things stay where you put them	Shakiness Proneness of physical state to accidental damage or change (Edge & Blackwell 2006)
Accidental mistakes are unlikely	Error-proneness (CDs classic)
Easier actions steer what you do	Unevenness – easy actions push ideas in a certain direction (Stacey 1995)
It is easy to refer to specific parts	Verbal reference – can you talk about it without pointing at the screen? (Church et al 2012)

You don't need to think too hard	Hard Mental Operations (CDs classic)
You can read-off new information	Facilitated Insight Are new insights generated as a result of the visualization form? (Bresciani et al 2008) Free rides (Shimojima 1996)
It makes you stop and think	Useful awkwardness - force the user to reflect on the task (discussion with Marian Petre/O'Hara & Payne 1998)
Elements mean only one thing	Specificity - The notation uses elements that have a limited number of potential meanings (irrespective of their defined meaning in this notation), rather than a wide range of conventional uses (Stenning & Oberlander 1995)
You are drawn in to play around	Tinkering, Flow and Attention Investment (Beckwith et al 2006, Nash 2012, Blackwell 2002)

The order of tasks is natural	Premature Commitment (CDs classic)
The steps you take match your goals	Work-Step Unit - How much of a programming task can/must be completed in a single task step? (Clarke & Becker 2003)
You can try out a partial product	Progressive Evaluation (CDs classic)
You can be non-committal	Provisionality (CDs classic)
Repetition can be automated	Automation/Repeats Easy Ability to redefine what actions are executed at what time (Edge & Blackwell 2006, Rode & Toye 2004)
The content can be preserved	Permanence Preservation or archiving of meaningful physical structures for later inspection (Edge & Blackwell 2006)

You can extend the language	Abstraction (CDs classic)
You can redefine how it is interpreted	Adaptability Ability to redefine how states are interpreted (Edge & Blackwell 2006) API Elaboration- To what extent can/must an API be adapted to meet the needs of a targeted developer? (Clarke & Becker 2003)
You can see different things when you look again	Creative Ambiguity - The extent to which a notation encourages or enables the user to see something different when looking at it a second time (Hewson 1991, Suwa & Tversky 1997)
Anything not forbidden is allowed	Construction Principle: impose only the necessary restrictions on what constitutes a well-formed statement (discussion with Harold Thimbleby, Stapleton & Delaney 2008)

References

Alexander, C., Ishikawa, S. and Silverstein, M. (1977). *A pattern language: towns, buildings, construction*. New York: Oxford University Press.

Beckwith, L., Kissinger, C., Burnett, B., Wiedenbeck, S., Lawrance, J., Blackwell, A. and Cook, C. (2006). Tinkering and gender in end-user programmers' debugging. In *Proceedings of CHI* 2006, pp. 231-240.

Blackwell, A.F. (2002). First steps in programming: A rationale for Attention Investment models. In Proceedings of the *IEEE Symposia on Human-Centric Computing Languages and Environments*, pp. 2-10.

Blackwell, A.F. & Fincher, S. (2010). PUX: Patterns of User Experience. *interactions* 17(2), 27-31.

Blackwell, A.F. & Green, T.R.G. (2000). A Cognitive Dimensions questionnaire optimised for users. In A.F. Blackwell & E. Bilotta (Eds.) *Proceedings of the Twelth Annual Meeting of the Psychology of Programming Interest Group* , 137-152

Blackwell, A.F. and Green, T.R.G. (2003). Notational systems - the Cognitive Dimensions of Notations framework. In J.M. Carroll (Ed.) HCI *Models, Theories and Frameworks: Toward a multidisciplinary science*. San Francisco: Morgan Kaufmann, 103-134.

Blackwell, A.F., Britton, C., Cox, A. Green, T.R.G., Gurr, C.A., Kadoda, G.F., Kutar, M., Loomes, M., Nehaniv, C.L., Petre, M., Roast, C., Roes, C., Wong, A. and Young, R.M. (2001). Cognitive Dimensions of Notations: Design tools for cognitive technology. In M. Beynon, C.L. Nehaniv, and K. Dautenhahn (Eds.) *Cognitive Technology* 2001 (LNAI 2117). Springer-Verlag, pp. 325-341

Blackwell, A.F., Rode, J.A. and Toye, E.F. (2009). How do we program the home? Gender, attention investment, and the psychology of programming at home. *International Journal of Human Computer Studies* 67, 324-341.

Bresciani, S., Blackwell, A.F. and Eppler, M. (2008). A Collaborative Dimensions Framework: Understanding the mediating role of conceptual visualizations in collaborative knowledge work. Proc. *41st Hawaii International Conference on System Sciences (HICCS 08)*, pp. 180-189.

Church, L., Rothwell, N., Downie, M., deLahunta, S. and Blackwell, A.F. (2012). Sketching by programming in the Choreographic Language Agent. In *Proceedings of the Psychology of Programming Interest Group Annual Conference*. (PPIG 2012), pp. 163-174.

Clarke, S. and Becker, C. (2003). Using the cognitive dimensions framework to measure the usability of a class library. In *Proceedings of the First Joint Conference of EASE & PPIG (PPIG 15),* pp 359-366.

Crilly, N., Blackwell, A.F. and Clarkson, P.J. (2012). Graphic elicitation: using research diagrams as interview stimuli. In J Hughes (Ed.), *SAGE Visual Methods*, SAGE Library of Research Methods (Vol. 4, Ch. 65, pp. 283-307).

Eckert, C., Blackwell, A.F., Stacey, M., Earl, C. and Church, L. (2012). Sketching across design domains: Roles and formalities. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 26(3), 245-266.

Edge, D. and Blackwell, A.F. (2006). Correlates of the cognitive dimensions for tangible user interface. Journal of Visual Languages and Computing, 17(4), 366-394.

Fincher, S. (2002). Patterns for HCI and Cognitive Dimensions: two halves of the same story? In *Proc. 14th Workshop of the Psychology of Programming Interest Group (PPIG 14)*, pp.156-172.

Green, T. R. G. (1989). Cognitive dimensions of notations. In A Sutcliffe and L Macaulay (Ed.) *People and Computers V* Cambridge University Press: Cambridge., pp. 443-460.

Green, T. R. G. (1990) The cognitive dimension of viscosity: a sticky problem for HCI. In D. Diaper, D. Gilmore, G. Cockton and B. Shackel (Eds.) *Human-Computer Interaction — INTERACT '90*. Elsevier.

Green, T.R.G. & Blackwell, A.F. (1998). Design for usability using Cognitive Dimensions. Tutorial session at *British Computer Society conference on Human Computer Interaction HCI'98*.

Green, T.R.G. & Petre, M. (1996). Usability analysis of visual programming environments: a 'cognitive dimensions' approach. *Journal of Visual Languages and Computing*, 7,131-174.

Hewson, R. (1991). Deciding through doing: The role of sketching in typographic design. *ACM SIGCHI Bulletin*, 23(4), 39-40.

Larkin, J. H. and Simon, H. A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11, 65-99.

Milic-Frayling, N., Sommerer, R., Rodden, K. and Blackwell, A.F. (2004). SmartView and SearchMobil: Providing overview and detail in handheld browsing. In *Mobile and Ubiquitous Information Access* (Lecture Notes in Computer Science), Springer, pp 158-171.

Nash, C. and Blackwell, A.F. (2012). Liveness and Flow in Notation Use. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, pp. 76-81.

O'Hara, K. P. and Payne, S. J. (1998). The effects of operator implementation cost on planfulness of problem solving and learning. *Cognitive Psychology*, 35, 34-70.

Richards, C. J. and Johnson R.D. (1980). Graphic codes for flow charts. *Information Design Journal* 1 (4) 261—270.

Rode, J.A., Toye, E.F. and Blackwell, A.F. (2004). The Fuzzy Felt Ethnography - understanding the programming patterns of domestic appliances. *Personal and Ubiquitous Computing,* 8, 161-176.

Shimojima, A. (1996). Operational constraints in diagrammatic reasoning. In G. Allwein & J. Barwise (Eds), Logical reasoning with diagrams. Oxford University Press, pp. 27-48.

Stacey, M. K. (1995). Distorting design: unevenness as a cognitive dimension of design tools. In G. Allen, J. Wilkinson & P. Wright (eds.), *Adjunct Proceedings of HCI'95*. Huddersfield: University of Huddersfield School of Computing and Mathematics.

Stapleton, G. and Delaney, A. (2008) Evaluating and generalizing constraint diagrams. *Journal of Visual Languages and Computing*, 19 (4), 499-521.

Stenning, K. and Oberlander, J. (1995). A cognitive theory of graphical and linguistic reasoning: Logic and implementation. *Cognitive Science*, 19(1), 97–140.

Suwa, M. and Tversky, B. (1997). What do architects and students perceive in their design sketches? A protocol analysis. *Design Studies*, 18, 385-403.

Wason, P. C. (1968). Reasoning about a rule. *The Quarterly journal of experimental psychology*, *20*(3), 273-281.

Whitley, K.N. and Blackwell, A.F. (2001). Visual programming in the wild: A survey of LabVIEW programmers. *Journal of Visual Languages and Computing*, 12(4), 435-472.