COMPUTER SCIENCE TECHNICAL REPORT

THE UNTRAINED EYE: WHAT MAKES A REPRESENTATION EASY FOR NOVICE READERS TO UNDERSTAND?

Carol Britton and Sara Jones

Report No 308

December 1997

The Untrained Eye: what makes a representation easy for novice readers to understand?

Carol Britton and Sara Jones
Department of Computer Science
University of Hertfordshire
College Lane, Hatfield, Herts. UK
AL10 9AB
Tel: 01707 284354 / 284370
Email: C.Britton, S.Jones@herts.ac.uk

Abstract

It is generally recognised that choice of notations can have a significant effect on the system development process, particularly in the early stages of requirements capture and validation. In the development of interactive systems it is essential that stakeholders are able to understand representations of key design ideas, in order that they can participate meaningfully in the development process. Although in some development projects this requirement is fulfilled by an early prototype, there are many situations in which paper-based representations have to be used. Some stakeholders, such as clients and potential users of the system, may be unfamiliar with the notations used by software engineers and therefore find it difficult to understand representations produced using such notations well enough to contribute effectively to system development. In this paper we discuss the problem of how to choose the most effective notation in a particular situation and identify understandability of representations as a key issue for interactive system development. We then consider ways in which the notion of understandability may be defined in this context, and discuss the possibility of evaluating notations in terms of ease of understanding.

1 Introduction.

Although the relationship between notations, representations and the quality of the system development process is not fully understood, the impact of the choice of notation on successful performance of many system development activities has long been recognised (Green 89 and 91a, McCluskey et al 95, Modugno et al 94, Roast 97). However, little is currently known about what notations are likely to be most suitable for use in which contexts. The choice of notations for particular projects often reflects the experience or preferences of the development team more than an objective consideration of possible alternatives (McCluskey et al 95).

In the case of interactive system development, it is essential that all those involved, including clients and users who may be untrained in the use of notations for modelling software, have access to a representation that they can readily understand. This is because successful development of interactive systems demands active user participation, and this can only be achieved through a shared understanding of representations of the system. Thus understandability of representations is a necessary (though not sufficient) condition for successful interactive system development.

In many interactive system development projects, the need for user understanding is addressed by building a skeleton prototype to offer options which can then be refined in line with the user's requirements. A prototype facilitates communication between developers and clients or users, allows users to relate what they are shown to their own experience of tasks, and enables them to give meaningful feedback on the design ideas which are embodied in the prototype.

There are, however, certain cases where the prototyping approach may be neither feasible nor adequate. For example, early design ideas may need to be discussed with clients and users before even a first prototype is built. For safety critical systems, development based on a prototype would be unlikely to gain the client's confidence, and for large complex systems of any kind the development of complete system prototypes is unlikely to be cost-effective. Even in development situations where prototyping appears to be desirable, problems can arise relating to the complexities of version control, difficulties in producing realistic task scenarios or the amount of time that is needed for a client to evaluate the prototype (Britton and Doake 96).

Where prototypes cannot be used, the main alternative is currently to use paper-based representations constructed using notations designed for modelling software. The question for the developer is how to select a notation which is easy for clients and users to understand. The problem of which modelling notation to choose is exacerbated for the developer of interactive systems by the fact that many notations designed to model this type of system are relatively immature, their theoretical underpinnings are often weakly documented and some are still undergoing evolution stimulated by practical experience. Since ease of understanding of representations is crucial for successful development of interactive systems, this paper aims to address the question of how to evaluate notations in terms of understandability and whether it is possible to predict the extent to which an untried notation will be readily understandable by novice readers.

The following section defines some of the key terms which we will use throughout the rest of the paper. Section 3 discusses the problem of choosing notations appropriate to a particular software development project, and section 4 contains a brief discussion of understanding and its significance in the context of representations for interactive system development. Section 5 considers ways in which notations may be evaluated in terms of ease of understanding, and we end with some conclusions in section 6.

2. Definitions of terms

No useful discussion can take place without a shared understanding of the meanings of key terms, and words such as 'model' are particularly prone to a variety of interpretations. Clear definitions of terms help to eliminate vagueness and to reveal and resolve ambiguity; even if the definitions used are not agreed by all readers, they nevertheless help to clarify what the authors are trying to say. We therefore suggest the following as working definitions in the context of this paper:

Notation:	a system of signs, symbols or characters used to represent concepts that are relevant to the development of a system; examples include the data flow diagramming technique, natural language, storyboards, the Z specification language.
Representation:	an expression of a concept or concepts relevant to the development of a system which is created using notations such as those listed above; examples of representations include particular data flow diagrams, Z specifications, and statements of requirements written in natural language.
Model:	the information content of a representation.
Stakeholder:	a person involved in any way in the development of a system; potential system users, clients and software developers are typically all stakeholders in the development of an interactive system.

3. The problem of choosing suitable notations

The problem of assessing and selecting notations, methods and tools for system development is one that has attracted interest from both academia and industry. The DESMET project (Kitchenham 94) addressed the question of how to determine objectively the effects and effectiveness of methods and tools in general, using both qualitative and quantitative approaches. More recent projects such as RESCUED (O'Neill et al 97) and the Evaluation Framework for Representations in Requirements Engineering (Sutcliffe et al 97) are currently investigating the choice of representations at various stages of the software development process.

Several authors have highlighted the importance of choosing an effective modelling notation, while others have set out general guidelines on how to choose appropriate requirements representations and methods for particular situations. Macaulay has identified the question of 'What representation format should be used?' as a significant one for practising software developers working on the elicitation and specification of system requirements (Macaulay 96) , and similar issues were discussed in a workshop reported in (Jones et al 97). Christel and Kang's report for the SEI on requirements elicitation identified a number of factors relating to the scope of a project, the need for understanding by various parties, and the volatility of requirements, which they suggested should influence the developer's choice of requirements techniques and notations (Christel and Kang 92). In 1995 Brun and Beaudouin-Lafon presented a taxonomy for the evaluation of formalisms for the specification of interactive systems (Brun and Beaudouin-Lafon 95). The taxonomy includes three types of criteria, relating to expressive power, generative capabilities, and extensibility and usability. More recently, Sommerville and Sawyer have listed some generic guidelines for choosing representations and methods (Sommerville and Sawyer 97) and the RESPECT project has made some general recommendations as to the stages of system development for which certain techniques and representations are most appropriate (Maguire 97). However with each of these authors taking a slightly different perspective on the problem, it is difficult for the practising software developer to know on what basis the choice of modelling notation for a particular project should be made.

In (Jones et al 97) the authors of this paper argue that an appropriate approach to the problem is to base the selection of notations as far as possible on existing knowledge and experience. A framework is proposed within which this knowledge can be structured, developed and exploited. The framework is presented in the form of a simple questionnaire and aims to help practising software developers make quick, but useful evaluations of available modelling notations in the context of particular development situations. It does not propose rigorous guidelines, but is intended rather as a tool to promote discussion and evaluation. Figure 1, below, gives an overview of the proposed selection process, showing how general criteria for modelling notations (identified from the industrial and academic literature) are considered in relation to a profile of the project, which is obtained from a detailed checklist of features of the project context provided in the questionnaire. Comparison of the general criteria for notations with the profile of the project context provides the basis for a refined list of criteria for notations which will be effective in the situation under consideration. Modelling notations available to the developer are then considered in relation to the refined list of criteria and from this the final selection of appropriate modelling notations is made.

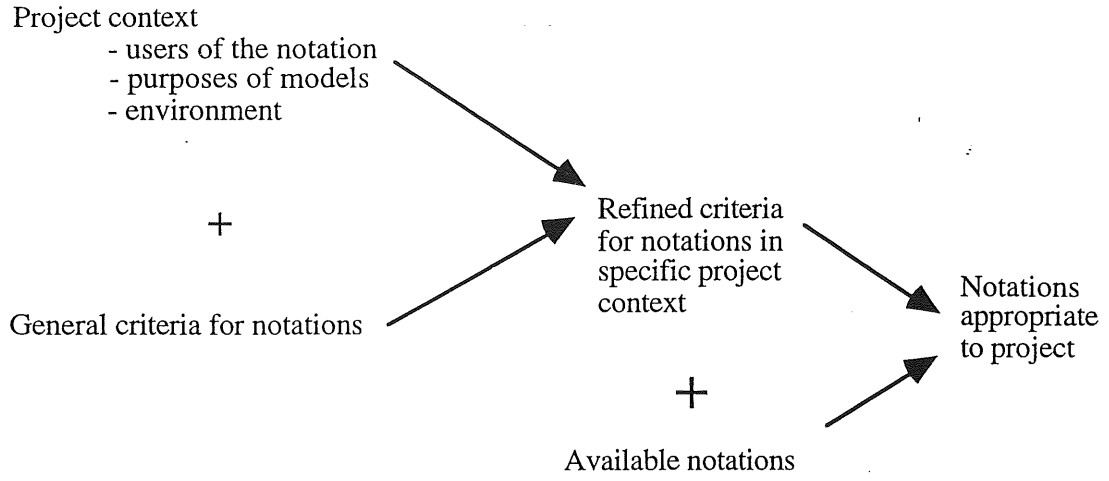


Figure 1: Overview of the selection process for modelling notations

At present, both the generation of a list of refined criteria, and the use of this refined list in choosing an appropriate set of notations can be done only on the basis of experience gained in past development projects. Our aim, however, is to investigate the possibility of providing a more objective basis on which to make such choices by defining key properties of notations in sufficient detail that at least some objective measures might eventually be used to compare them.

The general criteria for modelling notations, which are used in the first stage of the selection process, are those which are cited most frequently in the academic and industrial literatures. Farbey (93) covers criteria relating directly to notations, such as readability, modifiability and lack of ambiguity, and criteria relating to the notation in use, such as the production of a well-presented specification, the cost in time to produce the representation, and the amount of support available. Green (89) suggests that a notation should be able to support what he terms 'opportunistic planning', where high-and low-level decisions may be mingled, work may frequently be re-evaluated and modified and commitment to different decisions may be strong or weak. Although Green is writing about notations for programming, his point is equally relevant to the study of notations which are used earlier in the development process. In an article on hypermedia design, Garzotto (95) evaluates notations in terms of what can be described: a useful notation should be able to model information content and presentation, system structure, and interaction with the user. Davis (93) suggests a list of criteria pertaining to the effectiveness of representations and the choice of notations. Davis' list includes criteria relating directly to modelling notations, such as that the notation should permit annotation and traceability, facilitate modification, and some that are expressed in terms of the software requirements specification. These include the criterion that "proper use of the technique should result in an SRS that is understandable by customers and users who are not computer scientists".

Among publications from authors in industry, criteria for modelling notations in the STARTS guide (DTI and NCC 87) incorporate qualities such as rigour, suitability for agreement with the end-user and assistance with structuring the requirements. Rigour comprises four separate features: how precisely the syntax of the notation is defined, the extent to which it is underpinned by maths and logic, whether the meaning of individual symbols is defined, and the extent to which the notation supports consistency checking of the requirements themselves. Suitability for end-user agreement refers to ease of understanding of the notation by an untrained reader, and assistance with structuring the

requirements assesses the extent to which the notation supports hierarchical decomposition and separation of concerns in the representation. The STARTS guide also regards the range of requirements covered by the notation as important, including functional, performance, interface, system development and process requirements. Admiral Training's (95) guide to interactive multimedia development is similar to the STARTS guide in placing emphasis on what the notation should be able to model and on ease of understanding. Effective notations, according to Admiral, should be able to describe the current situation, the target audience, the actual and required level of user performance, the overall aim of the system, the environment, possible constraints and details of specific functions. Notations need to be easy to understand, not only for clients and users, but also for members of the multimedia development team who do not have a background in computer science.

Authors who have explored the topic of criteria for modelling notations differ in their approach and in the weightings that they give to different properties; in general, and as might be expected, publications from industry focus on practical criteria aimed at producing an effective representation, whereas criteria chosen by academics tend to be more theoretically based. There are, however, certain criteria that are generally agreed by most authors to be important for effective modelling notations; one of these is the criterion that a notation should be as easy as possible for untrained readers to understand. The following section explores the issue of ease of understanding, both generally and in the context of interactive system development.

4 Ease of understanding

4.1 A view of understanding

One perspective on understanding from the philosophy of language is described by Blackburn (84). The key elements in a discussion about the understanding of language or linguistic utterances may, according to Blackburn, be represented as in figure 2.

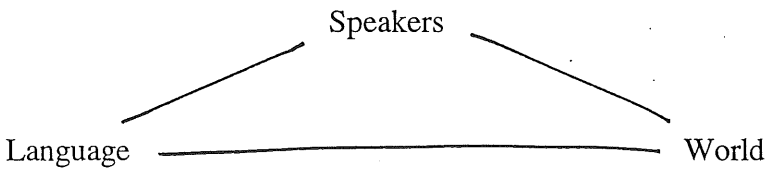


Figure 2: Key elements of understanding

Figure 2 denotes the fact that language is typically used by speakers to express their thoughts about the world. A person with a thought (such as 'I'd like a cup of coffee') can only communicate that thought to another person, such as the waiter in a café, through the use of some language - either verbal or visual (as in sign language). We say that the waiter is able to understand such a request if he can attach the intended meaning to it, or in other words, if he can know what thought or desire it expressed.

According to this view, understanding relies on a shared system of thoughts and meanings, one which all members of the linguistic community have (at least roughly) in common. In the above example, both the waiter and the original speaker would normally be familiar with the idea of a request and the fact that the words 'I'd like' may be used to express one. They would normally also have roughly the same idea about what kind of thing the words 'a cup of coffee' are used to denote. Thus Blackburn talks about a 'dog-legged' theory of meaning in which understanding between different members of a linguistic community is possible because that community shares a set of conventions about the mappings between words and things in the world, and also between words and thoughts.

For the purposes of our discussion, we may substitute the more general notion of 'representations' in place of 'language' in figure 2. We may limit ourselves to the domain of software systems as the part of the world in which we are particularly interested, and think of our speakers as being software developers, clients, and potential system users, in other words, the stakeholders in a system development (see figure 3).

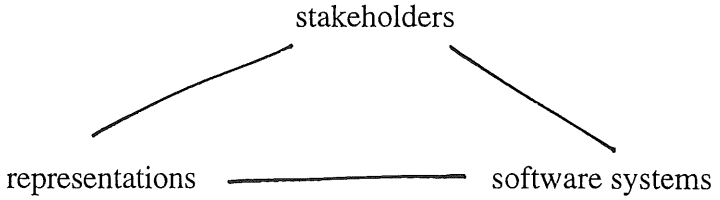


Figure 3: Key elements of understanding in interactive system development

Following on from the discussion above, we would say that understanding between the stakeholders would have to rely on a shared system of thoughts and meanings for representations; or in other words, a shared set of mappings between aspects of the software systems and their representations, and also between the representations and the thoughts of the stakeholders. But of course, this is likely to be problematic. Users and developers typically come to the development with vastly different experiences, both of the application domain (users will know it well, developers often hardly at all), and of software systems (users may have experience of system interfaces but usually do not have the developers' experience of the internal workings of such systems). Therefore our community of 'speakers' do not have a perfectly shared set of thoughts. There is also, at present, no common currency of representations whose elements are linked by any common agreement with particular aspects of software systems. Thus each of the links of the triangle in figure 3 is rather weak, and we can begin to appreciate why it will be difficult to tackle the problem of using representations effectively in the development of interactive systems.

The following paragraphs will return briefly to the question of why, despite being difficult, the question of understanding is important in this context, and will consider, in a little more depth what might be involved in understanding a particular representation. This leads to a discussion, in section 5, of whether it may be feasible to evaluate or predict the understandability of representations written using particular notations.

4.2 Understanding in the context of interactive system development

As we have already said, building interactive systems involves developers and users with vastly different experience and different ways of thinking who must be able to communicate effectively with each other if the system eventually delivered is to satisfy the users' requirements. It is assumed to be the developer's responsibility to present the users, who are likely to have little or no training in the field of computing, with representations of the intended system which they can understand well enough to give useful and meaningful feedback on design decisions proposed.

It is important that the representations used should be easy to understand so that untrained clients and users are not forced to put effort into deciphering them, rather than concentrating on their content. This problem is described most eloquently in (Green 89): "When a train of thought is broken again and again by the need to find something out the hard way, it is difficult to piece thoughts together into inspirations; it is difficult enough even to finish a simple train of thought without making a mistake, simply because of having to get the information in some tedious and error-prone way."

It is difficult to generalise about what makes representations written in one notation easier for an untrained reader to understand than those in another. Inevitably, users who are shown a representation will have personal preferences, different backgrounds and experiences and different attitudes to the representation and to the unfamiliar notation. One user may be instinctively attracted to diagrams, while another may feel more at home with a text-based notation. Without detailed knowledge of a particular user, it is not possible to predict whether he or she will find a specific notation easy to understand. However, some understanding of the needs of a 'typical' novice reader in understanding a representation may be possible, and may assist us in evaluating existing or proposed new notations in terms of their suitability for use with novice readers.

In basic terms, we may say that users presented with a representation of a proposed system would be expected first to extract information from the representation, either by simply absorbing and assimilating the information presented, or by searching for particular pieces of significance. We hope that they will then be in a position to relate the system described to the domain in which they are to use it, to imagine how it would be to perform the tasks of interest in that domain, and to comment on the proposed design, perhaps adding creative suggestions of their own as to how the proposed system might be modified. Thus we must permit users to relate the concepts denoted in our representations to concepts of their own, and encourage both reasoning and creativity.

Much work in cognitive psychology has addressed the extraction and use of information from spoken or written language, but there is less on the use of graphical representations such as those typically used in software system development. Sengler (83) has identified finding (or searching for), decomposing and abstracting information as three of the most important activities in reading a representation. Larkin and Simon (87) have also identified the key activities in reading representations as searching for and recognising relevant information, drawing inferences from that information and adding new (inferred) elements to the representation.

Given this basic view of what might be required in order for a representation to be usefully understood by an untrained reader, we now move on to consider the possibility of evaluating existing or possible new notations in terms of their likely understandability for novice readers.

5 Evaluating notations in terms of ease of understanding

5.1 Cognitive dimensions

One of the most useful concepts in the evaluation of notations for different purposes is that of cognitive dimensions (Green 89 and 91a). Cognitive dimensions are not a set of criteria which may be satisfied to various degrees by different notations; rather they are aspects of notations which may be important and useful in specific situations. Cognitive dimensions are tools for thinking about notations, rather than detailed guidelines. They are intended to support the evaluation of any type of information structure and can be applied to programming languages, musical scores or even telephone numbers (Modugno et al 94). The twelve dimensions described by Green aim to provide a broad-brush assessment of the information structures to which they are applied. All of the dimensions are useful in the overall evaluation of a modelling notation, but we mention here only those that are directly relevant to the particular feature of notations that is the subject of this paper: that of ease of understanding for novice readers.

5.1.1 Closeness of mapping.

The cognitive dimension which probably contributes most to ease of understanding is that of the closeness of the notation to the problem domain. In the terms of section 4, closeness of mapping corresponds to a strong link between language and world (see figure 2). If the reader can readily recognise symbols in the notations as representations of elements in the domain, this will facilitate the tasks of searching the representation and abstracting important information from it.

Close mapping between representation and domain is much more readily achieved in disciplines such as architecture and engineering where there are clear physical links. In software system development, where what is normally visible are the effects of processes and interactions between data elements, there is no obvious way of representing the actual processes or data elements themselves. It is therefore very difficult to achieve real closeness of mapping between representation and problem domain in the development of software systems. This is discussed further in section 5.2.2 on motivation of symbols in a notation.

5.1.2 Hidden dependencies.

A notation which allows hidden dependencies will produce representations in which important links between elements are not visible. This can be confusing for novice readers who do not know whether links exist, but can't be seen, whether they do not exist, or whether they are simply not part of the representation. The example most frequently cited is that of a spreadsheet where it is not clear that the value of a particular cell is dependent on the content of some other cell. Among notations for modelling software, data flow diagrams illustrate clearly the links between processes in terms of data, but do not show any timing or control dependencies. To achieve a complete representation, a range of notations has to be used in order to make explicit all the different dependencies in the system.

5.1.3 Diffuseness/terseness.

The degree of diffuseness or terseness of a notation is important for novice readers in two ways: a notation which is not succinct is likely to produce representations which are cluttered, lacking in organisation, or which have very little content; on the other hand a notation in which symbols carry complex information is likely to produce representations which are too dense to convey any meaning to novice readers. One example of this is the specification language Z where a single symbol, such as $\oplus$, represents a complicated mathematical operation.

5.1.4 Visibility

The ability to view components of a representation easily and to distinguish them from each other is valuable in providing the novice reader with confidence that he or she will be able to understand the representation. First impressions can have a significant effect on the way in which a novice reader approaches the tasks of searching the representation, recognising relevant information and validating the model. A textual representation which is presented in a single block is much more off-putting than the same text split into separate sections according to the different topics covered. Among mathematical notations, the schema boxes in Z help to increase visibility by physically packaging together associated parts of the representation. Graphical notations, such as entity-relationship or data flow diagrams, appear to offer the greatest potential for visibility, but frequently produce diagrams which are a cluttered mish-mash of icons and symbols, because not enough care has been taken to emphasise visibility of components in the representation. One of the most useful aspects of the cognitive dimensions approach to evaluation of notations is that it highlights the trade-offs that have to be made between dimensions in different situations. Visibility of components

is often a casualty in cases where priority is given to ensuring that the representation carries as much information as possible.

5.1.5 Redundant recoding

A cognitive dimension which is related to visibility is that of redundant recoding: the ability to express information in a representation in more than one way. Examples of redundant recoding in text-based representations include headings, font size, and use of upper and lower case. Colour or the shape of the representation itself may be used in graphical or textual representations to highlight separate parts of a system or to emphasise aspects of its functionality. In notations such as structured English and decision trees redundant recoding is provided by indentation and layout. Figure 4 below shows two versions of the same process description using structured English and a decision tree. The decisions to be taken are conveyed by both the content and the layout of the representations.

```
if not local Customer (*lives outside 20 miles radius*)
        charge P&P
else (*local customer*)
        if TotalCost (* of Order*) < £30 then
            set DeliveryCharge
        else if TotalCost (* of Order*) > =£30 then
            if Distance < = 5 miles then
                no DeliveryCharge
            else (*Distance > 5 miles*) then
                if CustomerTradeRecord > = 1 year then
                    no DeliveryCharge
                else (*CustomerTradeRecord < 1 year*)
                    set DeliveryCharge
```

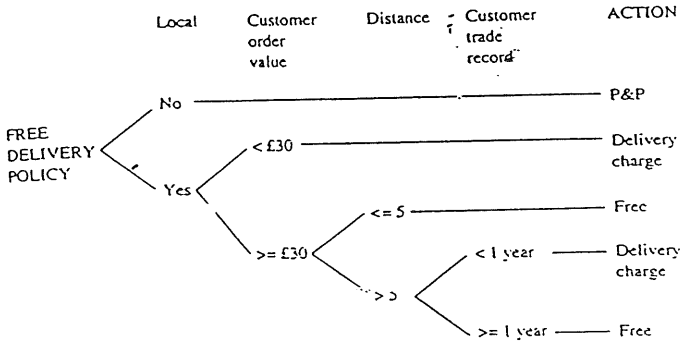


Figure 4: Process description using structured English and a decision tree

As with the cognitive dimension of visibility, graphical notations seem to have considerable potential for exploiting the concept of redundant recoding, but again this potential is no guarantee of a successful representation. The use of redundant recoding requires skill and creativity in order to make information more accessible without producing an over-crowded representation or misleading cues.

5.1.6 Abstraction gradient

One of the problems in designing representations to exploit the cognitive dimensions of visibility and redundant recoding is the risk of over-crowding; the cognitive dimension which addresses this question is the abstraction gradient: the availability and types of abstraction and structuring mechanisms provided by a notation. Reading representations of any size involves the activities of decomposition (to split the representation into manageable chunks) and abstraction (to identify the most important

features) as described in section 4. A notation which scores highly on the cognitive dimension of abstraction gradient is one which encourages developers to produce representations with clearly visible structures where the reader does not have to expend effort in working out how the representation should be split into sections or how to abstract the most important information from it. The amount of structure in a notation is discussed in section 5.2.5.

5.1.7 Consistency

A consistent notation is one where similar information is expressed in similar ways. This not only simplifies representations produced using the notation, but means that novice readers can make reasonable guesses at symbols which they have not seen before. One example of consistency is the use of different types of arrow to represent different types of function in the specification language Z.

5.1.8 Role expressiveness

In notations which score well on the dimension of role expressiveness the reader can readily infer the purpose of individual components in a representation. Closeness of mapping between notation and problem domain (see section 5.1.1 above) provides support for role expressiveness since the reader will recognise the purposes of different components of the representation based on his or her own experience of the problem domain. As already mentioned in 5.1.1, there is rarely a direct mapping between symbols used in notations for software development and the domain elements they represent. This means that role expressiveness depends largely on how names and labels are used in representations. In a data flow diagram, for example, the name given to a process or data store is crucial in conveying the purpose of that process or store to the reader. In a Z specification the purpose of a schema is encapsulated in its name, so it is important that care is taken to make sure this is meaningful.

Cognitive dimensions permit a broad-brush evaluation of a wide range of information structures. They are general principles, triggers for thought rather than rigid guidelines. Using the framework provided by cognitive dimensions, as discussed above, we now investigate the possibility of evaluating or predicting the understandability of representations on the basis of objectively measurable properties of the notations in which they are written. Some work has already been carried out on investigating ways in which the cognitive dimensions of repetitive and knock-on viscosity may be measured (Roast 97). In the section below we identify specific properties relating to ease of understanding that may, one day, be amenable to some form of objective measurement and thus provide a sounder basis for the evaluation of modelling notations.

5.2 Properties of notations which contribute to ease of understanding

Building on the work of authors such as Green (80, 89, 91a and b, 96), Petre (95) and Sampson (85) we have identified the following properties that may contribute to ease of understanding of representations:

1 The number of different symbols in the notation
2 The degree of motivation of the symbols
3 The discriminability of the symbols
4 The extent to which the notation is perceptual or symbolic
5 The amount or structure inherent in the notation

In the sections below each of these properties is described and an assessment is made of which cognitive dimension or dimensions it relates to, the extent to which it can be

measured, and how reliable an indicator it is of whether a notation will be easy for novice readers to understand.

5.2.1 The number of different symbols in the notation

One obvious property of notations which can be measured is the number of symbols they contain. Green (80) makes the point that programming languages with a large number of symbols and features are more difficult to learn and understand than languages with fewer features. If we apply this to modelling notations, it follows that a notation such as storyboard with 2 symbols, is apparently going to be easier for novices to understand than notations such as the Z specification language, which has 76 symbols (Myers et al 96). It is not surprising then that storyboards are widely used in the development of, for example, interactive multimedia systems, where not only the users, but also members of the development team, are likely to be unfamiliar with software modelling notations (Admiral Training 95, Britton et al 96). Table 1, below, shows the huge difference in the numbers of symbols found in modelling notations.

Notation	Number of entries in key or glossary	References and comments
DFD	4	Fertuck 1992
Structure Charts	5	Fertuck 1992
ERD (Chen)	5	Fertuck 1992
State Transition diagrams	4	Fertuck 1992
Z	76	Spivey 1987
1st order Predicate logic	12	Turski and Maibaum 1987
Data Dictionary	8	Yourdon 1989
Storyboard	2	Box and line

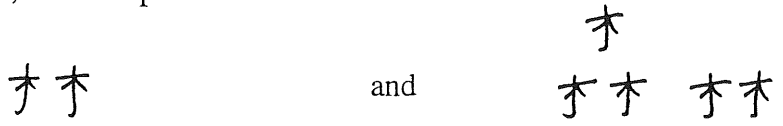
Table 1: Number of symbols in notations

Although superficially, it might seem that notations with fewer symbols are automatically easier for novice readers, this is too simplistic a view of the situation. The number of symbols in a notation is a crude measure. Consideration should also be given to the amount of information carried by each symbol; for example the function arrows in Z carry considerably more information than the arrows in a data flow diagram. Z and other mathematical notations are difficult for untrained readers to understand not only because they have large numbers of different symbols, but also because many of the symbols represent complex operations and concepts. Developers of systems also need to be aware of the trade-offs between a small and large number of different symbols in a notation. A small number of symbols will be easier for novices to grasp initially, but the restricted vocabulary of the notation may hamper their understanding in the long term. There is inevitably a trade-off between notations with few symbols and constructs, which may produce diffuse and disorganised representations, and notations where a larger number of symbols and constructs encourages representations which are well-organised and therefore more accessible to novice readers (Green 83).

The measure of the number of symbols in a notations has an impact on two of the cognitive dimensions described in section 5.1: diffuseness/terseness and the abstraction gradient, but does not give us a direct indication of understandability, due to the complicated trade-offs described above.

5.2.2 The degree of motivation of the symbols

This property is related to the cognitive dimensions of closeness of mapping (section 5.1.1) and role expressiveness (section 5.1.8). The concept of motivated and arbitrary notations has been described by Sampson (85). A notation may be considered to be motivated if there exists a natural relationship between the elements of the notation and objects or ideas that they represent. Many of the characters in Chinese script are motivated, for example:



meaning tree and meaning forest

Not only does the form of the symbols suggest trees, but the symbols also preserve the idea that a forest is made up of many trees.

In an arbitrary notation there is no natural relationship between the object and the representation; the symbol *, for example, bears no obvious relation to the notion of multiplication and the logic symbol ¬does not suggest negation.

As we noted already in section 5.1.1, disciplines where the artefact being modelled exists in physical form , such as a bridge or a building in civil engineering, are more likely than software development to have notations with motivated symbols, such as those used in the drawings of an architect or structural engineer. The problem in software systems development is that normally the effects and interface of the artefact are perceived rather than the thing itself. This makes it difficult to provide motivated symbols for use in representations: a shape can represent a room and a curve a bridge, but what realistically represents a process or an interaction?

Modelling notations for software systems tend to be arbitrary, although some diagrammatic notations do include motivated symbols, such as a line representing a link in entity-relationship diagrams, and an arrow signifying direction in data flow diagrams. Inclusive (such as Venn) diagrams and connected (such as state transition) diagrams may each be considered as motivated in certain situations. For example, a diagrammatic-inclusive notation can represent the positions of different elements on a screen, and a diagrammatic-connected notation may be effective for specifying possible routes through a system.

Motivation of symbols in a notations could be measured indirectly by means of experiments, such as those used in choosing icons, where subjects are asked to identify the images which map most closely onto concepts of interest. Although this would involve a considerable amount of effort, it does seem as though information obtained from such experiments would be useful in evaluating individual notations in terms of ease of understanding for novice readers. Not only does this property of notations relate to two of the cognitive dimensions that we consider to be relevant to understandability, but the situation is not confused by complicated trade-offs of the kind discussed in section 5.2.1 and below.

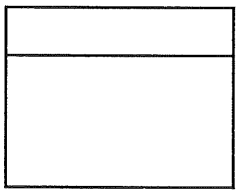
5.2.3 The discriminability of the symbols

Discriminability refers to the ease with which different symbols in a notation can be distinguished from each other; this depends largely on how physically distinct each symbol is from others in the notation. Discriminability has been identified by both Green (80) and Sampson (85) as an indicator of how easy an unknown notation will be for novice readers to understand. As an example, we can consider the Z symbols shown below:

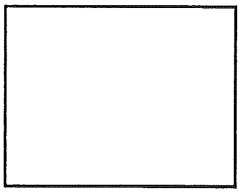
$\nrightarrow$ partial function

$\mapsto$ maplet

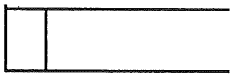
To someone who is not familiar with the Z notation the two symbols look very similar, yet they have completely different meanings. They are not easy to distinguish from each other and so are a potential cause of confusion for novices trying to understand a specification written in Z. Discriminability of symbols is a problem with many mathematical languages. In contrast, diagrammatic notations, such as entity-relationship and data flow diagrams, generally have symbols, such as lines, arrows and boxes, that are clearly distinguishable from each other. However, confusion may arise when someone who is unfamiliar with the notations has to look at more than one type of diagram. An untrained reader may well find that the symbol for a process in a data flow diagram is not easy to distinguish from the symbol for a data store or the symbol for an entity in the E-R diagram (see Figure 5 below).



Process in a DFD



Entity in an E-R diagram



Data store in a DFD

Figure 5: Symbols for process, data store and entity.

Different variations of standard notations, such as Yourdon and SSADM for data flow diagrams, may also bewilder novice readers.

The level of discriminability in a specific representation can be increased by use of techniques such as different sizes, fonts and use of colour, although these are typographic devices, rather than an intrinsic part of the notation itself. Figure 6, below, illustrates how even simple shading can help to distinguish data stores from processes in a data flow diagram; use of colour for shading or outline would be even more effective (Tufte 83) . This is an example of the use of redundant recoding (see section 5.1.5) to make the different components of the representation more visible and easier to distinguish from each other.

As with the property of motivation of symbols, it would be feasible to measure discriminability through experiments. These might involve situations where subjects are timed in picking out a particular symbol from amongst a collection of others. The property of discriminability is related to the cognitive dimension of visibility (section 5.1.4) and does not involve any complex trade-offs; therefore it appears to be a useful indicator of ease of understanding of a notation.

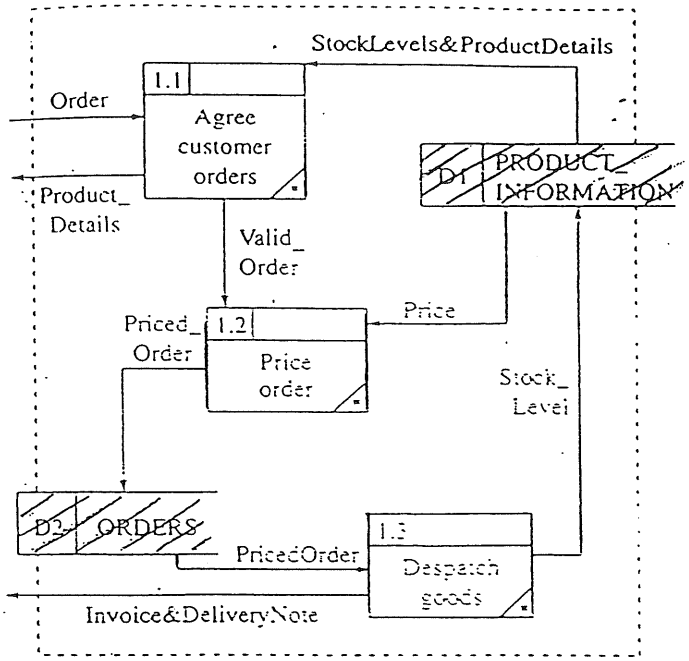


Figure 6: Use of shading to distinguish symbols in a data flow diagram

5.2.4 The extent to which the notation is perceptual or symbolic

Perceptual representations are those in which we perceive meaning directly without having to reason about them, for example the use of colour in electricity cables, or diagrams in various contexts, such as road signs. Most modelling notations contain both perceptual and symbolic elements. Graphs are perceptual in that they are diagrammatic, although they often contain labels (symbols) for the various nodes. Notations which are mainly symbolic rather then perceptual are frequently laid out to show certain aspects perceptually: Z's use of schemas, formatting of pseudocode and introduction of 'white space' to aid comprehension are examples of this. Text-based notations, which are symbolic, generally use perceptual devices to aid comprehension; these may include section headings, paragraphs, variations in size and font, bullets, emphasis and white space.

Perceptual and symbolic notations each have advantages and disadvantages in modelling. Representations in symbolic notations, such as mathematical specification languages, are able to hold much more information than perceptual representations, but are frequently beyond the understanding of untrained readers. Representations in diagrammatic notation often embody perceptual concepts, such as connectedness (as used in state transition diagrams) and inclusiveness (as used in Venn diagrams), which most people understand. However, without elaborate 'extras' the amount of information they can contain is restricted. A further problem with perceptual notations is identified by Green et al. (91b). Perceptual cues can be provided in a diagram by placing functionally-related items close together in order to emphasise the relationship; however, this can lead to diagrams which are cluttered and difficult to understand. There is also the risk with untrained readers that any adjacent items in a diagram will be seen as related, even when this is not the case.

Although we cannot currently measure the extent to which a notation is perceptual or symbolic, we present below in table 2 a tentative classification of notations according to their perceptual and symbolic content.

	perceptual		symbolic
	connected	inclusion	
DFD	yes	no	yes (labels)
STRUCTURE CHARTS	yes	no	yes (labels)
ERD- Chen	yes	no	yes (labels)
State Transition diagrams	yes	no	yes (labels)
Z	no	yes	yes
Predicate logic	no	no	yes
Data dictionary	no	no	yes
Story Board	yes	no	yes (labels)

Table 2: Perceptual/symbolic classification

The question of whether a notation is mainly perceptual or symbolic relates to the cognitive dimensions of redundant recoding (section 5.1.5) and diffuseness/terseness (section 5.1.3). However, the difficulty of assessing a notation in these terms and the complicated trade-offs involved mean that this is not a particularly reliable indicator of ease of understanding of a notation.

5.2.5 The amount of structure inherent in the notation

As discussed in section 4.2, finding, decomposing and abstracting information have been identified as key activities in reading a representation (Sengler 83, Larkin and Simon 87). Finding information involves searching the representation; this process will be easier for untrained readers to carry out if the notation used encourages a clear structure in the representation. Decomposition and abstraction are important because a reader can only cope with a small amount of information at a given time. An effective notation should encourage decomposition of the representation into sections or 'chunks', each of which is intellectually manageable by the reader. If the structure of decomposition is not clear, then the reader will be forced to devise his or her own decomposition of the representation, which may well be different from that which the developer had in mind.

Among software modelling notations, the context diagram and imposed hierarchical decomposition of data flow diagrams mean that it is relatively straightforward for untrained readers to find information at different levels and to concentrate on only a small part of the representation at any given time. Mathematical notations, such as Z, often provide structure at low levels - for example, separation of the specification into schemas can help to identify normal and error cases of an operation - but no mechanism is provided to give an overview of the system as a whole. This adds to the difficulties of novice readers of Z who wish to get a feel for the overall system before examining the details of the representation.

Although diagrammatic notations are often thought to encourage more explicitly structured representations then text-based notations, this is not always the case. In an experiment comparing textual and visual programming languages, Green and Blackwell (96) found that readers' understanding of programs depended more on the structure of the information in the program than on whether the program was written in a visual or text-based language. Green (83) also makes the point that the structure is only useful if

it is clearly visible; experiments have shown that simple structures are generally more effective than those that try to be 'natural' and mirror the way in which the reader thinks about the problem.

The amount of structure inherent in a notation is directly related to the cognitive dimension of abstraction gradient (section 5.1.6). It is difficult to see how notations could currently be measured, or even classified, according to the amount and types of structure they offer. However, if suitable measures were to be discovered, the possibility of evaluating notations in terms of structure would be extremely helpful in assessing how easy representations written in those notations would be for novice readers to understand.

6 Discussion

In order to make an informed choice of modelling notation, a developer needs first to identify the criteria for effective notations which are relevant in a particular situation. The second stage of the developer's choice consists of assessing available notations in the light of the criteria which are considered important. In the development of interactive systems, an important criterion will always be whether or not a notation facilitates the production of representations which are easy to understand. The aim of this paper has been to identify properties of notations that help to make representations produced using them easier to understand for untrained readers. We have also discussed the feasibility of evaluating ease of understanding through objective measures, in order to provide a sounder basis for choice of modelling notation.

For developers faced with a raft of modelling notations to choose from, many of which have been largely untried on 'real' problems, it is helpful to have some idea of whether a particular notation is likely to produce a representation which untrained readers can readily understand. Each of the properties identified in section 5.2 is, on its own, only a crude indicator of the overall ease of understanding of a notation, but taken together they may give some idea of how quickly and well a novice reader will be able to understand a representation developed in a particular notation. However, the assessment of notations in terms of ease of understanding is complicated by a number of issues.

Although there may be a theoretical case for suggesting that notations with certain properties will support ease of understanding, this can only be proved in practice through by empirical experiments involving novice readers attempting to extract meaning from representations. The situation is complicated by the fact that the effectiveness of a particular representation is influenced not only be the notation used, but by a number of different factors, including the nature and complexity of the problem to be modelled, the expertise and experience of the modeller, the purpose for which the representation is built, and the tools used to implement the notations. The number and variety of variables make it very difficult to compare different representations. A further problem in evaluating notations arises from the fact that most system developments require more than one modelling notation to capture all relevant information; this gives rise to added complications of consistency between the different notations and ways in which they can best be combined.

In addressing the question of ease of understanding of notations, this paper has suggested a possible role for objective measurement as a way of providing a sounder basis for decisions and choices made. Each of the properties of notations identified has been assessed in terms of whether it is currently amenable to direct measures, indirect measures, or whether we do not see any feasible way of measuring it. We have not found measures for properties corresponding to all of Green's cognitive dimensions; in the case of some of the dimensions we feel that advances in the science of measurement may one day produce useful metrics; however, some of the dimensions,

such as hidden dependencies and role expressiveness, are not simply concerned with the notation, but with the relation between the notation and the problem domain. Measures of notations alone will shed very little light on how well a particular notation scores on these dimensions. Even if it were possible to obtain meaningful measures and to locate different notations on the various cognitive dimensions, we have no obvious way of calculating the overall ease of understanding of a notation as a function of this, and in specific development situations we have no way of measuring the extent to which a reader has actually understood a representation.

Overall, it appears that a reductionist approach to investigation of ease of understanding that depends on measurement is currently neither feasible nor adequate. For the time being we must rely largely on the experience of experts concerning which notations to use in which development situations, and on the human skills of developers to judge whether particular representations have been understood.

In spite of all these caveats, we feel that it is useful to continue work in this field for three reasons. First, there is no doubt that the choice of modelling notation is extremely important for successful development of interactive systems and that ease of understanding by novice readers is an essential feature of a notation. Second, we do not know what measures will be possible in the future; rapid advances in measurement are being made in a wide variety of areas, such as acoustics and neurology, and who could have imagined two hundred years ago, that a child's fever, the heat of the day and chilled food could all be measured on the same scale? Finally, the process of attempting to measure something is valuable in itself, in that it forces the would-be measurer to understand and articulate exactly what is being measured. In the words of Lord Kelvin (1989) "When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it in numbers, your knowledge is of a meagre and unsatisfactory kind". We may never be able to evaluate understandability of notations in a precise and objective way, but we feel that working towards this ultimate goal will lead to a fuller appreciation, both of notations themselves and of whether representations produced using them will be readily understood by novice readers.

Acknowledgements

The authors would like to thank the organisers and participants in the International Workshop on Representations in Interactive Software Development (July, 1997) for their helpful and constructive comments on an earlier version of this paper.

References

Admiral Training Ltd (1995) *Multimedia Design - A Newcomer's Guide* Department of Employment, Sheffield.

Blackburn, S. (1984) *Spreading the Word: Groundings in the Philosophy of Language* Oxford University Press

Britton, C. and Doake, J. (1996) *Software System development. A gentle introduction.* McGraw-Hill

Britton, C., Jones,S., Myers,M. and Sharif,M.(1997) "A Survey of Current Practice in Multimedia System Development", to appear in Information and Software Technology

Brun, P. and Beaudouin-Lafon, M. (1995) "A Taxonomy and Evaluation of Formalisms for the Specification of Interactive Systems" in *People and Computers X, Proceedings of HCI'95,* M. Kirby, A.Dix and J. Finlay (Eds.), C.U.P.

Christel, N. and Kang, K. (1992) "Issues in Requirements Elicitation", Technical Report CMU/SEI-92-TR-12, Software Engineering Institute.

Davis, A.M. (1993) *Software Requirements. Objects, Functions and States* Prentice Hall International.

Department for Trade and Industry and National Computing Centre (1987) *The STARTS Guide,* second edition, volume 1, NCC Publications

Farbey, B. (1993) "Software quality metrics: considerations about requirements and requirement specifications" in *Software Engineering: A European Perspective* R. Thayer and A. McGetterick (Eds.), IEEE C.S.Press

Fertuck L. (1992) *Systems Analysis and Design*, WCB

Fitter, M.J. and Green,T.R.G. (1981) "When do diagrams make good computer languages?" in Alty,J. and Coombs,M. (Eds.) *Computing Skills and the User Interface* Academic Press.

Garzotto,F., Mainetti,L. Paolini,P. (1995) "Hypermedia Design, Analysis and Evaluation Issues" Communications of the ACM Vol 38, No.8

Green, T.R.G. (1983) "Learning Big and Little Programming languages" in Wilkinson, A.C. *Classroom Computers and cognitive Science,* Academic Press, New York

Green,T.R.G. (1980) "Programming as a Cognitive Activity " in Smith, H.T. and Green, T.R.G. (Eds.) *Human Interaction with Computers* Academic Press.

Green, T.R.G. (1989) "Cognitive Dimensions of Notations" in *People and Computers (HCI 89)* Sutcliffe and McCauley (Eds.) CUP.

Green, T.R.G. (1991a) "Describing Information Artifacts with Cognitive Dimensions and Structure Maps" in *People and Computers VI, Proceedings of the HCI'91 Conference*, Diaper, D. and Hammond, N. (Eds.), August 1991

Green,T.R.G., Petre,M. and Bellamy,R. (1991b) "Comprehensibility of visual and textual programs: A Test of Superlativism against the "Match Mismatch" Conjecture" in Koenemann- Belliveau,J., Moher,T. and Robertson,S. (Eds.) *Empirical Studies of Programmers* Fourth Workshop, Norwood NJ, Ablex pp121-146

Green, T.R.G. and Blackwell, A.F. (1996) "Thinking about Visual Programs " in *Thinking with Diagrams* IEE Colloquium Digest No: 96/010

Jones, S., Britton, C. and Lam, W. (1997) "Towards A Framework for Selecting Notations for Modelling Requirements" Technical Report number 303, Department of Computer Science, University of Hertfordshire, Hatfield, AL10 9AB

Kitchenham, B. (1994) *DESMET Guidelines for Evaluation Method Selection* DESMET Workpackage 2 (DES/WP2.2/7), NCC Services Ltd.

Larkin, J.H. and Simon, H.A. (1987) "Why a Diagram is (Sometimes) Worth ten Thousand Words", in *Cognitive Science*, 11, pp 65-99

Macaulay, L.A. (1996) "Requirements for Requirements Engineering Techniques", Proceedings of ICRE96, the Second International Conference on Requirements Engineering, IEEE Computer Society Press

Maguire, M. (1997) "RESPECT User Requirements Framework Handbook" Version 2.2, HUSAT Research Institute.

McCluskey, T.L., Porteous, J.M., Naik, Y, Taylor, C.N. and Jones, S. (1995) *A Requirements Capture Method and its use in an Air traffic Control Application.* Software practice and Experience, Vol 25 (1)

Modugno, F., Green T.R.G. and Myers B.A. (1994) "Visual programming in a Visual Domain: A Case Study of Cognitive Dimensions" in *People and Computers IX, Proceedings of HCI'94*, Glasgow, August 1994

Myers, M., Britton,C., Jones,S.and Sharif,M. (1996) "An investigation into the measurement of modelling notations" Technical Report No. 257, Faculty of Information Sciences, University of Hertfordshire, to be presented at The Eighth European Software Control and Metrics Conference, Berlin, May 1997

O'Neill, E., Johnson, P. and Johnson, H. (1997) "Representations in co-operative software development: an initial framework", Proceedings of the International Workshop on Representations in Software Development, 2-4 July 1997, Queen Mary and Westfield College, University of London

Petre, M.(1995) "Why Looking Isn't Always Seeing, Readership Skills and Graphical Programming" Communications of the ACM, June 1995, Vol 38, No.6

Roast, C.R. (1997) "Formally Comparing and Informing Notation Design" in *People and Computers XII, Proceedings of HCI'97,* H.Thimblely, B. O'Conaill and P. Thomas (eds.), Springer.

Sampson,G. (1985)*Writing Systems.* Hutchinson, 1985

Sengler, H.E.(1983) "A Model of Program Undersanding" in Green, T.R.G., Payne, S.J. and van der Veer, G.C.*The Psychology of Computer Use* Academic Press, 1983

Sommerville, I. and Sawyer, P. (1997) "Requirements engineering: a good practice guide", Wiley.

Spivey J M. (1987) *The Z Notation A Reference Manual*, Prentice Hall

Sutcliffe, A.G., Maiden N.A.M. and Bright, B. (1997) "An Evaluation Framework for Representations in Requirements Engineering", Proceedings of the International Workshop on Representations in Software Development, 2-4 July 1997, Queen Mary and Westfield College, University of London

Tufte, Edward (1983) *The Visual Display of Quantitative Information* Graphics Press, Cheshire, Connecticut.

Turski, W.M. and Maibaum, T.S.E. (1987) *The Specification of Computer Programs* Addison-Wesley

Yourdon E. (1989) *Modern Structured Analysis*, Prentice Hall