Cognitive Dimensions – An Experience Report

Maria Kutar, Carol Britton, Jonathan Wilson
University of Hertfordshire
College Lane, Hatfield, Herts, AL10 9AB
Email: M.S.1.Kutar, C.Britton, j.1.wilson@herts.ac.uk

Keywords: POP-III.C. Cognitive Dimensions, POP-I.B. Choice of Language

Abstract

The selection of appropriate notations for the specification of interactive systems is an important factor in successful system development. This creates a need for evaluation techniques which allow us to assess a notation's suitability for use in a particular setting. The cognitive dimensions framework provides an analytical tool for the evaluation of information artefacts, including specification languages. This paper presents our research into the evaluation of a real-time temporal logic, $TRIO_{\neq}$ using the cognitive dimensions framework. We show the results of the cognitive dimensions analysis, and discuss our experience of using the framework. The research suggests that the framework may be used not only to create a usability profile for a notation, but can also assist with determining whether a notation has appropriate semantics for use in a particular situation.

1 Introduction

It is generally accepted that the choice of notation used in the development of software systems can be a key factor in successful system development. (Green 1989, 1991; McCluskey et al. 1995) The ever increasing number of notations which may be used in the specification of software systems provides us with a wide variety of notations to choose from, but there is little guidance to assist us with choosing a notation, nor with evaluating that notation. Our research addresses the specification of multiple granularities of time (such as seconds, hours, days and so on), in interactive systems. The nature of the material to be specified restricts our choice of notation to those which incorporate appropriate semantics for the representation of time constraints. Having found a notation developed in the field of real-time systems which appeared to be suitable, $TRIO_{\neq}$, we wanted to assess its suitability for use with interactive systems. The approach we took was to use the notation to specify an interactive case study, and then to evaluate the notation. The Cognitive Dimensions framework (Green 1989, Green and Blackwell 1998) was selected for the evaluation as it has been used in the past with some success and appears to offer a pragmatic and analytical approach to the evaluation of information structures.

The cognitive dimensions framework allows a usability profile to be created, evaluating a notations suitability for use in a particular situation. Thus, our application of the case study would allow us to examine the suitability of the semantics for representing time constraints, and the cognitive dimensions evaluation would allow us to assess other aspects of the notation, those pertaining to usability. When using the framework however, it became apparent that some of the dimensions could be used to help determine whether the notation contained the appropriate semantics, as well as evaluating usability aspects of the notation. In addition we found that the application of the cognitive dimensions framework forced consideration of a number of issues which affect the way in which we produce specifications.

In the rest of this paper we discuss the reasons for our choice of notation (section 2), and present our initial findings when producing the specification (section 3). We then go on to consider approaches to the evaluation of languages, including an introduction to the cognitive dimensions framework (section 4), and present the results of the cognitive dimensions analysis of the specification (section 5). In section 6 we summarise our findings and consider directions for future work.

2 Specifying Time Constraints in Interactive Systems

The research reported in this paper investigates the specification of time in interactive systems. We are concerned with those systems where time occurs at a number of different time granularities, encompassing, for example, requirements which relate to time periods ranging from milliseconds to weeks or months. It is recognised that time plays an important role in interactive systems and temporal properties of interactive systems have two main areas of relevance. Firstly, there may be temporal functional requirements which relate to system correctness, and in the second case, temporal properties of interaction can have an important bearing on usability issues. Temporal issues arise in interactive systems at a wide range of time granularities, from those at small scales such as what constitutes a double mouse-click, to larger granularities where we may be concerned with the rate of turnaround of messages over hours, days or even weeks (Dix et al. 1998). The growth in interactive systems has resulted in increasing complexity with relation to time, and it is not uncommon for a number of granularities to be of importance in a single system. For example, in an e-commerce system we might be concerned with very small scale time constraints relating to system behaviour as the user browses the site, and also with turnaround times of minutes or hours for email messages. In addition we would need to consider periods of days or weeks for goods to be delivered, and much longer time periods of many years relating to the customer and stock databases.

The problem of incorporating a number of time granularities into a single specification has been addressed in the field of real-time systems, and a notation, TRIO$_{\neq}$ devised (Montanari 1996), which allows differing time granularities to be included within a single specification. TRIO$_{\neq}$ is a fragment of first-order logic for specifying temporal properties of systems. In TRIO≠ one can quantify over times of varying granularity and constrain the occurrence of system events with respect to these, using operators such as Futr (in the future), Past (in the past), and so on.

TRIO$_{\neq}$ was selected for the research described in this paper as it allows for the specification of multiple time granularities. This property was considered essential for our research for the obvious reason that, regardless of any usability criteria, a notation must have appropriate semantics to be used in a given situation. This distinction is well expressed by Mackinlay (Mackinlay 1986) who defines the criteria as expressiveness and effectiveness. Expressiveness criteria determine whether a notation is capable of expressing the right information. He states that “A set of facts is expressible in a language if it contains a sentence that (1) encodes all the facts in the set and (2) encodes only the facts in the set.” A suitably expressive notation must therefore be able to express exactly the desired information, nothing more and nothing less. We may regard expressiveness as being analogous to coverage of a notation.

Effectiveness criteria, on the other hand relate to how well the notation expresses the information. Effectiveness is defined (in the context of automated graphical representations) as “whether a language exploits the capabilities of the output medium and the human visual system”. In short we are asking the questions ‘can the notation say what I want it to say?’ (expressiveness) and ‘how well can it do it?’ (effectiveness). Common sense dictates that these criteria must be considered in order – there is no benefit in using an eloquent and usable notation if it is not possible to express the required information in that notation. Consequently, as stated above, our choice of notation for this research was based on its perceived expressiveness.

In order to assess the notation’s suitability for use with interactive systems, we have first used it for the specification of an interactive case study which contains a number of time constraints at varying time granularities. There is no specific framework which allows expressiveness to be assessed, but the application of a notation to a challenging case study allows us to examine whether its semantics provide the expressive power required. The case study used was an interactive pet fish, the behaviour of which is determined by the treatment that it receives. It must be fed and played with regularly in order to thrive. Less attentive treatment results in a neurotic and sulky fish, whilst more serious neglect of its feeding requirements results in death.

3 Producing the Specification with $\text{TRIO}_{\neq}$

As discussed above, the $\text{TRIO}_{\neq}$ notation was selected for its ability to express multiple time granularities. Whilst it is technically possible to specify all systems at a single granularity (usually the finest), this raises a number of difficulties. The first is simply that any notion of naturalness is lost, making understanding hard. Normally, we use the most appropriate granularity for a particular purpose, and using a different one can cause a great deal of difficulty. For example, it is possible to describe a day accurately as being 1440 (24x60) minutes, but if we were to talk about an event being 2880 minutes away, it would take a great deal of thought for most of us to understand that the event is two days away. A second difficulty is that temporal statements often carry a great deal of implicit information. We have no difficulty, for example in attaching temporal meaning to the two statements below, nor even with rephrasing them with respect to a finer granularity:

Every year I have a birthday.

There is one day in each year which is my birthday.

Every day I eat.

There are a number of minutes in each day during which it is true that I am eating.

The implicit information in the two sentences allows us to move to a finer granularity without difficulty, and therefore when formalising temporal statements we must be very careful to ensure that this is captured, particularly when moving between different granularities. Finally, we must be careful to ensure that when moving between the different granularities we don't alter the intended meaning of statements. For example if I make a promise to telephone someone a week from today I do not really mean that I will call them in exactly 168 hours – some flexibility would be assumed to be normal, in this case perhaps twelve hours each way.

The examples above illustrate that the ability to specify multiple time granularities in a specification is a valuable feature, allowing time to be represented in a natural way. One difficulty that we encountered when using $\text{TRIO}_{\neq}$ to specify the interactive fish, was that there are times when the notation can become cumbersome, and it is sometimes still necessary to specify at a flat granularity. For example, one aspect of the behaviour of the fish is that if it is fed at the same time each day for seven days, it will begin to wait for its food at the regular feeding time. In order to allow some flexibility a ten-minute feeding window is given, allowing five minutes either side of the initial feeding time. The specification of this aspect requires that we refer both to each of the seven previous days, and to the time to the nearest minute at which feeding occurred on each of those days. Consequently all temporal aspects of this part of the specification are referred to in minutes:

$\text{Regular\_feeding} \leftrightarrow \exists\, x_0 \ldots x_7$

$\quad 1440 \text{ x } 7 > x_0 \geq 1440 \text{ x } 6$

$\wedge\ 1440 \text{ x } 6 > x_1 \geq 1440 \text{ x } 5$

$\wedge\ 1440 \text{ x } 5 > x_2 \geq 1440 \text{ x } 4$

$\wedge\ 1440 \text{ x } 4 > x_3 \geq 1440 \text{ x } 3$

$\wedge\ 1440 \text{ x } 3 > x_4 \geq 1440 \text{ x } 2$

$\wedge\ 1440 \text{ x } 2 > x_5 \geq 1440 \text{ x } 1$

$\wedge\ 1440 \text{ x } 1 > x_6 \geq 1440 \text{ x } 0$

$\wedge\ [\text{Past, Eats, } x_0]_m \wedge \ldots . \wedge [\text{Past, Eats, } x_6]_m$

$\wedge\ |\, x_0 \ (\text{mod } 1440) - x_1 \ (\text{mod } 1440)\, | \leq 5 \wedge \ldots \wedge |\, x_0 \ (\text{mod } 1440) - x_6 \ (\text{mod } 1440)\, | \leq 5$

This loquacious expression states that the fish ate on seven separate days in the past week, and that the time at which this occurred was within the same ten minute time period on each day. We then need to show that, having been fed at a regular time for the past seven days, the fish will wait for it's food at the appropriate time:

$\text{Wait\_for\_food} \leftrightarrow \text{Regular\_feeding} \wedge \exists x_0 (1440 \text{ x } 7 > x_0 \geq 1440 \text{ x } 6 \wedge [\text{Past, Eats, } x_0]_m \wedge (x_0 \text{ (mod } 1440) \leq 5 \vee x_0 \text{ (mod } 1440) \geq 1435))$

Using Mackinlay's definition of expressiveness, it is clear that it is possible to express this requirement in TRIO$_{\neq}$, as there is a sentence in the language that encodes all of the required information, and no more. However, the use of a single granularity means that the statement is cumbersome and difficult to follow. This is an effectiveness issue which could be overcome if there was a concept of regularity within the notation, allowing us to express statements such as 'every day'. Overall however, we were able to express all of the temporal requirements contained within our case study using TRIO$_{\neq}$, although it should be noted that we were specifying only the temporal requirements of the case study. Having established that the notation was suitably expressive, consideration of effectiveness criteria could now be undertaken. This would allow us to examine in more detail issues relating to usability of the notation; in short, having determined that the notation allows us to say what we want to say, we could now assess how well it can do it. In the following section we identify approaches to the evaluation of specification languages and describe the cognitive dimensions framework.

4 Approaches to the Evaluation of Languages

It is generally accepted that the use of different languages for representation has an impact on the effectiveness with which a variety of tasks can be performed (Stenning & Oberlander, 1995). This is true especially in software system development, where the effect of the choice of language on successful system development has long been recognised (Green, 1989, 1991; Johnson, McCarthy & Wright, 1995; McCluskey et al., 1995; Modugno, Green & Myers, 1994; Roast, 1997). However, the relationship between languages, representations and the quality of the system development process is not fully understood. Little is currently known about what languages are likely to be most suitable for use in which contexts. The choice of languages for particular projects often reflects the experience or preferences of the development team more than an objective consideration of possible alternatives (McCluskey et al., 1995).

We believe that selection of notations would be more effective, were it influenced by consideration of whether a language is fit for a particular purpose, through an evaluation of that language. The difficulty of evaluating representations directly has been recognised by other authors. These include Stenning and Oberlander (1995) who note problems with an approach which emphasises "differences between token representations, rather than the differences of expressive power of the systems the tokens are drawn from". Scaife and Rogers (1996) also highlight the problem. In their survey of the literature on how graphical representations affect performance they note that it is difficult to draw general conclusions from the findings of specific studies, and that "It is often hard to separate general claims about graphical representations *per se* from factors that have to do with individual differences in ability in the subject or understanding of the domain-specific genre of the diagrams involved".

We found, in our search of the literature, only five evaluation approaches (Davis, 1988; Green, 1989; Brun & Beaudouin-Lafon, 1995; Haywood and Dart, 1996; von Knethen et al., 1998) that have been or could be applied to languages for software specification. However, none of them is well known in the software engineering community or appears to have been used by other authors. The work of Brun and Beaudouin-Lafon (1995) provides a way of evaluating formalisms for specifying interactive systems, but is concerned with formalisms that may be used to specify all aspects of interactive systems. It is not suitable for our purposes as we are concerned only with specification of the temporal aspects of interactive systems.

The framework of cognitive dimensions (Green, 1989, 1991) was developed for the analytical evaluation of information structures and has been used successfully in a number of cases, both by

Green himself (Green, 1991; Green, Petre & Bellamy, 1991; Modugno, Green, & Myers, 1994; Green & Blackwell, 1996), and by other authors (Shum, 1991; Petre, 1995; Roast, 1997; Yang et al., 1997). We have, ourselves, found cognitive dimensions to be useful and effective in evaluating two different mechanisms for structuring specifications written in Z (Britton, Jones & Lam, 1998). Cognitive dimensions are intended to support the evaluation of any type of information structure and can be applied to specification and programming languages, musical scores or even telephone numbers (Modugno, Green & Myers, 1994). The dimensions are not a set of criteria which may be satisfied to various degrees by different information structures; rather they are aspects of information structures which may be important and useful in specific situations. Green's work in this area has not, as yet, been theoretically validated (but see Blackwell & Green 1999), nor does it address the question of objective measures of cognitive dimensions. The framework does, however, provide a pragmatic and effective approach to considering information structures (in our case, specification languages).

4.1 Cognitive Dimensions – An Introduction

In this section we consider the concept and purpose of cognitive dimensions and give a brief overview of the individual dimensions. (For a full guide to cognitive dimensions the reader is directed to (Green & Blackwell 1998)). The aim of the cognitive dimensions framework is to provide tools which may be used to evaluate the usability of information structures. They are 'thinking tools' rather than strict guidelines, with a focus on usability aspects which make learning and doing hard for mental, rather than physical reasons. In Thomas Green's words, "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." (Green 1989). Cognitive dimensions are aimed at the non-HCI specialist and therefore comprise a broad-brush approach rather than detailed guidelines.

The cognitive dimensions framework may be applied to both interactive artefacts such as word processors, and non-interactive artefacts such as music notation, and programming or specification languages. An artefact may be analysed and a usability profile derived which can assist in determining the artefact's suitability for a particular task. It should be noted that the artefact is considered in conjunction with the environment in which it is to be used. We may think of the combination of an artefact and it's environment as a 'system', and it is this combination, the 'system', to which the analysis is applied. Consequently, a single specification language, for example, may be considered in a number of different environments, each 'system' resulting in a different usability profile. This is a key feature of cognitive dimensions as rather than providing a generalised analysis, they may be used to evaluate an artefact's suitability for a particular purpose. The ability to consider an information structure in the context in which it is used greatly enhances the usefulness of the dimensions.

There are fourteen dimensions in all, and they are shown in table 1 below with a brief description. Those that are the focus of this paper are considered in more detail in section five below, for a fuller discussion of the remainder see (Green 1998).

DIMENSION	**DESCRIPTION**
Abstraction	Types / availability of abstraction mechanisms
Hidden Dependencies	Important links between entities nit visible
Secondary Notation	Extra information in means other than formal syntax
Diffuseness	Verbosity of language
Premature Commitment	Constraints on the order of

	doing things
Viscosity	Resistance to change
Visibility	Ability to view components easily
Closeness of Mapping	Closeness of representations to domain
Consistency	Similar semantics are presented in a similar syntactic style
Error-Proneness	Notation invites mistakes
Hard Mental Operations	High demand on cognitive resources
Progressive Evaluation	Work-to-date can be checked at any time
Provisionality	Degree of Commitment to actions or marks
Role Expressiveness	The purpose of a component is readily inferred

Table 1 – The Cognitive Dimenisions

The relevance of each dimension is likely to vary according to the system being evaluated – sometimes a particular dimension may be considered to be of primary importance whilst at other times it may have little bearing on the evaluation. It is up to the user of the framework to assess this in relation to the system under consideration. It should also be noted that there are no satisfaction criteria for the dimension – in some cases for example, a language's diffuseness (verbosity) may be a positive aspect while in others it may be negative. Finally, the dimensions cannot be considered to be entirely orthogonal. Whilst they are to some extent independent, in that changes made to a given system in relation to a given dimension A, may not affect the system in relation to dimension B, it is likely that another dimension (or dimensions) will be affected by the change. Thus independence is only pairwise, and when considering changes to a system the relative trade-offs must be taken into account, with the resultant effect that compromises will often have to be made.

5 Cognitive Dimensions Analysis of the TRIO$_{\neq}$ Specification

Our intention during this research was to use cognitive dimensions to analyse the suitability of TRIO$_{\neq}$ to specify temporal requirements for interactive systems. The notation was chosen for its perceived ability to express temporal requirements at multiple time granularities, and it would become clear during specification of the case study whether the notation was suitably expressive (i.e. whether it was possible to express the desired information and no more). Therefore, by using the cognitive dimensions framework, we could analyse other aspects of the notation, concentrating on the usability aspects of the notation which relate to Mackinlay's effectiveness criterion. However, it became apparent during the cognitive dimensions analysis that some of the dimensions highlight factors which relate to both expressiveness and effectiveness. Below, we outline the cognitive dimensions analysis of TRIO$_{\neq}$, and highlight the dimensions which we believe relate to both expressiveness and effectiveness.

Abstraction

This dimension relates to the types and availability of abstraction mechanisms which the language allows. Different temporal granularities such as hours, minutes, days and so on, are abstraction mechanisms which we use in our everyday lives. As we have noted above, the ability to use differing granularities within a single specification, is an important feature, allowing greater temporal accuracy in specifications, as well as mirroring our natural understanding of time. We believe that it is a particular strength of TRIO$_{\neq}$ that this type of abstraction mechanism is available.

This dimension relates both to expressiveness and effectiveness – the number of abstraction mechanisms available will contribute both to expressive power of a notation and to other criteria such as understandability of expressions formulated using the notation. In TRIO$_{\neq}$ it is the abstraction mechanisms allowing varied time granularities to be considered within a single specification which contribute to its expressive power, making it suitable for the specification of those systems where time occurs at multiple granularities. For example, our specification included statements such as:

[NextTime, Assess_Behaviour, 1]$_{d}$

which states that the next occasion on which the behaviour of the fish will be assessed is one day from now, and

Click_Food $\rightarrow$ [Futr, Swim_to_Food, 3]$_{s}$

Which states that the fish will swim to its food three seconds after the food icon is clicked. The expressions 'NextTime' and 'Futr' are temporal operators, and may also be described as abstraction mechanisms. TRIO$_{\neq}$ includes a number of temporal operators which allow us to refer to past and future events, another aspect which contributes to the expressiveness of the notation – many real-time temporal logics (see for example, Alur and Henzinger 1989, Harel et al. 1990 and Ostroff 1992), contain only operators which refer to the future, restricting the type of temporal statements which may be expressed.

The example of regular feeding given in section three above, a part of our specification which occurs at a flat granularity, illustrates how understandability may be affected by abstraction mechanisms – it is very difficult for us to think of 1440 minutes as meaning one day, and in this case further abstraction mechanisms would contribute to the effectiveness of the notation. For example, a mechanism to allow us to refer to both a day and a time (in hours and minutes) within that day would have alleviated the problems which we encountered, making both production and comprehension of that part of the specification far easier.

Secondary Notation

This dimension also relates to both expressiveness and effectiveness criteria. It refers to the ability to provide information in means other than the formal syntax of a notation. In TRIO$_{\neq}$ there is no formal mechanism for secondary notation to be incorporated into a specification. It is certainly possible however, to use secondary notation in a way that contributes to the effectiveness of the specification produced. For example the biconditional operator $\leftrightarrow$, which represents necessity (it should be read as 'if and only if') is an important logical construct. It is not always easily distinguished from the implication operator $\rightarrow$ (which means 'if...then') despite the important difference in meaning. This could be emphasised through the use of a larger font size or bold for the biconditional operator, to highlight it's occurrence in a specification, which would draw attention to the difference between the two operators. Logical specifications may also be presented in a way that increases readability, through the careful use of layout and white space, using secondary notation to guide the reader through the specification.

In many other notations however, secondary notation may be used to add to the expressiveness of the notation (see for example Britton and Jones (1999) and Kutar et al (1998)). It is possible, for example to denote the ordering of events through the use of secondary notation, using layout or incorporating arrows to indicate succession and so on. This increases the expressive power of the notation without necessarily being incorporated into the syntax of the language.

Diffuseness

This dimension relates to the verbosity of a language. TRIO$_{\neq}$ is a fragment of propositional logic and consequently has a low level of diffuseness. This characteristic allows us to make very precise statements which have only one interpretation – they are not ambiguous in any way. In comparison natural language is very verbose, but lacks the precision of logically based languages. This relates to Mackinlay's notion of expressiveness where a notation is expressive if it is possible to express the desired information and no more than that. Natural language would fail to be suitably expressive as the ambiguity it contains means that more than just the desired information is frequently included.

For example, one requirement in our case study is that the fish must be fed every day in order to thrive. This raises an important issue regarding initialisation of the fish which is impacted upon by our interpretation of the phrase 'every day'. The phrase could be interpreted in two equally viable ways. Firstly, each day could be treated as a 24-hour time period from initialisation. In this instance, were the fish to be initialised at 3pm, a day would be considered to start at 3pm and end at 2-59pm. Alternatively, a day could be taken to initialise in tandem with the computer's clock, and run from midnight to 11-59pm. Here, the first day will be a short one. The choice made regarding initialisation could potentially affect the behaviour displayed by the fish. If the first option is used, and the fish is fed at, for example, 3-05pm on the day of initialisation, 2pm the next day, and 4pm on the day after, then it would effectively have not been fed on one day (as 26 hours elapse between the second and third feeding). Had the second option been taken, then feeding at these times would be counted as having happened every day. Although this allows periods of more than 24 hours to elapse between feedings, it is more in keeping with our understanding of 'daily feeding'. In TRIO$_{\neq}$ it is possible to specify either of the two options, but it is not possible to formulate a statement that contains the ambiguity of the English phrase 'every day'. Therefore it may be possible for the dimension of diffuseness to relate to both expressiveness and effectiveness, but it is less clear than with the dimensions above. Further research into this relationship using more notations is necessary for us to state with certainty that diffuseness may affect expressiveness.

Hidden Dependencies

This dimension asks whether important links between entities are visible. In the TRIO$_{\neq}$ example of regular feeding shown in section 3 feeding times are based on the time at which the fish was fed seven days ago. Over a period of weeks it is possible for the regular feeding time to drift so long as it is kept within the ten-minute time window, something which would affect the behaviour of the fish. For example if the fish were to be fed at 12 noon on the first day, and then at 12-05pm every day following this for six days regular feeding would be achieved. On the eighth day however, the ten minute time window would move, with 12-05pm becoming the centre point of that window. Over a period of weeks it would be possible for the time to move by a significant amount from the original time. This is not something which is immediately obvious from the specification - the dependency is hidden. Whilst it is possible to specify this aspect without allowing the feeding time to drift, it is not possible to incorporate the drift in a visible manner. It should be noted that the dependency actually relates to the expressiveness of the specification; in this particular case the hidden dependency forms part of the meaning of our expression. Certainly it is arguable that the notation would be more effective (in terms of usability for the specifier, and of understandability for the reader), were the dependency to be visible, although we cannot see how this could be achieved in our example, other than via annotation of the specification. It can be seen therefore, that this dimension may relate to both of Mackinlay's criteria.

The remaining dimensions relate only to effectiveness criteria, and we use our analysis of TRIO$_{\neq}$ to illustrate the usefulness of the cognitive dimensions framework in evaluating notations.

Visibility

In the TRIO$_{\neq}$ specification the visibility of components within the specification is determined by the specifier's use of secondary notation. This allows the specification to be shown in different ways which may be a useful feature for assisting the reader. For example it is possible to group sections of the specification which contain the same time granularity together. If temporal aspects of the system are being considered it can be helpful to consider the specification in this way. By contrast, the specification may be shown with different aspects of the fishes behaviour grouped together, so that it is clear which parts relate to, for example, feeding. It should be noted that the notation includes no particular method to enhance visibility although this could be a useful feature to enhance the notation.

Consistency

This dimension assesses whether similar semantics are expressed in similar syntactic forms. This is certainly the case with TRIO$_{\neq}$ where statements about events always occur in the form:

[Temporal_Operator, Proposition, Time]$_{\text{time granularity}}$

This contributes to the ease of use of the notation and to the understandability of specifications written using it. Once the format is understood it may be applied across the board.

Closeness of Mapping

This dimension was particularly relevant to our specification. The closeness of the representation to the domain is an important feature of temporal representations. TRIO$_{\neq}$ generally allows temporal statements to be made in any granularity, allowing the specifier to select that which reflects the domain. The statement

[NextTime, Assess_Behaviour, 1]$_d$

clearly represents the time domain day once we are aware that the subscript d denotes day. Were we to express the same construct at a granularity of seconds,

[NextTime, Assess_Behaviour, 86400]$_s$

the meaning of the statement is unchanged, but because the representation is no longer close to our understanding of the domain, it is much more difficult both to create and to understand. As we saw above in section three however, it is not always possible to use granularities in a way which reflects our understanding and in these areas comprehension of the specification becomes much more difficult.

Role Expressiveness

This dimension considers whether the purpose of a component is readily inferred. In a TRIO$_{\neq}$ specification this is generally dependent on the way in which the specifier uses the notation rather than any feature of the notation itself. The dimension remains a useful one however, in that it raises awareness of an issue which relates to the overall quality of a specification, and may encourage an approach to specification which results in greater understandability.

Premature Commitment

This dimension asks us to consider whether there are any constraints on the order of doing things. In this particular case there were none and the notation itself had little impact on the way that the specification was approached. This was influenced only by the case study itself which, in the cognitive dimensions framework, forms part of the system under consideration. Although a cognitive dimensions analysis of a particular notation evaluates that notation in the particular context in which it is being used, in this particular case it has allowed us to identify that the influence has come from the case study itself.

Progressive Evaluation

In specifications two forms of evaluation are of particular concern. The first of these is validation of the specification against requirements, to ensure that the requirements specified reflect those that the stakeholders of the system intended. It is possible to validate individual requirements at any time, but for obvious reasons this may only be comprehensively undertaken once a specification is complete. The second form of evaluation is verification of the specification to ensure that it contains, for example, no logical or temporal inconsistencies. Logically based notations allow parts to be checked for consistency at any time, although as with validation, verification of the whole may obviously be undertaken only when the specification is complete. This is a feature of the production of specifications rather than the notation or system, and does not contribute to the overall evaluation of TRIO$_{\neq}$.

Viscosity

There is little resistance to change in the notation, but it should be noted that changes to one particular statement in the specification may impact upon others. For example, the construct

Wait_For_Food $\leftrightarrow$ Regular_Feeding $\wedge \exists x_0 (1440 \text{ x } 7 > x_0 \geq 1440 \text{ x } 6 \wedge [\text{Past, Eats, } x_0]_m \wedge (x_0 \text{ (mod } 1440) \leq 5 \vee x_0 \text{ (mod } 1440) \geq 1435))$

refers to Regular_Feeding (shown in section three above). If changes were to be made to the definition of Regular_Feeding then it would be necessary to check whether there is any knock-on effect on the construct Wait_For_Food. This issue will always arise where a construct is used more than once within a specification, and is not a feature of the notation or case study under examination.

Provisionality

Whilst there is no particular commitment to expressions formulated in TRIO$_{\neq}$, care must be taken to ensure that if changes are made any parts of the specification on which the changed part depends are also changed, as was noted above where we considered viscosity. Again however, this is a feature of specifications in general rather than the notation under consideration.

Error-Proneness

There is a steep learning curve associated with both the use and reading of logically or mathematically based notations. Where the specifier or reader is familiar with such notations however, the background knowledge can usually be transferred to a new notation. Error-proneness was not found to be a problem when using TRIO$_{\neq}$,, indeed the ability to specify at the most appropriate granularity means that mistakes regarding the temporal aspects of the specification are less likely to be made than where a single granularity is used throughout.

Hard Mental Operations

There was a high demand on cognitive resources during the production of the specification, particularly when considering the relationship between the different time granularities. This is a feature of the material being specified as much as the notation used, and it should be noted that demand on cognitive resources would have been much higher had we been attempting to specify the system at a flat granularity. For example, the example shown in section three above relating to regular feeding was particularly difficult to produce due to the fact that an ‘unnatural’ granularity is used, and it remains hard to read. The ability to use more appropriate granularities in other parts of the specification, such as:

$[\text{NextTime, Assess\_Behaviour, } 1]_d$

where we are using the granularity of days, reduced the demand on cognitive resources, and this is most certainly a feature which enhances the usability of the notation.

6 Conclusions

The cognitive dimensions framework has proved to be a useful tool in the evaluation of TRIO$_{\neq}$, providing us with the means to gain thorough understanding of many aspects of the notation. Our selection of the notation was based around its perceived expressiveness, which we had hoped to assess through the application of the notation to a case study. This process highlighted some important areas where the notation was found to be less adequate for our purposes than we had hoped, but the ‘trial and error’ nature of this approach can be a laborious process. An interesting finding during our cognitive dimensions analysis was that some of the dimensions (abstraction secondary notation, diffuseness, hidden dependencies), relate to aspects of a notation that may impact upon the expressiveness of the notation as well as its effectiveness. This indicates that it may be possible for the cognitive dimensions framework to be used to assist in the initial selection of a notation as well as providing an evaluative framework. This is clearly an area where further research is needed.

A particularly useful aspect of the cognitive dimensions framework is the fact that it encourages evaluation of a notation in a particular setting – evaluating a ‘system’ that is comprised of the notation and the environment in which it is used. By considering the environment in which the notation is used, positive and negative influences from the environment can also be identified. For example when considering the dimensions of viscosity, provisionality and error proneness we could identify that our findings here were related to the nature of specifying systems rather than the notation under consideration. In addition, a number of the dimensions, such as secondary notation, role expressiveness and visibility, raised awareness of issues which in this case were unrelated to the notation or system. This allowed us to consider whether we felt we had produced the specification in such a way that reading and understanding are less difficult.

The main area of difficulty in our use of the cognitive dimensions framework was that they in themselves can require ‘hard mental operations’. This is unsurprising given that they are a thinking tool for considering a wide variety of aspects of a notation, but where the dimensions overlap (as we found, for example with role-expressiveness and visibility) then evaluation can itself become awkward. The overlaps and necessary trade-offs between the dimensions are not completely understood nor documented at this stage. We appreciate that these may vary according to the situation in which they are used but further research into these areas may provide the information which could contribute to ease of use of the framework itself. Overall however they have proved a valuable tool in our research into the suitability of $\text{TRIO}_{\neq}$ for use in the specification of interactive systems.

References

Alur, R. and Henzinger, T.A. (1989) A really temporal logic. IN *Proceedings of the Thirtieth Annual Symposium on Foundations of Computer Science*. Z. Galil (ed.) 164-169. Los Alamitos, Calif.: IEEE Computer Society Press

Blackwell, A F and Green, T R G (1999) Investment of attention as an analytic approach to cognitive dimensions. In T.R. G. Green, R. H. Abdullah & P. Brna (Eds.) Collected Papers of the 11th Annual Workshop of the Psychology of Programming Interest Group (PPIG-11), pp. 24-35

Britton, C., Jones, S. & Lam, W. (1998). Separating the system interface from its internal state: an alternative structure for Z specifications. *Proceedings of Formal Aspects of the Human Computer Interaction, BCS-FACS Workshop,* 87-102, Sheffield Hallam University .

Britton, C. and Jones, S. The untrained eye: how languages for software specification support understanding in untrained users. *Human Computer Interaction* 14 191-244

Brun, P. & Beaudouin-Lafon, M. (1995). A taxonomy and evaluation of formalisms for the specification of interactive systems. In M. Kirby, A. Dix & J. Finlay (Eds.), *People and Computers X, Proceedings of HCI'95,* 197-212, Cambridge University Press.

Davis, A. (1988). A comparison of techniques for the specification of external system behavior. *Communications of the ACM, 31 (9),* 1098-1115.

Dix, A.J., Ramduny, D. and Wilkinson, J. (1998) Interaction in the Large. In *Interacting With Computers, Special Issue on Temporal Aspects of Usability*

Green, T. (1989). Cognitive dimensions of notations. In A. Sutcliffe & L. Macaulay (Eds.), *People and Computers V, Proceedings of HCI’89*. Cambridge University Press.

Green, T. (1991). Describing information artefacts with cognitive dimensions and structure maps. In D. Diaper, & N. Hammond (Eds.), *People and Computers VI, Proceedings of HCI’91*. Cambridge University Press.

Green, T. & Blackwell, A. (1996). Thinking about visual programs. In *Thinking with diagrams* (IEE Colloquium Digest No: 96/010). Institute for Electronic Engineers, London.

Green T R G and Blackwell, A F (1998) A tutorial on cognitive dimensions. Available on-line at: http://www.ndirect.co.uk/~thomas.green/workStuff/Papers/index.html

Green, T., Petre, M. & Bellamy, R. (1991). Comprehensibility of visual and textual programs: A test of superlativism against the "Match Mismatch" conjecture. In J. Koenemann-Belliveau, T. Moher & S. Robertson (Eds.), *Empirical studies of programmers* 121-146. Norwood NJ, Ablex.

Harel, D., Lachover, H., Naamad, A., Pnueli, A., Politi, M., Sherman, R., Trachtenbrot, M.(1990) STATEMATE: a working environment for the development of complex reactive systems. *IEEE Transactions on Software Engineering* 16 403-14

Haywood, E. and Dart, P. (1996) Analysis of Software System Requirements Models. *Proceedings of Australian Software Engineering Conference,* 131-138. IEEE Computer Society Press

Johnson, C., McCarthy, J. & Wright, P. (1995). Using a formal language to support natural language in accident reports. *Ergonomics, 38 (6).*

Kutar, M., Britton,C. and Jones, S. (1998) A graphical representation for communicating sequential processes. *Proceedings of Formal Aspects of the Human Computer Interaction, BCS-FACS Workshop* Sheffield Hallam University

Mackinlay, J. (1986). Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics, 5 (2),* 110-141.

McCluskey, T., Porteous, J., Naik, Y., Taylor, C. & Jones, S. (1995). A requirements capture method and its use in an air traffic control application. *Software Practice and Experience, 25 (1).*

Modugno, F., Green T. & Myers B. (1994). Visual programming in a visual domain: A case study of cognitive dimensions. In *People and Computers IX, Proceedings of HCI'94*. Cambridge University Press.

Montanari, A. (1996) Metric and layered logic for time granularity. *ILLC Dissertation Series 1996-02, Institute for Logic, Language and Computation, University of Amsterdam, 1996*

Ostroff, J.S. (1992) Verification of safety-critical systems using TTM/RTTL. In *REX Workshop on real-time: theory in practice,* J.W. de Bakker, C. Huizing, W.P. de Roever, G. Rozenberg (eds.) 573-602 LNCS 600 Springer.

Petre, M. (1995). Why looking isn’t always seeing: Readership skills and graphical programming. *Communications of the ACM, 38 (6).*

Roast, C. (1997). Formally comparing and informing notation design. In H.Thimblely, B. O'Conaill & P. Thomas (Eds.), *People and Computers XII, Proceedings of HCI’97.* Springer.

Scaife, M. & Rogers, Y. (1996). External cognition: How do graphical representations work? *International Journal of Human-Computer Studies, 45*, 185-213.

Shum, S. (1991). Cognitive Dimensions of Design Rationale. In D. Diaper, & N. Hammond (Eds.), *People and Computers VI, Proceedings of HCI’91*. Cambridge University Press.

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

von Knethen, A., Kamsties, E., Reussner, R., Bunse, C. & Shen, B. (1998) A Comparative Case Study with Industrial Requirements Engineering Methods. *Proceedings of the 11th International Conference on Software Engineering and its Applications.* Paris, December 1998

Yang, S., Burnett, M., DeKoven, E. & Zloof, M. (1997). Representation design benchmarks: A design-time aid for VPL navigable static representations. *Journal of Visual Languages and Computing, 8*, 563-599.