Using cognitive dimensions to compare prototyping techniques

Andy Dearden **Jawed Siddiqi** **Amir Naghsh**

School of Computing & Management Science
Sheffield Hallam University
Sheffield, S1 1WB
a.m.dearden@shu.ac.uk
Tel: 0114 225 2916

Abstract

In this paper, we explore the characteristics of different prototyping techniques applied in interactive systems design. Our analysis applies the 'cognitive dimensions' framework, and is informed by an appreciation of four key activities within the design and development of software, namely: the authoring of design proposals; the validation of those proposals with users; implementations mediated by prototypes and specifications; and confirmation of developed systems. In previous discussions of prototyping, attention has focussed upon a concept of 'fidelity' of the prototype, discussing the relative merits of lo-fidelity and hi-fidelity prototypes. Our assessment offers a more fine-grained analysis of methods, helps to clarify important distinctions between prototyping methods, and may be used to inform the selection or development of prototyping tools and techniques.

1 Introduction

In this paper, we show how cognitive dimensions (Green & Blackwell, 1998) can be used to analyse different types of prototyping environment that can be used in the design of interactive systems. We shall argue that recent innovations in prototyping tools and techniques mean that the traditional concept of the 'fidelity' of a prototype, is not sufficiently fine-grained to highlight important similarities and differences between the available set of prototyping approaches. Our alternative formulation in terms of cognitive dimensions focuses on four important activities within the practice of software development that may be mediated by prototypes, namely: authoring of design proposals, validation of those proposals, implementation of those proposals in more developed systems and the confirmation of such implementations.

We hope that this more fine-grained analysis might be useful for developers when selecting or designing prototyping tools.

1.1 Background

It is generally accepted that user participation can contribute significantly to successful design for interactive systems. Many authors have argued in favour of using simple tools such as paper and pencils to create very early impressions of possible designs (Preece et al., 2002, Ehn & Kyng, 1991; Muller, 1993) and to encourage user participation in highly iterative design processes. Techniques that focus on creating and modifying executable software prototypes in order to obtain early feedback from users and clients, e.g. Rapid Application Development and eXtreme Programming, are widely used in commercial software development. Others have suggested that executable formal models, derived from specifications in languages such as Z, could be used to as part of a prototyping process for complex or safety-critical systems (Johnson, 1995; Ozcan *et al.* 1998; Fuchs, 1992). Typically, these different prototyping techniques have been distinguished by discussing the concept of 'fidelity', characterising prototypes as either 'lo-fidelity' or 'hi-fidelity'. Recent work (Landay, 1996; Damn et al., 2000; Lin et al., 2000; Wilson et al., 1997) has explored ways in which some of the benefits of paper prototypes might be realised in software based prototyping systems. In these systems designers can sketch user-interface elements or rough designs for web-pages and identify navigational links between states or pages by drawing arrows. These sketches can then be executed on the basis of the state transitions, allowing a user to experience the proposed navigational behaviour of the design, without the need for a designer to 'play the computer' as happens in traditional paper-prototyping methods (Preece *et al.*, 2002). The paperCHASTE project (Dearden *et al.*, 2002) aims to combine some of these methods to create innovative prototyping tools for interaction designers. In particular, paperCHASTE hopes to enable highly iterative design processes without loosing the potential benefits that formal notations could provide.

When considering such hybrid methods that combine computer animation with hand drawn sketching, or with formal specification, the question of how to characterise them arises. Are they, perhaps, 'medium-fidelity' techniques? Such a linear characterisation is unsatisfactory. They may offer different degrees of fidelity to different aspects of an envisaged system. It is also not clear whether 'medium fidelity' should be encouraged or discouraged in early prototype development. In what follows, we shall outline the possibility of using cognitive dimensions to develop a more discriminatory framework within which these techniques might discussed. Our initial investigation demonstrates that cognitive dimensions offer a useful tool for the analysis of prototyping techniques and we plan to conduct further analysis and investigations of this approach.

In the next section, we outline the main components of the framework. We then present our initial review of different approaches to prototyping and discuss the relationships uncovered. Finally, we offer some preliminary conclusions and our current intentions for further work.

2 A frame of reference

In this section, we identify four key activities that take place within the context of software development and are mediated by prototypes. Following this, we select a small subset of concepts from an established approach to the analysis of representational media in computing, namely the 'cognitive dimensions' described by Green & Blackwell (1998).

2.1 Activities mediated by prototypes

Prototyping approaches to software development pre-suppose cyclic and iterative development process models, e.g. the Spiral model (Boehm, 1988) or the Star model (Hartson & Hix, 1989) in which requirements and possible designs are repeatedly created, explored and validated prior to being further developed into deliverable software components. We can thus distinguish at least three very distinct actions within the activity of software design that are mediated by a prototype. We can label these as Authoring, Validation and Implementation.

- In *Authoring* a prototype may be produced and undergo various modifications. Authoring may involve users to different degrees. For example, in authoring a paper prototype, users may be invited to express their ideas directly onto paper, or using toolkits; alternatively, designers might discuss expectations, desires and requirements with users and clients before authoring a paper-prototype separately from the users.
- In *Validation*, the prototype is examined by stakeholders (designers, developers, users, customers etc.), to identify fit and misfit between the prototype and their needs. The degree of influence that different stakeholders have within the validation activity, in particular the relative importance of users' views, varies between projects.
- In *Implementation*, the prototype is used to guide the creation of a new artefact, or may itself be enhanced to 'evolve' into a new deliverable.

Of course these actions are not mutually exclusive, for example, when initially creating a prototype, the stakeholders are likely to make initial validation judgements and modify the prototype accordingly. During a validation session, additional elements of a design may be authored. During implementation, developers (or users) may identify validation issues that were not previously recognised, and further adjustments to a prototype representation may be authored. These three activities may be related to the activity types of exploration, modification, incrementation and transcription described by Green & Blackwell (1998). For example, our activity of authoring may involve exploration and modification, validation may involve modifications and incrementation, implementation may be viewed as being largely an exercise in transcription. However, we prefer the classification above because of its specific relationship to the particular context of software development.

An additional, though perhaps orthogonal, activity mediated (in part) by a prototype is the management of a relationship between two organisations (the commissioning organisation and the development organisation). We label this activity as: Confirmation

- In *Confirmation*, the implemented artefact may be related to earlier prototypes, and stakeholders may argue about similarities and differences between these different artefacts.

Of course, one way of resolving disputes within confirmation may involve authoring, validating and implementing alterations to either the prototype or the implemented artefact. Similarly, the way in which the activities of authoring, validation and implementation are conducted will influence the organisational relationships and could therefore impact on the confirmation activity.

2.2 Cognitive dimensions and prototyping notations

Green & Blackwell (1998) present the framework of 'cognitive dimensions' as a tool for analysing representational notations. Since prototypes used in software development can be considered as notational devices, we shall be using parts of this framework for the analysis.

2.2.1 Viscosity

Green & Blackwell (1998) define the cognitive dimension of *viscosity* as

> "Resistance to change: the cost of making small changes."

They further identify two types of viscosity:

> *"Repetition* viscosity: a single goal-related operation on the information structure (one change 'in the head') requires an undue number of individual actions.
>
> *"Knock-on* viscosity: one change 'in the head' entails further actions to restore consistency." (ibid. p12)

Green & Blackwell consider the desirability viscosity for different actions on a representation and suggest that high levels of viscosity are generally undesirable in exploration, modification or incrementation of a representation, but may be acceptable in transcription, where one does not want to make changes.
Clearly viscosity is an important issue in prototyping.

2.2.2 Provisionality

Green & Blackwell define *Provisionality* as:

> "the degree of commitment to actions or marks"

citing the example of an architect making "faint blurry marks", meaning that "something more or less like this goes more or less here" (ibid, p41).
Clearly, provisionality has a role in the analysis of prototyping representations. However, it appears that two different types of provisionality might be identifiable.
In any design process using prototypes, the current prototype may have different levels of status within the design process. This status might be evidenced by the way that the prototype is used within the confirmation activity. A prototype may be afforded a low degree of commitment, in the sense that the final design may be very different to this version, or a high degree of commitment in the sense that stakeholders expect the finished system to look and behave in a manner very similar to this version. In this paper, we shall refer to this as '*Process Provisionality*'. Even a detailed specification in a formal mathematical language might have a degree of process provisionality, in that the specification is understood to be open to further modification. Process provisionality is dependent upon the status of the representation within the confirmation activity, but the degree of process provisionality may influence the way that authoring and validation of the prototype proceed.
Alternatively, some representations invite multiple, alternate interpretations, and may be explicitly intended to stimulate design discussion. The architects blurry pencil marks have this property, but a formal specification in Z has specific semantics[1]. We shall refer to this as '*Interpretive Provisionality*'. Interpretive provisionality arises within the authoring of

[1] Note that the process of refinement might offer a degree of interpretive provisionality in a Z specification. For instance an initial specification might suggest a general relation, which could eventually be interpreted as a partial function to sequences in some later refinement of the specification.

representations, and may be significant for the activities of validation, implementation and confirmation.

2.2.3 Closeness of mapping

Closeness of mapping refers to the:

Closeness of representation to domain

Green & Blackwell (1998) provide two 'thumbnail illustrations' to expla in this dimension:
A *close* mapping: the visual programming language LabVIEW, designed for use by electronics engineers, is closely modelled on an actual circuit diagram, minimising the number of new concepts that need be learnt. A *distant* mapping: in the first version of Microsoft Word, the only way to count the characters in a file was to save the file to disc – whereupon it told you how long the file was.
In this paper, we interpret these illustrations as emphasing the visual and behavioural similarities between the symbols in a notation and their referents. Note that we are interpreting the 'domain' of prototyping tools to be software systems, rather than the domain in which the software might be deployed (e.g. banking, communications, air transport etc.). Hence the 'closeness of mapping' issue concerns the mapping between the prototype and the finally delivered software.

2.2.4 Using these dimensions

Prototyping brings out an important clarification of the concept of provisionality and its relationship to closeness of mapping. To clarify the distinction we are making, consider the following example. During the iterative authoring and validation of a prototype graphical user-interface, a button on a particular screen or panel can be represented by either:

1. a roughly rectangular mark on a piece of paper, or
2. a particular software object rendered with a defined border style positioned on an executable software prototype.

The eventually delivered artefact may or may not include that screen, that panel or that button. In either case, both the pencil mark and the software object appear to have the same level of 'process provisionality' and we might argue that the 'interpretive provisionality' is the same (both representations are understood to refer to a button). However, the software object has a greater 'closeness of mapping' because of the degree of similarity between the representational artefact (the rectangular pencil mark or the software object) and any software system that might be produced.

2.3 Summary

Our analysis now proceeds examining different tools and notations that can be used for prototyping, and exploring the four activities: authoring, validation, implementation and confirmation; and the three dimensions of viscosity (including knock-on and repetition viscosity), provisionality (distinguishing process and interpretive provisionality) and closeness of mapping.

3 A review of tools

Given the framework outlined above, it is now possible to examine a range of established prototyping techniques, and evaluate more recent developments that seek to combine benefits from different approaches.

3.1 Paper prototyping

Paper prototypes are closely associated with participatory approaches to software design. Rettig (1994) suggests that paper-prototypes provide a valuable and cost-effective means to evaluate and

iterate design options before a team gets committed to one implementation. Paper prototypes are usually described as an example of 'lo-fidelity' prototypes (Rettig, 1994; Preece et al., 2002). To develop a paper-prototype, interface elements such as menus, windows, dialogues and icons can be sketched onto paper. Blocks of textual output may be represented by squiggles. When the paper prototype has been prepared, a member of the design team sits before a user and allows the user to make selections and activate interface elements by using their finger as a mouse or perhaps by writing 'typed' input. The designer responds to the user's actions by moving interface elements around in accordance with the proposed behaviour of the computer system. A further person assists the session by providing task instructions and encouraging the user to express their thoughts and impressions. Notes may be made by other observers or a video record may be created. In participatory approaches to paper-prototyping, the users themselves may be encouraged to create or to make modifications to the prototype (Muller, 1993).

3.1.1 Examining paper prototyping

In terms of the cognitive dimensions discussed above, paper-prototyping relies on providing a low-viscosity representation to aid in the authoring of a design, and its incrementation and modification during validation activities. The use of paper & pencil representations is recommended by advocates of participatory design in particular because of the ease with which users can modify the design representation to express their desires. However, as Lin *et al.* (2000) points out, a paper prototype may suffer from repetition viscosity because changes to a design may need to be reflected across multiple sketches.

It is also important that a relatively close mapping is available between the domain (the appearance of a graphical user-interface) and the notation (paper sketches) so that the users can easily learn to interpret the notation. On the other hand, one stated benefit of paper prototypes is that because the detail of colours, precise positioning, fonts etc. is not available from a sketch, this avoids users focussing attention on such low level details of design. This might be interpreted as an argument against too close a mapping. Some authors have highlighted limitations in paper-prototyping in terms of its ability to faithfully present certain aspects of interaction. Rettig (1994) highlights the problem of representing aspects such as system response times. O'Neill *et al.* (1999) discuss the lack of an explicit representation of the navigational structure, which may result in users suggesting modifications to the layout of individual screens, but being unable to modify or critique overall dialogue structure. Hence, paper prototyping offers a closeness of mapping only in respect to certain aspects of the system being developed.

The use of squiggles or other vague lines in paper prototyping can give a paper prototype a high degree of interpretive provisionality. Many paper prototypes may be explored during early design with the intention of stimulating new ideas. In this case, there is often a clear view that the prototype is not intended as a representation of any finished system, i.e. a high degree of process provisionality is being supported.

3.2 Rapid software prototyping

As an alternative to the rapid iterative development of paper-prototypes, software prototypes may be developed and validated.

3.2.1 Examining software prototyping

Using such software tools is likely to produce a design that has a closer mapping to the finished software system than does a paper-prototype, especially in relation to the appearance and navigation of the software. However, it is often the case that the timing behaviours of large and distributed software are difficult to predict, and are not reflected by early software prototypes.

By applying tools such as GUI builders and visual programming languages, designers may seek to maintain a lower level of viscosity that would be the case in textual programming environments.
If a low level of viscosity is maintained, then designs might be treated as having a high degree of process provisionality, undergoing multiple iterations during authoring and validation. However, the interpretive provisionality of the design may be lower, since the prototype is rendered using the same medium as that intended for the finished system.
Because the toolkits used mirror the toolkits in which the finished system might be delivered, there is a close mapping between the appearance and local behaviour of the prototype and the finished system. Of course, other aspects of behaviour such as timing, or the precise detail of large data tables etc. may be still involve distant mappings.

3.3 Executable formal specifications

Using formal mathematically based notations is another way for examining aspects of the design of an interactive system. Such mathematical models of a proposed design can be used to verify detailed aspects of the semantic behaviour of a software system. The use of such formal methods has been particularly recommended in the context of designing safety or mission critical systems, but may also be appropriate where the precise behaviour of a system is hard to describe except mathematically, e.g. in validating the recommendations that an automated meeting scheduler might make where the constraints of the meeting (rooms, people, timing etc.) cannot all be satisfied.
Fuchs (1992) explains that executable specifications represents conceptual and behavioural model of the system that is going to be implemented and it could considered as a prototype which allows users and developers to interact with executable specifications. Siddiqi et al. (1998) propose a design process where executable formal specifications are used for the constructions of prototypes, which can then be used to validate software requirements with users. This process aims to combine the benefits of iterative prototyping with the advantage of using a formal specification that can be subjected to rigourous analysis of its behavioural properties. The process is demonstrated using TranZit, a tool that supports manipulation the Z specifications, and ZAL, an environment that supports the execution of a subset of Z. Özcan *et al.* (1998) show how ZAL models can be enhanced by animation facilities.

3.3.1 Examining executable specification

Khazaei & Triffitt (2002) provide an analysis of the Z notation, and its use within the TranZit tool in relation to cognitive dimensions. The analysis suggests that even within the TranZit environment, Z suffers from high levels of both knock-on viscosity and repetition viscosity. Khazaei & Triffit report that most of their subjects preferred for working with paper and pencil before attempting to input Z specifications into tools such as TranZit. Paper and pencil versions of Z specifications allow for representations that use the notational elements of Z, but do not adhere to the strict syntactic rules of the notation. This approach may help to reduce viscosity. Khazaei & Triffit also suggests that Z provides a distant mapping between notational elements and the domain described. Animation using the ViZ environment (Özcan *et al.*, 1998) offers one way in which a closer mapping might be enabled during validation.
Using a process in which a Z specification is repeatedly subjected to user validation and may be revised or discarded, can, in terms of the validation and confirmation activity be interpreted as providing for a level of process provisionality. On the other hand, the formal mathematical definition of Z (and the community of practitioners that utilize Z in practice) restricts the degree to which the representation is open to competing interpretations, i.e. the representation has low interpretive provisionality.

3.4 Hybrid methods

In recent years, Landay, Lin, Hong & colleagues have demonstrated a number of different tools that aim to provide interaction designers with prototyping tools that blur the boundaries between paper and software prototypes.

Landay (1996) suggests that paper-based methods have a number of inherent distadvantages:

- Paper sketches are hard to modify. When large numbers of sketches are used, a designer must frequently redraw common features of a design for all sketches when modifications are required.
- Paper sketches provide limited support for design memory. It is hard to search through sketches in the future to find out why a particular design decision was made.
- Paper sketches limit the quality of the interaction that can be simulated during validation activities. In paper prototyping, a designer (developer) needs to play the computer by moving sketches around in response to a user's actions.

From our own experience, another disadvantage can be identified, which is that paper sketches are difficult to transport between sites in a distributed development setting where not all stakeholders can be brought together at a single location to inspect the design.

In response, Landay proposes interactive user-interface design tools that support electronic sketching. These tools aim to provide designers with the freedom to sketch rough design ideas quickly, as well as the ability to test designs by letting user interacting with sketches, and to fill in the design details as users and designers make choices. Wilson *et al* (1997), proposed a similar tool that sought to emphasise the 'rough' 'sketchy' aspects of a paper-based prototyping. We examine three of these tools below.

- SILK (Landay, 1996) stands for Sketching Interfaces Like Crazy. It allows designers to sketch an interface using electronic stylus, and enables the designer to specify interactive behaviours by marking transitions on a storyboard. SILK then retain the sketchy look of the components.
- DENIM (Lin *et al.* 2000) extends the ideas in SILK to support the early stages of website design. It supports sketching inputs and allows designers to examine the design at different levels of detail varying from individual page layout to overall site navigation structures. Links between pages can be created by drawing an arrow from the starting point of the link within one page to the destination page. Like SILK, DENIM allows designers and users to execute the model to explore the navigational behaviour. Figure 1 shows a screenshot of the DENIM system.
- QUILL (Hong *et al*, 2002) further extends the functionality of SILK and DENIM by providing a gesture recognition system, so that particular marks on a sketch can be interpreted as in terms of common user-interface components such as buttons, drop-down list boxes, scrollbars or text input boxes. Thus the designer sketches the design, which is (semi) automatically translated into a software prototype that includes the selected components.

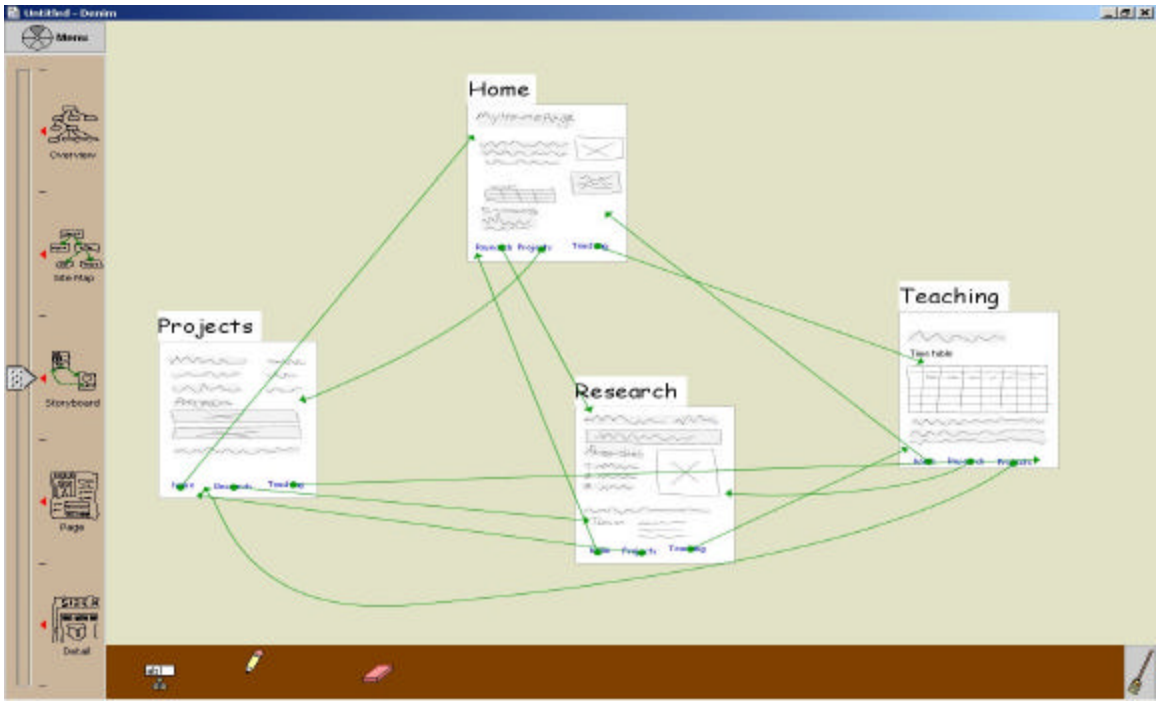

Figure 1: A design sketch produced using DENIM (Lin *et al.*, 2000).

3.4.1 Examining the hybrids

These hybrid systems all aim to provide a low viscosity representation for design. Indeed Landay's initial observations suggest the aim of reducing the repetition viscosity that may be encountered with paper-prototyping. On the other hand, there may be a risk in combining the state transition diagrams and paper representations of adding to the knock-on viscosity. Roast (2002) presents strong arguments to suggest that the knock-on viscosity of a combined notation may be higher than that for either of the original notations.

All the tools assume a degree of process provisionality. Because any of the prototypes can include rough pencil marks & squiggles, they are also open to elements of interpretive provisionality. On the other hand, as with paper prototypes, some symbols may have specific interpretations, such as tables, buttons or scroll-bars.

The paper sketching enables a close mapping to the screen layout and navigational behaviour of the completed software system, although the detailed appearance of user-interface components is not closely matched. QUILL's automatic translation to user-interface components leads to a closer mapping in this respect.

3.5 Discussion

The examination above suggests that cognitive dimensions offer a more discriminating account of prototypes and prototyping practices in user-interface design, than is afforded by the lo-fidelity, hi-fidelity distinction. Our analysis to date has applied the cognitive dimensions of:

- viscosity (both knock-on viscosity and repetition viscosity)
- closeness of mapping and
- provisionality (both process & interpretive).

The example of using executable Z specifications in an iterative design process, suggests that provisionality can be further subdivided to distinguish process provisionality and interpretive provisionality. The use of Z in an iterative design process (Siddiqi et al., 1998) demonstrates that process provisionality is independent of the notation used, and may be better considered as a characteristic of the design and development process. However, it should be recognised that using representations that include high interpretive provisionality will imply a degree of process provisionality.

In general, prototyping strategies are assisted by the use of low viscosity representations. Maintaining low viscosity lowers the costs associated with frequent iterations in authoring and validating designs. Hence, the development of toolkits and environments for rapid development of software prototypes, or the development of hybrid tools such as DENIM and SILK can be seen as attempts to reduce the viscosity of executable software prototypes.

For validation, being able to execute the prototype enhances the closeness of mapping between the prototype and behavioural characteristics of a system. This closeness of mapping may be further enhanced when actual components from user interface software toolkits replace paper based or sketch based techniques as in QUILL. This closeness of mapping, however, implies some sacrifice of interpretive provisionality which may adversely effect the initial design authoring activity. On the other hand, low interpretive provisionality may be desirable to simplify the implementation and confirmation activities.

The discussion of these different tools also indicates that often these cognitive dimensions are mixed in relation to different aspects of any given prototype. For example, a paper prototype may have a close mapping to the layout of a screen in the delivered system, whilst having a distant mapping to the timing behaviour of a system. It may be subject to high levels of repetition viscosity for some types of change, but have low levels of knock-on viscosity when one part of a screen is altered. Some areas of a paper-prototype may include squiggles to provide high levels of interpretive provisionality, whilst others might include detailed buttons and menus.

Similar contrasts can be found when investigating aspects of executable formal specifications, using software prototypes, or using hybrid prototyping tools.

With this more discriminatory view of prototyping tools, it may be possible to conduct a closer investigation of the advantages and disadvantages of different prototyping approaches. For example, lo-fidelity prototyping is sometimes advocated because of the ease of modification (low viscosity), and the argument is combined with a discussion of the value of extensive design exploration (high process provisionality). However, advances in the design of software and hybrid prototyping environments may be able to achieve similar levels of viscosity and process provisionality. A different set of arguments surround the desire to avoid discussion of low-level design details too early in the design process. These goals suggest prototypes where the closeness of mapping to some aspects of a software system should be kept low. Further, representations with high levels of interpretative provisionality might be useful in stimulating the discussion of new design ideas within design authoring activities.

4 Conclusion and further work

Previous discussions of the use of prototypes in user-interface design have contrasted lo and hi-fidelity prototyping. In this paper, we have developed an alternative characterisation of prototyping, based around an analysis of the activities that may be mediated by prototypes, and the framework of cognitive dimensions. Authors in HCI have previously argued for the importance of lo-fidelity prototyping, whereas our analysis suggests that a single dimension is insufficiently expressive to distinguish the many different prototyping approaches that have now been developed.

Our initial analysis has applied only a small number of the dimensions outlined in Green & Blackwell (1998). Other dimensions are clearly relevant. For example, all prototyping is clearly concerned with issues of Premature Commitment within the design process. Many prototypes are accompanied by secondary notation explaining aspects of their behaviour. Many issues in prototyping are concerned with the creation and re-use of abstractions. In future work we hope to conduct a detailed analysis of a range of hybrid prototyping tools, including an investigation of some of these other dimensions. We expect that such an investigation may reveal useful insights for the design of future prototyping tools, and may offer useful case-studies to clarify concepts within the cognitive dimensions framework.

4.1 Acknowledgements

This work was supported by EPSRC grant number GR/R87918, paperCHASTE. We should like to thank Mehmet Özcan, Babak Khazaei & Chris Roast for their helpful comments on earlier drafts of this paper.

4.2 References

Boehm, B. W., 1988. A Spiral Model of Software Development and Enhancement. *IEEE Computer*, 21(5), 61 – 72.

Damn C.H., Hansen, K.M, Thomsen, M. 2000. *Tool support for co-operative object-oriented design: Gesture based modelling on an electronic whiteboard.* In, Proceedings of CHI 2000. ACM Press.

Dearden, A. M., Özcan, M. B., & Siddiqi, J., 2002. paperCHASTE: Supporting design conversations by integrating formal and informal representations. http://www.shu.ac.uk/schools/cms/teaching/amd/paperchaste.html

Ehn, P. & Kyng, M., 1991. Cardboard Computers: Mocking-it-up or Hands-on the Future. In, Greenbaum, J. & Kyng, M. (Eds.) *Design at Work*, pp. 169 – 196. Hillsdale, New Jersey: Laurence Erlbaum Associates.

Green, T. & Blackwell, A. 1998. Cognitive Dimensions, A Tutorial. Available from: http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/

Fuchs, N.E., 1992. Specifications are (preferably) executable. *Software Engineering Journal*, 7(5), pp.323 - 334.

Hartson, H. R. & Hix, D., 1989. Toward Empirically Derived Methodologies and Tools for Human-Computer Interface Development. *International Journal of Man-Machine Studies*, 31, 477 – 494.

Hong, J., Landay, J, Long, C & Mankoff, J., 2002. Sketch Recognizers from the End-User's, the Designer's, and the Programmer's Perspective. In *Proceedings of AAAI 2002 Spring Symposium (Sketch Understanding Workshop)*. Stanford, CA 2002.

Johnson, C.W., 1995. The Economics Of Interface Development. In *Human Computer Interaction - Interact '95*. K. Nordby and P. H. Helmersen and D. Gilmore & S. A. Arnese (Eds.). London, UK: Chapman and Hall, pp. 19 - 25.

Khazaei, B & Triffit, 2002. Applying Cognitive Dimensions to Evaluate and Improve the Usability of Z formalism. In, *Proceedings of SEKE, 2002*, 571 – 577. ACM Press.

Khazaei, B & Roast, C., In Press. The Influence of Formal Representation on Solution Specification. To appear in Requirements Engineering Journal.

Landay, J., 1996. *Interactive Sketching for the Early Stages of User Interface Design.* Technical Report CMU-CS-96-201, Carnegie Mellon University, Pittsburgh, PA. 1996.

Lin, J., Newman, M.W., Hong, J.I. & Landay J.A., 2000. DENIM: Finding a tighter fit between tools and practice for web site design. In *proceedings of CHI 2000*, pp. 510 - 517. The Hague, Netherlands: ACM Press.

Lin, J., Thomsen, M & Landay, J., 2002. "A Visual Language for Sketching Large and Complex Interactive Designs." Proceedings of CHI 2002. CHI Letters 4(1): pp. 307-314.

Muller, M., 1993. PICTIVE: Democratizing the dynamics of the design session. In Schuler, D. & Namioka, A. (Eds.). *Participatory Design: Principles and Practices* pp. 211-238. Hillsdale, NJ. USA: Lawrence Erlbaum Associates.

O'Neill, E., Johnson, P. & Johnson, H., 1999. Representations and user-developer interaction in cooperative analysis and design, *Human-Computer Interaction*, 14 (1 & 2), pp. 43 - 91.

Özcan, M. B., Parry, P.W., Morrey, I. and Siddiqi, J., 1998. "Visualisation of Executable Formal Specifications for User Validation", *Lecture Notes in Computer Science*, Vol 1385, 142-157, Springer-Verlag.

Preece, J., Sharp, H. & Rogers, Y., 2002. *Interaction Design*. John Wiley & Sons.

Rettig, 1994. Prototyping for Tiny Fingers, Communications of the ACM, 37(4).

Roast, C., 2002. Dimension driven re-design - applying systematic dimensional analysis. In J. Kuljis, L. Baldwin, and R. Scoble, editors, *Proceedings of the 14th Psychology of Programming Interest Group workshop (PPIG 14)*, pages 173-185. Brunel University.

Siddiqi, J.I.A. Morrey, I. Ozcan, M. Roast, C. "Towards Quality Requirements via Animated Formal Specifications" *Annals of Software Engineering* Vol 3 Sept 1997, 131 - 155

Wilson, S., Bekker, M., Johnson, P. & Johnson, H., 1997. Helping and Hindering User Involvement - A Tale of Everyday Design, *Human Factors in Computing Systems, Proceedings of CHI97* pp. 178 – 185. Atlanta, GA. USA: ACM Press.