

Preprint. Final version will be available as: BLANDFORD, A., GREEN, T. R. G., FURNISS, D. & MAKRI, S. (forthcoming) Evaluating system utility and conceptual fit using CASSM. To appear in *International Journal of Human-Computer Studies*. That version may differ from the current one.

Evaluating system utility and conceptual fit using CASSM

Ann Blandford¹, Thomas R. G. Green², Dominic Furniss¹ & Stephann Makri¹

¹UCLIC, University College London, Remax House, 31-32 Alfred Place, London WC1E 7DP, UK

²University of Leeds, UK

A.Blandford@ucl.ac.uk

Tel. +44 20 7679 5288

ABSTRACT

There is a wealth of user-centred evaluation methods (UEMs) to support the analyst in assessing interactive systems. Many of these support detailed aspects of use – for example: Is the feedback helpful? Are labels appropriate? Is the task structure optimal? Few UEMs encourage the analyst to step back and consider how well a system supports users' conceptual understandings and system utility. In this paper, we present CASSM, a method which focuses on the quality of 'fit' between users and an interactive system. We describe the methodology of conducting a CASSM analysis and illustrate the approach with three contrasting worked examples (a robotic arm, a digital library system and a drawing tool) that demonstrate different depths of analysis. We show how CASSM can help identify re-design possibilities to improve system utility. CASSM complements established evaluation methods by focusing on conceptual structures rather than procedures. Prototype tool support for completing a CASSM analysis is provided by Cassata, an open source development.

Keywords

CASSM, usability evaluation methods, conceptual structures, co-evolution.

INTRODUCTION

Imagine you are planning a journey to another continent. Let us say you live in York, England and are travelling to San Jose, California. One thing you need to do is book a flight. But where from, and where to? Is Leeds/Bradford or London a better choice? You much prefer direct flights, but when you select flights from anywhere in the UK to San Jose, they all involve transfers. You search for a US map that includes airport information; there appear to be three not far away from San Jose, but then you need to know their names, you need ground transportation information, and you still need to know whether you can fly there direct from the UK (and, if so, from which airport). An apparently simple task of booking a flight has become rather complex. This is an example of a conceptual misfit between what users require and what current flight booking websites typically offer: sites work in terms of *flights between airports*; users work in terms of *journeys between places*.

This misfit between user and system conceptualizations is a kind of usability difficulty that is not readily identified using existing evaluation methods such as Heuristic Evaluation (Nielsen, 1994) or Cognitive Walkthrough (Wharton, Rieman, Lewis & Polson, 1994), for which the instruction materials typically encourage the analyst to focus on the device and draw on their own experience regarding the domain. Unless the evaluator is alert to the idea of 'misfits', it is also unlikely to emerge through empirical evaluation approaches such as think-aloud protocols (Ericsson & Simon, 1993); indeed, Nørgaard and Hornbæk (2006) report that think-aloud protocols are often used to confirm anticipated problems and typically focus on usability rather than utility of systems.

The identification of misfits typically represents a design opportunity: in the example above, a site that makes it easier to plan the journey as well as booking the flight (and possibly ground transportation too) could be attractive to users and create a bigger market for integrated travel arrangements. More generally, the identification of a user concept that is missing from the system representation is indicative of a possible design change (though not all user concepts are easily converted into elements of the system design). Under this view, evaluation is less about fixing bugs and more about identifying new design possibilities that better match user requirements. Evaluation directly informs re-design. It does not necessarily dictate, or even indicate, the precise form of that re-design, but highlights the possibility in the form of user-centred requirements.

The opening example was presented as a simple scenario, and only one misfit was highlighted; there might be many other misfits of interest in this domain. To identify them it is necessary to have a systematic approach. CASSM has been developed as such a systematic approach. It supports the analysis of *conceptual misfits* between the way the user thinks and the representation implemented within the system. The name, CASSM, stands for “Concept-based Analysis of Surface and Structural Misfits”, and is pronounced “chasm” to invoke the idea of a gulf (Norman, 1986) between user and system; the different kinds of misfits (surface and structural) are discussed below. This approach fills a niche in the space of analytical evaluation techniques by focusing specifically on conceptual fit rather than, for example, task structures, what the user knows about the system, user experience or the modalities of interaction. In particular, it encourages a high level view of system fitness for purpose, and hence supports reasoning about new design alternatives rather than incremental improvements.

BACKGROUND: ATTRIBUTES OF EVALUATION METHODS

When developing a new evaluation method, it is important to consider what properties that method should have (Blandford and Green, forthcoming). We have already stated that one of the aims in developing CASSM was that it should *focus on a system’s fitness for purpose*, and hence *lead to new design possibilities*. Wixon (2003) argues that this kind of utility should be central to choosing and using any evaluation technique. Hartson, Andre & Williges (2001) identify this criterion as ‘downstream utility’ in their discussion of how to evaluate evaluation methods; John and Marks (1997) consider an important aspect of downstream utility to be the persuasiveness of a method. Several other criteria for assessing evaluation methods have been identified and discussed in the literature.

Historically, simple *problem count* (the number and/or severity of problems an evaluation method helped identify) was considered an important criterion. Many studies through the 1980s reported on comparisons between different evaluation methods considering which helped analysts identify more usability difficulties; these studies were systematically reviewed, and their flaws highlighted, by Gray and Salzman (1998). Since that review, this criterion has been relatively de-emphasised. In the development of CASSM, we have been much more concerned with the quality and utility of insights than with their number.

More recently, *reliability* has emerged as a central concern. Reliability is a measure of consistency of results across different evaluators. Reliability between evaluators has been generally found to be low, even with evaluators who are given the same data (e.g. video data) and asked to apply the same analytical method (Hertzum and Jacobsen, 2001). Some methods, such as Heuristic Evaluation (Nielsen, 1994) and Cognitive Walkthrough (Wharton *et al.*, 1994), advocate employing a team of evaluators to mitigate against the evaluator effect. In the development of CASSM, we have not been concerned with reliability in this sense: if two different evaluators achieve different insights, that is not considered a problem as long as the insights are valid (see below). A single evaluator can identify some misfits, and that should be useful for informing redesign. Employing more evaluators, like conducting a more thorough analysis, increases costs, and a judgment has to be made about whether this will yield commensurate benefits. The CASSM approach does not include an explicit recommendation regarding number of evaluators.

A related criterion that has been considered is *thoroughness*. Hartson *et al.* (2001) define this as the proportion of real problems found (as compared to the real problems that exist in the system). This definition makes two assumptions that are, in our view, inherently problematic: firstly that it is possible to count problems (e.g. the number of problems counted will depend on the level of abstraction of problem descriptions); and secondly that it is possible to know how many ‘real’ problems there are in the system (a real problem could be one that is encountered by a particular proportion of users, that is very infrequent but catastrophic, that leads to errors in a certain percentage of interactions, or any of several other definitions). A third assumption implicit in this definition is that any method should be able to identify all classes of usability problems. In developing CASSM, we have explicitly not been aiming to achieve thoroughness, but to fill a niche. That is: to identify a kind of usability difficulty that is ignored by established evaluation methods.

In line with their approach of defining reliability and thoroughness in numerical terms, Hartson *et al.* (2001) also define *validity* and *effectiveness* numerically. In particular, they define the validity of a method as the proportion of problems found that are real. In our view, this definition suffers the same difficulties as their definition of thoroughness: that it is difficult – and ultimately not productive – to attach meaningful counts to these variables. We adopt a rather looser, but nevertheless stringent, definition of validity, which is that a method should support the analyst in correctly predicting user behaviour or identifying problems that users will have with the system. In particular, the approach should not result in many false positives (issues that are identified as problems that actually are not).

Of course, by this definition, a method that supported no predictions at all would achieve high validity (no false positives); however, it would clearly not be useful. A method also needs to be *productive*, in the sense of supporting the

identification of usability issues. Every usability issue will be outside the scope of some evaluation methods (e.g. a matrix algebra approach is unlikely to have much to say about the effectiveness of team working), so another requirement concerns *scope*. The scope of a method refers to the kinds of issues it does and does not address. This should be clear to users of the method so that analysts can both select an appropriate method for addressing their current concerns and also appreciate the limitations as well as the value of their analysis. CASSM has not been developed to identify all possible usability issues, but particularly those related to conceptual misfits between users and systems.

The criteria discussed so far have related to the outputs of evaluation methods; a second class of criteria relate to *learning* and *applying* a method. Surprisingly little discussion has focused on the idea that HCI methods must themselves be usable. This means that they must be learnable, cost-effective and also fit within design practice (one might want to be more specific about what particular kinds of design practice are to be addressed).

Learnability of a method will be a strong determinant of take-up and use in practice. We are not aware of many studies of the learnability of methods in HCI, although Blandford, Buckingham Shum and Young (1998) studied the practicalities of teaching (and learning) PUM, and John and Packer (1995) investigated the learnability of Cognitive Walkthrough. In our development of CASSM, ease of learning and ease of use were important criteria. In particular, the approach is intentionally ‘sketchy’ with iterative deepening, so that initial insights can be achieved with minimal investment and the analysis can be made more thorough over time if the situation demands it. Throughout the development, we have conducted informal studies of the use of the approach, which we discuss below.

As Wixon (2003) notes, there is still little agreement on what features an HCI method should possess. In developing CASSM, we have not been particularly concerned with productivity, reliability or thoroughness. Rather, when we started this project, our overall objective was to develop a method that was useful and usable.

For us, the important elements of being useful were that the method should:

- fill a niche in the space of evaluation methods (concerning conceptual misfits); and
- deliver valid insights: in particular, the analysis should be grounded in real user data wherever possible.

Similarly, elements of being usable were that the method should:

- support iterative deepening of analysis to enable analysts to choose how much effort to invest;
- provide tool support to facilitate analysis;
- be easy to learn; and
- fit with established design practice without disrupting it.

Figure 1 summarises the attributes discussed above. Those in **bold** are the ones we have identified as our objectives in developing CASSM; in the discussion section, we review the extent to which we have achieved these objectives.

| Stage of evaluation | Requirement | Aspects |
|---------------------|------------------------------|---|
| Outputs: useful | A: downstream utility | A1: persuasiveness A2: lead to new design possibilities |
| | B: problem count | |
| | C: reliability | |
| | D: thoroughness | |
| | E: validity | |
| | F: effectiveness | |
| | G: productive | |
| | H: have clear scope | H1: fill a niche |
| Process: usable | I: learnability | |
| | J: cost effectiveness | J1: quick and easy to apply J2: support iterative deepening |
| | K: fit with practice | |
| | L: tool support | |

Figure 1: Summary of properties of evaluation methods

CASSM: AN OVERVIEW

As noted above, we are not aware of any other usability evaluation approach that is designed to support the systematic analysis of conceptual fit between user and system. Moran's (1983) ETIT and the Yoked State Spaces (YSS) of Payne, Squibb and Howes (1990) strongly informed our thinking about CASSM, as they are based on the idea of comparing user and system representations of the same task; however, neither ETIT nor YSS was codified so that non-specialists could apply it. Our own earlier work on PUM (Blandford & Young, 1996; Young, Green & Simon, 1989) and ERMIA (Green & Benyon, 1996) also informed the design of CASSM, which has drawn on important representational elements of those approaches. While PUM and ERMIA were both codified (in tutorial materials), we perceive both as demanding too much precision and detail to apply routinely. The final approach that has heavily influenced our thinking is Green's Cognitive Dimensions of Notations (CDs: Green, 1989). CDs provide an informal vocabulary for describing various misfits between user and system. Precise definitions of many CDs are now encapsulated within the Cassata analysis tool described below.

CASSM focuses on the quality of fit between the user's conceptual model and that implemented within a system, encompassing a method for both data gathering and data analysis. Within evolutionary development, a recognition of poor fit highlights opportunities for re-design (Hsi, 2004).

As highlighted by Vicente (1999), misfits between user and system can originate from users having inaccurate mental models of the system (which should be corrected through improved system design or training) or from systems implementing an inaccurate representation of the user's domain of working. Some systems define their own domain; for example, the notion of 'links' depends on the notion of a hypertext; for these systems, the design challenge is often to help users to develop an appropriate mental model of the system. Then, the question is typically not whether there is a good conceptual misfit but rather how easy it is for users to acquire an appropriate mental model. In other words, a CASSM analysis would typically focus more on the ease of acquiring a good conceptual fit, with a view to considering how the system might be re-designed to support acquisition of an appropriate mental model, rather than re-designing to accommodate the users' existing mental models.

CASSM is a method and modelling representation, supported by a prototype tool, Cassata. CASSM compares the users' concepts with the concepts implemented within the system and interface. Conceptual analysis draws out commonalities across similar users to create the profile of a typical user of a particular type; the analyst can then assess the quality of fit between user and system. CASSM supports reasoning about two broad classes of misfits. *Surface misfits* are concepts salient to the user but partially or wholly missing from the system and, conversely, concepts imposed on the user by the system that might be difficult to learn or work with. *Structural misfits* are cases where the relationships between concepts in the user's model do not coincide with those in the system, resulting in situations where a change to the system representation causes user difficulties. Many structural misfits correspond to CDs such as *premature commitment*, *viscosity* and *hidden dependencies* (Green & Petre, 1996; Green, 1989; Green 1990).

Surface misfits go back, although not by that name, to Norman's (1986) analysis of whether the user of the system could understand the designer's conceptual model and whether the system appropriately implemented pre-existing user concepts. The advance made by CASSM is to provide a structured methodology with which to identify them. Structural misfits, in contrast, are hardly present in previous usability evaluation methods.

This recognition of different types of misfits between user and system is reflected in the name, CASSM, "Concept-based Analysis of Surface and Structural Misfits". Earlier papers on this approach (Blandford, Wong, Connell & Green, 2002; Connell, Green & Blandford, 2003) were published under the name OSM. This paper presents a more complete account of the method, including an account of how it can support substantive re-design (rather than the detailed usability problems that are the focus of many evaluation methods). In the terms of Nørgaard and Hornbæk (2006), CASSM considers both usability and utility because it supports the identification of both tasks that are more difficult than necessary (e.g. because something that is conceptually simple demands complex work-arounds using a particular system) and tasks that are impossible (such as the journey planning outlined above).

In the following sections, we summarize the approach to conducting a CASSM analysis and the kinds of insights the approach affords; briefly describe Cassata, the CASSM analysis tool; and discuss informal evaluations of the approach. We illustrate CASSM using three contrasting examples:

- a simple robotic arm interface that illustrates identification of and reasoning with surface misfits, and what can be achieved with minimal user data;
- a digital library interface that illustrates the use of verbal protocol data; and

- a simple drawing tool that illustrates a structural misfit.

These examples have been chosen to be simple enough to discuss in detail and to illustrate different aspects of the CASSM approach. We have completed a preliminary CASSM analysis of the journey planning / flight booking example with which we opened this paper, using think-aloud data from four participants. However, the resulting analysis is too complex for detailed presentation in this paper (since our purpose here is to illustrate the CASSM approach rather than discuss flight booking systems).

We close the paper by discussing the extent to which we have achieved our objectives in the development and testing of CASSM, and identifying future work.

CASSM: THE METHOD

The method of conducting a CASSM analysis is presented in detail in a downloadable tutorial (Blandford, Connell & Green, 2004). Here, we summarise the ideas underpinning the approach and important stages of analysis.

The focus for conducting a CASSM analysis is on *concepts*. A *concept* may be an *entity* or an *attribute*. Very early on in analysis, it is not necessary to distinguish between these two different types of concept, but as analysis proceeds the distinction becomes important. An *entity* is usually something that can be created or deleted within the system. An *attribute* is a property of an entity – usually one that can be set when the entity is initially created, or that can be subsequently changed. Attributes are always associated with an entity, so some entities are listed simply because they have attributes that can be changed, even if they themselves cannot be created or deleted (they are *fixed*).

For every concept, the analyst determines whether it is *present*, *difficult* or *absent* for the *user*, at the *interface* and in the underlying *system*. A concept is considered to be *present* for the user if users naturally and easily think in terms of it. Concepts might be *difficult* for the user if they are hard to learn, discover or work with: these difficulties may be of different kinds and levels of importance; it is up to the analyst, working with user data if possible, to assess the severity of difficulties. Concepts may be *absent* for the user if users show little recognition that the concept has been implemented in the system; this may result in users not using the system as intended (e.g. missing features of it) or having great difficulty in getting the system to work as desired. Similarly, for the interface and system, concepts may be present, difficult or absent. Concepts that are present in one place but absent or difficult in another are potential sources of misfits between the system and user, and might be triggers to consider re-design possibilities.

For every concept, the analyst also considers the *actions* that can be performed on it and whether these actions are easy, difficult in some way or impossible.

Finally, the analyst might want to consider *relationships* between concepts. These are particularly important when the analysis focuses on changing the state of the system, since changing one thing may change another; but there are other examples where the relationships between concepts play a minor role and can be put aside. In this paper, we do not discuss relationships in detail: such detail can be found elsewhere (Blandford, Green and Connell, 2005; Blandford, Connell and Green, 2004).

Data sources

CASSM does not fit into the classic mould of either an expert evaluation technique (such as Heuristic Evaluation or Cognitive Walkthrough, which assume that the analyst understands the user's perspective from the outset) or an empirical evaluation approach. A CASSM analysis generally requires access to users, in order to gather their perceptions of the domain in which they are working. It also requires access to some form of system documentation (and ideally the running system if it exists yet).

For gathering user concepts, some form of verbal data is helpful. This might be from a think-aloud protocol (of the user working with a current system), from Contextual Inquiry interviews (Beyer & Holtzblatt, 1998), from other kinds of interviews (e.g. Critical Decision Method interviews (Hoffmann, Crandall & Shadbolt, 1998)) or from user-oriented documentation – for example, describing user procedures for completing tasks. The more sources of data, the better, but this has to be balanced against any need for speed, efficiency or the practicalities of accessing different sources. While explicitly articulated concepts are the easiest to identify and record, tacit understanding can sometimes be inferred from observational data. For example, the user of a hypertext who never articulates the idea of a “link” but nevertheless navigates seamlessly between pages can generally be assumed to be comfortable with that concept.

Data sources should give information about how the system is used in its normal context of use. This is particularly important: to return to our opening example, if the user task had been presented as “select a direct flight from London

Heathrow to San Francisco”, few (if any) misfits would have been identified: it is important that the user task is the domain task of planning a journey rather than the device one of selecting a flight that is known to exist.

In this paper, we present examples of CASSM analyses that illustrate different data sources for the user side:

- A robotic arm example, where we did not have access to real users, so had to make do with user-oriented system documentation and some brief video clips of the system in use. This is not ideal, but shows what is possible in limited circumstances.
- A digital library, where we collected think-aloud protocols from four library users.
- A drawing tool, where we drew on our own experience of working with the system as well as earlier empirical studies of drawing tools (Connell *et al.*, 2003).

For gathering underlying system concepts, some form of system description is needed. If a prototype system or full implementation is available, that is clearly a useful source; other descriptions can include user manuals or system specifications.

- For the robotic arm, a prototype system existed, and the developer helped clarify our understanding of the design.
- For the digital library, we had access to the functioning system, but no access to documentation or developers.
- For the drawing tool, we had access to the running system and standard user documentation.

Finally, interface concepts can be gathered if the analyst has access to a working system or an interface description.

Who is the ‘user’ and what is the ‘system’?

Identifying the ‘user’ is harder than it might first appear. Notionally, the ‘user’ is a typical individual who interacts with the rest of the system. The user description is usually the result of merging the descriptions from several individual users who have similar perceptions of the system with which they work. In the digital library example, data from four users has been integrated into a single coherent account.

The ‘system’ is not just a computer system, but is the ‘rest of the system’ with which the user interacts. This can make describing the ‘interface’ challenging at times.

- In the robotic arm example, the system is the computer interface to the arm controller, and also the arm itself – everything that mediates between user and ‘real world’.
- In the digital library example, we have analysed the ‘system’ as being the user interface and underlying search engine, document database, etc., but not the broader environment within which the library might be located.
- The drawing tool has also been analysed as a stand-alone system.

By contrast, in an earlier analysis of an ambulance control system, in which the individual user interacts with several pieces of technology as well as colleagues (Blandford *et al.*, 2002), the ‘system’ was a much more complex set of interoperating sub-systems, each of which has an interface to the user group being studied.

As with many kinds of qualitative data analysis, it is not possible to lay down clear rules on how to group users together into a ‘typical user’ or where to draw the boundary of a ‘system’. The decisions taken should be appropriate to the purpose of the analysis. There is no unique right answer. As discussed below, this is an aspect of expertise in being a usability practitioner.

Analysis process

There are different ways of conducting a CASSM analysis, but we have found that the following generally works well. More detail can be found in the tutorial (Blandford *et al.*, 2004). Analysis can be terminated at the end of any phase, depending on the rigour required; this means that the costs associated with analysis can be determined by the analyst.

User–system concepts: The first step is to extract user and system concepts from the data and simply compare them to see how well they fit with each other. One way to conduct an analysis is to go through a transcription of users talking or documentation, highlighting words that might represent core concepts within the user’s conceptualisation of the system they are working with. For the system description, it is important to focus on the underlying system representation rather than interface widgets. For example, a CASSM description of a flight booking system would focus on concepts such as places, prices, carriers, dates and times rather than buttons and links. This first phase of analysis yields surface misfits.

Entities and attributes, and the interface: The next steps are to start distinguishing between entities and attributes and to pay attention to the interface. This step includes consideration of whether each concept is present, difficult or absent for the user, in the underlying system and at the interface. This deeper analysis helps identify more surface misfits.

Actions: Once the interface is sufficiently well specified, the next step is to consider actions and how the user changes the state of the system. For an entity, the actions are *creating* and *deleting*; for an attribute, they are initial *setting* of and *changing* the value. Considering actions serves two purposes. First, actions that are hard or impossible indicate further surface misfits. An action may be hard for various reasons: maybe it involves a long and tedious action sequence or it is difficult for the user to discover the affordance. Impossible actions fall into two categories: those which might be required but which the user *can't* perform (these are problematic) and those aspects of a system that are *fixed*, and rightly so (these are not problematic). Second, action information is used for identifying some structural misfits, as discussed below.

Structural misfits: The final step is to consider structural misfits. There are two ways to go about doing this. The textbook one is to add information about relationships to the analysis, and use the full CASSM specification to derive structural misfits (Blandford, Green & Connell, 2005). The 'unofficial' one, which often works better, is to take definitions of structural misfits (such as 'viscosity') and assess whether they apply to any aspect of the system as designed, then retrospectively work out what that must mean for the CASSM model, in terms of concepts, actions and relationships. This reverse engineering can help to highlight valuable concepts that would otherwise remain implicit.

As noted above, analysis can stop after any of these phases; also, a thorough analysis typically involves some iteration through the different phases as the analyst gets a deeper understanding of the system. However, not every analysis has to be thorough to be valuable. The additional costs need to be weighed against potential benefits. As discussed above, the concern is more with usefulness of analysis than inter-rater reliability or thoroughness.

CASSATA: THE TOOL

Cassata (Green & Blandford, 2004) is an open source tool to support CASSM analysis. It is implemented in the cross-platform language Tcl/Tk, and has been tested on both PCs and Macs. It has supported a range of CASSM analyses of varying levels of complexity, most of which have been made available in a library of worked examples (Green & Blandford, 2004). Cassata supports both data entry and limited forms of data analysis (i.e. checking a system description against definitions of various Cognitive Dimensions).

An example of a Cassata window is shown in Figure 2. Here, the main area of the grid shows entities and attributes, and their properties. The columns denote the following information: E/A indicates whether the description is of an entity or an attribute; the second column is the name of the entity or attribute (attributes are right-justified, and are those of the immediately preceding entity); the following three columns indicate whether the concept is present, difficult or absent for the user, interface and system (respectively); the next two columns describe how easy it is to create or delete an entity, or set or change an attribute; and the final column is for the analyst to enter notes – e.g. to explain something in the analysis or to record an insight about the system being analysed. This last column is included because much of the value of conducting an analysis comes from the insights that emerge while doing it, rather than from inspecting the resulting model: this is a place to capture those insights. The rows at the bottom of the table (below the obvious boundary) include information about relationships. For example, in Figure 2 we have recorded that the user-centric concept of 'position of an object', which is not available from the system, has to be deduced from ('maps-onto') the concept of 'position of gripper', a concept which *is* represented within the system. In a later example, we shall illustrate some other possible relationships.

| entities and attributes | | U | I | S | s/c | c/d | notes |
|-------------------------|------------------|---------|---------|---------|-------|----------|---|
| E | object in world | present | absent | absent | fixed | fixed | e.g. light switch, cup |
| A | position | present | absent | absent | fixed | indirect | any particular object may only have some attributes |
| A | configuration | present | absent | absent | fixed | indirect | may include orientation |
| A | speed | present | absent | absent | fixed | indirect | |
| E | gripper | present | present | present | fixed | fixed | not actually a problem |
| A | position | present | present | present | fixed | hard | |
| A | orientation | present | present | present | fixed | hard | not from documentation but clearly important |
| A | speed | present | present | present | fixed | hard | |
| A | openness | present | present | present | fixed | hard | not from documentation |
| E | joint | present | present | present | fixed | fixed | might include 'whole arm' |
| A | speed | present | present | present | fixed | easy | 5 possibilities |
| A | direction | present | present | present | fixed | easy | different directions apply to different joints |
| E | input device | present | present | present | fixed | easy | user has to choose (unless preconfigured). each has different features |
| E | menu | present | present | present | fixed | fixed | |
| A | current | present | present | present | fixed | hard | some menu transitions are easy, others hard |
| E | menu option | present | present | present | fixed | fixed | |
| A | current selected | present | present | present | fixed | hard | may be difficult -- either due to voice recognition problems or timing of gesture |
| E | cursor | present | present | present | fixed | fixed | this applies only to gestural interfaces |
| A | speed | present | present | present | fixed | easy | |
| A | option indicated | present | present | present | fixed | bySys | |
| E | vocabulary | present | present | present | fixed | fixed | user has to remember repertoire or gestures and/or commands |

| R | actor | type | acted_on | U | I | S | notes |
|---|---------------------|-----------|-------------------------------|---------|--------|--------|-------|
| 0 | gripper.position | maps_onto | object in world.position | present | absent | absent | |
| 1 | gripper.orientation | maps_onto | object in world.configuration | present | absent | absent | |
| 2 | gripper.speed | maps_onto | object in world.speed | present | absent | absent | |
| 3 | | | | | | | |

Figure 2: CASSM description of robotic arm. The upper table describes entities (left-justified) and their attributes (right-justified in the same column). The lower table describes relationships. The contents of the table are described in Example 1.

Using a tool turned out to have several virtues, some of which we did not anticipate. First, it made the recording of an analysis much faster. Certain columns of the analysis table can only contain a small range of entries, such as ‘present’, ‘difficult’ and ‘absent’, and Cassata allows these to be quickly selected from a click-through menu.

Second, the tool made the analyses much more consistent, in terms of encouraging consistency of approach. During development of the method, before a tool was developed, we found that each analyst subtly modified the analysis method, making it difficult to compare analyses.

Third, the tool encourages more complete analyses. This is a most important contribution. It is all too easy to forget to analyse whether a particular attribute can be changed as well as set, or to forget to consider whether it is as easy to delete entities as to create them.

Finally, the Cassata tool contains built-in algorithms for the detection of certain types of usability issue. Some of these are at the level of surface misfits, such as entities that can be created but not deleted; others are at the structural level, such as the viscosity problem illustrated below. At least one new user of Cassata declared that the built-in algorithms were much the most important contribution.

As discussed above, inter-rater reliability and thoroughness (requirements C and D in Figure 1) were of less concern to us in our development than validity (requirement E) and cost effectiveness (requirement J). However, the consistency encouraged by the tool made the scope of the method clearer (Blandford *et al.*, forthcoming); completeness of analyses contributed to cost effectiveness; and consistency contributed to quality (and comparability) of analyses.

EXAMPLE 1: THE INTERFACE TO A ROBOTIC ARM

Our first example considers a novel but simple device for which our only access to real users was possession of 6 video sequences showing the device in use (the device was subsequently destroyed in a flood, making it impossible to gather more user data); however, we did have contact with the developers. This example is included because of its simplicity and because it illustrates an analysis based on limited data. This shows what is possible with limited data; it also means that we can present all the data and ‘walk through’ the analysis in this paper. This analysis was completed as part of an exercise in comparing the scope of CASSM with that of other evaluation approaches (Blandford *et al.*, forthcoming).

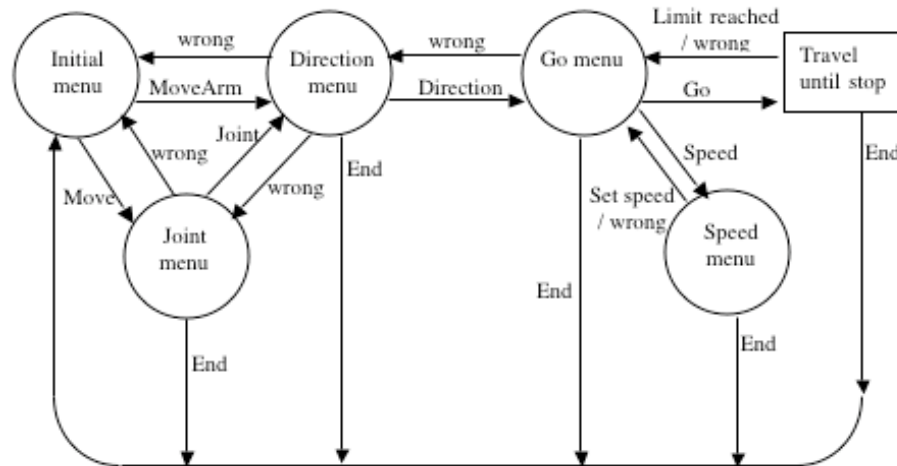


Figure 3: A State Transition Network diagram of the system.

The system was a robotic manipulator for use by wheelchair-bound people (Parsons, Warner, White & Gill, 1997). The manipulator was to be used in a domestic context for everyday tasks such as feeding and grooming, and was developed to prove that a sophisticated manipulator could be produced at reasonable cost: usability issues were considered informally, if at all.

The description of the system that was used for generating the CASSM analysis was as follows. This description is user-oriented, but was produced for us by the system developers. This is the only verbal data available about the system: the limited video data was used to validate findings, but could not be used to generate them.

Significant concepts are discussed more fully below. Some concepts, such as “motor control commands in a special command language to a dedicated microprocessor”, are not considered because it should not be necessary for the user to be aware of these details: one of the challenges of analysis is that the analyst has to reflect on what users actually need to know. While the motors and motor control commands will matter for maintenance and upgrade of the system, they should not matter to the user in normal operation.

Some less relevant sections of the original text have been paraphrased or omitted for clarity; these are marked in square brackets. The system is...

[...] a robotic manipulator for use by wheelchair-bound people. The arm is intended to be used in a domestic context for *everyday tasks such as feeding and grooming* [...] The arm consists of eight *joints*, powered by motors, which can move either *individual joints* or the *whole arm* at once, via the *input devices*.

The *input devices* interface with a Windows-based application which in turn sends motor control commands in a special command language to a dedicated microprocessor, which actually controls the movement of the arm. [Consider the task of moving] the robotic *arm* to a certain *position*, without making use of any pre-taught positions, as though it were to be used to turn on a *light switch*. [...] From the *main menu* of the application this covers the *options move and movearm*. Move allows the user to specify a *particular arm joint* and in what *direction* it can be moved, as well as controlling its *speed*. Movearm allows the user to move the *arm as a whole* in a particular *direction*. At present there is no feedback to the user other than that provided by the visual feedback of the arm's *position*.

The interface [...] is going to be implemented as a Windows application, using a menu format. *Menu options* will be selected in order to operate the arm. There are two *methods of input*, which can be used concurrently or as alternatives.

The *gesture input system* [...] allows a variety of unique gestures to form the *gesture vocabulary*. The *gesture system* is presently implemented so that a *cursor* moves along underneath the *menu options* continuously in turn, and if the *correct gesture* is made when the cursor is *underneath a particular option*, then that *option is selected*. Another gesture acts as a toggle between *high and low speed* of the *cursor*. A final gesture is an *escape option*, which automatically *stops the arm* if the arm is moving, and returns the user to the *main menu*.

The *voice recognition system* allows direct menu *option selection* simply by saying the menu option out loud.”

A high-level description of the menu options and possible transitions is provided in Figure 3. This has been derived from both the written description and discussions with the developers. The core concepts that can be extracted from this written description are as follows:

- Moving an *object* to a *position* (e.g. a light switch to the ‘on’ position, or a brush through the hair) is a core task. The *orientation* of the object might also matter, as might its *speed* (e.g. moving a hairbrush for grooming).
- *Joints* (powered by motors) – we do not distinguish between individual joints and the whole arm initially.
- *Input devices* (the user has to choose between these, depending on their physical abilities). These are also referred to in the text above as “methods of input”, of which instances are the “gesture input device” and the “voice recognition system”.
- Moving the “arm to a certain position” actually means having the *gripper* at the end of the arm in a certain *position*, and probably *orientation* and *speed*. Initially, we do not consider the case where the positions of individual joints on the arm matter (though this could be important in a restricted space).
- There are various *menus*, and *options* within those menus, from which the user has to make a *selection*.
- There is a *cursor*, which is *located on a menu option*, and which moves with a particular *speed*.
- For each input system, the user has to remember the *vocabulary* (the gestures or voice commands).
- The way to move an object is to move the gripper while it is holding that object (or pressing against that object), so there are *relationships* (which we call *maps_onto*) between the position, orientation and speed of an object and of the gripper while it is in contact with that object.

These ideas can be captured in Cassata as shown in the “entities and attributes” column of Figure 2. The final bullet point is represented as relationships between attributes (those of the gripper and corresponding attributes of the object in the world).

Having laid out the concepts, the next step is to consider whether they are present to the user, represented at the interface (considering the menu interface and the actual arm as together comprising the interface), and represented in the underlying system model. Without access to real users, it was necessary to use judgment to assign values for this. Since it was expected that users would be trained on the system, it was judged that all items would be known to the user. In contrast, information about objects in the world was not available to the system. Similarly, in the relationships table, the mappings between the gripper properties and those of the object are known to the user but not represented at the interface or in the underlying system.

Moving on to consider how the state is changed, we note that the user cannot create or delete any of the entities, but that this is not problematic (all are “fixed” in the system); we assume that the values of attributes are initially fixed too (they have some default setting when the user starts to use the system), but that the user can change the values of attributes (such as speed, position and orientation). Again, the judgment of whether they are likely to be easy, difficult or impossible was based on documentation and discussions with the developers. As shown in Figure 2, we judged some changes to be easy and other hard. The most interesting set of assessments is that pertaining to the object in the world – namely that it can only be moved indirectly (by manipulating the gripper rather than the object itself).

From the process of conducting this sketchy analysis, some potential difficulties of using the system became apparent, suggesting promising design changes:

- The user has to align the gripper with objects in the world (e.g. the light switch), in terms of position, speed and orientation. The mapping from the one to the other is non-trivial, particularly if the user has limited movement. In particular, the user may have difficulty judging how far away something is and getting the speed right on approach. While the use of pre-taught positions reduces this difficulty for routine and predictable activities, it will be a challenge for non-standard activities. A more sophisticated mechanism for supporting users in this detailed manipulation would greatly improve usability of the device.
- For grabbing objects, the user has to get the orientation and openness of the gripper right. Again, specific support should be provided for fine-tuning the relationship between the gripper and object.
- The positions and movements of most joints will usually be of no direct interest to the user except when there is some obstacle to be circumvented. The user’s main interest is likely to be in the properties of the gripper, and therefore the main task will involve moving the whole arm. The menu does make this task the easiest to perform (see Figure 3), but the requirement to re-select MoveArm before every direction change means that moving the whole arm is more

complex than strictly necessary, suggesting a re-design of the menu to enable the user to specify a series of directions one after the other.

- Both input devices pose some difficulties: of accurate voice recognition, or of timely and appropriate use of gesture. Users need to learn the vocabularies necessary to use the device(s) of their choice – probably not something that can be designed out without a radical design change.
- Some menu transitions are difficult. A task-oriented analysis would help to further probe this issue and propose detailed design solutions.
- An additional point emerged semi-independently of the CASSM analysis, which is that while the user is looking at the gripper to get its attributes right, (s)he cannot also be looking at the screen to work with gesture control. Although this point is not explicitly represented in the Cassata model, it emerged in the course of doing the analysis through thinking about the different entities (grripper and cursor) and how the user would divide attention between them, given the recognized importance of the gripper position.

Other, more focused, analysis methods, such as Cognitive Walkthrough (Wharton *et al.*, 1994), identify detailed interaction problems – for example, a possible confusion between the terms “move” and “movearm” as the user is making a selection at the interface – which do not easily emerge from a CASSM analysis. Conversely, issues concerning the mapping between the device and the real world objects would be less likely to emerge from a Cognitive Walkthrough than they have from the CASSM analysis, though it partly depends on the expertise of the analyst and their prior experience of this kind of system. The insights obtained from applying any method depend on both what features the method encourages the analyst to consider and the expertise of the analyst. A tool orients the mind of the analyst in a particular direction – to be more sensitive to some issues and less sensitive to others. This issue is explored in more detail by Blandford *et al.* (forthcoming), who report on a comparison of the findings from eight different analyses of this robotic arm. That study provided video evidence to support most of the findings from the CASSM analysis – for example, showing the arm moving in one direction, stopping, then being moved back, indicating difficulties with fine movement. It also showed that CASSM facilitates evaluation in terms of conceptual fit, which the other methods that were applied did not, but that other methods are better suited for reasoning about issues such as timing, task structure or the modalities used in interaction (criterion H1 in Figure 1).

EXAMPLE 2: A DIGITAL LIBRARY SYSTEM

The first example was chosen to illustrate the early stages of CASSM analysis. Our second example is drawn from an analysis of the ACM digital library (DL), based on observations of four library users. Our purpose in presenting extracts from this analysis is to illustrate the use of verbal protocol data to inform CASSM analysis.

Participants were all students completing a Masters course in Human–Computer Interaction and all working on their individual research projects at the time of the study. These participants are representative of one important class of users who make regular use of the ACM DL. Participants were invited to search for articles relevant to their current information needs, and to think aloud while doing so. Occasionally, the observer intervened with questions in the style of Contextual Inquiry (Beyer & Holtzblatt, 1998) to encourage participants to talk more explicitly about their understanding of the systems they were working with. Participants were invited to work with whatever information resources they chose; in practice, all four chose to work mainly with the ACM DL. Audio data was recorded and transcribed, and notes were made of key user actions. In this analysis, five categories of user concepts were identified:

- Concepts concerning the information they were looking for, relating to the topic of the search, the domain within which they were searching, particular ideas they were interested in and particular researchers who were pertinent to their interests.
- Features of the resources (the ACM DL, other digital libraries, the HCI Bibliography, Google and the Web of Knowledge) that they used.
- Concepts from the domain of publishing, relating to journals, articles, publishers, authors, etc.
- Concepts pertaining to electronic search, such as query terms, results lists and their properties.
- Features of themselves as consumers of information, including their interests, domain knowledge, search expertise, access rights to particular resources and research direction.

For the purpose of tracing how verbal protocol data feeds into analysis and helps identify new design possibilities, we focus on the first of these categories: how users described what they were looking for.

Participant 1 was looking for fairly focused material:

“I was looking at something on ‘language patterns’ yesterday on Google, so I’d like to have a look a bit more on that. It was originally thought up by Christopher Alexander, who is usually in design and architecture but I think there might have been studies applying the work to HCI.”

He regarded this as *“an unspecified topic”*.

This participant is talking not just about a topic (how pattern languages can be used in HCI), but also about a particular idea (pattern language), the researcher who originated that idea (Christopher Alexander) and domains of research (design, architecture, HCI). He reinforced some of this understanding, saying:

“Christopher Alexander is the guy that has been attributed to coming up with the idea.”

And *“I’d go to the ACM because it’s an HCI topic”*.

He illustrated his understanding of the cross-referencing of material between texts (that the work of one researcher would be referred to in a work by another), saying that he *“came across it by accident in a book by Heath and Luff.”*

He also recognized that there were different ways of expressing this topic as a search query. For example:

“I know the topic I want is either ‘language patterns’ or ‘pattern languages’.”

Participant two also started with not just a topic (focus groups) but the view that the material of interest would pertain to a particular domain (HCI):

“I was thinking of looking for how focus groups are used in HCI in terms of evaluating initial designs and stuff like that.”

She recognized that she was uncertain about how to describe this topic to the search engine:

“I’d put in ‘focus groups’ using quotes so that, well I’m guessing that it deals with it as one word and, I’m not sure. I’ll put a plus. I’m never quite sure how the search, I wouldn’t say random, but you seem to get funny stuff back from the ACM in terms of results.”

She also recognized that there was material returned in response to a search that was of more or less interest to her:

“I’d just go through some of this stuff and see what I find interesting. Year 2000 focus groups. I’m thinking that’s about the year 2000 bug, so it won’t be that relevant.”

Participant 3 was also concerned with the interestingness of particular topics:

“I’m specifically interested in designing interactive voice systems, so I wouldn’t be interested in a lot of these titles.”

He had some difficulty identifying suitable search terms and, like participants 1 and 2, was concerned about finding articles in the domain of HCI:

“I’ll change it to ‘voice response usability’ because ‘usability’ is fairly specific compared to ‘design’, which could be technology, or architecture: it could be in a different domain. And ACM’s big, it’s got all the different disciplines within it.”

Participant 4 expressed an understanding of broader and more narrow topics as a focus for search:

“I’m going to look for something on qualitative data analysis as a broad thing and see if I can get something to do with dialogue analysis or something to do with conversation.”

Participant 4 echoed themes from earlier participants concerning topics and usefulness (which, for the purposes of this analysis we assume is the same as interestingness: it is unlikely that differentiating between these concepts will be useful for the analysis):

“This one says something about dialogue structure and coding schemes and although it’s not really to do with my topic, it might be useful.”

She also indicated that she was exploring how to describe the interesting topic in search terms:

“I’m going to go back to my original search list and put in ‘dialogue’ and ‘coding’ because I hadn’t thought about looking for that term.”

| Entity / attribute | User | Interface | System |
|------------------------|---------|-----------|---------|
| topic | present | difficult | absent |
| specificity | present | absent | absent |
| features in domains | present | absent | absent |
| how expressed | present | absent | absent |
| interestingness | present | absent | absent |
| idea | present | difficult | absent |
| originator | present | absent | absent |
| domain | present | difficult | absent |
| researcher | present | present | present |
| works in domain | present | absent | absent |
| generated idea | present | absent | absent |
| wrote paper | present | present | present |
| name | present | present | present |
| ref'd in (other) paper | present | present | present |
| collaborated with | absent | present | present |

Figure 4. Concepts relating to information sought

These extracts show how the same concepts are expressed (albeit in slightly different ways) by multiple participants, giving assurance that they are significant concepts from a user perspective. These concepts are summarized in Figure 4. This data table has been exported from Cassata.

The concepts that are present for the user emerge directly from the analysis. To form the assessments relating to system and interface, it was necessary to look at the ACM DL and its interface. For example, there was no direct representation of topics, specific ideas or domains in the underlying system, so these are listed as being absent from the system representation. However, by reading the interface representation, such as the titles or abstracts of papers, the user can (probably) infer the topic, ideas and domain, so they have been labelled as ‘difficult’ in Figure 4. Nevertheless, most attributes of these entities are generally absent from the interface. The user is forced to find work-arounds for expressing these concepts. If we consider how they might be represented in the DL (e.g. exploiting existing meta-data), we realise that these ideas are similar (but not identical) to the ‘general terms’ and ‘index terms’ in the ACM classification system. We could imagine an implementation of the search facility that allowed the user to easily prioritize (for example) articles that included the general term “Human Factors”. The current implementation of the ACM DL hides this possibility, and none of our participants discovered it. Participant 3 expressed this succinctly:

“ACM’s big, it’s got all the different disciplines within it, so I’m just trying to focus in on usability related stuff.”

This indicates a promising design change: to make it easier for users to focus a search by particular general terms or classifiers within the ACM classification system (or, indeed, to be able to browse by classifier).

One concept that the DL represents comparatively well is that of a researcher (or at least, an author). Indeed, although information about what domain an individual works in and what ideas they have originated is poorly represented, other information about them – notably people they have co-authored with – is clearly represented through a link to “collaborative colleagues”. Although none of the participants in this study referred to the idea that an individual might collaborate with others, or hinted that that might be an interesting or important item of information, this information may be of use to other user groups of the ACM DL (this is an empirical question). In this case, the representation of a concept (who collaborates with whom) within the DL does not represent a difficulty to the user, because the use of this feature is discretionary.

In this example, we have shown how an analysis of verbal data and the existing DL implementation can highlight both strengths and weaknesses of the conceptual model underlying the current system design, and identify new design possibilities that – in this case – would represent incremental improvements to the interaction (criterion A2 in Figure 1).

EXAMPLE 3: A DRAWING TOOL

The two examples discussed so far have only considered surface misfits. Our final example focuses directly on structural misfits in a system where a central issue is change of state and its consequences. These misfits may exist in any system that permits users to design (or construct) new representations. Examples include word processors (which allow users to create and change documents), drawing programs, music editors, HTML editors and programming environments. Structural misfits exist if changes that are conceptually simple for the user are difficult or unwieldy to perform using a particular system. Systems that enable the user to engage in a design activity are typically more complex than our first

two examples, so this analysis has a different flavour from the two above: it omits the initial analysis of surface misfits and focuses directly on the more detailed analysis required to reason about structural misfits.

This example is taken from a study of a drawing tool. Surface misfits included users having difficulty learning about the ‘handles’ used to resize objects and the ‘layers’ that objects reside on. However, we focus on the difficulties of making changes to a drawing once it has been constructed. For example, adding a child onto a family tree may require many individual actions as illustrated in Figure 5: Freda, Graham and Maria all need to be moved (each by a different distance), as do the lines connecting them to their parents; the horizontal lines need to be re-sized; and Helen and her line need to be inserted. This particular example illustrates the CD of *viscosity* (Green, 1990), or resistance to change. Green (1990) distinguished two types of viscosity, repetition and knock-on. An informal example of repetition viscosity is going through a large batch of images by hand and changing the resolution of each one, without the benefit of any batch-processing macro; the user has to repeat the same action many times. An informal example of knock-on viscosity is changing the size of an image in a paper and then having to inspect the rest of the paper to check for any bad page breaks that result, and correct them, plus perhaps re-computing the table of contents if the pagination has changed; here the user has to clean up the mess caused by the first action. (In North America, a better-known phrase for the same concept appears to be ‘domino effect’.)

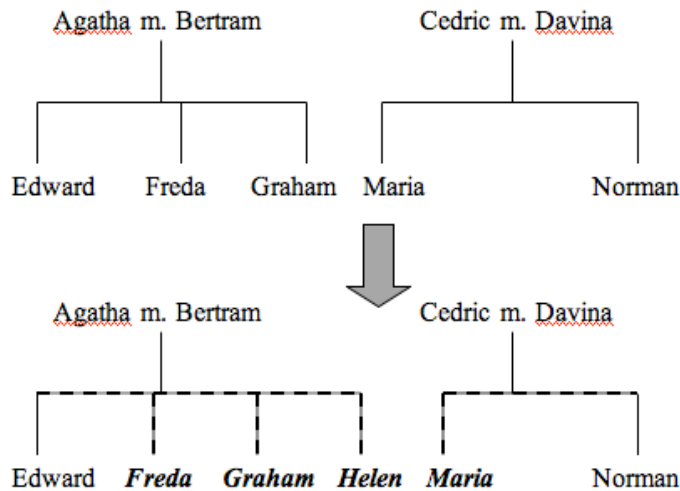


Figure 5. A simple conceptual action requires many device actions (dashed lines are those that needed to be moved or resized; names in italics had to be moved or inserted).

More formally, repetition viscosity occurs when what the user sees as a single action within his or her conceptual model cannot be performed as a single device action, but instead requires repetitive actions. This has been formally defined (and implemented in Cassata) as follows. Changing attribute P of entity A, A(P), needs many actions if:

- A(P) is not directly modifiable
- B(Q) affects A(P)
- B(Q) is modifiable
- A consists-of B

In the case of a family tree, the position of the family (as it appears on the printed page) cannot be directly changed (except if the whole family is selected and moved as a block, which cannot be done when new members are being added or there are space constraints). The position of the family is defined by the positions of all its members, which can be individually changed. In CASSM terms:

- family(position) is not directly modifiable
- member(position) affects family(position)
- member(position) is modifiable
- family consists-of member

In this case, the intuitive idea of ‘repetition viscosity’ is easier to work with than its formal representation, but the exercise of working out why such a drawing tool is viscous (for tasks such as preparing a family tree) has highlighted some of the important concepts (families, their members and their relative positions) that pertain in this case.

The second type of viscosity is knock-on viscosity: the property that changing one attribute may lead to the need to adjust other things to restore internal consistency. Changing A(P) has possible knock-on if:

```
A(P) is modifiable
A(P) affects B(Q)
there is a goal constraint on B(Q)
```

In the case of a family tree, we have intuitively the idea that the structure of a family is immutable (the order in which children arrive cannot be changed!), that the visually represented position of a person in the family tree reflects the family order, and that the family order as displayed should be the same as the family order ‘in the world’:

```
member(position) is modifiable
member(position) affects family(order)
there is a goal constraint on family(order)
```

Again, the informal understanding of viscosity seems to be easier to work with than the formal one, but the act of formalising highlights the fact that this example suffers from both kinds of viscosity, and highlights some of the important user concepts (position in family, etc.) that might otherwise remain implicit. Figures 6 and 7 show (respectively) the description of this example in Cassata and the output of automated analysis of this example. The relationships shown in Figure 6 are represented in a standardised format (‘actor’, ‘relationship’, ‘acted-on’) to express the ideas outlined above. As shown in Figure 7, the automated analysis identifies an additional, possibly spurious, case of repetition viscosity because the test does not include a check of whether it is reasonable to change the family order.

The screenshot shows the Cassata software interface for a model named 'family tree'. It displays two tables:

| entities and attributes | | U | I | S | s/c | c/d | notes |
|-------------------------|----------|---|---|---|----------|----------|-------|
| E | member | | | | hard | easy | |
| A | position | | | | easy | hard | |
| E | family | | | | easy | easy | |
| A | order | | | | indirect | indirect | |
| A | position | | | | easy | indirect | |
| E | | | | | | | |

| R | actor | type | acted_on | U |
|---|-----------------|-----------------|-----------------|---|
| 0 | | goal_constraint | family_order | |
| 1 | member.position | affects | family_order | |
| 2 | family | consists_of | member | |
| 3 | member.position | affects | family.position | |
| 4 | | | | |

Figure 6. Cassata description of the family tree example.

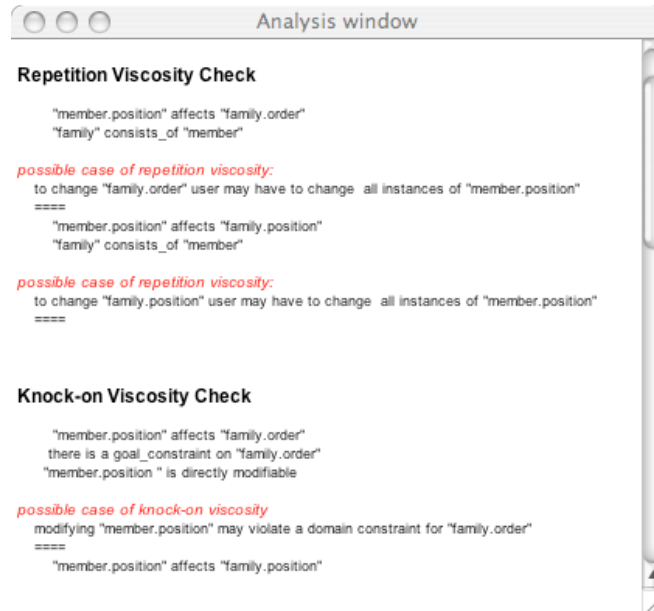


Figure 7. Automated analysis of the family tree example.

These insights might be used to help design drawing software that is better tailored to the requirements of tasks such as drawing a family tree. As argued elsewhere (Blandford *et al.*, 2005), viscosity can be found in many systems: for example, when we decided to re-order the figures in this paper, it was necessary to change each reference to any of the figures to maintain consistency. Blandford *et al.* (2005) present formalisations of other structural misfits, including premature commitment and hidden dependencies, that are not relevant to the example presented here.

LEARNING CASSM: INFORMAL FINDINGS

From the earliest stages of development, we have conducted informal evaluations of the CASSM approach, by including it in the curriculum of various HCI modules and setting class exercises that involve completing CASSM analyses. The findings have been mixed, with some individuals or cohorts finding CASSM very straightforward to apply in ways that have yielded powerful insights, and others simply not “getting it”. Students with a background in entity-relationship modeling typically find the basic representation obvious to work with whereas those without this background sometimes struggle initially with the idea of a ‘concept’. We have found that including a lecture on knowledge representation and mental models before teaching CASSM has been helpful for getting this basic idea across. In addition, sessions on qualitative data analysis based on Grounded Theory (Strauss and Corbin, 1998) help students to learn to identify concepts within verbal protocol data. Informally, it appears to us that learning to conduct a CASSM analysis is rather like learning to swim: there is often an initial period of floundering around, apparently getting nowhere, and then suddenly it all comes together and people understand the way of thinking. From then on, it becomes a question of refining an acquired skill.

As noted above, the key to the way of thinking is to step back from a particular system design or implementation to comprehend what people are doing at a deeper level, and how the system design is supporting or hindering that activity. This can be difficult to do because the tool mediates the activity (Bertelsen and Bødker, 2003) and hence frames the way the people think about the activity, so that people adopt device concepts to talk about domain tasks. This was particularly evident in one of our earlier analyses of an emergency medical dispatch systems (Blandford *et al.*, 2002), where ambulance dispatchers talked about “calls” (which were implemented in the dispatch system) when they were actually referring to incidents (about which there might be several calls). Teasing apart domain and device concepts in such situations is inherently challenging, but can yield important insights when done effectively. To continue with the emergency dispatch example: when the distinction between calls and incidents was fed back to controllers, they became better able to articulate their requirement that the system should help them relate calls to incidents and manage incidents – i.e. that both concepts should be explicitly represented in the system, not just calls.

DISCUSSION

Earlier, we discussed our aims in developing CASSM. We are now in a position to revisit those aims and discuss future work.

Filling a niche

As far as we can ascertain, no other evaluation methods are concerned with conceptual misfits, although the idea has been discussed in various contexts, such as Norman's (1986) work on 'gulfs', Moran's (1983) on ETIT and Payne *et al.*'s (1990) on Yoked State Spaces. Within software engineering, system concepts are represented – for example, in entity-relationship modeling – but are generally implicitly assumed to directly map onto user concepts. One exception, which is in the spirit of the work reported here, is that of Hsi (2004), who relates the system to the broader 'ecosystem' within which it sits.

We have conducted two scoping studies to compare the kinds of findings that emerge from a CASSM analysis with those from other evaluation techniques (Connell, Green & Blandford, 2004; Blandford *et al.*, forthcoming). These studies have provided evidence to support the view that CASSM fills a useful niche in the space of evaluation methods (criterion H1 in Figure 1). By focusing on concepts rather than tasks or procedures, CASSM supports the analyst in getting at the semantics and structure of the interaction rather than focusing on the details of procedures. Consequently, it supports evaluation of conceptual design rather than detailed design, and informs new design through a process of co-evolution (criterion A2 in Figure 1).

Ensuring validity

In spirit, the early stages of the CASSM approach share much in common with qualitative data analysis techniques such as Grounded Theory (Strauss & Corbin, 1998), in that insights emerge from systematic analysis of data. The validity of insights depends on the skill and integrity of the analyst(s) as well as the quality of the data on which the analysis is based. In contrast to many evaluation techniques including GOMS (Card, Moran & Newell, 1983), Cognitive Walkthrough and Heuristic Evaluation, a CASSM analysis is expected to draw on user data rather than relying on the intuitions of the analyst; to that extent, the validity of an analysis is less reliant on analyst insight than most evaluation approaches. However, all usability evaluation work is susceptible to analyst biases: this was one of the concerns raised by Hertzum and Jacobsen (2001), and was also found in the study reported by Nørgaard and Hornbæk (2006). Insofar as any qualitative data analysis is conducted by people with their own personal skills, biases and motivations, the analysis will be unique to the analyst; however, if the analysis can be related back to the data from which it is derived, and if it can be further validated by feeding the findings back to users and checking the findings with them, that is the best possible assurance of the validity of the analysis (criterion E in Figure 1).

Supporting iterative deepening

We do not regard CASSM as a highly prescriptive approach, but rather as a framework to support thinking about a design: a lens through which to view a system to bring out particular features and suggest re-design possibilities. In the three preceding examples, we have, as far as possible, walked through the process of conducting an analysis (of course, an actual walkthrough would involve many digressions and iterations that would confuse rather than illustrate). Because CASSM does not demand that the representation be complete and consistent before any analysis can be completed, it is possible to choose the depth of analysis, as appropriate to the situation (criterion J2 in Figure 1). This is consistent with the view of Wixon and Wilson (1997, p.655) that "while science is concerned with uncovering truth, regardless of time or cost, engineering is concerned with the 'art of the possible within constrained resources'": system evaluation and re-design is an engineering enterprise rather than a science (a point also discussed by Nørgaard and Hornbæk (2006)).

One important feature of the process is that it will yield insights that are not necessarily encapsulated in the resulting representation. For this reason, as discussed briefly above, we include a 'notes' section in Cassata. This makes explicit the view that the representation should support but not constrain reasoning. This is consistent with a view of design (e.g. Fitzgerald, 1996) that formalised system development methods are not, and should not be, rigidly adhered to, but that they can provide support as needed (criterion K in Figure 1).

Providing tool support to facilitate analysis

Cassata, has been implemented as an open source development (criterion L in Figure 1). It has been tested in-house, and used to support several CASSM analyses (Green & Blandford, 2004). A library of worked examples has been developed and made available. Future work includes subjecting Cassata to more thorough evaluation, involving people other than the method developers.

Facilitating ease of learning

To facilitate learning, a tutorial has been developed and tested iteratively (criterion I in Figure 1), and a library of worked examples has been constructed. Informal evaluations of the learnability of CASSM have been conducted during the development of the approach, as described above, but no formal study of learnability has been conducted to date. Our experience of teaching CASSM to students indicates that it is more difficult to learn to ‘go through the motions’ than (for example) Heuristic Evaluation. Our discussions with students indicate that the greatest challenges for novices are thinking in terms of concepts rather than processes, and recognizing that the same terms can be used in subtly different ways by users and systems; also that there are important user concepts that are not represented within the system, but that these concepts matter. This appears to increase the ‘entry cost’ in applying CASSM as compared to established approaches such as Heuristic Evaluation and Cognitive Walkthrough. However, we do not know whether the higher entry cost translates into a higher long-term cost or not: the ability to ‘go through the motions’ of applying any approach should not be confused with expertise in applying it (though a basic ability is a prerequisite for developing expertise). Our own experience suggests that the support for iterative deepening means that the higher entry cost is compensated by a lower application cost, but we cannot be sure without conducting a longitudinal study such as that of John and Packer (1995). A more formal evaluation study will yield greater insights into the nature of skill in applying CASSM, and hence how to facilitate the development of that skill.

Facilitating fit with design practice

Another area for further work is investigating how CASSM relates to software development practice. If developers assiduously gathered user requirements in advance of design, a method such as CASSM might be redundant: the necessary user concepts would be identified, represented and implemented. In practice, as illustrated by the fact that there is a rich world of examples on which to test CASSM, this rarely happens. Design and use co-evolve (Carroll and Rosson, 1992), and users work in ways that have not been fully understood or anticipated by developers. CASSM does not make use of established software development representations such as UML (Fowler, 1997) because the data gathering and analysis process are not an established element of such a development process; we envisage that the findings from a CASSM analysis should be synthesised and re-expressed for further use in design, but this has not yet been tested in practice. The CASSM approach has been taken up and used by others outside the development team (e.g. Mann, 2004; Roast, Dearden & Khazaei, 2004), but it inevitably takes many years to assess efficacy in practice.

Summary

In order to be used, any novel usability evaluation method should be shown to be useful and usable. Our criteria for CASSM being useful are that it supports the analyst in achieving valid insights (criterion E) and that it occupies a niche in the space of evaluation methods that is not occupied by any more established method (criterion H1); we have discussed above why these are important criteria and how they relate to criteria identified by others. Our work to date has shown that CASSM satisfies both of these usefulness criteria. Our criteria for CASSM being usable are that it should support iterative deepening (J2), have tool support (L), be learnable (I) and fit with design practice (K); the first two of these criteria have been satisfied, and work is ongoing on the latter two.

The contributions of CASSM can also be viewed from another perspective: in their discussion about the use of think-aloud data for assessing system utility, Nørgaard and Hornbæk (2006, p.216) argue that “Perhaps the lack of systematic analysis is understandable, given the scarce advice about analysis of usability tests.” As discussed above, verbal protocol data is an important source for a CASSM analysis, so CASSM can be regarded as an approach to analysing data from usability tests, and this paper presents “advice” on such analysis. However, as also discussed, the data that is collected and analysed needs to relate to users’ domain tasks, not just tasks with a particular device, to be useful for CASSM analysis, and this means that tasks for think-aloud testing need to be designed to gather rich domain data.

CONCLUSIONS

In this paper, we have presented CASSM, an analytical usability evaluation method that focuses on conceptual fit. We have described how to apply the approach and illustrated the use of the method with three contrasting examples. We have outlined our aims in developing the approach and discussed evidence that the aims have largely been met. CASSM fits a niche in the arsenal of usability evaluation methods; it supports incremental evaluation which allows the analyst to choose how much effort to expend relative to the thoroughness of evaluation needed; tool support helps to structure analyses; and the use of user data helps ensure the validity (though not necessarily the inter-rater reliability) of analyses.

By intentionally supporting sketchy analysis, CASSM avoids the death by detail that plagues some evaluation techniques. CASSM analyses do not have to be complete or consistent to be useful – though of course a thorough

analysis is likely to have these properties. Also, CASSM analyses are typically quite succinct, compared to checklist-based or procedurally oriented analyses.

Further work has been identified concerning the transfer of CASSM into practice. The development of the approach has been informed by iterative cycles of informal testing with HCI students. This testing has, at times, been discouraging: students often found it difficult to identify user concepts from data. Initially, we were worried about these difficulties – we were, after all, setting out to develop a usable method. We now realize, though, that CASSM’s focus on conceptual structures is both what makes it inherently harder to learn than more established methods and also what gives it its power and its ability to highlight design possibilities that are missed by established approaches. It demands a higher level of core expertise in evaluation than most established evaluation methods. However, with expertise it yields rich insights (concerning underlying conceptual representations).

Expertise in evaluation is necessarily built up over time, and each practitioner develops an individual expertise that is based on their accumulated experience and understanding. Any tool or method augments or focuses that expertise, but is never a substitute for it. CASSM supports the analyst to think in a particular way, which we have illustrated through the worked examples above. These analyses are not ‘correct’ or ‘incorrect’: they are valuable insofar as they yield useful insights for design (Wixon, 2003). If they had been conducted by a different analyst, with different experience and motivations, the results would doubtless have been different. We do not regard this as a ‘problem’: different analyses can be equally valid, in that they highlight features of users’ interactions with a system, without being identical. What distinguishes a CASSM analysis from any other is a particular stance (concerning conceptual misfits and co-evolution of design and use) with respect to design.

ACKNOWLEDGMENTS

The work reported here was funded by EPSRC (Grants GR/R39108, GR/S67494 and GR/S84798). We are grateful to the reviewers of an earlier version of this paper for helpful feedback on it.

REFERENCES

- Bertelsen, O. W. & Bødker, S. (2003) Activity theory. In J.M. Carroll (Ed.) *HCI models theories, and frameworks: toward a multidisciplinary science*. San Francisco: Morgan Kaufmann, 291-324.
- Beyer, H., Holtzblatt, K. (1998) Contextual Design. San Francisco : Morgan Kaufmann.
- Blandford, A. E., Buckingham Shum, S. and Young, R. M. (1998) Training software engineers in a novel usability evaluation technique. *International Journal of Human-Computer Studies*, 45(3), 245-279.
- Blandford, A., Connell, I. & Green, T.R.G. (2004) *Concept-based Analysis of Surface and Structural Misfits (CASSM) Tutorial notes*. CASSM working paper from <http://www.ucl.ac.uk/annb/CASSM/>
- Blandford, A. & Green, T. R.G. (forthcoming) Methodological Development. To appear in Cairns, P.A., & Cox, A.L. (eds.) *Research Methods for Human Computer Interaction*. CUP.
- Blandford, A., Green, T.R.G. & Connell, I. (2005) Formalising an understanding of user–system misfits. In R. Bastide, P. Palanque & J. Roth (Eds.) *Proc. EHCI-DSVIS 2004*. Springer: LNCS 3425. 253-270.
- Blandford, A., Hyde, J. K., Green, T.R.G. & Connell, I. (forthcoming) Scoping analytical usability evaluation methods: a case study. To appear in *Human–Computer Interaction*.
- Blandford, A. E., Wong, B. L. W., Connell, I. & Green, T. R. G. (2002) Multiple viewpoints on computer supported team work: a case study on ambulance dispatch. In X. Faulkner, J. Finlay & F. Détienne (Eds.) *People and Computers XVI*. Springer 139-156.
- Blandford, A. E. & Young, R. M. (1996) Specifying user knowledge for the design of interactive systems. *Software Engineering Journal*. 11.6 323-333.
- Card, S. K., Moran, T. P. & Newell, A. (1983) *The Psychology of Human Computer Interaction*, Hillsdale NJ: Lawrence Erlbaum.
- Carroll, J. M. & Rosson, M. B. (1992) Getting around the task-artifact cycle: how to make claims and design by scenario. *ACM Transactions on Information Systems*, 10(2), 181-212.
- Connell, I., Green, T. & Blandford, A. (2003) Ontological Sketch Models: highlighting user-system misfits. In E. O’Neill, P. Palanque & P. Johnson (Eds.) *People and Computers XVII, Proc. HCI’03*. Springer. 163-178.
- Connell, I., Blandford, A. & Green, T. (2004) CASSM and cognitive walkthrough: usability issues with ticket vending machines. *Behaviour and Information Technology*. 23(5). 307-320.
- Ericsson, K. A. & Simon, H. (1993) *Protocol Analysis: Verbal Reports As Data*, Revised Edition, MIT Press, Cambridge, MA.
- Fitzgerald, B. (1996) Formalised systems development methodologies: a critical perspective. *Information Systems Journal*. 6. 3-23.

- Fowler, M. (1997) *UML Distilled*. Addison-Wesley, Reading MA.
- Gray, W. D. & Salzman, M. C. (1998) Damaged Merchandise? A Review of Experiments that Compare Usability Evaluation Methods. In *Human-Computer Interaction* 13(3), pp. 203-261
- Green, T. R. G. & Benyon, D. (1996) The skull beneath the skin: entity-relationship models of information artifacts. *International Journal of Human-Computer Studies*, 44, 801-828
- Green, T. R. G. & Blandford, A. E. (2004) Downloadable package: CASSM tutorial, worked examples, Cassata tool and supporting documentation. From <http://www.ucl.ac.uk/annb/CASSM/>
- Green, T. R. G. & Petre, M. (1996) Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *J. Visual Languages and Computing*, 7, 131-174.
- Green, T. R. G. (1989) Cognitive dimensions of notations. In A. Sutcliffe and L. Macaulay (Eds.) *People and Computers V*. CUP 443-460
- Green, T.R.G. (1990) The cognitive dimension of viscosity - a sticky problem for HCI. In D. Diaper and B. Shackel (Eds.) *INTERACT '90*. Elsevier 79-86.
- Hartson, H. R., Andre, T. S., & Williges, R. C. (2001) Evaluating usability evaluation methods. *International Journal of Human-Computer Interaction*, 13.4. 373-410.
- Hertzum, M., and Jacobsen, N.E. (2001) The Evaluator Effect: A Chilling Fact about Usability Evaluation Methods. *International Journal of Human-Computer Interaction*, 13(4), 421-443.
- Hoffman, R. R., Crandall, B., & Shadbolt, N. (1998) Use of the Critical Decision Method to elicit expert knowledge: A case study in the methodology of Cognitive Task Analysis. *Human Factors*, 40(2). 254-276.
- Hsi, I. (2004) Measuring the conceptual fitness of a computing application in a computing ecosystem. *Proc. WISER 2004 (ACM Workshop on Interdisciplinary Software Engineering Research)* 27-36.
- John, B. & Marks, S. (1997) Tracking the effectiveness of usability evaluation methods. *Behaviour and Information Technology* 16, No. 4/5, 188-202.
- John, B. E. & Packer, H. (1995) Learning and using the Cognitive Walkthrough method: A case study approach. In *Proceedings of CHI'95*. pp.429-436. ACM Press: New York.
- Mann, P. (2004) Design for design: Requirements for graphical communication in computer-supported collaborative work (CSCW) in design. In A. Dearden & L. Watts (Eds) *Proceedings of HCI 2004 (Vol. 2)*. Research Press. 229-230.
- Moran, T. P. (1983) Getting into a system: external-internal task mapping analysis. In A. Janda (ed.), *Human Factors in Computing Systems*. 45-49.
- Nielsen, J. (1994) Heuristic evaluation. In J. Nielsen & R. Mack (Eds.), *Usability Inspection Methods*. New York: John Wiley. 25-62.
- Nørgaard, M. and Hornbæk, K. (2006) What do usability evaluators do in practice?: an explorative study of think-aloud testing. In *Proceedings of the 6th ACM Conference on Designing interactive Systems. DIS '06*. ACM Press, New York, NY, 209-218.
- Norman, D. A. (1986) Cognitive engineering. In D. A. Norman and S. W. Draper, Eds. *User Centered System Design*, Hillsdale NJ: Lawrence Erlbaum. 31-62
- Parsons, B., Warner, P., White, A., & Gill, R. (1997) An Approach to the Development of Adaptable Manipulator Controller Software. In: *Proc. International Conference on Rehabilitation Robotics*.
- Payne, S. J., Squibb, H. R. & Howes, A. (1990) The nature of device models: the yoked state space hypothesis, and some experiments with text editors. *Human-Computer Interaction*, 5, 415-444.
- Roast, C., Dearden, A. & Khazaei, B. (2004) Enhancing Contextual Analysis to Support the Design of Development Tools. In S. Fincher, P. Markopoulos, D. Moore & R. Ruddle (Eds.) *People and Computers XVIII*. Springer. 297-313.
- Strauss, A. L. & Corbin, J. (1998) *Basics of qualitative research: Techniques and procedures for developing grounded theory*. 2nd ed, Thousand Oaks, CA: SAGE Publications, Inc.
- Vicente, K. (1999) *Cognitive Work Analysis*. Mahwah, NJ : Lawrence Erlbaum
- Wharton, C., Rieman, J., Lewis, C. & Polson, P. (1994) The cognitive walkthrough method: A practitioner's guide. In J. Nielsen & R. Mack (Eds.), *Usability inspection methods* John Wiley. 105-140
- Wixon, D. (2003) Evaluating usability methods: why the current literature fails the practitioner. *interactions* 10.4. 28-34.
- Wixon, D. & Wilson, C. (1997). The Usability Engineering Framework for Product Design and Evaluation. In Helander, M., Landauer, T., and Prabhu, P. (Eds.) *Handbook of Human Computer Interaction* (2nd Ed.). Elsevier Science.
- Young, R. M., Green, T. R. G. & Simon, T. (1989) Programmable User Models for Predictive Evaluation of Interface Designs, in K. Bice. & C. Lewis (eds.), *CHI '89 Conference Proceedings*, 15-19.