Concept-based Analysis of Surface and Structural Misfits (CASSM)

Tutorial notes

Ann Blandford, Iain Connell & Thomas Green

September 2004

UCL Interaction Centre
University College London
Remax House, 31-32 Alfred Place
London WC1E 7DP
A.Blandford@ucl.ac.uk
http://www.uclic.ucl.ac.uk/annb/CASSM.html

Contents:

Introduction

Concept-based Analysis of Surface and Structural Misfits (CASSM) in a novel approach to usability analysis that focuses attention on misfits between user and system concepts. We believe that as an approach it has several desirable qualities:

- It focuses on concepts rather than tasks or procedures. Consequently, it complements the majority of existing approaches to usability evaluation. In particular, it analyses conceptual *misfits* between user and system.
- By intentionally supporting 'sketchy' analysis, CASSM avoids the 'death by detail' that plagues many 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 often quite succinct, compared to (for example) a Cognitive Walkthrough (Wharton *et al*, 1994), Heuristic Evaluation (Nielsen, 1994) or GOMS analysis (John & Kieras, 1996).
- As a notation, it provides a 'bridge' between the core ideas underpinning work on mental models and design issues, and may thus make prior work on mental models more readily accessible to design practice. [This should be regarded as a hypothesis that has not yet been tested.]
- The CASSM notation provides a relatively formal definition of many of Green's Cognitive Dimensions (see, for example, Green, 1989; Green & Petre, 1996; Blackwell & Green 2003). In this way, it further supports assessment of a system in terms of CDs. This is discussed in detail towards the end of this document.

Although the name (CASSM: Concept-based Analysis of Surface and Structural Misfits) emphasises the importance of *misfits*, you should be aware that there are other kinds of user–system misfits that are outside the scope of CASSM; for example, inconsistencies in procedures for similar tasks would be picked up by other techniques but are not directly addressed within CASSM. CASSM focuses on ***conceptual structures***.

Structure of document

This document is intended to be a reasonably thorough tutorial on how to conduct a CASSM analysis, though the main points are summarised in the Glossary at the end. In the remainder of this document, we aim to present the principles and process of conducting a Cassata analysis, illustrating the approach with examples taken from our own work. We start by outlining the process. We then discuss surface misfits (the easy kind of misfits), and illustrate the approach so far with two worked examples: a robotic arm (which is a single-user system) and ambulance control (a team activity).

To support CASSM analysis, we have developed a prototype analysis tool called Cassata. Before discussing more complex misfits (i.e. structural ones), we briefly describe Cassata. We then use Cassata as a vehicle for describing structural misfits. Finally, we illustrate the whole process with one further example: the Alhambra heating controller. The Alhambra example is the most completely worked through … which may make it a good reference source, or may mean you skip it completely because it's to detailed – depending on your learning style.

It is worth noting here that Cassata is available for download and use[1]. Since it is a demonstrator system, we do not guarantee that it is totally bug-free, and we know it has a couple of usability quirks itself, but all the models described here have been processed through Cassata without difficulty. It runs under Tcl/Tk, which is publicly available for both

[1] Download from http://www.uclic.ucl.ac.uk/annb/CASSM.html
This tutorial, the Cassata tool, the Alhambra and robotic arm examples discussed in this document and a few other bits-and-pieces can all be obtained together in the "Cassata shrink-wrap" available from the site.

PC and Mac. A separate document (Green & Blandford, 2004) describes the details of how to install and use Cassata. However, while Cassata makes life a bit easier, it is perfectly feasible to conduct a complete analysis without using it.

Overview

The approach taken in this tutorial is 'progressive deepening', so we start with a quick overview, then expand each set of ideas in subsequent sections. Here's the quick overview…

History

CASSM started life as UUUM – Usable, Useful, Used Modelling, a title that captures our core aspirations in developing this approach. Its parentage can be identified in ERMIA (Green & Benyon, 1996), PUM (Young, Green & Simon, 1989; Blandford & Young, 1996) and Cognitive Dimensions (CDs: Green, 1989; Green & Petre, 1996; Blackwell & Green 2003). In developing CASSM, we have aimed to retain the strengths of ERMIA and PUM, as we see them, while avoiding the 'death by detail' that those approaches demand. As discussed below, the work has been strongly informed by CDs, and now provides semi-formal definitions of many CDs.

The first published work on CASSM was under the title 'OSM' (e.g. Connell *et al*, 2003), which stands for Ontological Sketch Modelling – 'ontological' because the approach involves capturing the core essence, or nature, of the system in terms of concepts and relationships; 'sketch' because it is possible to work with sketchy, or partial, descriptions; and 'modelling' because what is produced is an abstract description that supports reasoning (i.e. a model). As work has progressed, we like to think that our understanding of the approach has matured, and that the latest name (CASSM) is more pronounceable, less prone to causing confusion and better captures the nature of the approach.

The CASSM concepts

The focus for conducting a CASSM analysis is on *concepts*.

- A ***concept*** may be an ***entity*** or an ***attribute***. Very early on in analysis, we don't necessarily even bother to distinguish between these two different types of concept, but as analysis proceeds it usually becomes necessary.
    - An ***entity*** is usually something that can be created or deleted within the system. Sometimes, entities are things that are there all the time, but that have attributes that can be changed. In the medical dispatch example discussed below, entities include emergency calls (which are created whenever someone reports a new emergency) and vehicles (which cannot be created or deleted[2], but have attributes such as location and current activity that are important).
    - 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.
- For every concept, the analyst determines whether it is ***present***, ***difficult*** or ***absent*** for the ***user***, at the ***interface*** and in the underlying ***system***. Concepts that are present in one place but absent from another are, broadly speaking, sources of misfits. So are concepts that are *difficult*. This idea is central to CASSM, and is discussed in much more detail below.
- For every concept, the analyst also considers the ***actions*** that can be performed on it. For an entity, these are ***creating*** and ***deleting***; for an attribute, these are (initial) ***setting*** and ***changing*** the value. Again, this idea is expanded on below.

[2] In a more detailed analysis, this would not be strictly true: vehicles become available or unavailable ('created' / 'deleted') at the beginning and end of every crew shift.

- Finally, the analyst might want to consider ***relationships*** between concepts. These are typically only considered in a pretty detailed analysis, so we leave discussion of them until much later.

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 requires access to users, in order to gather their perceptions of the domain in which they are working and – if it exists – of the system that they use to complete tasks in that domain. It also requires access to any available system documentation (and ideally the running system if it exists yet).

For gathering user concepts, some form of verbal data is required from users. 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. CDM: Hoffman, Crandall & Shadbolt, 1998) or from user-oriented documentation – for example, describing formalised user procedures for completing tasks[3]. Basically, 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. Data sources should give information about how the system is used in its normal context of use. In this tutorial, we include three examples of CASSM analyses: a robotic arm, ambulance control and a heating controller. The three examples illustrate different data sources for the user side:

- For the robotic arm example, we did not have access to real users, so had to make do with user-oriented system documentation. This is not ideal, but shows what's possible in limited circumstances.
- For the ambulance control example, we had access to procedure documents; we also conducted Contextual Inquiry interviews with staff in the control room and Critical Decision Method interviews with off-duty controllers, so that we had the ambulance controllers' own descriptions of what they do.
- For the Alhambra heating controller, we relied on two interviews of people describing their own heating controllers (which were different from Alhambra) and how they use them.

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 we also had access to the developer to clarify our understanding of the system design.
- For the ambulance control example, we had access to the fully functioning system, and also conducted an interview with a member of the development team.
- For the Alhambra heating controller, we had access to a system simulation[4] and limited system documentation.

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

[3] A detailed description of data-gathering techniques is beyond the scope of this document. Find yourself a good book on qualitative data gathering and analysis!

[4] Available from http://homepage.ntlworld.com/greenery/workStuff/devices/controllers/HeatingA2.html and also included in the "Cassata shrink-wrap".

Who are the 'user' and the 'system'?

Yes, this is a hard question! Basically, 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.

- As noted above, we did not have access to users of the robotic arm, so had to base the user description on existing documentation.
- In ambulance control, people work in teams, and there are several similar teams in the centre. We found that the different team members had very different perceptions of the system and their tasks, but that people with the same roles across the different teams had similar perceptions. We therefore constructed a separate user description for each different role within the team (Blandford, Wong, Connell & Green, 2002).
- For the Alhambra heating controller, although the two users interviewed had experience of very different types of heating controller, it was still possible to merge their descriptions in terms of key concepts.

In principle, if there were several users, it should be possible to construct separate, but overlapping, user descriptions and let commonalities emerge. In practice, we have never tried doing it this way.

Now for a very important point: the 'system' is not just a computer system, but is the sum total of the 'rest of the system' with which the user that is the focus of analysis interacts. This can make describing the 'interface' a bit tricky at times.

- In the robotic arm example, this is the computer interface to the arm controller, and also the arm itself and other things in the room that the arm might come into contact with.
- In the ambulance control example, for any 'user', it is the computer system and bits of paper and map book and other team members with whom they interact.
- For Alhambra, it is the heating controller itself and the core parts of the heating system with which that interacts.

In each example, we have tried to discuss how we have defined the boundaries of the 'system', but we recognise that this is a bit of a black art. Sorry!

Analysis process

There are different ways of conducting a CASSM analysis, but the following generally works well. Here, we just give an overview; detail is provided in following sections.

1. ***User–system concepts***: The first step is to simply compare user and system concepts and see how well they fit with each other. As described above, the main sources of user concepts are interviews, think-aloud protocols and user-oriented documentation. And the main sources of system concepts are system descriptions and maybe a running system. Put simply, the way to conduct an analysis is to go through the words (transcription of users talking or documentation) highlighting nouns and adjectives, then using intelligence to decide which of those words represent core concepts within the user's conceptualisation of the system they are working with. For a running system, you might have to put your own descriptors on the things you see and experience. One thing you should work hard to avoid is extensive descriptions of interface widgets: really focus on the underlying system representation. Interface widgets are a means to an end, not an end in themselves. For example, if you were describing a telephone, you would focus on calls, people, numbers and the line state, not on the keys on the keypad or the receiver as an object. This will yield you your first level of surface misfits.
2. ***Entities and attributes… and the interface:*** Depending on what matters most to you, you might start distinguishing between entities and attributes to start to achieve clarity in your model, or you might focus attention on the interface. For user, system and interface, you consider whether the concept is present, absent or difficult (meaning 'present, but likely

to cause problems in some way'). This deeper analysis is likely to help identify more surface misfits.

3. ***Actions:*** If the interface is sufficiently well specified, you can also start to consider actions and how the user changes the state of the system. Considering actions serves two purposes. Firstly, actions that are *hard* or *can't* indicate further surface misfits. An action that is *hard* is typically one that is difficult to perform for some reason: maybe it involves a long and tedious action sequence or it is difficult for the user to discover. An action that the user *can't* perform is one that is impossible, but that the analyst thinks the user might want to do under some circumstances. [This contrasts with *fixed*, which is an action that is impossible, but not problematic, as discussed more fully below.] Secondly, action information is used for identifying some structural misfits, as discussed properly below.
4. ***Structural misfits:*** The final step is to consider structural misfits. There are two ways to go about doing this. The 'official' one is to add information about relationships to the CASSM analysis, as described below, and use the full CASSM specification to derive structural misfits. 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 (optionally) retrospectively work out what that must mean for the CASSM model, in terms of concepts, actions and relationships. This unofficial approach is illustrated in the Alhambra example below. It is only when considering structural misfits that relationships really come into play, so they can often be ignored until quite late on into analysis.

An analysis can stop after any of these phases, and a thorough analysis typically involves some iteration through the different phases as the analyst gets a deeper understanding of the problem. But as we have tried to emphasise, we do not believe that every analysis has to be thorough to be valuable. We only push one of our examples through to structural misfit analysis; the earlier two focus on surface misfits.

Surface misfits

First, we consider surface misfits, which should be relatively easy to spot if you're looking at the problem the right way. Within 'surface', as outlined above, there are three further levels: first, just identifying system and user concepts, with little reference to the interface between the two (see, for example, Blandford *et al,* 2002); second, more detailed analysis in terms of how well each concept is represented by the user, interface and system; third, analysis in terms of what actions are needed to change the system, and whether there are problems with actions. At this point, concepts are usefully sorted into entities and attributes to support thinking about what can be set, changed, created or deleted. These three steps correspond to the first three phases outlined above.

Any kind of system can suffer from – or avoid – surface misfits, because they are concerned with the surface-level mappings between the users' conception, the interface representation, and the underlying system model. We now consider the three levels of detail in turn.

Misfits between the user and the system:

Misfits between user and system are probably the most important surface-level misfits. These misfits fit into three classes:

User concepts that are not represented within the system, and hence cannot be directly manipulated by the user. A very simple example is the use of land-line telephones: the user is normally interested in speaking to a particular individual, but the only means of doing that it to place a call to a location where they are likely to be (e.g. their home or office), because the telephone system does not map handsets to individuals. Mobile telephones, that are generally assigned to one individual, and for which the user only has to make the simpler mapping of

name to number, partly overcome this misfit, though they can suffer from a converse difficulty which is the user distinguishing between 'work calls' and 'domestic calls', and treating these differently – whether in the way incoming calls are responded to or outgoing calls are billed.

These concepts are often the most difficult for users to deal with, leading to structural difficulties such as viscosity (see below).

They also often force users to introduce workarounds, as users are unable to express exactly what they need to, and therefore use the system in a way it wasn't designed for – e.g. using a field in an electronic form to code information for which that form was not actually designed, or keeping paper notes alongside an electronic system to capture information that the system does not accept.

System concepts that the user has to know about but that are not naturally part of their initial understanding, and therefore need to be learned. At a trivial level, these might include strictly device-related concepts like scroll-bars: it doesn't take long but they do need to be learnt. A slightly more complex example is the 'layers' of most drawing tools that users need to be aware of and work with as they manipulate drawing objects – these can cause substantial user difficulties, particularly for novice users.

For users, these misfits may involve simply learning a new concept, or they may involve the users constantly tracking the state of something that has little significance to them.

User- and system concepts that are similar but non-identical, and which are often referred to by the same terms. This could be considered as an amalgamation of the two categories above (a user concept that the system doesn't represent and a system concept that the user has to know about), but has a particular set of implications, in terms of how the user has to mould their understanding to the system. One example in the domain of diaries is the idea of a 'meeting'. When a user talks about a meeting, they usually mean a pre-arranged gathering of particular individuals at an agreed location with a particular broad purpose (and perhaps a detailed agenda). Within some shared diary systems, a meeting has a much more precise definition, referring to an event about which only other users of the same shared diary system can be kept fully informed, and which has a precise start time and precise finishing time, and possibly a precise location and precise resources available too. The difference between these concepts is not great, but the user of the system who has lots of external meetings does not find the meetings facility useful. Another example, within the domain of ambulance dispatch, is the difference between a call and an incident; this is discussed below.

These misfits may cause difficulties because the user has to constantly map his / her natural understanding of the concept onto the one represented within the system, which may have a subtly different set of attributes that the user then has to work with.

A more detailed taxonomy

The broad user–system misfits listed above are the first level of analysis – you don't even need to look in detail at the interface to spot those. The next level down is the one where CASSM starts to draw out issues concerning the interface between user and system. We start with a table enumerating all the possible combinations (Table 1):

User	Interface	System
Present	Present	Present
Difficult	Difficult	
Absent	Absent	Absent

Table 1: User–Interface–System concepts

In all cases, 'present' means clearly represented. We assume that underlying system concepts are either present or absent, whereas for the user or at the interface there are concepts that are present but difficult in some way.

For users 'difficult' concepts are most commonly ones that are ***implicit*** – ideas they are aware of if asked but not ones they expect to work with. An example would be the end time of a meeting in a diary system: if you look at people's paper diaries, you will find that many engagements have start times (though these are often flagged as 'approximate' – e.g. '2ish') but few have end times, whereas electronic diaries (which are sold as diaries, but are better described as scheduling systems) force every event to have an end time (or a duration, depending on how you look at it). This forces users to make explicit information that they might not choose to. Another source of difficulty might be that the user ***has to learn*** the concept, or that it is perceived as ***irrelevant*** by the user.

Similarly, there are various reasons why a concept may be represented at the interface but in a way that makes it difficult to work with. Difficulties that interface objects may present include:

- ***Disguised***: represented, but hard to interpret;
- ***Delayed***: represented, but not available to the user until some time later in the interaction;
- ***Hidden***: represented, but the user has to perform an explicit action to reveal the state of the entity or attribute; or
- ***Undiscoverable***: represented only to the user who has good system knowledge, but unlikely to be discovered by most users.

Which of these apply in any particular case – i.e. why the interface object might cause user difficulties – is a further level of detail that can be annotated by the analyst. At the end of this document, we include a brief discussion of all 18 possible combinations – only recommended as bed-time reading if you want to get to sleep!

Adding in information about actions

Because CASSM is primarily concerned with conceptual and structural misfits, actions are of only secondary concern. The analyst is encouraged to define how actions change the existence of entities or the values of attributes, as a further step of analysis.

In particular, for every setting or changing, creating or deleting the analyst should define whether the action is:

- ***Easy***: no user difficulties.
- ***Hard***: difficult for some reason (e.g. undiscoverable action, moded action, delayed effect of action). For example, on a Mac it is possible to select a sentence in MS Word by pressing the 'apple' key and clicking anywhere in the sentence; not many people know this!
- ***Indirect***: effect has to be achieved by changing something else in the system; for example, you cannot directly change how many pages a document has, but you can change it by altering the font size, page margins or page layout, or by adding or deleting text. Do not get *hard* and *indirect* confused, at least not if you want to do structural misfit analysis!
- ***Can't***: if particular state change is impossible, and this is likely to cause problems, then it should be noted as 'can't'.

- ***Fixed***: many cases of 'impossible' are not, in fact, problematic; for example, an entity may be listed simply because it has important attributes that can be changed. *Fixed* means that something cannot be changed, but that this is unlikely to be a problem.
- ***Done by 'the system'*** (which may include other agents – e.g. over a network, or simply other people): again, many of these cases are not actually problems, and it is up to the analyst to consider implications. Again, this should not be confused with *indirect*: *indirect* changes are made by the user, but by changing something else, whereas *done by the system* is done by some agency other than the user being modelled.

Some aspects of the action descriptions should alert the analyst to possible difficulties – most notably actions that ***can't*** be done or are ***hard***. Others – such as ***indirect*** – are particularly important for identifying structural misfits, as discussed below.

As well as describing actions in the above terms, the analyst should also be alert to:

- Changes that are impossible. For example,
  - Are there actions whose effect is irreversible, or very difficult to undo?
  - Are there times when the user has to commit to some value earlier than would be natural and is then unable to change it? This is a form of *premature commitment* (see below).
- Side-effects of actions on other aspects of the system state. This leads to the identification of 'affects' relationships that are considered in more detail below. It also relates to *indirect* actions.
- Actions that have unpredictable effects for the user (and if so, why?).

The analyst should make a note of misfits identified while they are working through the analysis.

Cognitive Dimensions that relate to surface features

Some of Green's CDs relate to features of a system that are only surface-level misfits. We list the most important here to bring out particular aspects that an analyst might otherwise overlook as they relate to sometimes non-obvious features of the interactive system.

Abstraction level: devices may be classed as imposing the use of abstractions ('abstraction-hungry'), rejecting the use of abstractions ('abstraction-hating'), or allowing but not imposing abstractions ('abstraction-neutral'); further, the abstractions themselves may be domain-based or device-based. CASSM can express these distinctions reasonably well and can therefore detect some of the misfits, among them:

- domain abstractions that are part of the user's conceptual model but are not implemented within the device; and
- device abstractions imposed upon the user.

Imposed device abstractions have to be learnt in order to work effectively with the device, such as style sheets or graphics masks, and are therefore easy or difficult to learn according to how well they are represented at the interface (as discussed above).

Visibility: This refers to the user's ability to view components readily when required, preferably in juxtaposition to allow comparison between components. CASSM cannot at present express either inter-item juxtaposability nor the number of search steps required to bring a required item to view ('navigability') but captures the essence of visibility by designating those concepts that are hidden, disguised, delayed or undiscoverable as 'difficult' in the interface representation.

To illustrate many of the ideas presented so far (though not particularly these CDs!), we present two worked examples that illustrate the CASSM approach in different ways.

Worked example: robotic arm

Our first example is the CASSM analysis of a robotic arm as described fully by Blandford *et al* (2004b). As described more fully below, the robotic arm is a manipulator for use by people with limited movement, who control the arm by means of a computer interface.

In this case, we did not have access to real users of such devices, so the CASSM analysis is based solely on an existing written description of the robotic arm. Additional system information was provided by the developer. In the following, key user concepts and relationships are highlighted.

"The AMMC at Middlesex University is currently developing 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, and has been developed primarily to prove that a sophisticated manipulator can be produced at a reasonable cost, with usability issues being considered informally if at all. 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. For the purpose of the analysis, only one task is being considered, which uses only a small part of the interface. However, the task is one that will be very common to all users, and therefore will give valuable information on the usability of the interface. The task is to move 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. It is this kind of task that the developers of the arm consider to be a basic task, and that should be part of the core functionality of the interface. 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 has not yet been fully implemented, but it 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 is based on a baseball cap with two sensors: one allowing movement forwards and backwards to be detected, the other allowing movement left and right to be detected. This 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. It is designed to be trained to individual voices, and needs resetting over time to do the way that voices change."

There are 8 arm joints, different possible directions depending on which joint is selected (usually 2, or 6 directions for the whole arm) and 5 speeds.

Analysis

Some of the concepts highlighted are in fact not worth enumerating in more detail. For example, in our analysis, we did not bother to list the arm itself as an entity because it does not have interesting attributes – it is the joints and the gripper that are interesting.

The description highlights three kinds of concepts: things in the world that the user is trying to manipulate, components of the arm and components of the computer interface through which the arm is manipulated. We used the Cassata tool (described below) to describe the robotic arm, as summarised in table 2. In this table, U=User; I=Interface; S=System; P=Present; A=Absent; S/C=set (attribute) / Create (entity); and C/D= Change (attribute) / Delete (entity).

Most concepts are present for user, interface and system; the exceptions are objects in the world, of which the robotic arm system has no representation. Most actions are also unproblematic (either fixed or easy). The exceptions are changes to the properties of objects in the world, which can only be achieved indirectly (by manipulating the arm to change them), manipulating properties of the gripper, many of which we assessed as being hard, and a couple of detailed menu manipulations, that we also judged to be hard.

	Concept (Entity / Attribute)	U	I	S	S/C	C/D	Notes
E	object in world	P	A	A	fixed	fixed	e.g. light switch, cup
A	position	P	A	A	fixed	indirect	any particular object may only have some attributes
A	configuration	P	A	A	fixed	indirect	may include orientation
A	speed	P	A	A	fixed	indirect	
E	gripper	P	P	P	fixed	fixed	
A	position	P	P	P	fixed	hard	
A	orientation	P	P	P	fixed	hard	not from documentation but clearly important
A	speed	P	P	P	fixed	hard	
A	openness	P	P	P	fixed	hard	not from documentation
E	joint	P	P	P	fixed	fixed	might include 'whole arm'
A	speed	P	P	P	fixed	easy	5 possibilities
A	direction	P	P	P	fixed	easy	different directions apply to different joints
E	input device	P	P	P	fixed	fixed	user has to choose (unless preconfigured). each has different features
A	selected	P	P	P	fixed	Not Sure	
E	menu	P	P	P	fixed	fixed	
A	current	P	P	P	fixed	hard	some menu transitions are easy, others hard
E	menu option	P	P	P	fixed	fixed	
A	current selected	P	P	P	fixed	hard	may be difficult -- either due to voice recognition problems or timing of gesture
E	cursor	P	P	P	fixed	fixed	this applies only to gestural interfaces
A	speed	P	P	P	fixed	easy	
A	option indicated	P	P	P	fixed	By Sys	
E	gesture	P	P	P	fixed	fixed	user has to remember repertoire

Table 2: CASSM description of robotic arm

The process of producing the CASSM description highlighted some potential difficulties, as follows:

- 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.
- For grabbing objects, the user has to get the orientation of the gripper and its openness right.
- Considering domain and device concepts, we see that the only domain-relevant concepts are those relating to an object in the world. Everything else the user has to do is about manipulating the device. 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. It is likely that the user's main interest is in the properties of the gripper, and that therefore the main task will involve moving the whole arm.

- Both input devices pose some difficulties: of accurate voice recognition, or of timely and appropriate use of gesture. This is likely to be simply something that users have to practice and learn to work with, but is nevertheless likely to pose initial difficulties.
- Some menu transitions are difficult.
- It may be difficult for the user to judge the direction to be set when the arm is contorted.
- Finally, an additional insight emerged, though not directly from the CASSM analysis: 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.

Blandford *et al* (2004b) compare this analysis to six other usability evaluations of the same robotic arm, and also to video data of the arm in use, and show that CASSM covers different 'evaluation territory' from all the other techniques considered.

Worked example: Ambulance Dispatch

The robotic arm is a fairly simple, single-user system (as is the heating controller which we consider in detail below). CASSM can also be applied to much larger, multi-user systems, as we will illustrate here with an outline of the analysis for ambulance control. More details of this example are presented by Blandford *et al* (2002).

At the centre studied there are four main ambulance dispatch roles: call taking, telephone dispatching, radio operator and allocator. Each of these roles was analysed separately. For illustration purposes, we consider just the allocator role (the allocator is the person with overall responsibility for deciding which vehicles attend each incident in their area). Three sources of information were worked through to identify core concepts:

1) Organisational documentation specifying the responsibilities of the allocator;

2) Transcripts of contextual inquiry interviews; and

3) Transcripts of Critical Decision Method interviews.

We started by listing entities and their attributes, making notes to set the context, and noting which source each concept had been identified in. Since this document extends to several pages, we focus on just two related entities, an incident and a call, here:

Entity	Attribute	Notes	Source
Incident	Type	e.g. Major Incident, RTA, maternity / BBA	1, 2, 3
	Requirements	e.g. maternity needs a 'hotel crew', maybe midwife; Major needs many crews	2, 3
	Nearest vehicle	As the crow flies or on the basis of expected journey time	2, 3
	Hospital reqts	E.g. maternity may be under a particular hospital.	2
	Nearest hospital		1
	Nearest stand-by point or station		1
	Number of casualties	Affects number of vehicles sent	2, 3
	Number of vehicles sent	A paper copy of the ticket is printed for each.	2, 3
	Who's there	For safety reasons	3
	First on scene	For major incident: "silver", in charge until manager arrives	3
	Manager	For major incident	3

	Likely extra factors – e.g. time of day, traffic conditions		3
Call	Refers to incident	Duplicates are a problem (or might confirm the seriousness of an incident)	1, 3
	Allocated / dispatched / not yet / held	Need information on ticket if delayed beyond STA	1, 2
	Priority	Or urgency (e.g. red, amber, green)	1
	Progress	An extended version of allocated / dispatched. NB case where an ambulance is diverted from a low priority call to a red one.	1
	Type (emergency / urgent / urgent immediate / white		2
	STA	Time of arrival at hospital (for urgents)	2
	Caller type	Doctor / public / police / LFB	2
	Ongoing / completed	Allocator can listen (but doesn't often)	2
	Responsibility of someone	An allocator. E.g."Officially Addington is Biggin Hill call, which is my vehicle, right. But because all the managers are over that side, in the Bromley area, which is nearer to Biggin Hill, the incident goes over to that sector."	2, 3
	Patient ready time	For (not so) urgents. "what the computer doesn't know is that the police have phoned up and said that the RV point – the rendezvous point for the crew isn't 1500 now, it's now 1600"	2
	Start and destination of job	For urgents	2

In addition to identifying entities and attributes, two constraints emerged. The first is a domain constraint – something the users have to work to achieve – namely that the allocator is responsible for maintaining coverage in their area, making sure that there are vehicles near enough to any possible incident location. The second is a device constraint: that the allocator always has to work with available resources. However, in this sketchy analysis, we don't consider these constraints any further, just noting that they are examples of relationships, which are discussed fully below.

In conducting the analysis, we judged some of these attributes to be of limited importance in the allocator's job and – more pertinently – in their interactions with the local system. For example, information about who is first on scene has minimal effect on their planning. One of the features that emerged as very important in this analysis is that 'calls' and 'incidents' are treated very similarly – that properties of one might equally be considered to apply to the other in some cases. Although the mapping is not strictly one-to-one (there can be several calls for an incident), the system the allocators use forces them to deal with each call explicitly, rather than consolidating information about a particular incident.

In this case study, we have not even got as far as constructing a complete CASSM description in Cassata, but just enumerated important (and sometimes less important) concepts to spot misfits between user and system (stage 1 of analysis as discussed above). Blandford *et al* (2002) compare and contrast the perspectives of the different roles within the overall system.

These two examples (the robotic arm and ambulance dispatch) are intentionally incomplete, illustrating the 'sketchy' nature of many CASSM analyses. They are also intentionally presented in slightly different forms, to illustrate the flexibility of analysis. The Cassata tool,

presented next, constrains the forms in which analysis can be represented, but paper analysis can be a little more flexible.

Using Cassata to support analysis

We have already mentioned Cassata 'in passing'. This prototype tool is available from the project website (www.uclic.ucl.ac.uk/annb/cassm.html), together with supporting documentation. Since the tool has helped clarify our thinking about CASSM, we present it here and use it to support our discussion of structural misfits. Much of this material is replicated from Blandford *et al* (2004a).

As described above, an ***entity*** is a concept that can be created or deleted, or that has attributes that the analyst wants to enumerate, and an ***attribute*** is a concept that is a property of an entity. In figure 1, entities are shown in the left-hand column, left-justified. Note also the 'E' in the left margin. Attributes are shown right-justified in the left-hand column. Note also the 'A' in the left margin. There should not be 'floating' attributes.

Figure 1 shows the Cassata window for a partial description of a word processor document. For clarity, the picture is cropped from the right. It is a description of a set of figures (pictures or diagrams) in a document, which consists of one or more individual figures. For the user, there is the important idea that the figures should be sequentially numbered – so the *number-sequence* is important, and is an attribute of the *set-of-figs*. Each *figure* has an attribute which is its particular *number*, and changing a figure number changes the overall sequence of figure numbers.

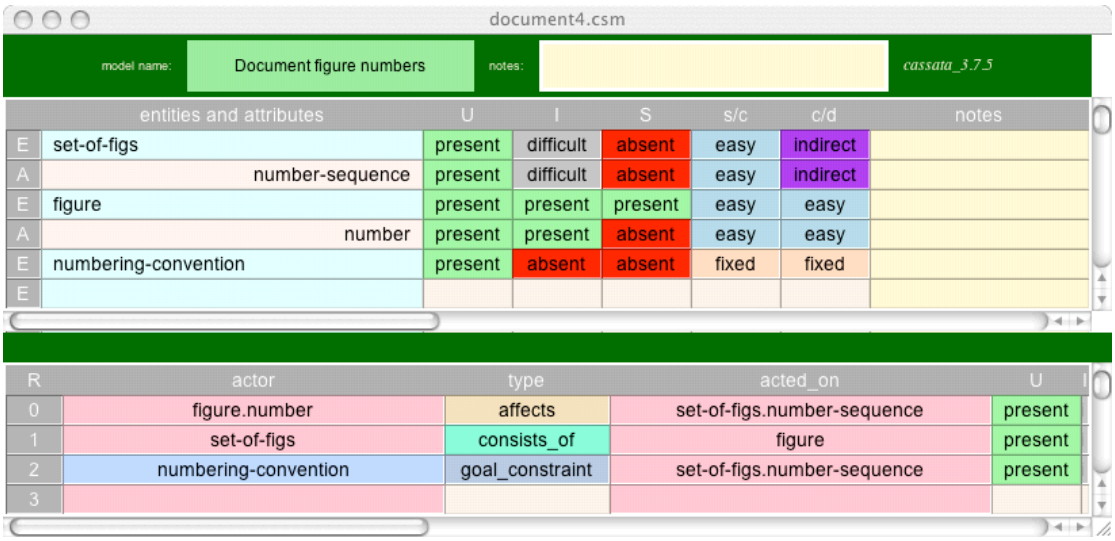

Fig. 1. Cassata data table for a partial description of a document. The upper table describes concepts (i.e. entities and their attributes); the lower describes relationships between those concepts.

The top half of the window shows information about concepts (entities such as *figure* and attributes such as *number*): for each concept, three columns show whether it is present, difficult or absent for the user, interface and system respectively; the next two columns show how easy it is to …

- (S/C) set the value of an attribute, or to create an entity; and
- (C/D) change the value of an attribute, or to delete an entity.

The final column is a notes area in which the analyst can add comments. To take the first row as an example: the `set-of-figs` is a conceptual entity that is meaningful to the user, is not clearly represented at the interface ('difficult') and is absent from the underlying system model. It is easy to create a set of figures, (because this happens automatically as the user

adds figures) but harder to delete it (done indirectly because that requires deleting *all* the individual figures).

The bottom half of the window shows information about relationships (such as *affects* and *consists_of*) between concepts. In this particular case, the three lines of input state that changing any *number* (of a *figure*) affects the *number-sequence* (of the *set-of-figs*), that a *set-of-figs* consists of (many) *figures*, and that there is a (socially imposed) domain constraint that figure numbers should be sequential.

Practical aspects of working with Cassata are presented in the companion document (Green & Blandford, 2004). One of the features of Cassata is that it supports the automated analysis of various structural misfits, so we now move on to describe these in more detail.

Structural misfits: taking account of relationships

Structural misfits refer to the structure of information, and how the user can change that structure. This may be done by working from a CASSM description to identify misfits – possible automatically. However, as shown in the heating controller presented below, for analysts it is often just as easy to 'work backwards', asking explicit questions about whether a particular kind of misfit might apply and then working out what that means in terms of concepts, actions and relationships.

Here, we present the structural misfits of which we are currently aware. These are a subset of Green's Cognitive Dimensions. It is worth noting that structural misfits only apply to systems where the system state can be changed in a meaningful way by the user; thus, systems such as web sites or vending machines do not generally suffer from structural misfits. One of the consequences of this is that such structural misfits can often be identified by an analyst with less extensive user knowledge acquisition: the analyst's own expertise is often enough to identify many of these misfits.

Another point to note is that although structural misfits are much finer-grained than the surface-level misfits discussed above, they can be immense sources of user frustration and inefficiency. But first, we define the relationships that underpin the definitions of structural misfits, and that are implemented in Cassata.

Defining relationships

Five kinds of relationships are currently defined. These are: consists_of, device_constraint, goal_constraint, affects, and maps_onto (and there is a 'not_sure' option in Cassata, which might be used for other relationships that the user wishes to note).

Consists_of takes two arguments, Actor and ActedOn, which are both concepts. This means that the first consists_of the second: chapter consists_of paragraphs; set-of-paragraphs consists_of paragraphs (e.g. sharing a paragraph style); etc. In the figure-numbers example above, the set of figures consists_of many individual figures.

Device_constraint also takes two arguments, both concepts. The value of Actor constrains the possible values of ActedOn. For example, considering drawing a map on the back of an envelope, the starting_position (for drawing) constrains the location of a particular instruction. An easier example is that the field-width for a data entry field constrains the item-width for any items to be put in that field.

Goal_constraint expresses the idea that there is some domain-based constraint. For example, when writing a conference paper, it is common to have a limit on the length of a document. In the figure-numbers example there is a goal-constraint that figure numbers should follow the normal numbering convention.

Affects is concerned with side-effects. That changing the value of one concept will also change the value of another. For example, changing the number of words in a document will change its length. In the figure-numbers example above, changing an individual figure number affects the whole sequence of numbers.

Maps_onto is a simple way of expressing the idea that two concepts are very similar but not quite identical. These are most commonly a domain-relevant concept and a device-relevant one. In particular, the user typically has to use one (the device concept) in place of the other (the domain concept). For example, a (user) meeting maps_onto a (diary-entry) meeting, but depending on the form of the diary, the two meeting types may have importantly different attributes.

As for entities and attributes, it is possible (though not always necessary) to define whether relationships are represented at the interface, to the user, or in the underlying system.

Also, as for entities and attributes, there is the facility to add notes, either as alerts to usability difficulties or as explanations to the reader of why a system has been described in a particular way.

We now present three important classes of structural misfits: viscosity, premature commitment and hidden dependencies. In what follows, we take A to be an entity of interest with an attribute P, and B to be some other entity with attribute Q.

Viscosity

As discussed above, "viscosity" captures the idea that a system is difficult to change in some way. Green (1990) distinguished two types of viscosity, repetition and knock-on, which can be defined as follows.

1) ***Repetition viscosity*** occurs when a single action within the user's conceptual model requires many, repetitive device actions.

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
```

For example, we get repetition viscosity on figure numbers in a document because whenever a figure is added, deleted or moved, a range of figures need to be re-numbered one by one. Stated more formally:

```
set-of-figs(number-sequence) is not directly modifiable
figure(number) affects set-of-figs(number-sequence)
figure(number) is modifiable
set-of-figs consists-of figure
```

Figure 2 shows the basic requirements on a model for it to exhibit Repetition Viscosity. Note in particular the use of 'indirect' to denote something that can be changed, but not directly. Figure 3 shows the output when this particular model is assessed by the automated analysis tool in Cassata.

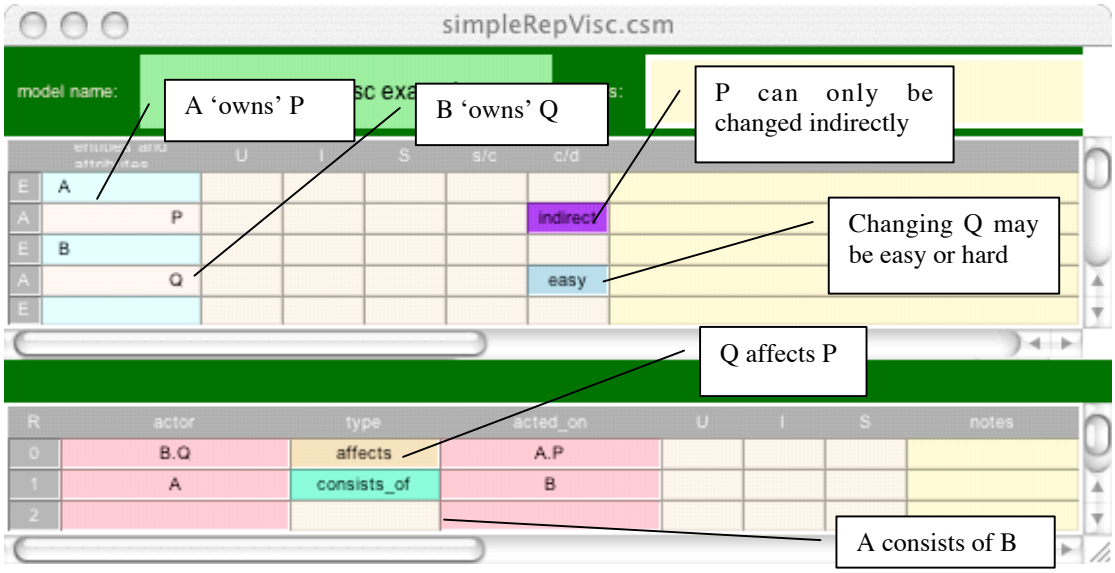

Fig. 2. Repetition Viscosity

Repetition Viscosity Check

"B.Q" affects "A.P"
"A" consists_of "B"

possible case of repetition viscosity:
to change "A.P" user may have to change all instances of B.Q"

Fig. 3. Output from Repetition Viscosity analysis in Cassata

2) ***Knock-on viscosity***: changing one attribute may lead to the need to adjust other things to restore the internal consistency. (In North America, a better-known phrase for the same concept appears to be 'domino effect'.)

Changing A(P) has possible knock-on if:

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

Timetables and schedules typically contain high knock-on viscosity; if one item is re-scheduled, many others may have to be changed as well. Figure 4 shows the conditions for a model to exhibit Knock-on Viscosity.

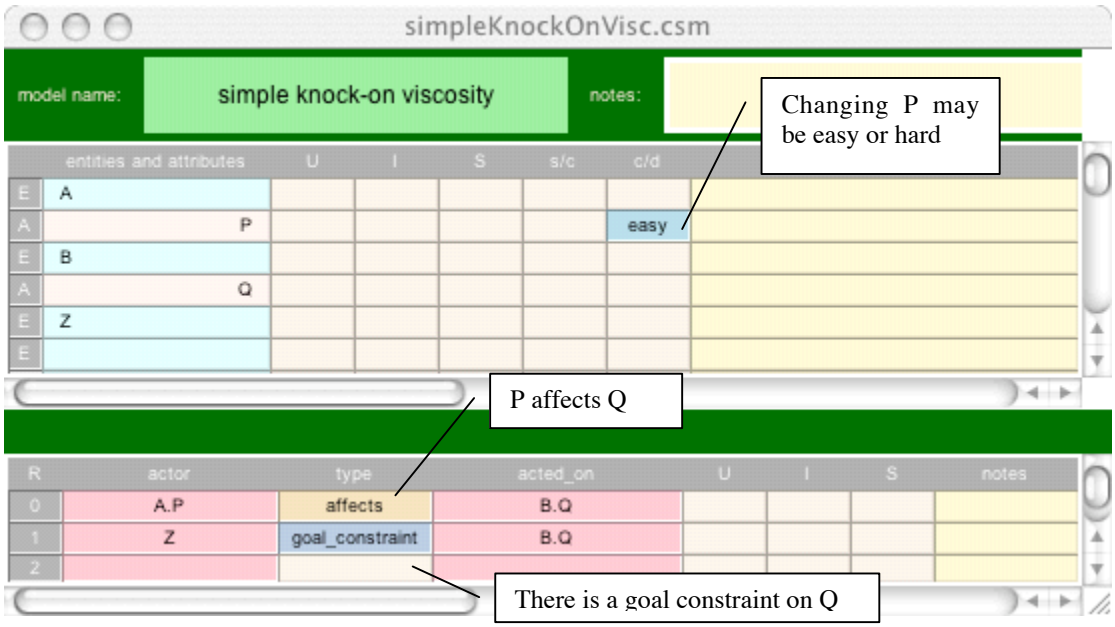

Fig. 4. Knock-on Viscosity

Premature Commitment

Informally, premature commitment occurs when the user has to provide information to the system earlier than they would wish or are prepared for. We have several sets of conditions that alert to possible premature commitment.

1) ***Non-modifiability premature commitment***: As discussed above, if an attribute cannot be changed after it has been set then the system possibly demands premature commitment:

```
A(P) is settable
A(P) is not modifiable
```

Some painting tools exhibit this type of premature commitment: that the width and colour of a line cannot be changed once it has been set.

Extending this to entities, we may get potential non-modifiability premature commitment if entities can be created but not subsequently deleted:

```
A is creatable
A is not deletable
```

In principle the converse may hold too, but there are few situations in which that would class as premature commitment (rather than simply an irreversible action). Figure 5 shows the conditions for a model to exhibit this kind of Premature Commitment.

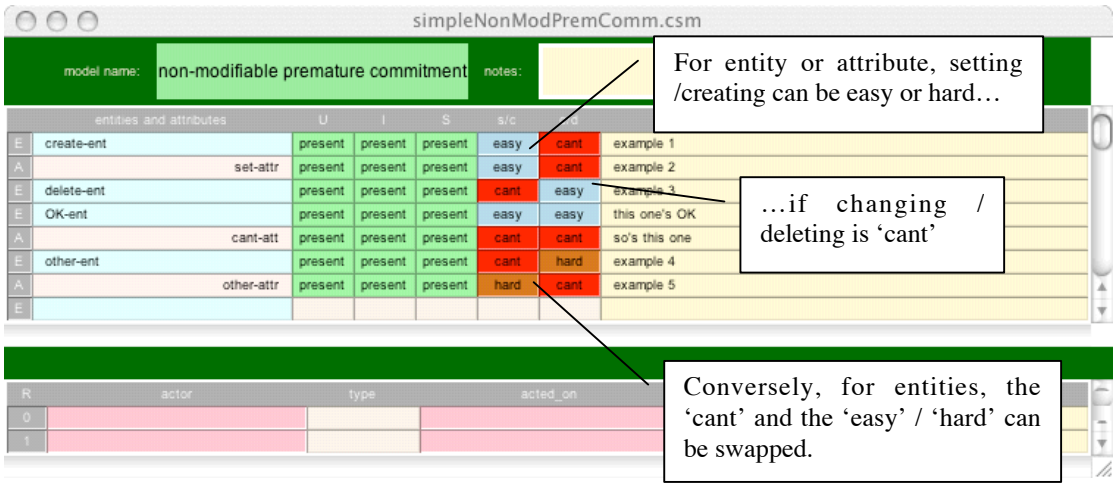

Fig. 5. Non-modifiability Premature Commitment

2) ***Abstraction-based premature commitment***: If a user has to define an abstraction in order to avoid repetition viscosity, and that abstraction has to be defined in advance, then the system potentially creates abstraction-based premature commitment. Frequently that abstraction will be a simple grouping. A common example of potentially premature commitment to abstractions is the defining of paragraph styles before starting to create a technical document. The purpose is to avoid repetition viscosity by allowing all paragraphs of one type to be reformatted in one action, but the problem is to foresee the required definitions. A more technical example would be the creation of a class hierarchy in object-oriented programming.

The conventional analysis in the Cognitive Dimensions framework is to treat the abstraction management components of the system as a separate sub-device, which may have its own properties of viscosity, hidden dependencies, etc (Blackwell, Hewson & Green, 2003). In CASSM we take a simplified approach such that this type of premature commitment is highlighted if:

```
A consists-of B
A(P) is directly modifiable
A(P) affects B(Q)
```

The paragraph styles case would be represented thus:

```
Paragraph has attribute style
Set-of-paragraphs has attribute style-description
Set-of-paragraphs consists-of paragraph
Style-description is directly modifiable
Changing style-description causes style to change
```

Figure 6 shows the basic requirements on a model for it to exhibit Abstraction-based Premature Commitment.

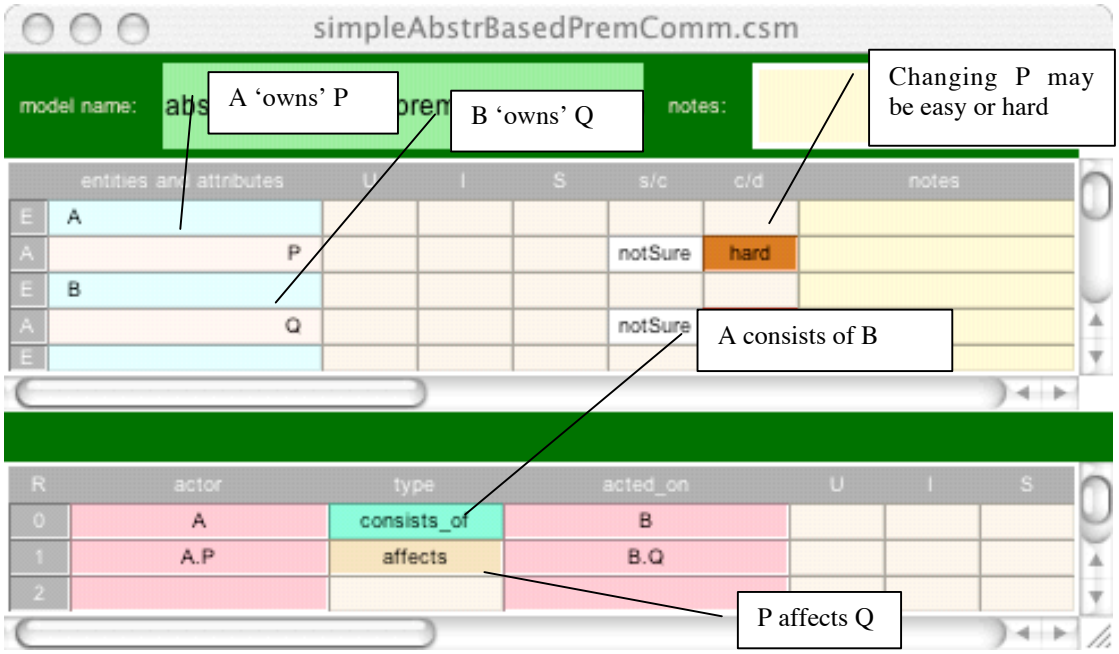

Fig. 6. Abstraction-based premature commitment

3) ***Device-constraint premature commitment***: Here, setting an attribute of one entity constrains the way that new instances of another entity can be created:

```
B(Q) is settable
A(P) is not settable
There is a device constraint between B(Q) and A(P)
It is possible to add more As
```

As mentioned above (when defining *device constraint*), one example of this is drawing a map on the back of an envelope; another is that of setting the field width in a data structure when the size of all items to be entered in that field is not known (here, ">=" is an example of a device constraint):

```
field(width) is settable
item(width) is not settable
field(width)>=item(width)
more items can be added
```

Figure 7 shows the basic requirements on a model for it to exhibit Device-constraint Premature Commitment.

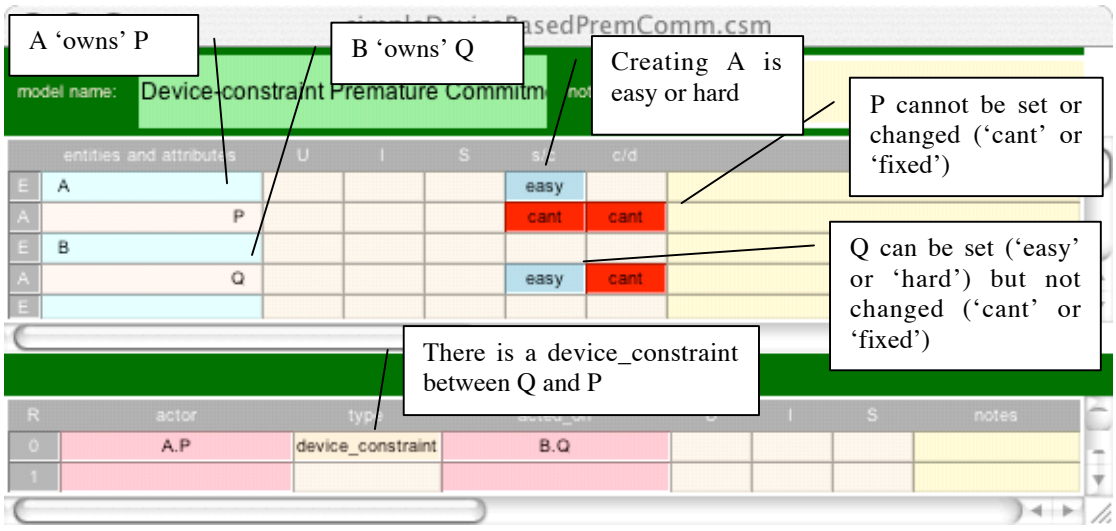

Fig. 7. Device-constraint premature commitment

Hidden Dependencies

A hidden dependency occurs when important links between concepts are not visible (or otherwise readily available to the user). Spreadsheets contain many hidden dependencies, so that changing a value or formula somewhere in a sheet can have unanticipated knock-on effects elsewhere in the sheet. Similarly, changing a style in MS Word can have unexpected knock-on effects on other styles through the style hierarchy. This is formalised simply:

```
Changing C affects D
The relationship is not visible
```

Here, C and D are concepts (entities or attributes). They may even be the same concept. For example, in the word processor because the concept 'style definition' denotes an aggregate of styles formed into a hierarchy, changing any one definition potentially changes other definitions that refer to it, so we have the reflexive relationship:

```
Changing style-definition affects style-definition
The relationship is not visible
```

Figure 8 shows the basic requirements on a model for it to exhibit Hidden Dependencies.

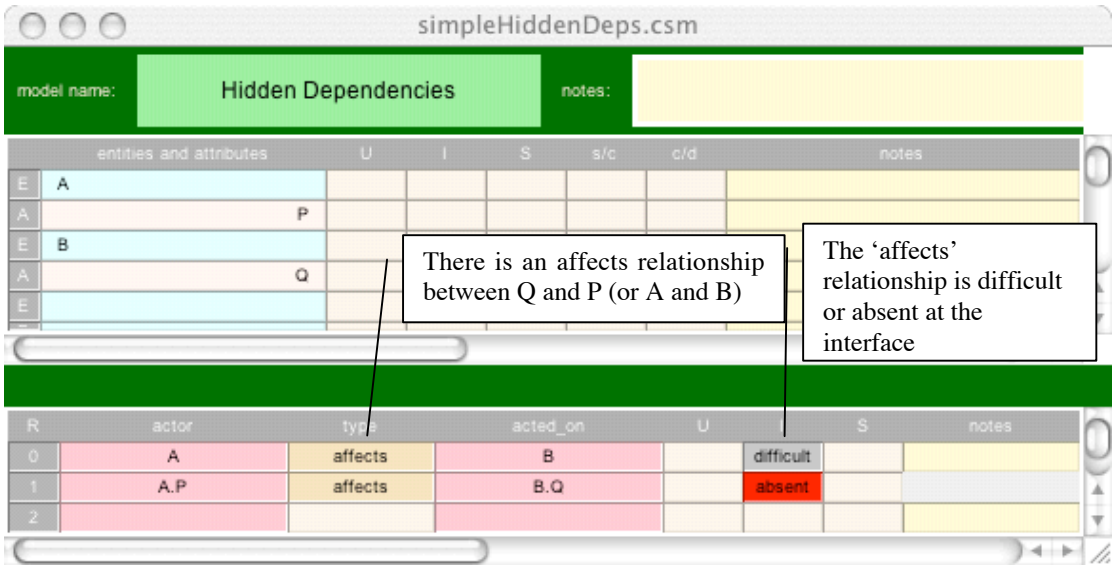

Fig. 8. Hidden Dependencies

Worked example: the "Alhambra" heating controller

In this section, we exemplify one possible approach to analysing a system using CASSM, using a heating controller simulation ("Alhambra"). The simulation is on Thomas Green's web site (http://homepage.ntlworld.com/greenery/workStuff/devices/controllers/HeatingA2.html) and also included in the "Cassata shrink-wrap". Figure 9 shows a screen-dump of the simulation.

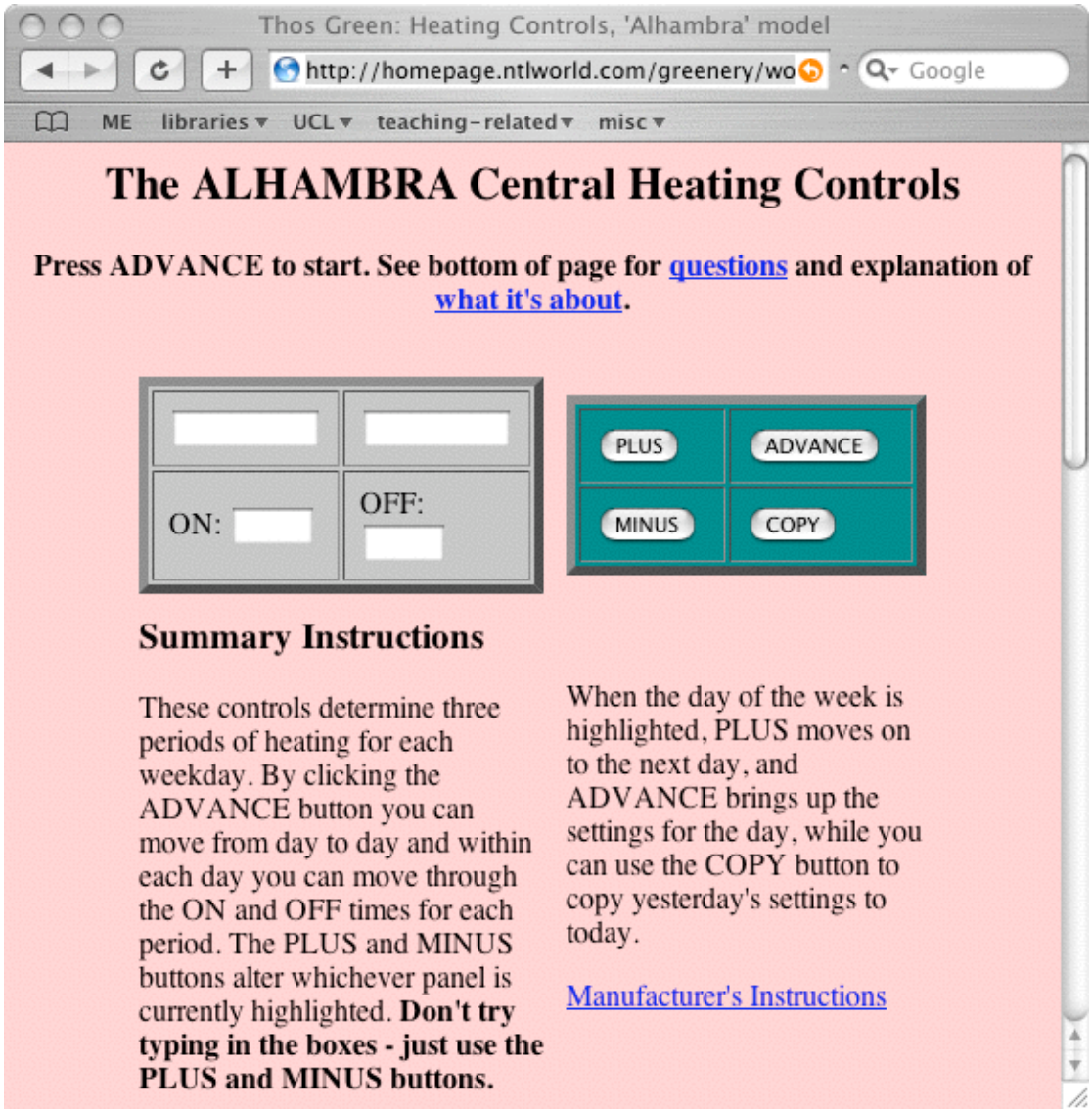

Fig. 9: The Alhambra simulation

This analysis follows to procedure outlined above: first get users' conceptions from interviews, then analyse the (simulated) system, then bring the two together, and finally push a bit deeper to consider structural misfits.

Heating System Interviews and preliminary analysis of user domain concepts

Each interview started by getting the interviewee to describe their current heating controller and how it works, as a starting point for them talking about heating controllers more generally, and what they would like from one. This was done to enable interviewees to focus on concrete knowledge in the domain. Their experience of particular controllers will heavily influence their perceptions of controllers in general.

Relevant concepts are marked like this and annotations are marked *like this*. For each interview, the interviewer is denoted by 'I' and the respondent by 'R'.

Interview 1

I (interviewer): Could you describe to me the overall function and design of your heating control ?

R (respondent): Ours is actually split into two separate things, there's a thermostat which is in the hallway, and which has a dial which allows you to set the desired temperature - in actual fact you can also tell the actual temperature because when you twiddle it it clicks on and off, which is a feature we use, but I don't think we're meant to. So the temperature control happens in the hallway, and the timing control happens in the boiler, and it's an integral part of the boiler, and what there is on it is just a rotary dial, a mechanical dial, and it's got just four little blippy things on it two of which mean on and two of which mean off - I think they're colour coded such that the on ones are orange and the off ones are yellow - I really can't remember to be honest. So the way you set the time on it is by moving the little mechanical things round the dial - the time it comes on and goes off - and the way you set the current time is by actually turning the dial itself - so it's all basically mechanical switches that turn the thing on and off, and then there's one separate switch, no two or three separate switches, one of which says that you can set it so that it comes on: not at all (basically it's an override for all the timer); twice a day, so it comes on at the first blip and goes off at the first negative blip, on at the next blip, off at the last blip; having it coming on once a day, so that it comes on at the first blip, ignores the next two and goes off at the last one; or have it on continuous, which just completely overrides the whole thing, which means you've got your heat in the house at 3 am as well.

In this discussion, we see reference to several concepts that are outside the scope of the heating controllers being evaluated – in particular, temperature and how it is controlled. Because they are outside the scope of the analysis, we will note that they were mentioned, but not include them in the analysis. Of course, this involves looking ahead to the systems being analysed and using some system knowledge to define the scope of the user analysis.

One key concept in this paragraph is time: the time the heating comes on, the time it goes off, and the current time.

There is also a concept that is hard to name: whether the heating is on always, once, twice or never. This concept is specific to the particular heating controller being described, so will not be carried further.

Note that in this analysis we have not named device features (such as 'blippy things') as concepts because they are interface tools that simply mediate action: the user doesn't need a deep concept of a 'blippy thing'.

...

R: So that's one of them, and then there's two separate little manual overrides for the central heating and the hot water, so you can choose to have the hot water, to override and have the hot water come on even though the rest of it's off, and you can override and have the central heating come on though the rest of it's off. And past experience leads me to believe that if you try running the central heating for too long and without any hot water, then it goes cold, because the central heating water gets heated from the hot water, so choosing central heating only is only a short term solution - if there's not hot water then there's no central heating either - but that's just based on past experiences - thinking you've got the central heating on and somebody's running a shower for a bit, and all of a sudden the house is going cold.

The interviewee has introduced two new ideas here. The idea of 'manual override' isn't a concept in itself, but it indicates another idea: that the heating is on or off (and this may be independent of the timing system). A second idea is that of the central heating and the hot water being separately controllable. The systems being studied do not support this functionality, so we simply note it, but do not build this into the analysis.

....

R: ... we fiddle with the timer things, obviously twice a year we have to change the physical clock bit to get it back onto the correct time zone, but also it's set up normally so that it goes on and off twice during the day, so we warm up in the morning and then we warm it up again in the evening. And so when we're at home all day, whether because it's the weekend or for

any other reason, we just manually flick both the heating and the hot water things across. You have to wait till it's turned off, you have to wait about twenty minutes, cos you can't flick it in the first twenty minutes after it's gone off, then flick them both into override mode so it will stay on for the rest of that off period, and at the start of your [next] on period it will reset itself back to normal

Here, we have the idea of time zone, which may or may not be important – we'll hang on to that idea for a while.

Morning and evening are probably best thought of as 'periods of the day when people are at home', or 'home periods' for short. Similarly, the respondent refers to 'days when we're at home all day', and the 'weekend', so there is some concept of 'home days'. Maybe it's an attribute of a day.

The respondent also refers to 'override mode', which initially sounds like a concept, but on further reflection isn't: it's just talking about whether the system is on or off (overriding the default timer settings).

I: You can't do that for the days of the week ?

R: No, there's no control over different days of the week, which is why we always manually switch it at the weekend if we're in - it has no concept of different days of the week - we do, but it doesn't

This restates the position about different kinds of days, rather more clearly than it was stated above.

…

R: in the summer we tend to turn the whole thing off and just run on the hot water, we just turn the heating right off. [...] Actually thinking about it, there must be separate controls, so that the heating can come on once or twice or on continually, and the hot water must be separately controllable, because in the summer we only have hot water and no heating.

This introduces a new concept – of an extended period of time for which heating is not required (and conversely another period where it is).

…

I: So what would be your main requirements for an imaginary new version of a system, were you about to acquire one ?

R: ... I would like to able to specify the difference between weekends and weekdays; I like the fact that I can manually override it, because for instance if I'm working at home I just flick it and then I don't have to worry about the heating if I'm in all day. But we're predictably in at the weekends, and I can't override it. … I don't think I actually want anything much more complicated than this. This children I know would like it so that it's really warm at the time when they have to get up, so they'd actually like it warmer in the morning than in the evening, because their body clocks are such that in the mornings they're very sluggish and they feel cold very easily, and in the evening it doesn't have to be so warm - but that doesn't worry me, my body's not awake enough to know whether it's warm or cold outside

The interviewee is reiterating concepts already introduced earlier. Temperature (and the ability to control it more finely) has also been reintroduced.

…

I: Ok, thank you.

Interview 2

I: Could you describe to me what [your heating controller] looks like - you can sketch if you like - just your memory of it, it doesn't have to be accurate, I'm not going to go round your house and check ....
R: Ok, it's in two parts, there's a device part, ten centimetres by five or six, it's divided into two parts, the left-hand part is time, and mid-day .... the other part is .... a cycle of dates .... [drawing while talking]

Again, we have the concept of time. We also have 'mid-day' mentioned as something apparently special, and a 'cycle of dates'. We will need to find out later what the interviewee means by these ideas.

I: Ok, so it presumably doesn't look like that [refers to drawing], it's got LCDs or LEDs or something ...

R: The time part has a LED and small button ... very small buttons to cycle through the time of day and date ... of the week, and the right hand part is a dial which allows you to set particular on and off patterns for the day that you're currently looking at

Time is repeated. We also have days of the week, and an 'on and off pattern'.

…

I: You said that was a mechanical .... is it a dial ?

R: It's a dial and you press a button to move a pointer around a dial and then you set on and off at the end of each phase the pointers show - so it's kind of like an electronic representation of the timer that you [move] buttons round a rim.

Phase appears to be a concept, probably the same as 'on and off patterns'.

I: So can you draw what it looks like on a particular daily setting ?

[Long pause while drawing proceeds]

R: So it's rather like a raised clock face, and it's set to am or pm, [so that] 12 o'clock indicates which 12-hour cycle you're in ...

A new concept here is whether the displayed time is morning or afternoon.

I: Ok, so that works on a 12-hour cycle ?

R: Yeah ... if it's am it's 12 midnight to 12 midday, if it's pm it's 12 midday to 12 midnight … and then you move an indicator around the rim of this .... device, and then you set on or off for each position

I: So conceptually obviously it would naturally be on-off on-off on-off

R: yeah, you can't set two ons, two offs, you can only have an on followed by an off

I: right, so once you've set one on, the rest of them automatically go off-on off-on

R: well until you re-set it ....

I: So can you have lots of these time settings ?

R: As many as you want

We seem to be struggling towards an idea of an on-off sequence with as many phases as desired.

…

R: And in addition to that there is a manual override switch, separate from the timing device, which allows you to set the temperature, and the temperature can be showing the current temperature in the room, in the air adjacent to the thermostat .... there's the temperature which you set the threshold value to .... and there's the current temperature which the machine is registering

As in the first interview, there are important concepts concerning temperature that we will not pursue further.

…

R: for every on or off part of the cycle, you are allowed to set whatever the threshold value - the separate threshold value [is] - and move the pointer by the on-off setting to the start of the [...] phase, and you set the threshold value for that phase, and that's not shown [...], it's only shown when you're in that threshold [phase], and it shows the current setting ... by pressing it, by querying it, it gives you the current threshold, it shows the threshold value that it's currently working at. But the display shows the current temperature, and then there's a small flame symbol - it's gas - which shows when [the heating] is on, by virtue of the current temperature

being below the threshold temperature. Once the current temperature reaches the threshold temperature it [the heating output] goes off.

The interviewee is talking about a 'cycle' to refer to a single phase of the heating coming on then going off. However, the interviewee is also using the terms 'on' and 'off' to refer to something slightly different, which is whether or not the boiler is actually on at that instant.

I: right ok ... you clearly wouldn't be changing all of that every day

R: No, that's set up once and once it's set you leave it until you need to reset it

I: What kinds of reasons might you have to reset it ?

R: Temperature change - when the temperature moves from a cold snap to a warm snap ....

I: and then what do you change ?

R: If it's a mild snap then it can go off; if it becomes too cold, then the threshold value may be too low, so rather than continually adjust the threshold values, you might manually up the temperature - at least I do, by half a degree or so, so it will come on and warm the flat for a bit, rather than it being constantly off - and that becomes a problem around now, when the cold weather is finished, and there's this changeover from cold to mild, when you find that the heating is off and it's quite cold in the flat

This section refers to external factors such as the outside temperature, and also to some other 'temperature' that is associated with the heating controller. Let us hope we can ignore these!

…

R: You can override the absolute temperature as well, but I don't do that, I just adjust the threshold for whatever number of hours it is ... for the current time [period]

I: so it's always for the current time period, whatever that is ?

Again, we have repetition of temperature and current time period.

R: yeah

I: do you have to change it any other time ?

R: No

I: it automatically switches from winter time to summer time ?

R: oh you mean the times - the time has to be set from GMT back to BST, otherwise it's an hour out

I: is it that time there ? [indicates on sketch]

R: yes - the time

I: so you change that twice a year ?

R: yes

We have the concept of different time zones.

I: presumably you've got it set up so that some days like weekdays are different from weekends

R: yeah, that's right .... the time, days of the week, you have to tell it that today's Tuesday

I: so do you have it different for each day of the week ...

R: you could do

I: No I'm asking about what you do

R: … it's set to be lower during the day during the week than at weekends, so that during the weekday it's set to, er 18 degrees day time, except at lunchtime when it comes back to 21, and 21 in the evenings [and] at weekends, [when] it's on all day

We have different 'kinds' of days. We also have different times of day (when the interviewee or flatmates might be at home).

…

I: So what would your needs for a controller be ?

R: Well I'd like one which has individual settings within days of the week - it would be nice if it had some kind of automatic control as well, but that's a long way down the preferences - it's a nuisance when you come to the third week of March and [you] forget [to change the time setting]. I would definitely like it to be sensitive to the temperature, not of the air but the actual ambient temperature in some important part of the house - this one is sensitive only to air temperature - even if you warm through the rest of the house it's only air temperature, so I'd like it to be [..] cleverer

I: you'd like a controller that you only had to set up once

R: definitely yes - and maybe adjust twice a year for summer and winter

I: and the main things that you would like to set on that are ...

R: weekday and weekend differences, on and off times during the weekday, and on/off times during the weekends, depending on how much money I had and who's paying the bill, and how much it costs to run it all weekend, all day at weekends. And I'd like therefore to be able to put if off, to be able to override - if I'm going to be out all day and if I'm going to be in all day during the week - not just [to be able to] override the threshold, but to just have it on or off and timed (or time sensitive), and then threshold values for temperature within those - so it would be just on at the temperature I give it, regardless .... so I really need not just time of day but temperature control within time of day ...

I: OK. Thank you.

This repeats concepts already introduced.

Bringing the two interviews together into one set of user concepts.

Reading back through the two interviews, even though the interviewees clearly were familiar with very different kinds of heating controllers, nevertheless there are largely common themes that come through:

- There are days of the week, and within that there are different kinds of days: there are days when the default is that the house will be occupied and days when it won't, but then within those classes there are exceptions (e.g. 'working at home' or 'out for the day').
- There are times: when the heating switches on and off, and the current time. A related concept is that there are periods, or phases, when the heating is on.
- There are times when the user is commonly at home (morning, lunchtime, evening) and times when they are not.
- The heating has a current state: off, on, or 'really on' (where the difference between the two 'on' states is whether or not the current temperature is below the set temperature as defined by the thermostat). The heating may be on because the current time is within a heating phase, or because the timer has been over-ridden. Because the Alhambra heating controller does not incorporate a thermostat, we will not push this distinction further in the analysis.
- Both referred loosely to external weather and the effect that has on use of heating system, but it is probably outside the scope of this analysis.
- Both also discussed time changes (GMT to BST).

The resulting set of concepts is actually pretty small, but captures the key user ideas about heating controllers. Obviously, depending on the purpose of the study and the time available, more than two interviews might be used to verify the core concepts – we have used just two for illustrative purposes.

We now consider the core concepts in terms of entities and their attributes. In this study, no obvious important relationships have emerged… yet.

Entity	Attributes
Day of the week	Default type (working-day / weekend) Actual type (at home / out)
Current time	
Heating-period	Start time End time
Period of day	At home or out? (by default) Actually at home or out?
Heating system *(this is an implicit entity)*	Off or on?
Current temperature	Above/below thermostat value

Device concepts: Alhambra

Just as, for the user side, we started from interview data, so, for the device side, we start from an existing description of the system. This was supplemented with a test of the simulation so that the analyst was confident about how it actually worked.

Summary Instructions for the Alhambra central heating controller

The following description is taken from Thomas Green's site (URL above).

These controls determine three periods of heating for each weekday. By clicking the ADVANCE button you can move from day to day and within each day you can move through the ON and OFF times for each period. The PLUS and MINUS buttons alter whichever panel is currently highlighted. Don't try typing in the boxes - just use the PLUS and MINUS buttons.

There are 'periods of heating', each with an 'on time' and an 'off time'.

Both 'days' and 'weekdays' are mentioned, but these are not different concepts. So we will use the single concept of a 'day'.

You can 'move from day to day', so there is a currently selected day – we will have an attribute of 'day', which is whether or not it is selected.

There is also a concept of 'panel' which, although it is only a device concept, is key to being able to use the device, because data can only be changed in the 'panel' that is currently selected.

Note that there are other device concepts mentioned in this paragraph – notably different buttons and boxes – that we do not enumerate for CASSM because they are directly associated with individual actions (as in 'press this button') and no more knowledge about them is needed. So they're not interesting to model.

When the day of the week is highlighted, PLUS moves on to the next day, and ADVANCE brings up the settings for the day, while you can use the COPY button to copy yesterday's settings to today.

'Yesterday' may be a new concept; 'day of the week highlighted' refers to when the 'day' 'panel' is highlighted.

Summarising the device concepts

Just as we did for the user concepts, we can summarise the device concepts in terms of entities and their attributes:

Entity	Attributes
Day of the week	Selected *(note: this may be for operation – i.e. today – or it may be for modification)*

	Previous *(i.e. yesterday)*
Panel	Currently selected
Heating-period	Start time End time

Bringing the two together

We are now ready to construct a first joint description of both user and device sides. For this, we also take into account the information about *how* to change the state of the device and what is available at the interface. We try to state explicitly how each concept is represented within the system, at the interface and by the user.

For Alhambra, we have the following. In each of these sets of tables, the entity and its set of properties is followed by its attributes (if any).

Entity	*User*	*Interface*	*System*	*Create by*	*Delete by*	*Notes*
Day of the week	Present	Present	Present	Fixed	Fixed	

Attribute	User	Interface	System	Set by	Change by	Notes
Default type	Present	Absent	Absent	Fixed	Hard	Common values are weekdays / weekends
Actual type	Difficult	Absent	Absent	Fixed	Fixed	On some systems, this would be achieved with 'manual override'
Selected	Present	Present	Present	Fixed	Easy	You can only select the day before or after the current day
Previous	Difficult	Absent	Present	Fixed	Fixed	User may not remember all settings for the previous day

Entity	*User*	*Interface*	*System*	*Add by*	*Delete by*	*Notes*
Current time	Present	Absent	Present	Fixed	Fixed	On Alhambra, there isn't an obvious rep'n.

Entity	*User*	*Interface*	*System*	*Add by*	*Delete by*	*Notes*
Heating period	Present	Present	Present	Hard	Hard	Alhambra has 3 heating periods per day. This is not changeable, except indirectly.

Attribute	User	Interface	System	Set by	Change by	Notes
Start time	Present	Present	Present	Fixed	Easy	Settable in 10 minute chunks
End time	Present	Present	Present	Fixed	Easy	Settable in 10 minute chunks

Entity	*User*	*Interface*	*System*	*Add by*	*Delete by*	*Notes*
Period of day	Difficult	Absent	Absent	Fixed	Fixed	

Attribute	User	Interface	System	Set by	Change by	Notes
Default whereabouts	Difficult	Absent	Absent	Fixed	Fixed	Should match with heating periods (such that home is warm when default is 'in')
Actual whereabouts	Present	Absent	Absent	Fixed	Fixed	Sometimes, user may remember to 'override' the heating setting to mirror their whereabouts

Entity	*User*	*Interface*	*System*	*Add by*	*Delete by*	*Notes*
Heating system	Difficult	Absent	Absent	Fixed	Fixed	This isn't a 'real' entity – it's the 'system' … but it has an attribute:

Attribute	User	Interface	System	Set by	Change by	Notes
On or off?	Present	Absent	Absent	BySys	Can't	On many systems, the setting can be overridden. It's not obvious that this is the case for Alhambra.

Entity	*User*	*Interface*	*System*	*Add by*	*Delete by*	*Notes*
panel	Present	Present	Absent	Fixed	Fixed	This is a device entity that has no real-world significance.

Attribute	User	Interface	System	Set by	Change by	Notes
Selected?	Present	Present	Absent	BySys	Easy	The user can only change particular parameters when the correct panel is selected.

Whew! What a lot of tabulation! This lot is now presented at the level of detail that enables us to describe it to the Cassata analysis tool. You should note that tables like this are really for writing, not reading. The kind of reflection you do when you're doing an analysis yourself is different from what you do when you're reading someone else's analysis. So these tables are, in a sense, easier to write than they are to read.

By this point, we can clearly see some misfits:

- The device has no direct representation of 'work days' and 'home days'.
- The 'copy' function depends on the user remembering the previous day's settings, which may be a valid assumption in many situations, but not all.
- Periods of the day when the user is in or out should correspond to the heating periods.

At this level of analysis, none of these difficulties seem particularly problematic, so Alhambra is likely to be a reasonably usable design for most users.

Taking this a level further: repetition viscosity

With our current understanding of Alhambra, we can look at it a bit more closely and ask ourselves: if we want to change settings in cognitively meaningful ways, are there likely to be any deeper problems?

One of the things we can do at this point is look at the definitions of the deeper-level misfits, such as some of the Cognitive Dimensions, and see whether any of them apply. If we do this, we'll find the definition of *repetition viscosity* – that 'changing an attribute that has a simple meaning to the user involves repetitive device actions'.

In this case, we've noted that users seem to think of 'sets of days' (although we haven't actually stated that as an entity explicitly so far), and that those days have the same default heater timings. If the user wants to change the heater timings for that set of days, that will involve changing them for each of the days in the set, which is actually very tedious.

This further analysis suggests additional entities, attributes and relationships, as follows:

Entity	*User*	*Interface*	*System*	*Create by*	*Delete by*	*Notes*
Day of the week	Present	Present	Present	Fixed	Fixed	

Attribute	User	Interface	System	Set by	Change by	Notes
Heating-period	Present	Difficult	Present	Easy	Easy	

Entity	*User*	*Interface*	*System*	*Create by*	*Delete by*	*Notes*
Set-of-days	Difficult	Absent	absent	Fixed	Fixed	

Attribute	User	Interface	System	Set by	Change by	Notes
Heating-period	Present	Difficult	Present	Fixed	Fixed	

Relationship	Actor	Acted-on
consists-of	Set-of-days	day
affects	Day(heating-period)	set-of-days(heating-period)

For our purposes we've gone far enough now. Of course, an analyst with greater insight might have spotted these entities and relationships earlier and thence derived the repetition viscosity from the representation, rather than the other way around, but there is no unique 'right way' of doing these things.

Cassata analysis

We have entered this data into the prototype Cassata CASSM analysis tool (see Table 3). When this model is analysed by Cassata, repetition viscosity is highlighted as a possible problem.

		U	I	S	S/C	C/D	Notes
E	day-of-week	present	present	present	fixed	fixed	
A	default-type	present	absent	absent	fixed	indirect	common values are weekdays / weekends
A	actual-type	present	absent	absent	fixed	fixed	on some systems this would be with manual override
A	selected	present	present	present	fixed	easy	you can only select the day before/after the current one
A	previous	difficult	absent	present	fixed	fixed	user may not remember settings from previous day
A	heating-period-day	present	difficult	present	easy	easy	
E	current-time	present	absent	present	fixed	fixed	NB user and system values may be different
E	heating-period	present	present	present	indirect	indirect	Alhambra has 3 periods per day
A	start-time	present	present	present	fixed	easy	
A	end-time	present	present	present	fixed	easy	
E	period-of-day	difficult	absent	absent	fixed	fixed	
A	default-whereabouts	difficult	absent	absent	fixed	fixed	Shoudl match heating periods
A	actual-whereabouts	difficult	absent	absent	fixed	fixed	user may remember to override settings to match their whereabouts
E	heating-system	difficult	absent	absent	fixed	fixed	
A	on-or-off	present	absent	absent	bySys	cant	Overriding is not possible in the simulation, although it is in the real thing
E	panel	present	present	absent	fixed	fixed	device-entity -- has no real-world significance
A	selected-panel	present	present	absent	bySys	easy	user can only change parameters when the right panel is selected
E	set-of-days	difficult	absent	absent	fixed	fixed	
A	heating-period-set	present	difficult	present	fixed	indirect	

set-of-days	consists_of	day-of-week	present	absent	absent
day-of-week.heating-period-day	affects	set-of-days.heating-period-set	notSure	notSure	notSure

Table 3: Alhambra data represented within Cassata

Some of the 'red alerts' in the user/interface/system columns are unlikely to cause particular user difficulties – they simply highlight a high-level misfit between what the user is actually doing (controlling the comfort of the home while also minimising cost by not heating the home when they are out) and the device (Alhambra) that is available as part of the overall system for achieving that. As a final exercise with Alhambra, we suggest that you work through these 'alerts' and consider both why they have been flagged and also whether it would be within the scope of the controller design to correct each of these misfits.

Summary of Alhambra example

This worked example has included two levels of analysis for the Alhambra heating system. The higher level has simply involved comparing user, interface and system representations in terms of core concepts. As shown, even this simple heating controller has several misfits – many of them caused by the fact that it *is* just a simple heating controller that has no information about the user's whereabouts or normal patterns of being in or out of the home. Thus, there are many concepts that are present or difficult for the user but absent from interface and system. These are of less immediate concern than the difficult interface representations of heating period for both day and set-of-days.

The more detailed level has involved relating the system descriptions to Green's Cognitive Dimensions, which has led to the identification of more detailed concepts and relationships, and also to the identification of a structural misfit – repetition viscosity.

Summary of CASSM

Just to recap: CASSM focuses on *misfits* between the user's conception of what they are doing and those implemented in the system. The primary concern is with entities and attributes and their properties. Actions are considered later, and relationships (which help identify various structural misfits) last of all. There is no unique 'right answer' in a CASSM analysis: CASSM is intentionally sketchy, with the intention that the analyst should be able to focus on the things that seem to matter most. Consequently, CASSM analysis demands a reasonable degree of general HCI skill. All the evidence we have indicates that CASSM fills a useful gap in the repertoire of HCI evaluation techniques. We have illustrated some of our own analyses, and the way we think about CASSM, through worked examples – we hope that these give enough of an idea of how to go about doing it yourself.

End note: details and examples of Present–Difficult–Absent combinations

For the reader who loves detail, we include here a summary of the possible combinations of user–interface–system representations. In this section, we use the code:

(user)**P**resent/**D**ifficult/**A**bsent - (interface)**P**resent/**D**ifficult/**A**bsent - (system)**P**resent/**A**bsent.

However, most people can skip this section with little loss.

P-P-P: No difficulties – good fit between user and system.

P-P-A: The analyst may choose to encode interface objects that only affect the display but not the underlying system representation in this way. For example, a 'zoom' function that is clearly represented at the interface with a recognisable magnifying glass might be represented this way. Similarly, a drawing of something that is clearly visible to the user at the interface, but where the 'something' is not represented in the underlying system might be encoded this way. Such a combination in practice rarely causes user difficulties, unless the user wants to manipulate the object in some way.

P-D-P: This combination is likely to cause some user difficulties, depending on the exact reason why the interface representation causes difficulties. As noted above, there are different reasons for interface difficulties, and it is up to the analyst to think through consequences in each situation.

P-D-A: This is an unlikely combination, and it is up to the analyst to consider why they have encoded a concept in this way and what the likely difficulties might be.

P-A-P: This is another unlikely combination. Where it exists, the lack of interface representation means that users need to manipulate system concepts indirectly, which is likely to cause serious difficulties.

P-A-A: This is a common situation and is discussed above as one of the three first-level cases. It causes difficulties.

D-P-P: An example might be that the user is forced to be explicit about some concept that they would naturally not mention. The example of the end times of meetings in an electronic diary has been discussed above. These are only problematic if the user is required to set or change values, not if the user only views preset system settings.

D-P-A: This is another unlikely combination, and it is up to the analyst to consider how it might cause user difficulties.

D-D-P: This combination is likely to cause user difficulties, depending on the exact reason why the interface representation causes difficulties. As noted above, there are different reasons for interface difficulties, and it is up to the analyst to think through consequences in each situation. Also, in this case the user is probably required to make explicit some information they would not normally work directly with.

D-D-A: This is an unlikely combination, and it is up to the analyst to consider why they have encoded a concept in this way and what the likely difficulties might be.

D-A-P: This is another unlikely combination. If this is a valid combination then the lack of interface representation means that users need to manipulate system concepts indirectly, which is likely to cause serious difficulties.

D-A-A: This is another unlikely combination. If it occurs, the analyst should consider the consequences.

A-P-P: As discussed above, these are concepts that the user has to learn; however, because they are clearly represented at the interface, they are unlikely to cause users serious difficulties. For this and other codings that start with 'A', the analyst might prefer to use 'D's if the user has to learn the concept, rather than 'A's, which indicate that the concept is absent. This is a matter of analyst preference.

A-P-A: This is an unlikely combination, unless the analyst chooses to encode interface objects that only affect the display but not the underlying system representation in this way. These will be interface objects that are (presumably) easy to learn and only affect surface aspects of the interaction anyway, and are therefore unlikely to cause great difficulties.

A-D-P: This combination is likely to cause user difficulties: these are important system concepts that are poorly represented at the interface in some way, but that the user has to learn to work with. The example of layers in a drawing package has been discussed above.

A-D-A: This is such an unlikely combination that it is equivalent to being absent from all three situations, which should never arise.

A-A-P: This is a source of user difficulties: something the user has to learn about if they are to work effectively with the system, but which cannot be accessed or manipulated through the interface. This is a fairly unusual situation, but does occur; for example, some drawing packages implement Bezier curves that can be drawn but not subsequently manipulated. The user therefore has to work with the Bezier curve points, but they are not accessible through the interface. Similarly, the '9.30 rule' on the London Underground system affects ticket purchases but it not explicitly represented at the interface of all ticket machines.

A-A-A: There is absolutely no point in encoding these concepts within CASSM! The only situation in which you might include them is where you are modelling several different classes of user with the same interface, where some classes of user do not have a particular concept but others do.

To summarise, few of the combinations where a concept is not represented in the underlying system are common. And overall, five combinations are both common and likely to cause difficulties, for reasons explained in more detail above.

Glossary (summary of key terms and concepts)

- An ***entity*** is usually something that can be created or deleted within the system. Sometimes, entities are things that are there all the time, but that have attributes that can be changed.
- An ***attribute*** is a property of an entity.
- For every concept, the analyst determines whether it is ***present***, ***difficult*** or ***absent*** for the ***user***, at the ***interface*** and in the underlying ***system***.
- For users ***difficult*** concepts may be ***implicit***, ***hard to learn,*** perceived to be ***irrelevant*** or difficult for some other reason (that we haven't thought of yet!).
- Difficulties that interface objects may present include:
  - ***Disguised***: represented, but hard to interpret;
  - ***Delayed***: represented, but not available to the user until some time later in the interaction;
  - ***Hidden***: represented, but the user has to perform an explicit action to reveal the state of the entity or attribute; or
  - ***Undiscoverable***: represented only to the user who has good system knowledge, but unlikely to be discovered by most users.
- For every concept, the analyst also considers the ***actions*** that can be performed on it. For an entity, these are ***creating*** and ***deleting***; for an attribute, these are (initial) ***setting*** and ***changing*** the value.
  - ***Easy***: no user difficulties.
  - ***Hard***: difficult for some reason (e.g. undiscoverable action, moded action, delayed effect of action).
  - ***Indirect***: effect has to be achieved by changing something else in the system.
  - ***Can't***: if particular state change is impossible, and this is likely to cause problems, then it should be noted as 'can't'.
  - ***Fixed***: many cases of 'impossible' are not, in fact, problematic. *Fixed* means that something cannot be changed, but that this is unlikely to be a problem.
  - ***Done by 'the system'*** – i.e. done by some agency other than the user being modelled.
- In Cassata, the s/c column refers to setting attributes and creating entities. The c/d column is about changing attributes and deleting entities. [We know this is hard to learn: Cassata does itself have misfits!]
- Finally, the analyst might want to consider ***relationships*** between concepts
  - ***Consists_of*** takes two arguments, Actor and ActedOn, which are both concepts. This means that the first consists_of the second.
  - ***Device_constraint*** also takes two arguments, both concepts. The value of Actor constrains the possible values of ActedOn.
  - ***Goal_constraint*** takes two arguments that express the idea that some (Actor) domain concept (such as a social convention) constrains some other (ActedOn) concept.
  - ***Affects*** is concerned with side-effects. That changing the value of one concept will also change the value of another.
  - ***Maps_onto*** is a simple way of expressing the idea that two concepts are very similar but not quite identical. The user typically has to use the second argument as a surrogate for the first.

Examples of every concept and classification are as follows. We hope these examples are familiar – if not, don't worry, and think of your own examples for yourself. You also have the right to disagree with some of our classifications, especially if you have data to back up your view!

entity	document, paragraph, set of figures, etc..
attribute	length, colour, etc..
User: ***present***	document
User: ***difficult***	***implicit***: end time of meeting; ***hard to learn***: layers in a drawing package; ***irrelevant***: account type when user only has one type
User: ***absent***	the points underlying a Bezier curve in a drawing package (probably!)
Interface: ***present***	paragraph
Interface: ***difficult***	***Disguised***: obscure screen widgets; ***Delayed***: railcard types (on most ticket machines); ***Hidden***: word count in a typical word processor; ***Undiscoverable***: points underlying a Bezier curve….
Interface: ***absent***	ticket pricing rules (for most known ticket machines)
System: ***present***	paragraph
System: ***absent***	the scene a user is drawing (e.g. 'a house')
Action: ***Easy***	Creating words by typing
Action: ***Hard***	***Undiscoverable***: selecting a sentence in MS Word (it's option-click on a Mac); ***Moded***: in MS Word, tab has a different effect if a paragraph is selected to that which it has 'normally'; ***Delayed effect***: correcting source html in Dreamweaver only has effect when user clicks again in WYSIWYG window. ***Non-obvious precondition***: some html forms only accept input after the user has explicitly selected the field (even the first one).
Action: ***Indirect***	Changing the length of a document (by typing, changing font size, etc.)
Action: ***Can't***	Some forms have fixed-width (too small!) fields.
Action: ***Fixed***	The user of the robotic arm cannot create or delete robotic arms (but can change arm attributes)
Action: ***Done by 'the system'***	Number of messages in user's mailbox can be increased by the system automatically checking the mail server.[5]
Consists_of	A paragraph consists of sentence
Device_constraint	Disk capacity imposes a device constraint on how many files can be stored
Goal_constraint	Figure numbers should form a sequence.
Affects	Changing the number of words in a document affects its length
Maps_onto	For the robotic arm, the user has to use the gripper position as a surrogate for the position of an object in the world.

[5] This one is slightly problematic in Cassata because the user can also change the number of mail messages by deleting them, so responsibility is shared between agents – the tool isn't perfect!

Acknowledgements

This work is supported by EPSRC grant GR/R39108. We are grateful to the various people (too many to list) who have given feedback on earlier versions of this tutorial and on the design of CASSM.

References

Beyer, H., Holtzblatt, K.: *Contextual Design*. San Francisco : Morgan Kaufmann. (1998).

Blackwell, A. & Green, T. R. G. (2003) Notational Systems – The Cognitive Dimensions of Notations Framework. In J. Carroll (ed.), *HCI Models, Theories and Frameworks,* pp. 103-134. Morgan Kaufmann.

Blackwell, A., Hewson, R., Green, T. R. G.: The design of notational systems for cognitive tasks. E. Hollnagel (ed.) In E. Hollnagel (Ed.), *Handbook of Cognitive Task Design*. Mahwah, N.J.: Lawrence Erlbaum. (2003) 525-545.

Blandford, A. E., Wong, B. L. W., Connell, I. W. & Green, T. R. G. (2002) Multiple viewpoints on computer supported team work: a case study on ambulance dispatch. In X. Faulkner, J. Finlay & F. D. Étienne (eds), *Proc. HCI 2002 (People and Computers XVI*), pp. 139-156. Springer.

Blandford, A. E., Young, R. M.: Specifying user knowledge for the design of interactive systems. *Software Engineering Journal*. 11.6, (1996) 323-333.

Blandford, A., Green, T. & Connell, I. (2004a) Formalising an understanding of user–system misfits. To appear in *Proc. EHCI-DSVIS 2004*. Preprint available from CASSM project website.

Blandford, A., Hyde, J., Connell, I. & Green, T. (2004b) *Scoping Analytical Usability Evaluation Methods: a Case Study*. Working paper available from http://www.uclic.ucl.ac.uk/annb/CASSMpapers.html

Connell, I., Green, T., Blandford, A.: Ontological Sketch Models: highlighting user-system misfits. In E. O'Neill, P. Palanque & P. Johnson (Eds.) *People and Computers XVII, Proc. HCI'03*. Springer. (2003) 163-178.

Green, T. R. G. & Blandford, A. E. (2004) Install and use Cassata. Working paper included in the "Cassata shrink-wrap" available from http://www.uclic.ucl.ac.uk/annb/CASSM.html.

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

Green, T. R. G., Benyon, D.: The skull beneath the skin: entity-relationship models of information artifacts. *International Journal of Human-Computer Studies*, 44 (1996) 801-828

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

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

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.

John, B. & Kieras, D. E. (1996) Using GOMS for user interface design and evaluation: which technique? *ACM ToCHI* 3.4. 287-319.

Nielsen, J.: Heuristic evaluation. In J. Nielsen & R. Mack (Eds.), *Usability Inspection Methods*, New York: John Wiley (1994) 25-62.

Wharton, C., Rieman, J., Lewis, C., Polson, P.: The cognitive walkthrough method: A practitioner's guide. In J. Nielsen & R. Mack (Eds.), *Usability Inspection Methods*. New York: John Wiley (1994) 105-140.

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.), *Wings for the Mind: CHI '89 Conference Proceedings*, pp.15-19. ACM conference on human factors in computing systems, Austin, Texas, April-May 1989. Reading, MA: Addison-Wesley.