Understanding structural misfits between user and system: CASSM as an approach to reasoning about selected Cognitive Dimensions

Ann Blandford[1] & Thomas R G Green[2]

[1] University College London Interaction Centre, UCL, Remax House, 31-32 Alfred Place, London WC1E 7DP
[2] University of Leeds

Summary

One perspective on Cognitive Dimensions (CDs) is that they provide a language for reasoning about a poor fit between the way a user thinks about what they are doing and the way a system forces them to behave in the interaction to achieve their goals. These 'poor fits' can have many different sources, including the following:

- There may be concepts that are important to the user that are simply not represented within the system, and for which the user has to find a work-around. Conversely, there may be system concepts that users are unaware of, and which hamper their work if they cannot learn about them easily. These 'surface' misfits, which do not refer to the underlying structure of the system representation, are important, but are not well covered within the CDs vocabulary.
- Other misfits result from constraints – often hidden -- in the underlying structure of the system representation that do not conform to the user's conceptualisation. These typically result in tasks which are conceptually simple for the user being difficult or tedious to achieve with the existing system implementation. CDs have been developed to support reasoning about many such structural misfits.

We have been developing CASSM (www.uclic.ucl.ac.uk/annb/CASSM/) as a methodology for reasoning about misfits of the two types outlined above. The approach to a CASSM analysis is to take data from naturalistic use, to construct a descriptive model from that data, and then to reason from the model about surface and structural misfits. CASSM analysis is supported by a prototype data representation and analysis tool called Cassata, which includes routines to check for a subset of CDs which have been formally defined.

Consider a small example: a standard calculator. We can describe some of the central features of a calculator and its use in CASSM terms as follows:

- The user knows about a **formula** (e.g. $(3+2)*5/2^2$), to which they want to know the answer. There is no direct system representation of this.
- They have to convert this into a **calculation** expressed in machine terms.
- There is a requirement (domain constraint) that the calculation should return the value corresponding to the formula. Conversely, the value returned will determine (affect) the perceived formula value. In addition, the two concepts 'formula' and 'calculation' map onto each other, in the sense that the calculation is the system representation of the user's formula.
- The user has to be aware of the contents of **memory** and how to manipulate it.

These ideas are captured in the CASSM model below.

entities and attributes:

	Entity/attribute	User	Interface	System	Create or set	Delete or change	Notes
E	memory	difficult	present	present	fixed	fixed	
A	contents	difficult	absent	present	easy	hard	
E	calculation	difficult	difficult	present	easy	hard	A device concept
A	value	difficult	difficult	present	easy	hard	
E	formula	present	absent	absent	easy	easy	A user concept
A	value	difficult	absent	absent			

relationships:

				User	Interface	System
0	calculation.value	goal_constraint	formula.value			
1	calculation.value	affects	formula.value			
2	formula	maps_onto	calculation	difficult	absent	absent

In terms of CDs, such calculators typically suffer from:

- **Viscosity**: a minor change to the formula results in a major change to the calculation: usually a total reworking;
- **Visibility**: users cannot see the values of all items, such as calculation history or the contents of memory.

Knock-on viscosity is reported by Cassata for this model as follows:

```
Knock-on Viscosity Check

 "calculation.value" affects "formula.value"
 there is a goal_constraint on "formula.value"
 "calculation.value " is directly modifiable

possible case of knock-on viscosity
modifying "calculation.value" may violate a domain constraint for "formula.value"
====
```

Additional calculation is not required to identify **visibility** problems: such difficulties are represented directly in the Cassata model; for example, the fact that memory.contents are absent from the interface indicates a problem with their visibility.

The Cassata model highlights some additional misfits that are not explicitly captured by CDs. One such example is highlighting the fact that, for the traditional calculator, making errors is easy but error recovery is difficult. This is captured in the statement that the calculation.value is easy to set but hard to change.

The development of CASSM (as hence the Cassata prototype) has been highly exploratory. This applies both to the development of the ontology of the model, in terms of what features of the user's conceptualisation and the system representation to incorporate in the model, and to the definitions of CDs that have emerged through the work (and are now implemented in the Cassata tool). In contrast to the work of Roast *et al* on CiDa (as reported at this workshop), our definitions of CDs have emerged from analysis of examples. Thus, the quality of the definitions is highly dependent on the quality and consistency of the examples from which they are derived. Three classes of CDs (2 variants of Viscosity, 3 of Premature Commitment and one of Hidden Dependences) have been tested thoroughly, and are therefore included in the Cassata test suite.

For this workshop, based on the calculator example presented above, we highlight an unresolved issue of definition: more constrained than those raised by Green's paper on the Dark Corners of CDs (presented at this workshop), but nevertheless symptomatic of an interesting difficult. In that paper, Green succinctly defines Premature Commitment as:

“environments that constrain the order of actions against the flow of the dependencies in the notation”. With this definition, it would appear that traditional calculators are prone to premature commitment: the user has to translate the natural order of a formula into the imposed order of a calculation. It would be possible to extend the model above and also to propose (yet another) definition of premature commitment – for example, as follows:

Calculation-elements have a sequence

Formula-elements have a sequence

Calculation-elements.sequence affects calculation.value

Calculation-elements.sequence ≠ Formula-elements.sequence

However:

- This does not conform to any of the existing definitions of premature commitment, indicating that either the existing definitions are based on an inappropriate representation or that the (informal) concept of premature commitment is quite weakly defined.
- Within this definition, the relationship “not equal to” is central, as is the notion of a sequence, and neither of these concepts is directly represented within CASSM. This may indicate a limitation of CASSM as it is currently specified.

In the nature of the way that CDs have been developed over time, the only way to validate or refine such a definition is by testing on further examples. It might be found, for example, that some of the existing repertoire of premature commitment examples can be reframed in the terms outlined above, providing more support for the view that the CASSM representation would gain power if modified slightly. Conversely, it might be found that each new example demands a new definition, which would be further evidence that premature commitment is itself weakly defined.

Just as the development of CASSM has been iterative and reflective, so the development of most models has involved cycles of iteration. In particular, while it would be satisfying to give a clear account that the identification of CDs (such as the viscosity described above) emerged directly from the model, in practice the reasoning often goes the other way: for example, a system is believed to be viscous, so what has to be true of the model for that viscosity to be identified? This, in turn can be valuable in that it helps in the articulation of concepts and relationships that might otherwise go unrecognised. The recognition of these features could be central to the design of systems that better fit their users.