Human-Centered Computing and Representation: A Framework

Clayton Lewis
Department of Computer Science
and Coleman Institute
University of Colorado
Boulder, CO 80309 USA
clayton.lewis@colorado.edu

CHI 2009, April 4 – 9, 2009, Boston, MA, USA
ACM 978-1-60558-246-7/09/04.

Abstract
The CompRep framework is a way of thinking about computation that clarifies the relationship of human-centered computing to other aspects of computing.

Keywords
Human-centered computing, representation, theory

ACM Classification Keywords
H1.2. [User/Machine Systems]: Human factors; H.5.2. [User Interfaces]: Theory and methods.

Introduction
The CompRep framework is a *perspective project*, a way of thinking about computation that clarifies the relationship of human-centered computing to other aspects of computing. The organizing theme of the framework is, "computational systems are *representational* systems". By taking representation to be the purpose of computational systems, and building on theoretical work on the nature of representations, we get insights into those aspects of computing that are distinctively human-centered, and material for agendas in education and research. The aims of this work in progress presentation at CHI are to get feedback from the CHI community on the framework, and to solicit participation in developing the ideas.

What is Computer Science about?

"Computer science is no more about computers than astronomy is about telescopes," is a view widely attributed to Edsger Dijkstra. But then, what *is* it about? We computer scientists haven't been very good at explaining the intellectual basis of our field, and this may be one of the reasons we are struggling to interest students, and potential collaborators, in our field, even while the importance of computing in all aspects of life is growing by leaps and bounds.

Computational systems are important because they are used to *represent* all kinds of things people care about, like airplane wings, employees, or social networks, and representations are really useful. Further, as we'll see, *computational* representations have a number of properties that make them *especially* useful.

The nature of representations.

So, what is a representation? There's some theory of representation that we can draw on, best developed for measurement systems, a special case (see [3], for the broader theory of representations, see [6,7]).

A representation has to be understood in the context of a representational *system*, which consists of a target domain, in which there is something we want to accomplish, and a representation domain, with mappings connecting them. The point of representations is that we map work in the target domain into work in the representation domain, where it can be done more easily, or faster, or better in some other way. Then the results are mapped back to the target domain, where we need them.

Suppose we have two cars, and we want to see which has greater fuel economy. We could compare the cars directly in some way, but life is much easier if someone provides a representation, say in the form of a bar chart. The bar chart corresponds to the real situation with the cars, in that if one car is more economical than the other, then its bar is longer. We map the difficult question, which car is more economical, into the easier question, which bar is longer?

Figure 1 shows the representational system here. The target domain contains the cars, and the representation domain the bars.

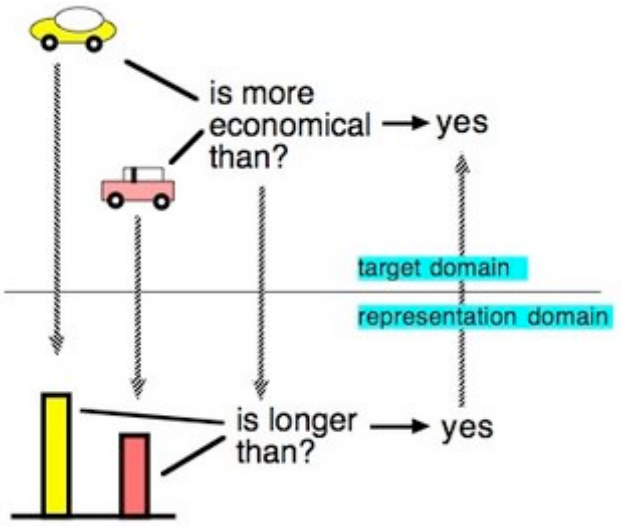

figure 1. Representational systems connect a target domain to a representation domain.

Our question in the target domain is "Which car is more economical?" We map this question onto a question in

the representation domain: "Which bar is longer?" Finally, we map the answer we get in the representation domain into the target domain.

Cost structure is crucial in representations.
As stressed earlier, the whole point of having a representation is to be able to do something more easily, or cheaper, or faster, or better in some way, and computational representations often have huge advantages over others. Here are a few examples.

- People can use a computational representation of something to do work on the other side of the world, quickly and cheaply.
- Computational representations can easily be accessed in other *times*, as well as in other places. That is, they are easy and cheap to store and retrieve.
- Many operations on computational representations can be *automated*, meaning that they can be carried out by a machine, rather than a person. This often offers huge advantages in speed, accuracy, and cost.

We'll treat all of these advantages (speed, effort, etc.) under the general heading of *cost*.

If computational representations didn't offer these cost advantages they would rarely be used. But many, in fact most, computational representational systems include operations that are implemented not by computers but by people, the users. This means that the overall cost structure of the representational system, the structure that determines whether or not the system is useful, includes not only the attributes of the computational operations that are included, but also the attributes of the operations performed by people.

The cost structure of human operations: An example.
Figure 2 shows a bar chart as in Figure 1 along with two alternatives, in which the bars are displayed differently. All three charts express the same information, but the alternatives B and C present it in ways that people find difficult to process, because of the way their perceptual apparatus works. So, design of effective information presentation depends on understanding this cost structure.

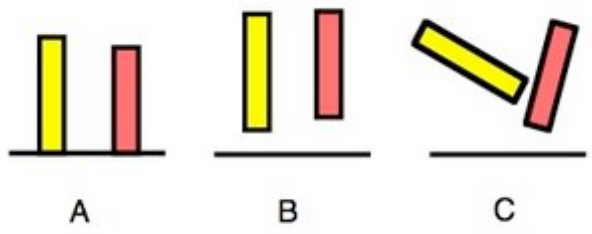

figure 2. Alternative bar presentations.

Human-Centered Computing.

Here is the synopsis of the program description for the Human-Centered Computing Cluster at the US National Science Foundation [8]:

"This cluster, Human-Centered Computing (HCC), encompasses a rich panoply of diverse themes in Computer Science and IT, all of which are united by the common thread that human beings, whether as individuals, teams, organizations or societies, assume

participatory and integral roles throughout all stages of IT development and use."

The description goes on to list a wide range of topics, including multi-modal interfaces, intelligent user interfaces, visualization, multi-agent systems, collaboration and conferencing, accessibility and assistive technology, affective computing, studies of social organizations, and many more.

Can this broad area of work, defined mainly by example, usefully be given more conceptual coherence? Focusing on the role of people in the representational work of computational system suggests that it can. Further, the representational perspective shows that HCC is not, as some have thought, a peripheral aspect of computing, but rather a central aspect, not just in its practical importance, but also in the intellectual content it shares with other areas of computing.

For a representational system to work, the cost structure of its human operations has to meet the same conditions as the cost structure of its automated operations, including conditions on accuracy and reliability. Here are more examples that illustrate factors that influence the cost structure of operations performed by people in representational situations.

Recognizing objects or people in a scene, or judging aesthetic qualities, can be done from pictures. That is, if a real scene isn't at hand, it can be represented by a picture, and judgments made from the picture can substitute for the judgments that would be made from the real scene. Conveniently, suitable pictures can be created and displayed by computer, taking into account the specific characteristics of human vision, including color perception (for Homo sapiens the brightness of just three colors can be chosen to reproduce perfectly any color; a display for dogs would need only two colors, because dogs discriminate only short from long wavelengths); limited spatial resolution and blending (closely spaced dots merge into a smooth-appearing area); and depth perception. Presentation of movies and videos relies on further facts about how people see: images closely spaced in time merge into the appearance of smooth motion.

These examples bring out what the point of human centered computing is. Effective displays are not based on faithful physical and geometric reproductions of the signals available from real scenes. Rather, they systematically exploit specific facts about the human perceptual apparatus. Analogous considerations apply to input apparatus, like keying, drawing, and speaking.

HCC and social systems.
Humans are social animals: most things that people do are done in groups. Designers have to understand not just what individual people are likely to do, and can do, but what people working in groups will and can do. For example, sometimes systems fail because some users don't do things, like entering information into a data base, that are needed to support other users. But sometimes, as in Wikipedia, people working as volunteers put huge amounts of effort into really useful contributions. If you are designing a system for a lot of people to use, you have to try to understand what causes contrasts like this.

Programming, representations, and HCC.

Computational representational systems are built up by *programming*, the process of piecing together

computational operations so as to reflect the structure of a target domain. Programming starts with a repertoire of primitive operations in a programming language, and uses notations that specify how these operations should be put together, or *composed*, to represent the operations needed in the representational system. These notations have to be used by people, and the cost of using them to create a representational system has to be included in the overall cost structure of the system. So a critical question is, what is the cost structure of the associated human operations? A small example can illustrate the considerations.

Suppose we are working on the specific problem of adding an echo to a digitized bird song. Some preliminary programming will give us operations that make a sound softer, that delay it, and that mix together two sounds so that we hear them together. Here are two ways of specifying how to compose these operations to produce the song with an echo:

```
#Version 1
mix(originalSong,delay(makeSofter(originalSong)))
#Version 2
softerSong=makeSofter(originalSong)
echo=delay(softerSong)
mix(originalSong,echo)
```

Version 1 uses *nesting*, a notational device borrowed from mathematics, to specify the composition. Version 2 has no nesting of operations; composition is specified entirely by the use of variables that hold the results of earlier operations and allow them to be reused. What are the merits and demerits of each?

Human-centered analysis of programming notations.

The best developed framework for this is Cognitive Dimensions analysis, developed by Green, Petre, Blackwell, and others [1,2] Here are a few dimensions:

Hard mental operations: high demand on cognitive resources.

Both versions require hard mental operations. In 1 one has to parse the expression to determine what is composed with what, and in what way (when mappings take more than one parameter the possible compositions multiply). In 2 one has to trace the history of the variables to recover the same information. Neither task is easy in general.

Closeness of mapping: closeness of representation to domain.
Role-expressiveness: the purpose of a component is readily inferred.
These two dimensions point to advantages of Version 2 over 1. The introduction of named variables in 2 picks out correspondences between entities in the program and entities in the target domain, an example of closeness of mapping. At the same time, these correspondences help the reader understand the roles of parts of the program. Perhaps we can create notations that ease the hard mental operations in both versions.

Why does a representational perspective matter? Agendas for education and research.

The CompRep framework has been useful in teaching an introductory Computer Science course to nonmajors.

Because both the target domain and the representation domain are in focus, the approach clarifies what Computer Science is about, in a way that invites interest from students for whom the computer itself, like Dijkstra's telescope, may not be fascinating. When applied to programming CompRep helps students use what they know about target domains, like sound, to structure their programming tasks, which are new to them. See detailed discussion in [4]. Can this educational beginning be developed?

On the research side, the CompRep perspective suggests that better understanding of the cost structure of human operations would be fruitful in driving innovation in systems design. More research should combine studies of psychophysics and perception with experimentation on computer presentation of information. An even larger opportunity offers for work on the cost structure of human operations in programming. The few beginnings in this area show that the critique of Card and Newell [9] almost 25 years ago still holds: "Millions for compilers but hardly a penny for understanding human programming language use."

The cost structure of social operations also badly needs inquiry. Our ability to predict what operations will and will not be effectively carried out by self-organizing volunteers is weak, yet this kind of work is already of decisive importance in many real-world activities.

The theory of representation itself needs more development. There are potential connections to category theory (in mathematics) and philosophical work on representation by Cummins, Millikan, and others. See [4,5].

Acknowledgements

I thank the NSF Broadening Participation in Computing program. George Furnas, Brian Cantwell Smith, Peter Polson, and Randolph Bias provided valuable feedback.

Citations

[1] Blackwell, A. Human Computer Interaction Notes, http://www.cl.cam.ac.uk/Teaching/2000/AGraphHCI/HCI/hcinotes.html#cds (2000).

[2] Blackwell, A. Cognitive dimensions resource site. http://www.cl.cam.ac.uk/%7Eafb21/CognitiveDimensions/ (n.d.).

[3] Krantz, D., Luce, D., Suppes, P., and Tversky, A. *Foundations of Measurement*. Dover Publications (2007).

[4] Lewis, C.H. CompRep Blog http://comprep.blogspot.com (2008).

[5] Lewis, C.H. A theoretical view of Human-Centered Computing. http://spot.colorado.edu/~clayton/HCCNotes1.pdf (n.d.).

[6] Mackinlay, J. and Genesereth, M. Expressiveness and language choice. *Data & Knowledge Engineering* (1985), 17-29.

[7] Mackinlay, J. D. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics*, 5, 2, (1986), 110-141.

[8] National Science Foundation HCC Cluster Description http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=500051 (n.d.)

[9] Newell, A., and Card, S. The prospects for psychological science in human-computer interaction. *Human Computer Interaction* (1985) 209-242.