Experiences with Cognitive Dimensions

Margaret Burnett, Jason Dagit, Joseph Lawrance, Laura Beckwith, Cory Kissinger
Oregon State University
{burnett, dagit, lawrance, beckwith, ckissin}@cs.orst.edu

As a researcher often in the situation of designing programming language or programming environment features, I (MMB) became acquainted with the Cognitive Dimensions framework (CDs) [Green and Petre 1996] around 1995. I quickly became one of its biggest fans. Since then, our group has used CDs in almost every research project we've conducted. After these 10 years of using CDs for a number of purposes, we briefly reflect on our experiences.

Using CDs for evaluating existing and emerging systems

A group of us began by using CDs to evaluate two existing systems that we had written. We gained immediate insights into issues we had neglected to consider before. We were immediately so impressed with this aspect that, ever since that time, we have used CDs to evaluate any new subsystems and languages that we design.

Still, in that first experience, we quickly ran into trouble, and we've later seen other researchers have the same trouble. The trouble we had was this: what can be accomplished with CDs for the purpose of evaluating existing systems is easily misunderstood. Many researchers do not realize that CDs are limited in the same way that testing is limited. In neither case is it possible to "prove" that a system is acceptable; rather, it is only possible for these mechanisms to find the existence of problems. Yet, people are often tempted to use CDs (and testing) in exactly this way, as though they really can be used to "prove usability."

This problem is exacerbated when the system being evaluated is one's own system, because then a conflict of interest arises: there is a strong incentive to decide that the system has no usability issues. Thus, as soon as the system's designer finds an opportunity to approve some aspect of a CD, he or she may be tempted to pronounce that CD as being satisfactorily considered, rather than thinking a little longer before moving on to the next CD.

On the other hand, when the system being evaluated is an emerging system, these conflicts and temptations disappear. In an emerging system, the designer is highly motivated to find every single problem possible, so as to quickly address problems early in the design stages rather than having them crop up at a later, more expensive time, such as after implementation.

Because of these motivational issues, it has been our experience that researchers are much more effective and productive using CDs on emerging systems than they are at using CDs on existing systems.

CDs as a basis for Representation Design Benchmarks

As a result of our early experiences, we devised Representation Design Benchmarks [Yang et al. 1997]. These are an adaptation of *some* of the Cognitive Dimensions to make them quantitative. (Only some CDs were applicable to the problem Representation Design Benchmarks were devised to address, namely the static representation of a visual program in a visual programming environment.) Because they are quantitative, we have found Representation Design Benchmarks to be more objective than CDs, and therefore not so readily misused. Also, they are especially well suited for determining progress and movement through the design space as a system's design decisions are changed.

Nowadays, we use both Representation Benchmarks and CDs to evaluate emerging systems. Despite their common basis, they are different enough that each always succeeds at finding problems the other misses.

CDs for educating CS students and CS researchers

Cognitive dimensions are a low-cost way of quickly making Computer Science language and environment designers aware of critical human-oriented issues in the systems they are creating. we view them as an extremely valuable resource for starting the education of Computer Science researchers in human-oriented issues. Those who become more serious about the human side of languages and environments will go beyond CDs to learn other analytical evaluation techniques, and eventually will move to empirical work as well. But even those who never go beyond CDs in their study of the human side of programming systems will gain enough insights from CDs to be more successful than they were before at designing systems that have some possibility of being actually useful to humans.

References:

[Green and Petre 1996] T. Green and M. Petre, Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework, *Journal of Visual Languages and Computing* 7(2), June 1996, 131-174.

[Yang et al. 1997] S. Yang, M. Burnett, E. DeKoven, and M. Zloof, Representation Design Benchmarks: A Design-Time Aid for VPL Navigable Static Representations, *Journal of Visual Languages and Computing* 8(5/6), October/December 1997, 563-599.