Chapter to appear in: M. Banich & D. Caccamise (Eds.) *Generalization of Knowledge: Multidisciplinary Perspectives*. New York: Psychology Press.

# Beyond Human-Computer Interaction: Meta-Design in Support of Human Problem-Domain Interaction

**Stefan Parry Carmien**
Department of NeuroEngineering
Fatronik Technological Foundation
San Sebastian, Spain
scarmien@fatronik.com

and

**Gerhard Fischer**
Center for Lifelong Learning and Design
University of Colorado, Boulder
gerhard@colorado.edu

## Keywords:

universe of one, Turing tar pit, meta-design, human-computer interaction, human problem-domain interaction, domain-oriented design environments, division of labor, design trade-offs

## Introduction

The history of interactions between computational machinery and humans has been one of increasing specificity, from the complete generality of machine languages to specialized domain oriented design environments. However, as designers, we need to balance the ease of use of specific tools with the flexibility and scope of general ones. Our research over the last two decades has been focused on creating socio-technical environments to empower human beings as designers and users of computational artifacts. This chapter explores the issues involved with creating environments that broadly support domain activities yet are open enough to accommodate changing user and task needs.

The development of computational environments has been driven by the fundamental design tradeoff between generality and specificity. In this context, generality and specificity refer to an axis of plasticity in the composition of computationally based tools intended for end users (in contrast to tools intended for professional programmers). On one end of he continuum there is assembly and machine language, the basic instruction set for instructing a computer with which everything is possible but the expertise needed to use them is reserved for systems programmers, high tech scribes. At the other end are computer devices and systems that are specifically designed to do one thing; ATM machines are good examples, as are modern elevator controls. These systems are easy to use and efficient in supporting the user's desires (e.g.: getting 100 euros from an ATM, going to the 13th floor) by supporting the demands of specific tasks. Generality seems to be a highly desirable goal because the same tool could be used in many different contexts; however being broadly applicable for all kinds of users and all kinds of tasks comes with a substantial cost which can be characterized by the Turing Tar Pit (Perlis 1982):

> *"Beware of the Turing Tar Pit, in which everything is possible, but nothing of interest is easy."*

These environments are based on a level of representation that is too far removed from the conceptual world of the knowledge workers in specific domains. They emphasize *objective computability* (i.e.: what can be computed in principle), but they pay little attention to *subjective computability* (i.e.: what can people do with a reasonable amount of effort and with limited detailed knowledge about the computational environment).

The other end of the design spectrum can be characterized by the *Inverse of the Turing Tar Pit*:

> *"Beware of the over-specialized systems, where operations are easy, but little of interest is possible."*

These systems can be so fitted to the tasks they are designed for that doing anything besides the exact task in the specific way it is tailored for is difficult or impossible. Modifying these systems to do things differently than the way provided leads to frustration or abandonment.

In our research over the last two decades, we have explored different approaches to create more usable, useful, effective, and enjoyable human-centered computational environments (Norman 1993) with the following research activities and developments:

- creating a deeper understanding of design and its support with *domain-oriented design environments* (Fischer 1994);
- advancing human computer interaction to *human-problem domain interaction (Fischer and Lemke 1988)*;
- supporting not only reflective practitioners (Schön 1983), but *reflective communities (Fischer 2005);* reflective practitioners engage in reflection-in-action meaning reflection that is triggered by breakdowns that occur in the action mode of design. *Domain-oriented design environments* support reflection-in-action with critics (to interrupt action and notify the designer of a possible breakdown) and argumentation (to support reflection). Supporting reflective practitioners is important, but it is not enough. Complex design problems need *reflective communities* because they require more knowledge than any single person possesses and the knowledge relevant to a problem is usually distributed among stakeholders
- exploring *meta-design* as "design for designers" (Carmien and Fischer 2008).

In this chapter we analyze these developments from the perspective of the design trade-offs between "domain-specific" and "domain general". This differentiation can be applied to the computational environments constructed and to the knowledge background of the people using these environments. We will be discussing two specific applications: (1) a socio-technical environment supporting and empowering people with cognitive disabilities and (2) an educational framework illustrating that the principles applied to the design of computational systems can be successfully applied to the design of social systems. A social system can be considered as a web of interrelated communications which support the unity, the existence and the evolution of the system (Luhmann, Bednarz et al. 1996). We will conclude by presenting several possible lessons learned and challenges ahead that our research and projects have illuminated.

## From Human-Computer Interaction to Human Problem-Domain Interaction

When computer systems first emerged, users were required to express themselves in the machine language of that system. These languages were completely general: their semantics was not tied to any specific problem domain. The conceptual distance for a human (working in a certain domain) who wanted to model a problem was very large as indicated in Figure 1.

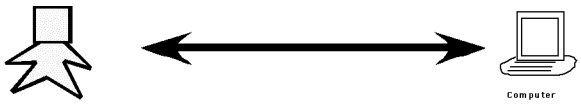


**Figure 1: Programming — in the Very Early Days**

The first fundamental development was the creation of assembly languages and high-level programming languages as indicated in Figure 2. These developments still retained the generality, but they facilitated and supported specific domain-oriented operations (high level computer languages such as APL, LISP and Smalltalk were designed to make certain abstractions easier to implement and manipulate; APL was focused on matrices, LISP on dynamic data structures and recursion, and Smalltalk on object-oriented programming). At the same time, these developments led to a division of labor (Levy and Murnane 2004): compiler developers emerged as a new class of computer professionals that developed compilers thereby allowing most of the professional programmers to program in higher-level languages.

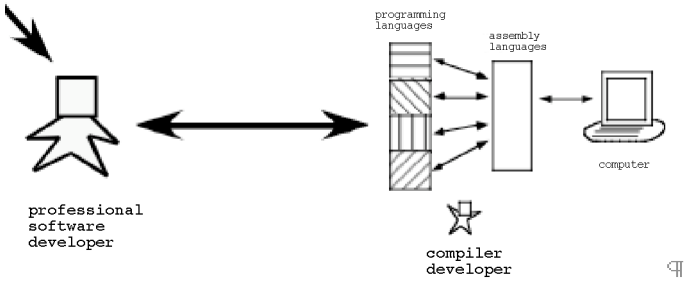


**Figure 2: Support for Human-Computer Interaction with High-Level Programming Languages**

The initial developments (moving from the architecture in Figure 1 to Figure 2) represented the beginning of creating computational environments to support users engaged in specific domains. In the history of computers and computer languages, the concept of "domain" has always been an important dimension, albeit most often an implicit one. In this chapter domain indicates a grouped set of human actions in the world that are together aimed at a common goal; the domain of kitchen design consists of all the design activities that lead to the plan (and implementation) of a kitchen, similarly the domain of supporting activities of daily living by persons with cognitive disabilities consists of persons with cognitive disabilities, caregivers and the artifacts that aid them in this process. Domains have specific (and bounded) areas of interest and expertise. The names of early high level programming languages such as COBOL and FORTRAN reveal their focus on specific domains: (1) COBOL is an acronym for **C**ommon **B**usiness-**O**riented **L**anguage, defining its primary domain in business, finance, and administrative systems for companies and governments, and (2) FORTRAN, derived from the phrase "The IBM Mathematical **For**mula **Tran**slating System" was designed with features especially suited to numeric computation and scientific computing.

Our research objectives to support not just human-computer interaction but *human problem domain interaction* (Fischer and Lemke 1988) extended the architectures of Figure 2 further by introducing *domain-oriented design environments (see* Figure 3*).* Domain-oriented Design Environments are both human-centered and domain-oriented, their domain orientation is a result of embedding accumulated meta-knowledge about the task into the design environment. An example of this in a kitchen design system might be the rule that windows should not be located over kitchen ranges, or that the spatial relationship between the work centers should stay within certain ranges to facilitate smooth workflow in the kitchen. These developments were driven by the objective that if the most important role for computation is to provide people with a powerful medium for expression, then the medium should support them in working on the task, rather than require them to focus their intellectual resources on the medium itself. When users suffer from a *tool mastery burden*, their tasks fade to the background because most of their efforts are put toward mastering the tool. To bring tasks to the forefront, computers must become "invisible" (Norman 1998) and disappear into the background.

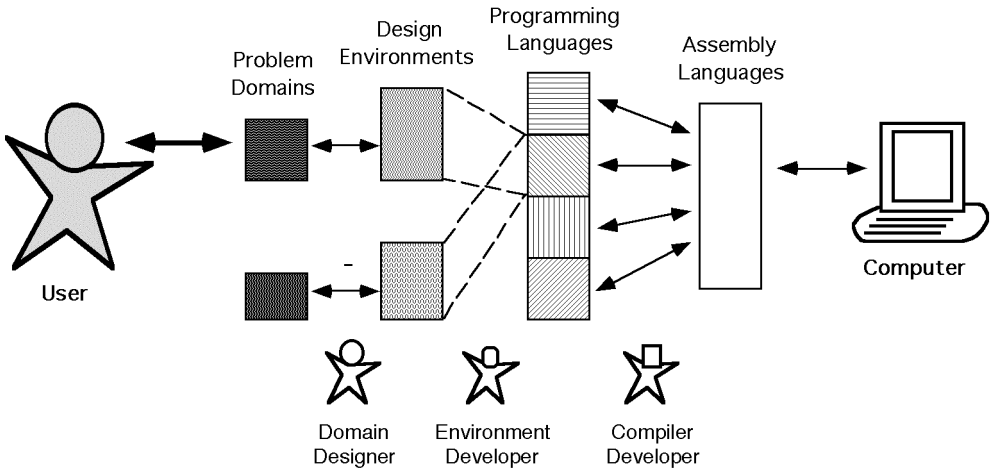


**Figure 3: Layered Architecture in Support of Human Problem-Domain Interaction**

The shift from human-computer interaction to human problem-domain interaction is a movement toward putting users, the owners of problems, in charge. As long as users must interact with the computer rather than the problem domain, they are not in charge of their problems, and must instead rely on computer specialists. In analogy to writing and its historical development, a major goal of HCI should be "to take the control of computational media out of the hands of high-tech scribes."

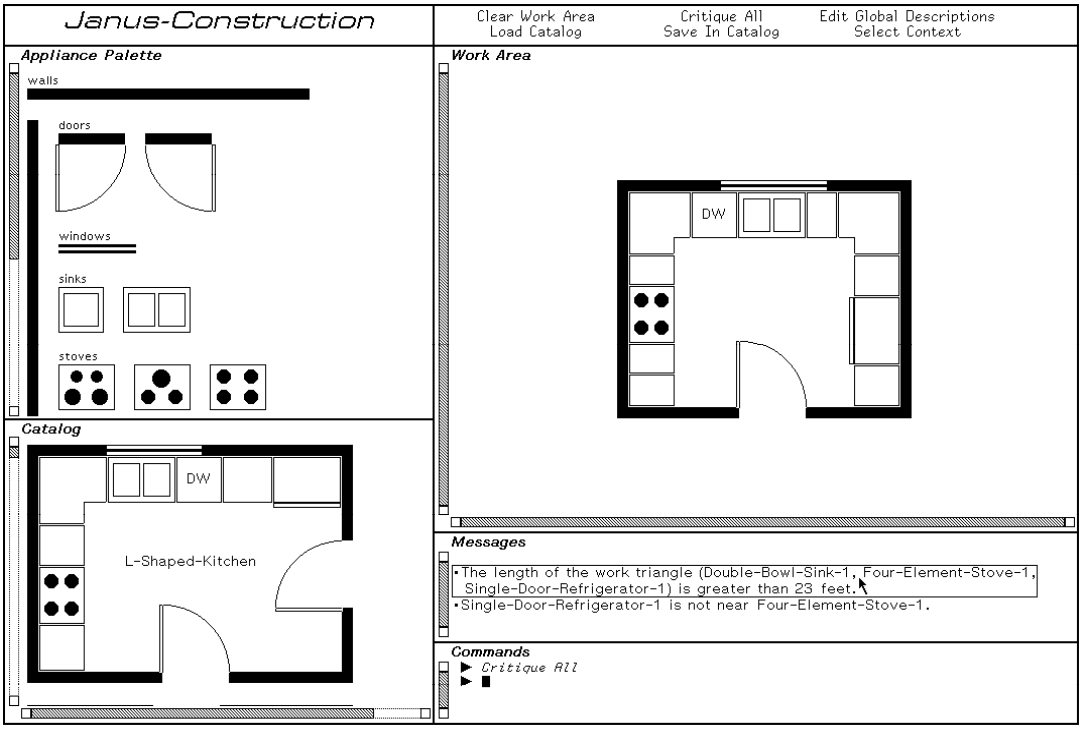


**Figure 4: A Domain-Oriented Design Environment for Kitchen Design**

Domain-orientation requires the existence of specialized features in the design environment that support the specifics of the particular domain for which it was created. Examples of specific domains that we have explored in our research are:

- kitchen design supporting professional kitchen designers (see Figure 4) and empowering them to create new design (in the "Work Area") by engaging in (a) design by composition (using domain-oriented parts from the "Appliance Palette" - see the upper left section of Figure 4 ), and (b) design by modification (modifying existing designs contained in the "Catalog" – see the lower left section of Figure 4) (Fischer 1994);
- construction and use of multimedia scripts to support task accomplishment by persons with cognitive disabilities. The MAPS system (Carmien 2007) helped caregivers provide appropriately configured instructions to young adults with cognitive disabilities.

These domain-oriented design environments provided extensive support for certain problem contexts but created the fundamental problem of how their users (being knowledge workers in the domain rather than computer programmers) could extend the systems to fit unanticipated new requirements and tasks. End-user development and end-user modification was required to find effective mixes between domain-specific support and generality (Eisenberg and Fischer 1994; Myers, Ko et al. 2006).

The next section will describe a research project in the area of "design for all", an approach to design that emphasizes usability for as many differently abled people as possible, in contrast to requiring add-on adaptations, often described as universal design; in this case to create environments to support and empower people with cognitive disabilities. The discussion will explore how new mixes between specificity and generality can be supported.

## Socio-Technical Environments for People with Cognitive Disabilities: Facing the Challenge of a "*Universe of One*"

In evaluating the impact and utility of a computationally supported tool in use, it is necessary to look at the artifact, the user(s) and the interaction between them, particular the changes caused and required in workpractice by the adoption of such systems. One way of doing this is to study the system from the perspective of socio-technical systems (Mumford 1987). Socio-technical systems have both technical and human/social aspects that are tightly bound and interconnected. Socio-technical design is an approach to complex organizational work design that recognizes the interaction between people and technology in workplaces.

People with cognitive disabilities represent a *"universe of one"* problem (Carmien and Fischer 2008). Persons with cognitive disabilities often will have several different disabilities and each specific combination of cognitive, motoric, sensory and psychological impairments together define a need for deeply customized assistive technology such that a solution for one person will rarely work for another. The "*universe of one*" conceptualization includes the empirical finding that (1) *unexpected islands of abilities* exist: clients can have unexpected skills and abilities that can be leveraged to ensure a better possibility of task accomplishment; and (2) *unexpected deficits of abilities* exist (Cole 2006). Unexpected missing abilities often occur in otherwise high functioning individuals. Accessing and addressing these unexpected variations in skills and needs, particularly with respect to creating task support, requires an intimate knowledge of the client that *only caregivers* can provide. Compounding this problem is the fact that currently a substantial portion of all assistive technology is abandoned after initial purchase and use—as high as 70 percent in some cases (Martin 1999; Reimer-Reiss 2000). This fact results in the consequence that the very population that could most benefit from technology is paying for expensive devices that end up in the back of closets after a short time.

The fundamental challenge derived from supporting the "universe of one" requirement is that it requires highly specific systems -- so how can we achieve generalization? Our answer to this challenge is a design methodology based on meta-design.

### *Meta-Design*

*Meta-design*, or "design for designers" (Fischer and Giaccardi 2006), is grounded in the basic assumption that future uses and problems cannot be completely anticipated at design time, when a system is developed (Suchman 1987). It is an emerging conceptual framework aimed at

defining and creating social and technical infrastructures in which new forms of collaborative design can take place.

The rationale to apply meta-design to address the "universe of one" was identified by analyzing the VISIONS (BAESMAN AND BAESMAN 2003) system which assisted people with cognitive disabilities by supporting task completion using a PC and touch screens located in the residence of a client with cognitive disabilities. While the VISIONS system was effective in supporting tasks which were anticipated by the designers, it was often abandoned because caregivers were not provided the tool appropriate to their limited computer programming ability to personalize the environment to the needs of their clients. The *Memory Aiding Prompting System* (MAPS) (Carmien 2006) described below eliminated these shortcomings of the VISION system by creating a socio-technical environment based on a meta-design framework by providing the caregivers the design power to modify and evolve the technical systems according to the needs of individual clients.

Meta-design extends the traditional notion of system design beyond the original development of a system to include the co-adaptive processes in which the users become co-developers or co-designers. It is a design method that can address the need of situatedness to account for changing tasks, defining when and what computer artefacts to embed in daily life and practices. This extension is supported by a metadesign distinction between metadesign tool design time and design time; there are two design processes here. First is the creation of a tool that supports metadesign activity; second, in what would more commonly called use-time, is the use of the metadesign tool to create a socio-technical environment that is more fitting to the domain of use. Co-development has the end-user ***involved*** in the ordinary design process; metadesign supports the end-user in a continuous refinement process of designing a tool. Another way to differentiate between the two is to note that one way of taking about metadesign is *design over time* (Carmien 2007) the design of a co-developed application ends at some point and use time of that application ensues; the design time of a metadesigned tool never ends.

Meta-design is the antidote to strong-specific (i.e. deeply focused to a niche in a particular domain) tools becoming unusable due to change. It tries to break new ground between generality and specificity by identifying different objectives for *design time and use time* of an environment. In all design processes, these two basic stages can be differentiated. At design time, system developers (with different levels of user involvement), develop environments and tools creating complete systems for a "world-as-imagined." This "world-as-imagined" is a result of the systems designers understanding of their system in use in specific contexts and doing specific tasks. At use time, users use the system, but because their needs, objectives, and situational contexts could only be broadly anticipated at design time, systems require modification to fit the real needs of users. To accommodate unexpected issues at use time, systems need to be underdesigned (Brand 1995) by providing a context and a background against which situated cases can be interpreted thereby allowing the "owners of problems" to create the solutions themselves at use time. By underdesigned we mean that a functional system or building is delivered, but that room is

internally provided for the user to tailor it to her precise needs. A common example of underdesigned is the German approach to rented flats. Most often a rented apartment is not provided with a finished kitchen; the designing, purchasing and installing of it is left to the tenant. While many might find this an overly onerous burden to moving in (and sometimes the old tenant sells the kitchen to the new tenant in a separate transaction not involving the landlord), it does mean that the boundary between *my* home and a *rental* home becomes blurred.

### *Memory Aiding Prompting System (MAPS)*

The *Memory Aiding Prompting System (MAPS)* (Carmien 2006) provides an environment in which caregivers can create scripts that can be used by people with cognitive disabilities ("clients") to support them in carrying out tasks that they would not be able to achieve by themselves. The challenge that MAPS addressed was the need for deep customizing of scripts. Prompting is an established technique used for both learning and performing a task by adults and older children with cognitive disabilities (Snell 1987). Independent living transition professionals teach Activities of Daily Living (ADL) performance by prompting the person with cognitive disabilities through a task by verbally instructing them through each step, either with or without instructional cards, until it has been internalized by the promtee, such that she could successfully perform the task unaided. Prompting has been historically part of instructional ADL techniques for persons with cognitive disabilities: being prompted through tasks in a rehearsal mode, and then using the memorized instructions at use time (Aist 1973; Reed 1989). Prompting consists of breaking down a task into constituent parts and creating prompts, consisting of pairs of images and verbal instructions, for each step. A prompting script is a sequential set of prompts that when followed perform a task. Special education and rehabilitation studies focus on comparing techniques and creating a principled understanding of prompting techniques with a perspective of maximizing internal recall and the *unaided* performance of the steps to complete a task by persons with cognitive disabilities (Reed 1989). With the arrival of tools like MAPS, the memorization and decision making elements of the task could be offloaded to the device and the system that supported it. Prompting studies provide a background for the study and design of computationally based prompting (Lynch 1995; Lancioni, Van den Hof et al. 1999) .

Key to the production of efficacious task scripts is the appropriate segmentation of the chosen task into subtasks of the suitable granularity (Snell 1987; Saskatchewan Learning - Special Education Unit 2003). The prompts must be scaled to the cognitive level of the user. For some users that may be as complex as 'go to the post office and get stamps" where for others 'get out two slices from the open bag of bread' may be an optimal segment size. The size of the triggered segment is dependent on the set of existing internal scripts that the prompt can trigger. The smaller the size of the internal scripts the larger the external prompting script must be. Figure 5 below illustrates the relations between internal scripts and external scripts in MAPS.

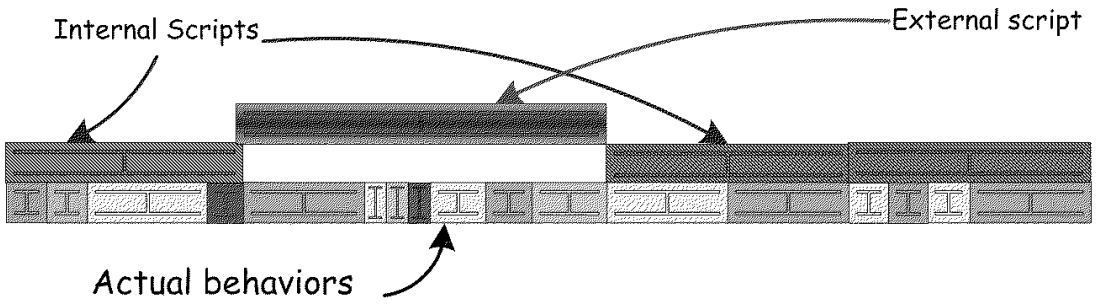


**Figure 5: Behaviors, Internal and External Scripts**

To account for the great diversity among clients, MAPS was developed as a meta-design environment, empowering the caregivers to develop personalized prompting systems for the specific needs of individual clients. It is based on a co-design of social and technical systems, and it uses models and concepts that focus not only on the artifact but exploit the social context in which the systems will be used. For instance, in the field study of MAPS adoption, one of the end-users, a teenage girl, used a script made by her mother to allow her to shop at the local supermarket by herself for the first time. In this script the steps and stages of the shopping process, including an explicit photo and voice prompt shopping list, were made by the mother, which she then put on her daughters PDA using MAPS. Near the end of the script there was a checkout section that included a photo of the check out clerk that the mother and daughter had been going to for many years. That specific photo both anchored the physical context (i.e. go to that checkout line and get ready for the steps in the checkout/payment process) and social environment (e.g. 'here is the nice lady you have known all these years who will help you, just like with mom'). Both of these social context triggers and reinforcements made a difference in the actual use of the device, reducing anxiety and directing her to the one checkout person most likely to be sensitive to the special needs of the end-user.

The two end-users of MAPS were tightly connected. Every young adult with cognitive disabilities had a caregiver, be she family or institutional. MAPS was designed from the beginning to present a script designer interface to the caregiver (who was assumed to be not more than basically skilled with PCs) and a person with cognitive disabilities who would use the system to 'play' the scripts in pursuing activities of daily living. From the perspective of the meta-design tool designer the end-user was one person in two roles, sometimes in the role of script designer (the caregiver) and sometimes in the role of a script user (the client).

MAPS consists of two major subsystems that share the same fundamental structure but present different *affordances* for the two sets of users (an affordance is an aspect of an object which makes it obvious how the object is to be used, for example buttons or scroll bars in a computer application). One of the attributes of good design is that the affordances presented to the user are intuitively useable.. **MAPS-DE** for caregivers (see Figure 6) employs web-based script and

template repositories that allow content to be created and shared by caregivers of different abilities and experiences.

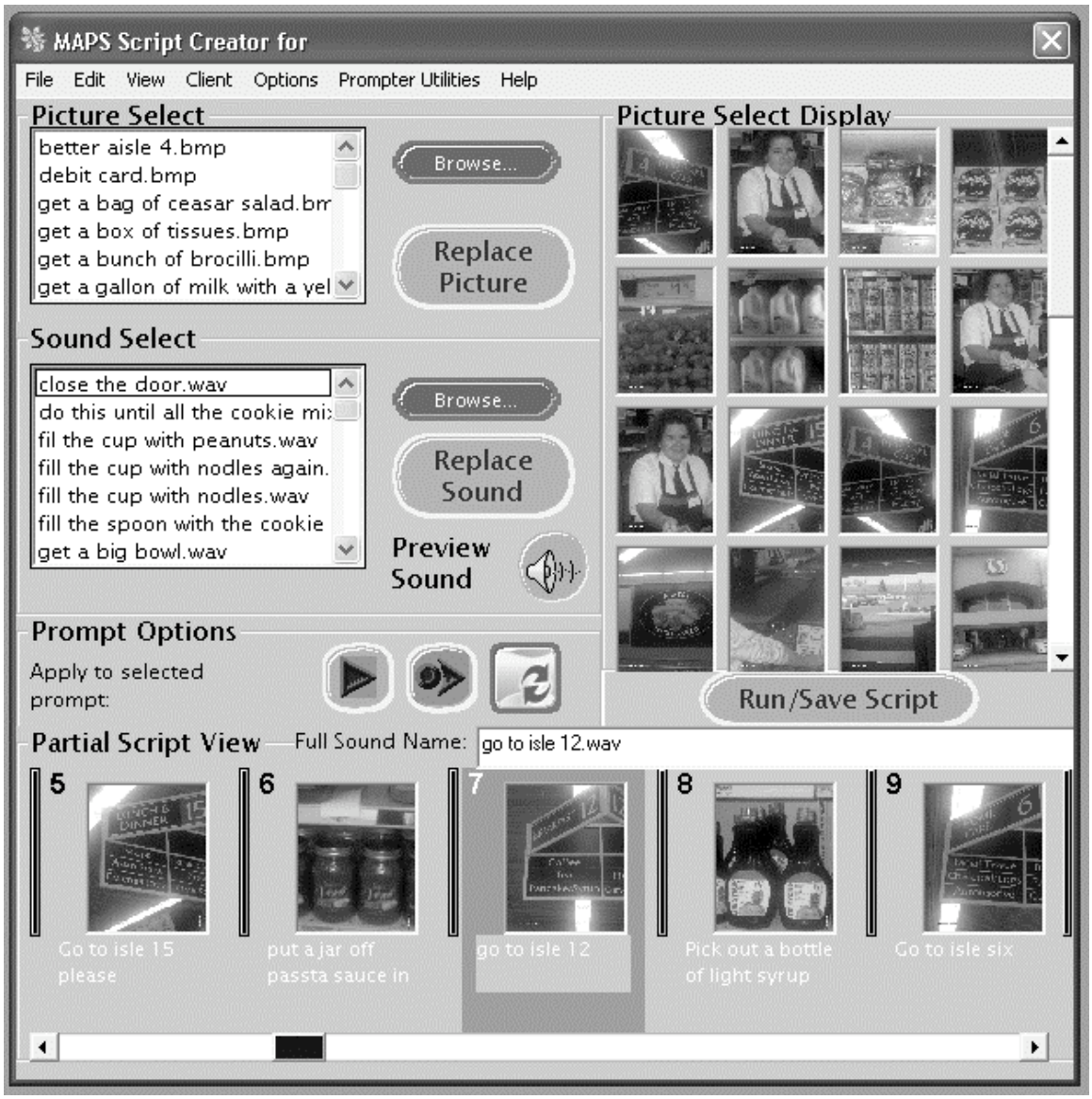


**Figure 6: MAPS-DE — a Design Environment for Creating Scripts by Caregivers**

**MAPS-PR** (see Figure 7) for clients provides external scripts that reduce the cognitive demands for the clients by changing the task from learning and remembering what to buy and how to maneuver through the supermarket, picking items, to following the instructions provided by the prompting systems.

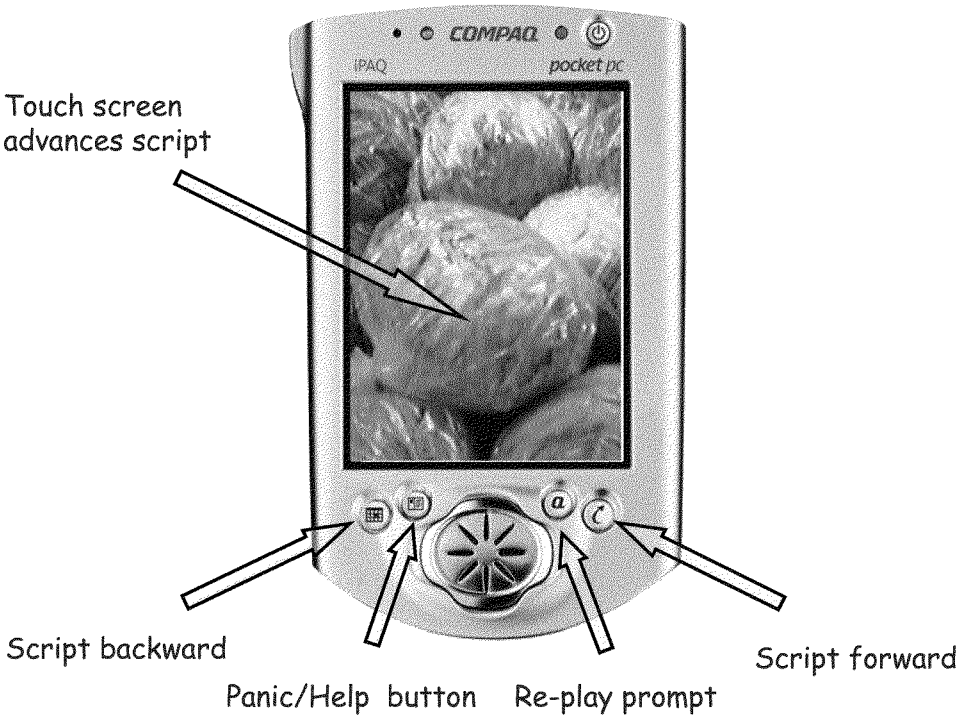


**Figure 7: MAPS-PR — a Personal Assistant for Clients**

### The Importance of Representation

Because of the needs and abilities of the end-users with cognitive disabilities, abstract representations of the components of goals such as icons are often not effective. Used as indicators to identify specific objects in the world, icons, simplified abstract visual symbols of objects, are not as accurately nor as quickly identified with their referent by persons with cognitive disabilities, even those icons that are specifically designed for this purpose. In a test with typical young adults and with young adults with cognitive disabilities, use of actual photos enabled quicker and more accurate identification of the target for those subjects with significant cognitive impairments (Carmien 2008). So in designing the MAPS system, the specifications called for support for photos to act as one part of the prompt and specific verbiage recorded by specific voices as the other part. MAPS evaluation collected several illuminating examples of this. In one instance a prompt for a teenage girl was not recorded by her mom, because of typical teenage power issues. In another a young adult woman used the prompter system to support employment in a used clothing store as part of the process of transitioning from high school to employment; to support this transition the state provided her with a certain amount of assisted employment. Assisted employment meant that she would have a job coach with her for the first several months of employment, guiding her through internalizing the scripts of the tasks that constituted her prospective job. At the end of her job training with the coach she expressed

anxiety on 'graduating' to unsupported employment; she was told by her coach that if she needed a hand to hold, her (the job coach's) voice would be with her in the form of the recordings of the prompts. As a result, when supporting caregivers act as multimedia programmers they had to take into account, on one hand, the need for arbitrary images and sounds to be used; and on the other hand, the need to produce scripts in a particular format.

The need for adaptation of the scripts over time as the young person with cognitive disabilities using the MAPS-PR prompter came from the situated use of the scripts. In some cases internalized sub-tasks or the discovery that, for *this* person, certain sub-tasks needed more support, provided a functional requirement that could be satisfied by a metadesign approach. From the perspective of the meta-design tool designer the end-user was a dyad, both generating and consuming scripts. Mediating between these two roles is feedback (or system backtalk) in the form of information about how the script was used in the specific task. This backtalk came in the form of observations by the caregiver and also a log of script use that the MAPS-PR prompter provided at the end of each session. By supplying a tool to the caregivers that supported both script modification (expanding or contracting task support scaffolding) as well as support script re-use among these user dyads composed of caregiver/person with cognitive disabilities, a correct balance between the two tarpits could be navigated. The underlying architecture of MAPS provided local and networked databases of scripts that could be modified to suit current needs and saved so that the original script was still available for use as a template of successful scripts by other dyads (after appropriate anonymisation) or to regress to earlier versions of the same script if needed.

A unique challenge of meta-design in the domain of cognitive disabilities is that the clients themselves cannot act as designers, but the caregivers must accept this role. Figure 8 illustrates the entire process, from metadesign tool design to its use in the world, with the caregiver creating scripts, the scripts use by the person with cognitive disabilities and then the revision and updating of the scripts on the basis of the script's match to the person/task/environment in use.

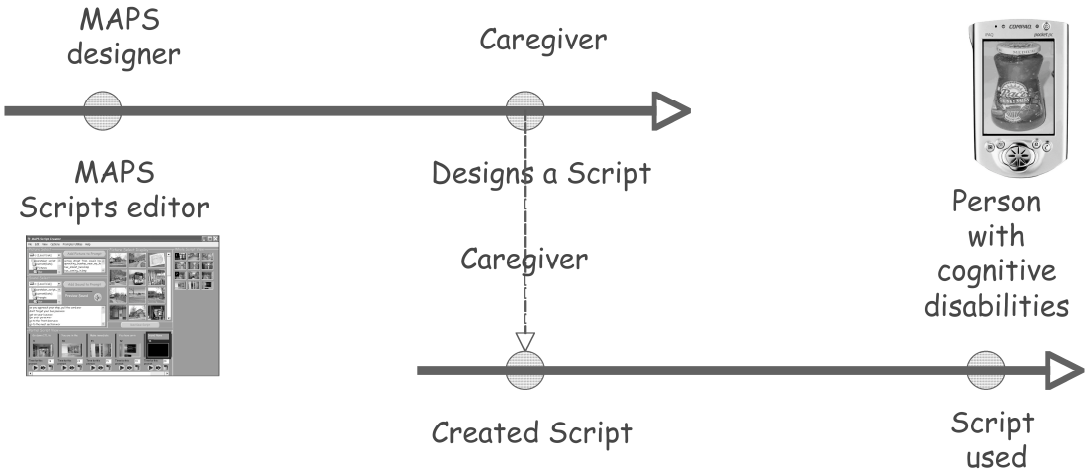


**Figure 8: Empowering Caregivers to Act as Designers**

### *Potential Pitfalls and Shortcomings of Domain-Orientation*

Opponents and critics of domain-oriented approaches have argued that the following problems exist with domain-oriented design approaches:

> (1) *The lack of a profound theory of what domains are will restrict the development of domain-oriented design environments on ad-hoc intuitions about what constitutes a domain.*

We argue that domains are not fixed, given entities that should be regarded as well-defined and unchanging(Rittel and Webber 1984). It is one of the foundations of our approach to understand domains as being "constructed" by a design community as opposed to existing in some objective, super-general form. This has set our work apart from other approaches such as domain analysis and modeling (Sutcliffe 2004) and has provided the foundation for a process model (called the seeding, evolutionary growth, reseeding (SER) model (Fischer and Ostwald 2002)) of how domains and the *domain-oriented design environments* supporting them should evolve over time.

> (2) *Domain specificity poses a major limitation for design productivity; every domain will have to have its own purpose-built environment.*

As indicated above, we are aware of the tension and the design trade-off between *the Turing Tar Pit* and the inverse of it. Referring back to human organizations and domain expertise: our society educates its members in domains, and switching from one domain to another is a non-trivial undertaking. So why should we expect that we will get a new domain-oriented design environment in a different domain for free?

Making a tool domain specific does not necessarily mean shackling its flexibility in that domain. To a large extent the designer can mitigate the error of forcing the user to solve every problem in the domain in the same way, as Tufte pointed out about PowerPoint users becoming limited in their expressivity (Tufte 2003) . One way to avoid this is to carefully choose the metaphor that the tool is based upon; in the kitchen design environment that metaphor was an architects desk and stencil templates, in the case of MAPS it was a sequential filmstrip. In MAPS case this was the result of a summer spent making various lo-fi mockups of possible design metaphors and 'playing' scenarios with them, a sort of supported cognitive walkthrough (Lewis and Rieman 1993) . While the filmstrip metaphor was successful, for one of the dyads a further elaboration was needed. For the dyad of the job coach and young woman in assisted employment referred to above, the job coach/ student employee pair needed the ability to fork and loop over sets of scripts in order to accommodate the need for 'bridging' tasks to give the client support for 'soft skills'. Soft skills are the auxiliary, human politeness behaviors that young adults with cognitive disabilities often lack; it is soft skills that more often cause job termination than simple poor job performance. The job coach needed to provide the opportunity for the client to select one of three tasks at use time. The MAPS-PR system was modified to successfully support this (see Figure 9), however it was clear that the lack of time spent in designing the underlying metaphor led to a

design that was more constrained (to just this set of tasks) than the more general original MAPS design.


**Figure 9: MAPS-PR with the optional 3-script screen interface**

No one with any concerns for productivity would limit her/his learning efforts and work products to a specific domain, if they could be applied more generically. To address this issue, in addition to meta-design we have developed a general-purpose software architecture that can be instantiated for different application domains. This architecture, called the *multifaceted architecture (Fischer 1994)* provides support to avoid rebuilding domain-oriented design environments from scratch for a new domain, while maintaining the domain-specific orientation and support. Without paying the price of working in a domain, computational environments will be severely limited in the amount of support they can provide, specifically to capture the interest of end-users and providing them with control over their tools.

Human–centered design is focused on making the human the focus in the design of technology. While technology has grown exponentially, internal human capabilities (without taking advantage of external supporting socio-technical environments) have not fundamentally changed over time (our neurons do not fire faster and our memory has not increased in capacity). The fundamental challenge to improve human learning, working, and collaboration requires external environments that complement more effective human abilities and knowledge in specific tasks. There are interesting trade-offs between generic systems (providing breath and generality in the form of super-appliances such as Swiss Army Knives) and very specific domain-oriented tools (such as the elaborate tool sets used by mechanics) (Buxton 2002).

The next section will discuss this design tradeoff between generality (breath) and strong specific (depth) systems in the context of creating innovative educational environments in which the minds of tomorrow are educated.

## Educational Implications: How do we educate the "Renaissance Scholar" of the 21st Century?

*"The smartest people in the world do not generally look very intelligent when you give them a problem that is outside the domain of their vast experience."* — Herbert Simon

The design principles that we have explored in the preceding sections about computational environments are equally relevant for social design as applied to education and collaboration in groups and communities. There is an ongoing debate in education about how to cope with the design trade-off between *discipline specialization (depth)* versus *general knowledge (breath)* in a world in which there are too many things to learn and to know. For example: given the constraints on human ability how can we expect an individual in the context of developing a socio-technical environment such as MAPS to maintain the requisite specialist knowledge in their technological discipline, while at the same time have the needed competence and understanding about people with cognitive disabilities?

The "Renaissance Scholar" is not a reasonable model for the 21st century because *"nobody knows who the last Renaissance man really was, but sometime after Leonardo da Vinci it became impossible to learn enough about all the arts and the sciences to be an expert in more than a small fraction of them"* (Csikszentmihalyi 1996). Simon (Simon 1996) argued that when a domain reaches a point at which the knowledge for skillful professional practice cannot be acquired in a decade, specialization increases (Kaufman and Baer 2004), collaboration becomes a necessity, and practitioners make increasing use of media supporting distributed intelligence (Salomon 1993; Hollan, Hutchins et al. 2001). One of the potential answers to these challenges is (as argued before) to educate people to become members of *reflective communities* (Fischer 2005) rather than acting only as reflective practitioner (Schön 1983).

The *fish-scale model* (Figure 10) (Campbell 1969) is a qualitative model illustrating an interesting structure of reflective communities: it tries to achieve *"collective comprehensiveness through overlapping patterns of unique narrowness."* The model depicts a competence that cannot be embodied in a single mind. The inevitably incomplete competence of an individual (sometimes referred to as *"symmetry of ignorance"* (Rittel 1984; Fischer 2000)) requires reflective communities in which there is the right mixture between sufficient overlap and complementary competence. The fish-scale model provides a viable path toward a new design competence, based on the integration of individual and social creativity (Fischer, Giaccardi et al. 2005).

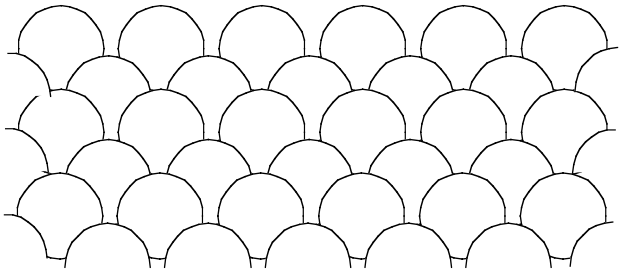

**Figure 10: The Fish-Scale Model**

This abstract model can be contextualized to the application of creating computational environments for people with cognitive disabilities and in this way we can create a team in which depth and breath are achieved simultaneously (CLever 2004). In this model, teams consist of software professionals and domain experts who are engaged in an interdisciplinary collaboration (left pane of Figure 11). In CLever's case the team included (1) special education teachers working with people with cognitive disabilities and (2) software professionals developing environments such as MAPS. To make this collaboration productive and effective, the specialists acquired some knowledge in the other disciplines that created a common language and a shared understanding (right pane of Figure 11).

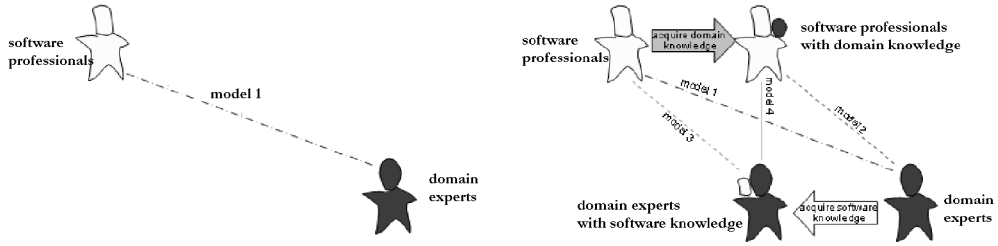


**Figure 11: Interdisciplinary Collaboration and Reflective Communities**

## Conclusions

As cultures evolve, specialized knowledge will be favored over generalized knowledge (Csikszentmihalyi 1996; Simon 1996) and specialized artifacts and technologies will come to the forefront of common usage (Basalla 1988; Buxton 2001). Will this cultural fragmentation lead to a new "Tower of Babel" and how can new frameworks and approaches be developed which overcome some of these limitations? The conceptual frameworks presented in this paper (domain-oriented design environments, meta-design, reflective communities) explore new synergistic efforts to support development of specialized and generalized knowledge. In doing so, we worked under the basic assumption that there are no decontextualized "sweetspots" but only *design-trade-offs* (1) between specific environments well suited to a task and general environments applicable to a large set of tasks, and (2) between breath-first and depth-first education.

## References


Aist, E. H. (1973). The Effect of Two Patterns of Visual Prompting on Learner Achievement in Industrial Arts, Arizona State University.

Baesman, B. and N. Baesman (2003). Visions System website. **2003**.

Basalla, G. (1988). The Evolution of Technology. New York, Cambridge University Press.

Brand, S. (1995). How Buildings Learn: What Happens After They're Built. New York, Penguin Books.

Buxton, W. (2001). Less is More (More or Less). The Invisible Future — the seamless integration of technology in everyday life. P. J. Denning. New York, McGraw-Hill**:** 145-179.

Buxton, W. (2002). Less is More (More or Less). The Invisible Future - the seamless integration of technology in everyday life. P. J. Denning. New York, McGraw-Hill**:** 145-179.

Campbell, D. T. (1969). Ethnocentrism of Disciplines and the Fish-Scale Model of Omniscience. Interdisciplinary Relationships in the Social Sciences. M. Sherif and C. W. Sherif. Chicago, Aldine Publishing Company**:** 328-348.

Carmien, S. (2006). Socio-Technical Environments Supporting Distributed Cognition for Persons with Cognitive Disabilities. Department of Computer Science. Boulder, Colorado, University of Colorado at Boulder (Available at: http://l3d.cs.colorado.edu/~carmien/).

Carmien, S. (2007). Leveraging Skills into Independent Living- Distributed Cognition and Cognitive Disability. Saarbrücken, VDM Verlag Dr. Mueller e.K.

Carmien, S. and G. Fischer (2008). Design, Adoption, and Assessment of a Socio-Technical Environment Supporting Independence for Persons with Cognitive Disabilities. Proceedings of CHI 2008: ACM Conference on Human Factors in Computing Systems. New York, NY, USA, ACM Press**:** (in press).

Carmien, S., Wohldmann, E. (2008). "Mapping Images to Objects by Young Adults with Cognitive Disabilities." *Research In Developmental Disabilities* **29**: 149-157.

CLever (2004). CLever: Cognitive Levers -- Helping People Help Themselves, L3D Center.

Cole, E. (2006). Patient-Centered Design as a Research Strategy for Cognitive Prosthetics: Lessons Learned from Working with Patients and Clinicians for 2 Decades, Montreal, Canada.

Csikszentmihalyi, M. (1996). Creativity — Flow and the Psychology of Discovery and Invention. New York, NY, HarperCollins Publishers.

Eisenberg, M. and G. Fischer (1994). Programmable Design Environments: Integrating End-User Programming with Domain-Oriented Assistance. Human Factors in Computing Systems, CHI'94 (Boston, MA). New York, ACM**:** 431-437.

Fischer, G. (1994). "Domain-Oriented Design Environments." Automated Software Engineering **1**(2): 177-203.

Fischer, G. (2000). "Social Creativity, Symmetry of Ignorance and Meta-Design." Knowledge-Based Systems Journal (Special Issue on Creativity & Cognition), Elsevier Science B.V., Oxford, UK **13**(7-8): 527-537.

Fischer, G. (2005). From Reflective Practitioners to Reflective Communities, Proceedings of the HCI International Conference (HCII), Las Vegas, July 2005 (published on CD).

Fischer, G. and E. Giaccardi (2006). Meta-Design: A Framework for the Future of End User Development. End User Development: Empowering People to Flexibly Employ Advanced Information and Communication Technology. H. Lieberman, F. Paternò and V. Wulf. Dordrecht, The Netherlands, Kluwer Academic Publishers**:** 427-457.

Fischer, G., E. Giaccardi, et al. (2005). "Beyond Binary Choices: Integrating Individual and Social Creativity." International Journal of Human-Computer Studies (IJHCS) Special Issue on Creativity (eds: L. Candy and E. Edmond): (in press).

Fischer, G. and A. C. Lemke (1988). "Construction Kits and Design Environments: Steps Toward Human Problem-Domain Communication." Human-Computer Interaction **3**(3): 179-222.

Fischer, G. and J. Ostwald (2002). Seeding, Evolutionary Growth, and Reseeding: Enriching Participatory Design with Informed Participation. Proceedings of the Participatory Design Conference (PDC'02), Malmö University, Sweden, CPSR.

Hollan, J., E. Hutchins, et al. (2001). Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research. Human-Computer Interaction in the New Millennium. J. M. Carroll. New York, ACM Press**:** 75-94.

Kaufman, J. C. and J. Baer (2004). Hawking's Haiku, Madonna's Math: Why It is Hard to be Creative in Every Room of the House. Creativity: From Potential to Realization. R. J. Sternberg, E. L. Grigorenko and J. L. Singer. Washington, DC, American Pyschological Association**:** 3-19.

Lancioni, G., E. Van den Hof, et al. (1999). "Evaluation of a Computer-aided System Providing Pictorial Task Instructions and Prompts to People with Severe Intellectual Disability." Journal of Intellectual Disability Research **43**(1): 61-66.

Levy, F. and R. J. Murnane (2004). The New Division of Labor: How Computers are Creating the Next Job Market. Princeton, Princeton University Press.

Lewis, C. and J. Rieman (1993). Task-Centered User Interface Design: A Practical Introduction. Boulder, Colorado, University of Colorado, Boulder.

Luhmann, N., J. Bednarz, et al. (1996). Social Systems, Stanford University Press.

Lynch, W. (1995). "You Must Remember This: Assistive Devices for Memory Impairment." Journal of Head Trauma Rehabilitation **10**(1): 94-97.

Martin, B., and McCormack, L. (1999). Issues surrounding Assistive Technology use and abandonment in an emerging technological culture.

Mumford, E. (1987). Sociotechnical Systems Design: Evolving Theory and Practice. Computers and Democracy. G. Bjerknes, P. Ehn and M. Kyng. Aldershot, UK, Avebury**:** 59-76.

Myers, B. A., A. J. Ko, et al. (2006). Invited Research Overview: End-User Programming. Human Factors in Computing Systems, CHI'2006 (Montreal)**:** 75-80.

Norman, D. A. (1993). Things That Make Us Smart. Reading, MA, Addison-Wesley Publishing Company.

Norman, D. A. (1998). The Invisible Computer. Cambridge, MA, The MIT Press.

Perlis, A. J. (1982). Epigrams on Programming. SIGPLAN Notices**:** 7-13.

Reed, R. W. (1989). An Investigation of Two Prompting/Fading Procedures to Teach Independent Dire Evacuation Behaviors to Individuals with Severe/Profound Mental Retardation, University of New Orelans.

Reimer-Reiss, M. (2000). Assistive Technology Discontinuance. Technology and Persons with Disabilities Conference.

Rittel, H. (1984). Second-Generation Design Methods. Developments in Design Methodology. N. Cross. New York, John Wiley & Sons**:** 317-327.

Rittel, H. and M. M. Webber (1984). Planning Problems are Wicked Problems. Developments in Design Methodology. N. Cross. New York, John Wiley & Sons**:** 135-144.

Salomon, G., Ed. (1993). Distributed Cognitions: Psychological and Educational Considerations. Cambridge, United Kingdom, Cambridge University Press.

Saskatchewan Learning - Special Education Unit (2003). Task Analysis.

Schön, D. A. (1983). The Reflective Practitioner: How Professionals Think in Action. New York, Basic Books.

Simon, H. A. (1996). The Sciences of the Artificial. Cambridge, MA, The MIT Press.

Snell, M. E. (1987). Systematic Instruction of Persons with Severe Handicaps. Columbus, Ohio, Merrill Publishing Company.

Suchman, L. A. (1987). Plans and Situated Actions. Cambridge, UK, Cambridge University Press.

Sutcliffe, A. G. (2004). The Domain Theory: patterns for knowledge and software reuse. Hillsdale NJ (in press), Lawrence Erlbaum Associates.

Tufte, E. (2003). The Cognitive Style of Power Point, Graphics Pr.