0068

Regarding Software as a Material of Design

E. Blevis[1], Y.K.Lim[1] E. Stolterman [1]
[1]Indiana University at Bloomington, Bloomington Indiana, United States
eblevis@indiana.edu

The Issue

Design as a discipline has a long history with origins in art, architecture, and industrial design. It has a pragmatic experience in products, environments, and artifice that is informed by theories that range from engineering and cognitive psychology to philosophy to aesthetics to marketing and business. Whereas it is possible as hypothesis to regard software design as a distinct design discipline from other forms of design, it is also possible as hypothesis to think of software as a material of design just in the same way that other materials are used in design. The former hypothesis is common. The latter one is less common. The notion of regarding software as a material of design owes to the purposefully rhetorical notion of digital artifacts as being composed from *materials without qualities* as described in Löwgren & Stolterman's recent book, "Thoughtful Interaction Design" (2004). By *materials without qualities*, Löwgren & Stolterman mean simply that even though digital artifacts do not have physical qualities in the same sense as fabric and concrete, they may still be regarded as material. Of course there are many ways to think about software—as a discipline of expertise, as a kind of product, as a material of products. Our goal here is to try to understand how best to integrate software design with other design disciplines and the specific technique of this paper is to examine how regarding software as a material of design can help inform this understanding, other ways of thinking about software notwithstanding.

Regarding software as a material of design in some ways alike to and in some ways distinct from other materials—comparable in the same way that any materials may be compared—creates an opportunity to bring the wealth of design experience in other design practices and theoretical understandings to the practice and theory of software design. Such a hypothesis invites synthesis of design practice and theory with the practice and theory of software design and engineering—our goal is to make it easier for software and interaction designers to understand and incorporate the best practices of design in general.

As a metaphoric material, software can be understood as creating new design possibilities when it itself is novel. More importantly, it can be understood as having an effect on environmental coherence when it is

treated as a fabric of artifice. The pursuit of invention that drives much of the computer sciences in the sense of creating new materials of design is important. Also important in design is invention in the sense of the use and arrangement of materials to create environments that better the human condition, or even the condition of the biosphere if you prefer.

For something to be designed well, it needs to have been designed in consideration of more than mathematical integrity, cognitive models of "users", or usability—it needs to have been designed in consideration of contexts, environments, inter-relations, markets, emotions, aesthetics, visual forms, semiotic references and a whole host of considerations that are part of the assumed nature of successful designs. It needs to be construed as part of a dialogue between product, anti-product (i.e. reclaiming old things as new), and lifestyle and notions of ecology and futures.

In several texts, Donald Norman has pioneered the art of describing why things sometimes are not designed well from the perspective of people who need to use them (1990; 1998; 1999). The literature on human-computer interaction (HCI) contains many examples of individual artifacts—formed from software or otherwise—that fail in the context of usability, or even affect or experience. A very early description of this kind of failure in context owes to Wingrad and Flores' notion of *breakdown* in their text, "Understanding Computers and Cognition: A New Foundation for Design" (1986). The notion of breakdown is perhaps a larger notion in that it describes the unintended consequences of complex interactions between various aspects and artifice in the world. In what follows, we give our own personal example in the style of Donald Norman intended to illustrate the notion of breakdown that results from failure to understand software as material.

An example that illustrates the conception of software as a material of design

As an example, one of the authors recently acquired a new PDA—a "smartphone"—which also includes a phone, a camera, and a GPS navigation receiver. Setting up, understanding use, and installing GPS software on the phone took two full days—imagine how long it takes if you don't have a Ph.D. in computer science! One day, the smartphone seemed to have stopped working as a phone. The phone section of the smartphone would ring, but once answered the author could neither hear nor be heard by the other party. The software indicated in its call log that phone calls were being received correctly. An hour of analysis with the smartphone manufacturer's technical support yielded a verdict that the smartphone was defective and needed to be replaced. Further tinkering by the author revealed that dust had gotten into the headset jack receptacle. This occurred because this new smartphone doesn't have a cover over the headset jack receptacle. The dust made the phone section of the smartphone sense that it had a headset attached and the software turned off the speaker and microphone native to the smartphone. Blowing into the headset jack receptacle restored the smartphone to working order.

Usability, models of cognition, understanding software in isolation from the context of other product elements cannot help us avoid this kind of *breakdown*. From the user's point of view, it matters little or perhaps not at all which part of the product is software and which is hardware and which is form and which is design—it matters only that the product has stopped working and its complexity of interactions makes it very difficult for

the manufacturer's trained technicians to adequately diagnose problems. The design flaw is the absence of a cover over the headset jack receptacle. From the user's point of view, the software that acted according to an erroneous sensor is not inherently different from the other materials of the PDA. An architect takes responsibility for an entire artifice, including the choice of materials and their coherence one-to-another. A car designer who provides software to control a breaking system that fails because it fails to detect that the car is moving faces serious liability exposure. There is no reason to believe that software designers in general should take less responsibility.

We believe that one way to avoid this kind of breakdown is to regard software as a material of design and that thinking of software in this sense allows software design to be just like other kinds of design. The privileged status that some technologists might ascribe to software over other aspects of the artificial is rightly lost on ordinary people who use products designed with the material of software.

It remains to show how to operationalize this notion of design with the material of software. To do so, we first need to recognize that there are many different and distinct disciplines involved in design that can benefit from the notion of regarding software as a material of design. Next, we need to show how collaboration between these distinct backgrounds and cultures can be supported. We develop our understanding of how such collaborations can be supported along two dimensions, one relating to a notion of design philosophy which integrates various viewpoints—namely, *values, methods*, and *reasoning* (VMR)—and one relating to a notion of design practice which also integrates various viewpoints—namely, *mind sets*, *knowledge sets, skill sets*, and *tool sets* (MKST). For each of these dimensions we provide frameworks that design teams can use as instruments of collaboration, allowing individuals and sub-teams within teams to express their own understanding of design as philosophical and practical profiles and to foster productive collaborations by sharing each others' profiles. We call this system involving philosophical and practical profiles the *SoftMat* paradigm, denoting "Software as Material".

The SoftMat Paradigm—Philosophical and Practical Basis

The SoftMat paradigm is ongoing research. In this paper, we give a detailed hypothesis informed by our design philosophy and experience. We describe how we propose to iteratively develop our hypothesis further using empirical and philosophical means.

We explain how we are constructing the SoftMat paradigm in what follows. Figure 1 diagrams the relations between the various elements of our research.

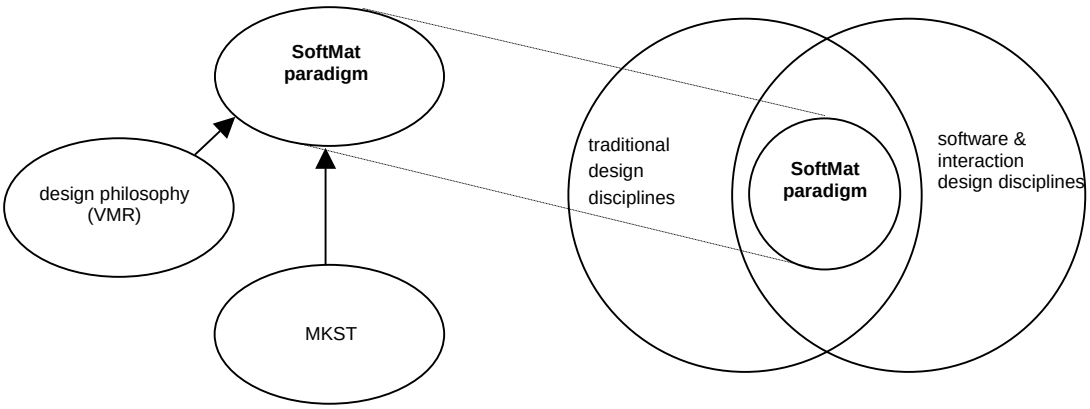

Figure 1. The SoftMat paradigm and its foundations

Figure 1 shows that the SoftMat paradigm is intended as a tool for integrating software and interaction design disciplines and traditional design disciplines. That the SoftMat paradigm as illustrated in Figure 1 is properly contained within the intersection of the circles denoting traditional design disciplines and software and interaction design disciplines is intended to illustrate that the SoftMat paradigm is only one instrument for creating this integration—there are certainly others. The SoftMat paradigm is our original thinking about an approach to this integration which is informed by our collective years of practice and reflection in software design, and design in general. On the left side of Figure 1, the SoftMat paradigm is illustrated in a manner which shows that it is informed by and composed of a philosophical component which we call VMR and a practical component which we call MKST.

Our design philosophy is based on a triumvirate of notions relating to design, namely *values, methods,* and *reasoning*. We believe that few designers consciously operate in recognition of all three notions and that conceptualizing design in all three of these terms serves as a platform for understanding the differences in emphasis between one group of designers and another. Moreover, we believe that encouraging designers to operate in recognition of all three levels can improve the reflective practice of any designer.

The SoftMat paradigm is intended to be useful as an instrument for design collaboration, since it characterizes the differences in understanding design among different groups of people. The intent is that bringing clarity to such differences can provide the key to effective inter-disciplinary collaborations based on mutual understanding, rather than the tensions that accrue from a multi-disciplinary morass.

Philosophical Basis for the SoftMat Paradigm: The VMR Framework

As a thematic, theoretical foundation for understanding design, we distinguish between *what design is about* from *what design is* from *what designs are*. Discourse about *what design is about* informs a choice of value systems that serve as notions of appropriateness for design. Discourse about *what design is* informs notions of the utility and

intrinsic limitations of various methods, including design collaboration and design processes. Discourse about *what designs are* informs a vision for representations and interpretations of design knowledge as objects of reasoning—as a large-scale activity which gives form to design knowledge so that it can be shared and serve as a basis for design collaboration.

We have called the philosophical framework that supports the SoftMat paradigm *VMR* for *values*, *methods*, and *reasoning*. The terms *values* and *methods* are reasonably straightforward. The term *reasoning* is much less universal. In a formal sense, by *reasoning*, we mean the act of constructing representations and interpretations. In a less formal sense, by *reasoning*, we mean the construction of any prototype or sketch or explanation that is used to denote a design or promote creative discussion about a design. It is important to note that we are using the word *reasoning* to denote all of these constituents, namely *representations, interpretations, prototypes, sketches*, and *explanations*.

Our approach is informed by a long collective experience in the world of design and an equal experience in incorporating our knowledge of design communities into our work in informatics—HCI in particular, as well as substantial backgrounds in interaction design, both in scholarly and enterprise venues. We characterize our approach as a design philosophy, which itself is comprised of theoretical elements informed by our practice and analysis. Since our experience spans from the pragmatics of design and development of software through the design-oriented theoretical understanding of software-intensive and software-embedded systems as materials of design, we have constructed and continue to refine a design philosophy which can serve to explain the sensibilities of design. By such sensibilities, we mean understanding design as inextricably tied to the contexts of human condition, and even the biosphere. Furthermore, we seek to operationalize such sensibilities in a pragmatic way that informs and ameliorates the design and development of software in a way that creates coherent improvements for the human condition and biosphere.

Values: What design is about

By *values*, we mean notions of *what design is about*. A cursory survey of popular media convinces us that many people in the popular culture think of design as decoration as illustrated in Figure 2—MTV's "Pimp My Ride", BBC's "What Not to Wear" and "Changing Rooms" are examples from the world of television. Of course, almost no one who considers her or himself to be a professional designer thinks of design as decoration. Certainly, software designers, industrial designers, communication designers, and architects do not think of design as decoration. We created Figure 2 as a caricature to illustrate this disconnect between the notions of what design is about among professional designers and the notions of what design is about in the general public.

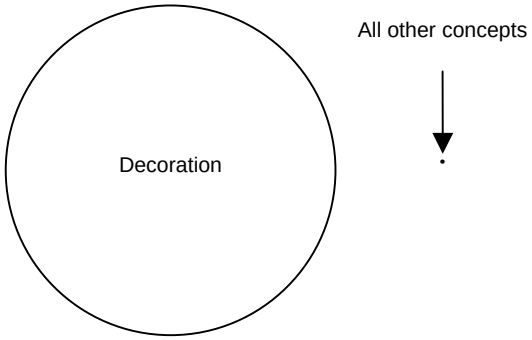

Figure 2. A caricature of the popular conception of design and all other concepts

In informatics in general, the notion of what design is about may vary widely. Some notions of design in informatics, such as human-centered design may in some interpretations be deeply integrated with notions of values and ethics, see Kling & Star (1998) for example. Some notions of design are more process-oriented, such as object-oriented design or models of design process like waterfall, star, rapid applications development, or spiral models and as such are possibly abstract from or apparently neutral in terms of values and ethics. See Preece, Rogers, & Sharp (002) for an inventory of software design models of process.

In design schools or firms, the notion of design as decoration is likely to be construed as actually antithetical to a thoughtful notion of design—in these venues, one is more likely to think of design as *being about endowing form with meaning* at the least. Endowing form with meaning is a core competence of designers, but it is not the most profound notion of design. One hopes for a way to get past an obsession with objects of form and to construct a deeper understanding of the nature of design as an agency in the world.

In Figure 3, we give a diagram to try to illustrate a possible progression of understanding of what design is about, from values-neutral notions of objects and features to values-critical notions of ecologies and futures:

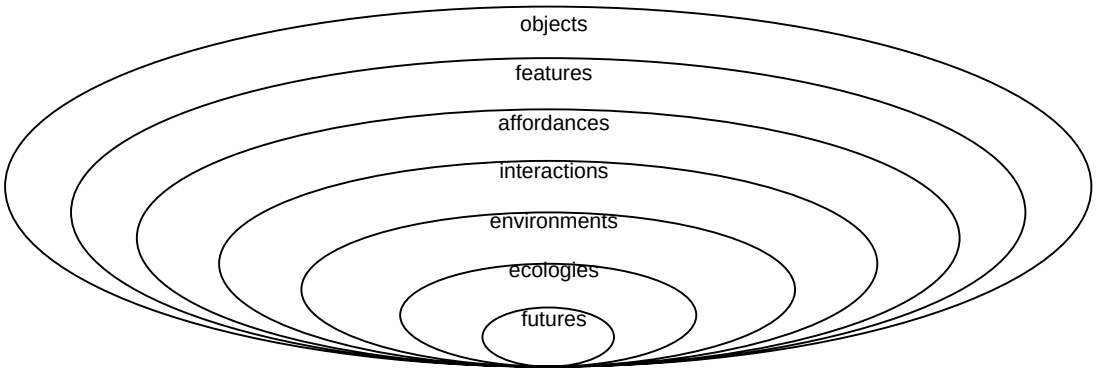

Figure 3. Expanded view of all other concepts: Design values as notions of what design is about

It is important to note that the layers of Figure 3 denoting levels of understanding what design is about are not mutually exclusive and furthermore, nearly any design enterprise can be motivated by many of these levels of understanding. Moreover, it would be entirely possible to name other layers. Each of these layers has many references in various literatures which we will presently sample in this section.

The understanding that some objects are more "designerly" than some others is a starting point for the discourse implied by this diagram. From this point, we may move through a progression of ever more thoughtful notions of design being in (i) the features of objects—an engineering and technology-centered view, (ii) the affordances of objects—visual cues of form that reveal the underlying operational semantics of objects, and elements of visibility of form that predict usability or affect, (iii) the interactions between people and objects—the pervasive utterance in design circles that the design is in "verbs", (iv) whole environments—the notion of design as intervention in an environment, (v) whole ecologies—the notion of design as a balanced, systemic organization, and finally (vi) futures—issues of sustainability and the idea that design is a choice among future ways of being.

The view that design is about objects or features of objects pervades the methodological literature on interaction design and software design in informatics. For example, see Fitzgerald, Russo, & Stolterman (2002) which describes in detail exactly this observation. The notion of a progression from user needs and requirements to specifications in terms of objects and features is understandably targeted at achieving a natural, values-neutral approach to design in an engineering-centered context. Such a view of design serves well to create notions of modularity that allow for scale in the construction of computer systems.

As a refinement of a notion due to Gibson (1977), Donald Norman (1990) adapted the word "affordances" to the context of human-computer interaction, in a manner not dissimilar to how we have defined it above. Norman's refinement includes the idea of affordances as being perceptual in nature. The idea that design is at least in part about affordances—about visibility as a predictor of usability—yields an understanding of what design is about that corresponds in some deep sense to designerly notions of design being about endowing form with meaning—meanings that facilitate and clarify the interactions between people and objects. Once we have moved away from notions of objects and features to notions of affordances as catalysts to interactivity, we have created fertile ground for discourse on values and ethics. Norman (1999) has since complained about the ubiquity of the term affordance which greatly reduces the specific meanings he intended.

Herbert Simon's (1996, p.111) notion of design as *"everyone designs who devises courses of action aimed at changing existing situations into preferred ones"* is perhaps the single-most quoted definition in informatics circles. Although Simon (1996, p.113) intended this definition to motivate a notion of a science of design, as *"a body of intellectually tough, analytic, partly formalizable, partly empirical, teachable doctrine about the design process"*, it is the notion of what it means for a situation to be "preferred" that concerns us here, as such a notion of preference must certainly be informed by some understanding of values and ethics. Also, phrases like "courses of action" and words like "situations" take us beyond notions of objects and features and into the realm of interactions and

interventions in environments. Another important work that treats design in the context of interactions and environments is Lucy Suchman's seminal writings on Situated Actions (1987).

In what is not necessarily a contrast to Simon's notions, Winograd and Flores (1986) have described how any kind of design changes the designer in a way that makes rational objective understanding of design interventions less than perfect. Our friend and colleague, Jeff Bardzell, summarizes Winograd & Flores' notions as follows:

> "Winograd & Flores emphasized that the traditional conceptualization of technology as a tool subject to the will of the rational mind is flawed, replacing it with a conceptualization that emphasizes human action in the world rather than abstract reflection, as well as the agency of designed objects once they are put into the world, specifically their ability to change the relationships between people and their environments".

Such an understanding of the role of design implies that values and ethics are inextricably tied to any design action in the world in a manner which precludes values-neutral design. Similar notions of ontological designing are central to the work of the EcoDesign foundation as described by Willis (1999). Winograd and Flores' (1986) work is an apparent inspiration for Tony Fry's (1999) notions of design as a choice among potential futures.

If design values can be construed in the deepest sense in terms of choosing among potential futures, then our notion of software as a material of design potentially spans these categories of values. It does so in the sense that software may be understood as a fabric *of* the future and a fabric that *is* the future. The choices we make about how we deploy software in the world as elements of artifice profoundly changes the world in ways that are irrevocable, for better or worse. This unavoidable change and choice demands that we act thoughtfully in the use of software as a material of design—a theme that has been greatly developed in Löwgren & Stolterman's (2004) Thoughtful Interaction Design. For example, the world wide web has enabled transactions and business models that were not possible 10-15 years ago, and it also creates needs for thinking otherwise about issues of security and privacy—that is, the world wide web as a material and fabric of our present condition creates peril together with advantage. As we stated in the discussion of the issue we address in this proposal, thinking of software as a material or fabric of design—even metaphorically—compels us to consider software design as a values-rich design discipline.

In what precedes, we have referenced some of the giants of the computational sciences, Norman, Simon, and Winograd. There are a great many other authors who merit reference here. Many of these are familiar in the world of design and are becoming familiar within the world of informatics, especially in the now accepted confluence of HCI and design. Victor Margolin and Victor Papanek have written extensively about values in design (Papanek, 1985; Margolin, 1989; Margolin & Margolin, 2003). Batya Friedman (1997) has contributed a seminal edited volume about human values and computer technology. The volume divides its contributions into a triumvirate of concerns, namely the concept of human values in design, the notion of computers as

anthropomorphic agency, and the practice of "value-sensitive" design. A number of others have written about values in HCI, notably Bonnie Nardi and Vicki O'Day (1999) who have described information ecologies as a values-oriented approach to technology.

Methods : What design is as an activity

Oftentimes, people conceive of design not in terms of objects or values but rather in terms of activities and processes. Once there is a conception of activity and process there is a need for codification with the purpose of making these activities repeatable and transferable. This is our understanding of what method is. The level of formality that is applied to the codification actually varies quite dramatically between design disciplines and even within software design. For example, some software needs to be written at least with attempts at formal proofs of correctness as for example in safety critical applications, whereas other software can be built on notions developed with storyboards. This range of formality can actually serve to isolate various software design communities one from another.

Thinking of design as an activity introduces the question of if and how design can be made to be a science. This question has been treated extensively in the design literature in a way that makes it curious how the notion that there is a science of design can be taken as axiomatic as is the case with the US National Science Foundation's present funding of research on the "Science of Design" under the directorate for the computer sciences. We believe that there is some part of design—and indeed some part of design with the material of software—that can be treated as a science. Nonetheless, we also believe that it is important to understand the controversy this position creates in design disciplines outside of software design and to profit from the understandings that this controversy reveals. For example, if you believe that design can be made to be a science, you are likely to believe that methods and processes in software engineering such as star, rapid application development (RAD), joint application development (JAD), eXtreme programming, and so forth — see Preece, Rogers, & Sharp (2002)—can lead to results that are reproducible by different groups employing the same methods. We know that this is hardly true for software in the sense that it is true for other sciences such as chemistry or physics.

A notion of Science of Design for software-intensive systems and software-embedded systems could have two possible notions of activity, namely (1) the application of information technologies to the theory and practice of design, and (2) the theory and practice of design under some notion of appropriateness in which the core expertise is interactive information technologies. Appropriateness is determined by a value system such as technology-centeredness, enterprise-centeredness, user-centeredness, human-centeredness, humanity-centeredness, or sustainability. Clearly, the choice of a value system relates to one's thinking about *what design is about*.

The very idea of "Science of Design" is on the surface dissonant to many designers. Simon's (1996) conception of design as a science of the artificial is founded in the notion of thoughtful use and interpretation of knowledge from the natural sciences. Still, the phrase "Science of Design" is a compounding of over-loaded

meanings. Nigel Cross (2001)—a cornerstone figure of the design research community—clarifies and enumerates these meanings in a manner which we diagram as in Figure 4.

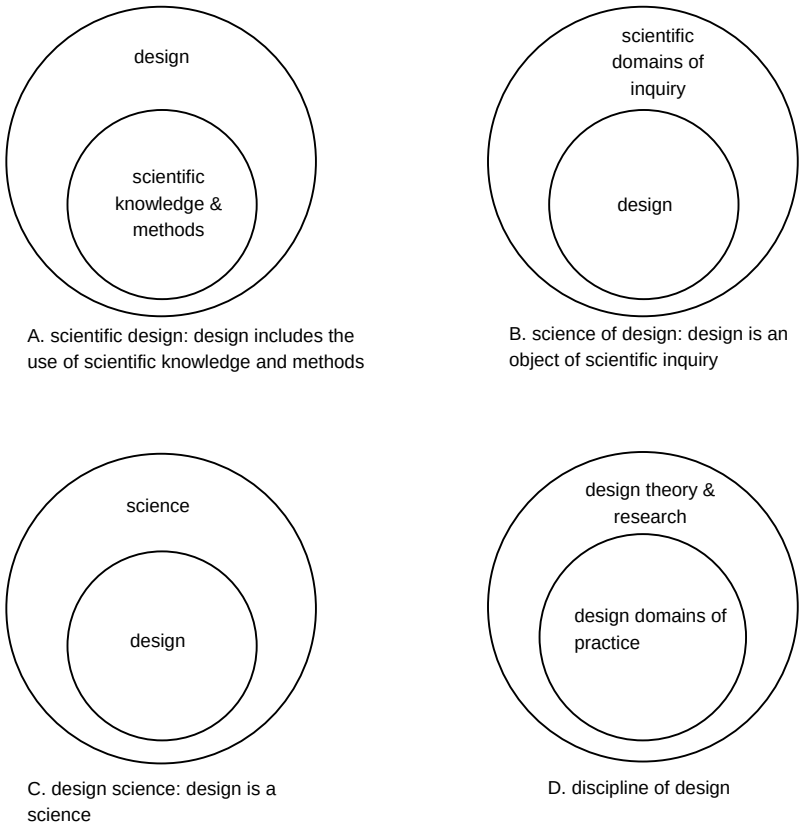

Figure 4. Our diagram of Nigel Cross' account of notions of science and design.

Cross disambiguates *scientific design* from *science of design* from *design science*, but most importantly he emphasizes a notion of an independent *discipline of design* as a reflective practice after Donald Schön (1987).

Sometimes the notion of design as reflective practice is set in opposition to the notion of design as a science in the design literature. This should not be. We believe that reflective practice is a necessary part of design that complements and makes sound any attempt at being formal and scientific. Particularly in our pedagogy, we have created an atmosphere in which science of design and design as reflective practice are both integrated into the design processes. One of ways we do this is simply to prefer notions of design frameworks to notions of design procedures. We claim that much confusion owes to the fact that prescriptions for design processes are seldom followed in practice. In fact, this may occur because design tends to be messy and strict ordering of steps—however iterative—is not how things tend to come to be which need to incorporate creative reasoning. See Fitzgerald, Russo, & Stolterman (2002). Ideas, insights, experience, feedback from prototypes, needs and requirements from observations, concepts come to any design process in often inopportune orders

from the point of view of what could otherwise be a strict production process. This means that to be truly thoughtful in our design with the material of software we need to be continuously receptive to any and all relevant events that cause us to better our understanding. A framework functions as a container for design elements and an instrument of reflection—a record of an organically unfolding design plan, rather than a prescriptive step-by-step linear process.

Reasoning, Representations, and Interpretations: What designs are

Aside from values as motivations for design, aside from methods and frameworks as ways of creating designs, a third important element of our design philosophy concerns *what designs are* as objects of reasoning composed of representations and interpretations.

For us, it is at least as important to focus on *what designs are* as it is to focus on *what design is,* or *what design is about.* As a matter of practical necessity, one way to think about what designs are is the notion that *designs are explanations.* We may define the notion of *what a design is* as a plan or explanation—an ex**plan**ation about why things—objects, features of objects, affordances of objects, interactions between people and environments, ecologies of people and environments, futures and collective futures—are a certain way or why they should best be another way. Design explanations are motivated by a notion of values, which for us is a deeply held, personal understanding of humanity-centered design, one in which designs are judged on the basis of how they have created or will create coherent improvements in the collective human condition. This point is described in Blevis & Siegel (2005).

Some readers will link this notion of designs as explanations to Moran and Carroll's (1996) notions of design rationale in HCI, or to Alexander's (1977) notions of pattern languages in architecture, or to Simon's (1996) notions of *"courses of action aimed at changing existing situations into preferred ones"*. This notion that designs are explanations is related to all of these; nonetheless, it is not always practical in many contexts of design to embrace some of the mathematical formality of much of the literature on design rationale, even though such formality may be within the training of some of the participants of a design team. One way to think about representing design is not to emphasize accounts of how to do design, but rather to emphasize ways of encoding descriptions of individual designs in a manner that allows them to be compared as explanations one to another. Discovering a representation for design explanations that is easily understood by a multi-disciplinary design team with highly variant facility with formality is an open problem. This is one of the primary problems we propose to ameliorate by means of our current and ongoing research.

As it turns out, patterns in the sense of Christopher Alexander's (1977) early writings on pattern languages have formed the basis for attempts to create reusable, codifiable knowledge in software design for a long time. See Gamma et al. (1995), for example. As an interesting side note, Alexander's (1977) pattern languages as well formed part of the experience that allowed Ward Cunningham's (2001) invention of the wikiwikiweb. The differences between notions of formalism and patterns in architectural design and patterns in object-oriented design are significant and profound. Alexander's patterns are not the same as patterns in the object-oriented

programming sense, rather the propositions themselves reflect human behaviors rather than program behaviors and are not easily reducible to formal objects in the strictest logical sense.

The problem of creating representations and interpretations for design with the material of software is the confusion that accrues from these very different levels of formality of representations and spheres of interpretations. In an object-oriented, programmatic sense it is possible to be quite formal about representations and interpretations. However, in the sense of embedding software as a material of the environment, such formality may not be possible and at the very least, too much formality can actually create barriers to collaboration.

Practical Basis for the SoftMat Paradigm: The MKST Framework

In addition to this philosophical framework, we seek to operationalize the SoftMat paradigm in terms of a practical framework for improving collaboration between various design teams who design with the material of software.

Continuously improving ways of working in practice is difficult. Oftentimes, the need for change that is widely recognized along one dimension of collaboration is thwarted in practice by failure to understand the corresponding needs for change along other dimensions. Oftentimes, team members do not recognize all of the dimensions that they need to take into account in order to establish effective, multi-disciplinary team collaborations. We claim *by way of hypothesis* that the necessary dimensions that need to be considered include as highly distinguished elements the dimensions of *mind sets, knowledge sets, skill sets*, and *tool sets*.

By **mind set,** we mean the design philosophy of a team member, including attitudes towards values, appropriate methodology, and the role of reasoning—especially *apropos* of levels of formality. Clearly, a visual designer and an engineering-oriented software designer will be able to work together more effectively if the differences in underlying mind sets are made to be highly visible in order to establish common grounds. Moreover and at the best, these differences in mind set can be used to enhance the scope of issues considered by the team rather than force compromises.

By **knowledge set,** we mean the established background and experience of a team member, including technical expertise, issues of visual literacy, content or domain-area expertise, and other forms of specialized knowledge. Oftentimes, the jargon that is used in individual knowledge areas can lead to confusion. Sometimes this is comical—for example, computer scientists use the word "formal" to refer to mathematical rigor whereas visual designers use the word "formal" to refer to surface features or the shape of something.

By **skill set,** we mean what individual team members know how to do, including programming, problem solving, specification, architecting, sketching, prototyping in high-fidelity, low-fidelity, or appearance modes, communication, presentation, and so on and so forth. Skill sets are particularly "hard won"—the result of substantial time and experience—and therefore require perhaps the most delicacy in negotiating mutual respect and authority in inter-team collaborations.

By **tool set**, we mean the instruments of external cognition that team members use to apply their skills, including physical tools, methods and techniques, research protocols, and so on and so forth. Knowing how to use particular tools contributes to the sense of identity and contribution of individual team members.

In order to continuously improve multi-disciplinary design team collaborations, it is important to recognize the interdependence of these MKST dimensions. For example, a common approach to change is to develop a new tool with the expectation that doing so will make team members work in a different, improved way. If the introduction of a new tool is not combined with a corresponding recognition of the other three dimensions, the team members are likely to try to use the new tool to achieve only what they expected from the old tool, rather than the new possibilities that the tool designers may have had in mind. We hypothesize that any serious attempt to influence practice has to address every dimension.

Operationalizing the SoftMat Paradigm

In order to operationalize the SoftMat paradigm, we propose two frameworks based in the foundations we have described above, namely the values, methods, and reasoning (VMR) philosophy profile framework and the mind sets, knowledge sets, skill sets, and tool sets (MKST) practice profile framework. The frameworks consist of tables with questions related to the key concepts of each profile. The intent of the tables is to guide people who are working in interdisciplinary design teams in the construction of each team member or single-disciplinary sub-group's individual philosophy and practice profiles. The hope is that clearly characterizing each individual according to these frameworks will allow the team to function more effectively as a whole, given the transparency that accrues from this exercise.

In Table 1, we give examples of the kinds of questions that an individual team member or a single-disciplinary sub-group can ask her or himself or themselves in order to guide the construction of a philosophy profile.

values	what do X designers believe that design is about as a value system? what are the objects of design? what are the contexts of use? what are the primary concerns? what can be measured? what are the goals? for what are designers responsible? how does design improve life? how does design integrate with the environment?
methods	what do X designers believe that design is as an activity in terms of methods and frameworks? what do you do when you design?

	what frameworks are used? what methods are used? how is iteration used? how is prototyping used? how is user research conducted? how do you collaborate? how do you measure success?
representations	what do X designers believe designs are in form of plans or explanations? what is the form of a design? how are designs expressed? how are designs compared? how do you reason about designs? how are existing designs distinguished from planned ones? how is one designed compared to another?

Table 1. VMR philosophy profile questions

In addition, we think of any individual software design team member as having a particular MKST profile, given in terms of individual mind set, knowledge set, skill set, and tool set. Our hypothesis is that knowing the details of particular MKST profiles and knowing how to allow team members to combine their talents based on making these profiles visible can make design collaborations with the material of software more effective. Such transparency of the profiles may allow for collaboration in a manner which creates a true union of attributes, rather than breakdowns in communications that owe to hidden differences in perspectives.

In Table 2, we give examples of the kinds of questions that an individual team member or a single-disciplinary sub-group can ask her or himself or themselves in order to guide the construction of an MKST practice profile.

Mind set	what do X designers value as the outcomes of their practice? what is the most important aspect of design? which part of a system or a thing is the design? how can design be distinguished from lack of design? who is responsible for design? what makes design good?
Knowledge set	what do X designers understand as being important knowledge? what do you need to know about in order to be the designer? what is the subject of design research?

	what constitutes design knowledge? how is design knowledge acquired?
Skill set	what are the core skills that every X designer should have? what do you need to know how to do in order to be the designer? how do you work with others? what are the qualifications to be a designer?
Tool set	what are the core tools that every X designers should know how to use? what kinds of things do you use when you design? which tools are professional tools? which tools are appropriate at which times? which tools are appropriate for which purposes?

Table 2. MKST practice profile questions

Examples of the Profiles in Action

In Tables 3 and 4, we give some hypothetical answers intended to characterize the types of responses we imagine we will get if we ask different kinds of designers from different backgrounds the questions we pose in Tables 1 and 2, respectively. Tables 3 and 4 provide a hypothetical conceptualization of design with the material of software in terms of four characteristic design groups, namely visual designers, interaction designers, content designers, and software designers. By *visual designers*, we mean designers who are primarily concerned with the look and feel of designs with the material of software. By *interaction designers*, we mean designers who are primarily concerned with user experience and usability. By *content designers*, we mean designers who are primarily concerned with the production and architecture of information in designs with the material of software. By *software designers*, we mean designers who are primarily concerned with the engineering and coding of design with the material of software.

	visual designers	interaction designers	content designers	software designers
values (what design is about as a value system)	aesthetics affect culture	interactivity experience transparency	message credibility	performance correctness function
methods (what design is as activity)	creating form and image	designing for usability and user experience	understanding discourse and culture	programming, specification, testing, capability maturity model (CMM), object-oriented programming (OOP)

representations (what designs are as plans or explanations)	sketches, look and feel, visual artifacts, appearance prototypes	prototypes, demonstrations, task models	text, images, narratives	programs, specifications, requirements, unified modeling language (UML)

Table 3. Hypothetical example of four VMR philosophy profiles

	visual designers	interaction designers	content designers	software designers
Mind set	appearance	interactivity	message	performance
Knowledge set	visual form	cognition	narrative	algorithms & data structures
Skill set	drawing, sketching, brainstorming, illustrating, …	processes: contextual design, interaction design process, formative evaluation, iterative design, participatory design, …	secondary research, and analysis, précis, narrative, indexing, tagging	processes: star, spiral, waterfall, joint application development (JAD), rapid application development (RAD), …
Tool set	image and illustration tools, photography, video, cultural artifacts ...	usability labs, rapid ethnography, low & high fidelity prototypes ...	classification, reportage, secondary research ...	Software Development Kits, open source ...

Table 4. Hypothetical example of four MKST practice profiles

It is important to emphasize that Tables 3 and 4 are just characteristic examples of how the profiles may be instantiated. As important as the actual values is the process of each individual or sub-team filling out the table as a way of self-description that leads to a method for shaping shared understanding among different groups.

Future work on the SoftMat paradigm

We plan to continue to research the SoftMat paradigm directly by conducting interviews with a wide range of people involved in software design teams and analyzing these interviews to construct a detailed notion of MKST and VMR profiles that owes to a bottom-up research approach. With this analysis in hand, we hope to construct hypotheses about how different profiles can be made visible and combined. Once this is accomplished, we hope to find software design teams that are willing to work with us, construct their MKST

and VMR profiles, and then apply our hypothetical understanding of how to aggregate MKST and VMR profiles to their specific circumstances. We have already found several suitable teams. We expect to refine the SoftMat paradigm based on this future study.

Summary

We argue that regarding software as a material of design facilitates a notion of software design as a multi-disciplinary, potentially inter-disciplinary design discipline. We introduce the SoftMat paradigm as an instrument for fostering such inter-disciplinary collaborations about design with the material of software. The SoftMat paradigm operationalizes as frameworks (i) a design philosophy that seeks to integrate notions of values, methods, and reasoning, and (ii) a notion of design practice that seeks to integrate notions of mind sets, knowledge sets, skill sets, and tool sets as a basis for clarity and scaffolding of collaborations. As an instrument of collaboration, the goal of the SoftMat paradigm is to provide a way to open up and expose tacit, invisible value systems and practice habits, and to make these transparent in order to foster design environments based on mutual understanding.

Acknowledgements

We gratefully acknowledge Martin Siegel, Yvonne Rogers, & Jeffrey Bardzell.

References

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A Pattern Language : Towns, Buildings, Construction*. New York. Oxford University Press.

Blevis, E., & Siegel M. (2005). The Explanation for Design Explanations. *Proceedings of 11th International Conference on Human-Computer Interaction: Interaction Design Education and Research: Current and Future Trends*. Las Vegas, NV

Blevis, E., Lim, Y. K., & Ozakca, M. (2005). The Design Exchange: A Collaborative Online Community for Designers based on Shared Construction of Design Knowledge. *Proceedings of 11th International Conference on Human-Computer Interaction: Online Communities and Social Computing*, Las Vegas, NV.

Cross, N. (2001). Designerly Ways of Knowing: Design Discipline Versus Design Science. *Design Issues* (MIT Press), 17(3), 49-55.

Cunningham, W., and Leuf, B. (2001). *The wiki way: Quick collaboration on the web*. Addison-Wesley Professional.

Fitzgerald, B., Russo, N., & Stolterman, E. (2002). *Information Systems Development – Methods-in-Action*. McGraw-Hill.

Friedman, B. (Ed.) (1997). *Human Values and the Design of Computer Technology*. Stanford CA: CSLI Press.

Fry, T. (1999). *A New Design Philosophy: An Introduction to Defuturing*. New South Wales, Australia: NSWU Press.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

Gibson, J. J. (1977). The theory of affordances. In R. E. Shaw & J. Bransford (Eds.), *Perceiving, Acting, and Knowing*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kling, R., & Star, S. L. (1998). Human centered systems in the perspective of organizational and social informatics. *SIGCAS Computers and Society* (ACM Press), 28(1), 22-29.

Lim, Y., & Sato, K. (2001). Development of Design Information Framework for Interactive Systems Design. *Proceedings of the 5th Asian Design Conference, International Symposium on Design Science*, Seoul, South Korea.

Lim, Y., Rogers, Y., & Mehta, K. (2005). Designing an Environment for Co-located Collaborative Analysis of User Experience, *Proceedings of Workshop of In-Use, In-Situ: Extending Field Research Methods* (BCS-HCI), London, UK, October 27-28.

Lim, Y., & Sato, K. (2006) Describing Multiple Aspects of Use Situation: Applications of Design, *Design Studies*, 27 (1), pp. 57-76.

Löwgren, J. & Stolterman, E. (2004). *Thoughtful Interaction Design*, .MIT Press.

Margolin, V. (Ed.) (1989). *Design Discourse: History, Theory, Criticism*. Chicago: University of Chicago Press.

Margolin, V., & Margolin, S. (2003). A "Social Model" of Design: Issues of Practice and Research. *Design Issues* (MIT Press), 18(4), 24-30.

Moran, T. P., & Carroll, J. M. (Eds.). (1996). *Design Rationale: Concepts, Techniques, and Use*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Nardi, B.A. & O'Day, V.L. (1999). *Information Ecologies: Using Technology with Heart*. Camdridge, MA: The MIT Press.

Nelson, H. & Stolterman, E. (2003). *The Design Way -- Intentional Change in an Unpredictable World*. Educational Technology Publications. New Jersey.

Norman, D.A. (1990). *The Design of Everyday Things*. (2nd ed.). New York: Doubleday.

Norman, D.A. (1998). *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution*. Cambridge, MA: MIT Press.

Norman, D.A. (1999). Affordances, Conventions, and Design. *Interactions* (May), 38-43.

Papanek, V. (1985). *Design for the Real World: Human Ecology and Social Change* (2nd ed.). Chicago: Academy Chicago.

Preece, J., Rogers, Y., & Sharp, H. (2002). *Interaction Design: Beyond Human-Computer Interaction*. New York: John Wiley & Sons, Inc.

Schön, D. (1983). *The Reflective Practitioner*. London: Temple Smith.

Simon, H. (1996). *The Sciences of the Artificial* (3rd ed.). Cambridge, MA: MIT Press.

Suchman, L. (1987) *Plans and Situated Actions: The Problem of Human-Machine Communication*. New York: Cambridge University Press.

Willis, A. M. (1999). Ontological Designing. *Proceedings of the Design Cultures, Conference of the European Academy of Design*, Sheffield Hallam University.

Winograd, T. & Flores, F. (1986). *Understanding Computers and Cognition: A New Foundation for Design*. New York: Addison-Wesley, Inc.

Winograd, T. (Ed.). (1996). *Bringing Design to Software*. (1st ed.). New York: Addison-Wesley, Inc.