Issues in Multimedia Authoring
80-990

Keith Douglas
kdouglas@andrew

Art, Technology and the Role of Aesthetic Values in HCI

Introduction

Brenda Laurel's recent (1993) book concerns itself with arguing for a new way of understanding human computer interaction, namely through the analytical categories of the theatre. This suggests a fundamentally aesthetic dimension to computing.

Taking the latter as a suggestion, in this paper I will argue that technology in general and human-computer interaction (hereafter, HCI) in specific relies upon and must make use of aesthetic values. This (apparent) truism is not without its problems; I shall sketch some and propose a resolution.

To these ends I shall first sketch a précis of my own philosophy of technology and illustrate how HCI (as commonly understood) is a technological (and technical) discipline. I then show how it necessarily involves the concept of various sorts of value. Since it is easier to demonstrate the value-ladenness of technology in the ethical domain, I shall discuss ethical values first briefly and then move to aesthetic ones, demonstrating how dangerous and problematic it is to import them into technology and yet how necessary they are.

I shall then apply the general framework sketched briefly in previous sections to HCI in specific to see how aesthetic values may be necessary in computing by means of carefully examining the user. Next, I turn to possible objections to this approach. I close with some remarks on the future directions of this research.

A Sketch of the Aims and Goals of Technology, HCI as a Technology

This paper does not have the time and space necessary to expound the entire philosophy of technology of the present author. We shall focus primarily on the value-theoretic portion of this system, and take it as given that it (a) exists and (b) is secured by epistemological considerations developed elsewhere (Douglas

2000).

Understanding the background philosophy of technology of the present paper is best done by contrast. For this purpose, I shall sketch the author's views on basic science, applied science, technology and technics (Again, for more, see Douglas 2000). We shall also focus on factual science; formal science (e.g. mathematics) is not in our purview here.

Basic science is research into nature, society or things (e.g. brains) at the interface between nature and society for the sake of knowledge. Applied science is research into such knowledge that may eventually lead to applications. Technology is the application of scientific findings (i.e. law statements[1]) to plans of action to form organizations, proposals for artifacts, etc. Technics is the repair and actual manufacture of said artifacts, etc. Example of pure science: polymer chemistry. Example of applied science: polymer chemistry of wearable fibres. Example of technology: designing highly insulating clothing out of polymer fibres for space suits[2] . Example of technic: repair maintenance and maintenance of space suits. Note that the same individual may (at different times in her professional activities) do several of these stages. It is rare, however, for an individual to do all of them to any great extent. Many researchers focus on applied science and technology, or pure science and applied science, for instance. Note that these definitions makes the practice of medicine a technic, and that management and the law (particularly in the study of punishment), etc. into socio- and psycho- technics and technologies (to the extent that they are informed by psychology, sociology, etc). The existence of other psycho-technologies (however undeveloped) will play a brief role in the sequel when we discuss testability in HCI.

[1] A theory *strictu sensu* in this view is "simply" a hypothetico-deductive system of said statements.

[2] Note that on this conception, the pre- or non-scientific design of artifacts is called "craft."

Implicit Value Judgments in Technology

The origin of value judgments implicit in technology has been studied by many. However, due to our further interest in the testing of technological ideas (i.e. science based plans for action mediated perhaps by artifacts), it is prudent to make use of a formulation that will allow the furthering of this interest in testing without further ado. This will become clearer as we proceed.

This formulation is due to Mario Bunge, and is called the "rule based on law" principle. For the purposes of this paper, we may adopt the view that a (scientific) law statement is a sufficiently true factual generalization. (Note that these may of course be probabilistic, strictly causal, etc.) In turn, a rule is a specific instruction of the form "To accomplish goal G with means M perform (or refrain from) action A." Thus a "rule based on law" is a statement of the form "If D is desired, then do C" or "if D is not desired refrain from doing C." These are grounded on an appropriate law statement. Since a law statement (for our purposes) has as principle connective a material conditional, it is thus the ground for two possible rules, depending on whether the consequent of the conditional is desirable or not. This is why a technological proposal, based on rules involves a value judgment.

It is not the place to discuss the origin of our desires, value theoretic and otherwise. We take them as given, but note that depending on whether the consequent is desirable or not is what determines the value of the rule as a whole. A technological proposal thus makes a value judgment as it is a proposal to (so to speak) enact the antecedent of a conditional, or prevent one from being enacted. For more on this principle, see Bunge's *Social Science Under Debate* (1998), pp. 331-334.

From Ethics to Aesthetics

As originally formulated, the "rule based on law" principle applies to ethical judgments. Since (as we shall see) aesthetic judgment is also important to technology, I propose that the principle be extended to this domain as well. Both ethics and

aesthetics are part of the traditional domains of value theory within philosophy, so this move should not be too surprising. What might be viewed as problematic is the subjectivity involved.

The view that ethical values are subjective has some currency as well, but there is even more agreement that aesthetic ones are such. Assuming this is true is not damaging to my point. We shall later see that in fact aesthetic values are precisely the ways to meet the user's needs because they are subjective.

However, there is no intrinsic reason why the "rule based on law" principle cannot be extended. Bunge assumed that because only ethically salient situations would be desirable/undesirable; we simply extend this to aesthetic situations.

The Danger of Aesthetic Values

Despite the above, there is some danger that some aesthetic judgments will not be able to play a proper role in HCI. This is illustrated by two sorts of examples. We have already met some subjectivity related ones. A painting of some important bit of religion, (e.g. Titian's Assumption of the Virgin) for instance, will be only beautiful, or worthy of awe, etc. to believers[3]. To an outsider it may evoke nothing, or worse, hostility to those who might find reminders of cultural imposition and so forth.

Another possibility is that an aesthetic judgment might be so crass as to offend virtually everyone (except, presumably, the speaker), or it might reflect a hostility to technology generally. Both of these issues are echoed in a line from Heidegger's (1949) lecture course[4] The Question Concerning Technology:

[3] I do not mean that only a Christian may get a positive aesthetic reaction from this painting. However, if in HCI one wants specific aesthetic reactions, one needs to consider specific cases of reactions. "Do we want this reaction? What users will experience it?" are the sort of questions that should be kept in mind.

[4] Tom Rockmore (1992) has pointed out that this line was modified to be less crass in Heidegger's published version of the lectures.

> "Agriculture is now the mechanized food industry, in essence the same as the manufacturing of corpses in gas chambers and extermination camps, the same as the blockade and starvation of nations, the same as the production of hydrogen bombs."

I only bring the above up to show the extreme of what we wish to avoid. If someone really believes that all technology is of the same kind, little we can do (here) can convince him otherwise. However, that said, it may be that we are developing an application for which a technological solution involves the participation of those who are perhaps skeptical of the effectiveness of such measures. So we may in some cases have to deal with a reaction almost as extreme as Heidegger's.

For instance, if public health measures are regarded as a technology (and by the above account they are if informed by social science, pathology and so forth) a vaccination plan might a specific instance of the above. (I.e. there are a lot of people who are hostile to vaccination programs/mandatory vaccinations.)

While of course nothing as far reaching as a vaccination plan proper applies to HCI, an automated information kiosk in a public health center that describes the procedure, its motivations, and so forth might be a component. A person skeptical of vaccinations might be afraid or concerned about computers as well, so designing the information kiosk to be accessible is thus vital. This leads us into the next section. So, even something apparently unrelated to HCI can indeed result in its application.

<u>The Necessity of Aesthetic Values and HCI as a Specific Case</u>

But despite these dangers, I would like to discuss some values that people have that seem to be aesthetic in nature and are relevant to HCI. The present author has informally polled users, developers and computer scientists on what they consider to be

important aesthetic values[5] in computing.

This informal investigation serves two purposes, one straightforward and one a metapurpose. The straightforward purpose is simply to demonstrate a prima facie case for aesthetic values in computing. The metapurpose is to avoid the crass remarks of the Heideggerian sort above. We accomplish this task by asking the stakeholders in a technological system what values they wish to have "embodied" in it, and then, most critically, test to see if the system met their needs. This of course was not done in the informal discussion I set up for I was not actually designing a computing system.

I thus discuss some of the values that were brought up in conversation to serve both of the aforementioned purposes. These are: accessibility, user-friendliness, avoiding unnecessary cuteness, starting points, integration of form and function. There are of course others possible; the remarks above about this being a two-level argument should put any objections to this limited list in that regard to rest.

A fair number of users talked to me about accessibility. A user of a computer system for a given task has to be able to complete that task with a minimum of interference from outside the task, and with a minimum of instruction. This is not meant to say that programs or hardware with steep learning curves are forbidden, just that the learning curve should be related to the task[6]. This comes up in the traditional argument against commandline interfaces: they make even simple tasks inaccessible in this sense. Another, more specific example is Jasik's Debugger. Many

[5] I take it as given that these <u>are</u> values, because they are identified as such by my colleagues and friends. I will not consider the objection that users may be mistaken about their aesthetic values as being way off the present subject. My supposition is that a species of aesthetic fallibilism would just add another layer to the testing ideas under discussion, but that itself has to be born out by future research.

[6] I realize that this phrase is a "weasel word." Future research in this would have to delineate carefully what this is to mean. I take it as a primitive (or perhaps intuitive) concept for the time being.

developers praised this product for its thoroughness. However, all of them griped tremendously about the UI, which is regarded as idiosyncratic (to say the least) and unfriendly, even for a debugger.

A related point is user-friendliness. Error messages and feedback should be clear and written in correct language. Further, if possible, text should be localizable. There very little worse than a program designed for a given market suddenly producing an error in a foreign language. Fascile translations of programs might produce this if the runtime library used has not been translated. For instance, if one uses a RAD tool like RealBasic, and the exception handling is not translated this will happen. Of course, feedback is not limited to the verbal: highlights and shadows and springy action is also appropriate if done right.

“Unnecessary cuteness” was hard to pin down in detail amongst my interviewees, though it was echoed in similar terms by many. Generally speaking users do not want an interface to be “cute” for its own sake. Kai Krause’s software (e.g. Power Goo) is regarded as being such. Note of course that this is relative to an expected audience. Kindergarten students using KidPix obviously have a different standard of “cuteness” than an adult using Photoshop or Illustrator despite the (superficial) similarity in function.

“Starting points” relate to accessibility. The user should be able to “see where to go.” AppleWorks opens up with the user able to pick some literal starting points - it is this from where I have gotten the term I am using. In AppleWorks’ case, the user can select several document types and templates from a list; if the user develops her own “stationary” she can include this in the starting points too. This latter feature was very handy when I was doing nearly weekly assignments for some logic courses I did once. The now defunct (alas!) Apple Guide literally showed a user how to do things, which was a very valuable system for this concern of “starting points”.

Our final example of a user-determined aesthetic value is “integration of form and function.” This is best illustrated by a

development tool, Apple's Interface Builder, although it should certainly apply to many user applications. Interface Builder is used to develop the interface to applications for MacOS X. Developers tell me it both illustrates the form-function integration people suggest is an important value, as well as allows developers to more easily create it.

Interface Builder illustrates this value because its own UI is literally that of laying out components and tying their functionality together. If one wants a menu, one simply makes a new menu object and puts the menu items on it. In that respect, Interface Builder is like Apple's previous interface design tool, ResEdit. However, it takes the drag and drop and so on to an even greater degree. It also allows the developer to create form-function integrated apps as it hooks the source code part of development more tightly to the UI than some other interface development tools. (It is similar in that respect to Metrowerks' Constructor.)

Before moving on to objections, I would like to note that this approach to aesthetics seems to answer Mumford's criticisms of the aesthetic in technology. (Here I am reading this (ambient) use of technics to mean technology[7] in my sense.) He writes (1952, pp. 80):

> "You do not make a machine more human by painting it with flowers."[8]

Mumford is telling us that superficial characteristics do not change the status of artifacts. He is right; we should move to deeper aesthetic understandings of them and build those findings into them. I hope to have at least sketched a two-fold way in that direction.

[7] Like many critics of technology, Mumford uses his *mots d'art* unfortunately loosely. He elsewhere uses the term to include scientific research, in a manner that is most unfortunate. But that is another story for another time.

[8] One can only speculate what Mumford would have thought of the recent (2001) "Flower Power" iMac, a machine which is literally painted with flowers. Of course, it is (arguably) much more than that.

Objections

Here I would like to deal with seven objections.

The first of these concerns the plurality of values question we discussed above writ large. My objector asks: "Suppose you do all this investigation, and find that you get a Heideggerian type remark from some of your users, and this from another group, and that from a third and so on. What do you do? Try and average them out? Ignore part of the users and hope they will learn to use and appreciate your system regardless?" As we shall see, I have no hard and fast answers to many of these objections. Many work in a similar style to this answer, so it is vital that we look at this response carefully.

I suggest that again testing is part of the answer. Does group X find it easier to move towards Y's position or conversely? Is a compromise reasonable? Is the "user segment"[9] that will find your system inappropriate a significant one? Each kind of application will have to decide whether a given group is appropriately excluded. The blind not having access to Photoshop is clearly not as critical as them not having access to a computerized voting station, for instance.

Testing is not only a way to overcome an apparently extreme subjectivity in values, it is also a way to find out how users respond to other values than their own. For instance, some users might find they like something after all. (This suggests that surveys and questionnaires to users cannot replace but at most supplement actual usability testing.)

Another objection to consider concerns the nature of the testing I have proposed. Some technologists have designed technological systems "intuitively" and thus someone may object that by focusing on testing I am not allowing intuitive design and engineering. This objection possibly conflates two levels of testing and

[9] For commercial products, this would be a market segment. I use the term "user segment" because I do not intend to suggest that all computing systems are commercial products.

groundedness, and may misunderstand the nature of intuition. The Swiss engineer and architect Robert Maillart is a good example to focus on that does not deal with HCI, to avoid poisoning the well. As reported by Billington (1979), Maillart was not fond of the then common practice of designing by working out constraints and so forth, i.e. using mathematical modeling to the extent that was popular at the time. Instead, Maillart guided by aesthetic intuition in his bridge designs and minimized calculation - not used none.

That said, however, he did test his designs. Load tests are vital for bridges if only for safety reasons, for instance. He also did not rule out calculations and planning prior to design, just did not feel they are all important. (See Billington 1979, particularly pages 46-47, for more)

Another objection that one could raise concerns what one might call exaptation in computing systems[10]. I coin this term out of analogy to exaptation in biological systems. In biology (see, e.g., Campbell 1993), a trait of an organism is exapted if the function it is used for is different from the one that was originally selected for. For instance, it has been suggested that the early bacterial flagella were exaptations of protein extrusion systems. We can thus say that an exaptation of a computing system feature is the use of a feature in a way that the designers of the system did not have in mind. For instance, early versions of MacOS (prior to 7.0 or so) were designed around the Motorola 68000 processor. Since this processor only has a 24 bit address bus, certain developers made use of the upper 8 bits of 32 bit values in memory whose lower 24 bits were pointer types for other purposes. Of course, this scheme broke when the operating system started to enforce proper values on pointers in anticipation for use of a full 32 bit address bus. Needless to say that these sorts of features can occur at the user end of things, too. The objection I am considering concerns the testability of systems keeping in mind that exaptations are almost certainly going to

[10] I use the term “computing system” herein to avoid poisoning the well over whether I am discussing a hardware device, a software package, or as is very common, the integration of both.

occur in any sufficiently large system. My critic asks: “How do I test for these, given that there are in principle so many? What if a user tells me he finds that a certain way of doing things is the clear way to do things, but doesn’t quite work right, and I find he’s exapted my design?” (Or any number of similar questions.)

The answer to this family of objections can be summarized in the slogan: “Remember the time variable.” This can spell itself out in several ways. The first way is several rounds of testing prior to release. This way many possible exapted uses of features in your system will come to light and be handleable early in the development cycle. The second is to remember that one often has a second version of a product to make changes and corrections. Obviously in some cases a product does not (e.g.) sell well enough to require a new version, but in most cases one at least should plan to upgrade a product. In this case, the exaptations can be reinforced if this goes along with the “spirit” of the program, or removed if they are too awkward to continue “supporting.”

Of course, both have to be user-tested to see if the balance of the decision is in the right direction. In this sense exaptations can be “artificially selected” for much as one can breed a strain of E. coli with desired features simply by applying selective pressure. Developers should of course be hesitant on applying pressures analogous to selective pressure on their users, but it should not be ruled out *a priori*, particularly as other features users want may depend on the one being pressured. In the case of the Apple developer case, users (and developers) wanted systems with more memory. So 32 bit addressing had to be brought in, with pressures to change away from 24 bit. The pressure here was simply “We won’t support it forever, and the users are going to hate rebooting to use your program, and only with 8 megabytes of RAM,

too."[11]

Another possible objection would be to my starting point - am I not saying that Laurel has forgotten science and evidence in the name of art? Laurel (1993) does tell us that (pp xix):

> "Adopting a point of view that is relatively more 'artistic' than 'scientific' does not mean that logic, specificity or disciplined thinking must be abandoned - quite the contrary."

We commend her willingness not to sacrifice the *ésprit du système* or the *ésprit géometrique* to aesthetic goals. But we contend that she has forgotten one piece of the scientific (and technological) puzzle: testing[12]. Laurel's book is full of interesting ideas to revitalize HCI, but there is little attempt to validate those ideas beyond persuasive argument[13]. Worse, there are attempts at exactification (necessary but not sufficient for rigourous testing) that involve pseudoquantities. Consider (for example) her figure 3.7, on page 86. This plots the "complication" of dramatic action versus time. While this graph is useful metaphorically (i.e. it tells us broadly speaking what to expect in a good dramatic presentation) if we wished to test this network of ideas, we would have to find some means of "operationalizing" her variable of complication. This is not done anywhere in her text. She even goes so far as to claim that a fractal metaphor might be

[11] An example of this would be to somehow continue the support of the 8 high bits for non-pointer data in 32 bit clean versions of the MacOS. An emulation layer could handle this, for instance. But this would produce more unfortunate circumstances than it would solve. As is well known, any design decision involves tradeoffs. Apple solved this problem by temporarily allowing some machines to boot in a 24 bit mode so that older applications could run.

[12] Testing ideas (or rather their 'embodiment') in technology is rather different from testing ideas in science (see, e.g. Bunge 1999, particularly chapter 11.). Nevertheless the technological method for testing technological proposals can make use of scientific methods.

[13] I do not mean to suggest that argument is not necessary, merely that it is far from being sufficient.

apt to describe the shape of this curve. Laurel has absolutely no grounds for suggesting this - an exact model would be needed.

This should not be read as a nitpicky philosopher of science/technology comment: it affects the entire merits of the work, for the usual reasons. This reason is simply that without exactness (and testability) improvability is impossible, except ad hocly. If you don't know where you are (lack of exactness) one cannot see how wrong you are (testability) and hence cannot tell how to improve and refine.

Another possible objection comes from the developer's perspective[14]. This objection concerns what might be dubbed the "skins" or "themes" family of problems. These objections concern several related issues: (a) what is the role of the user in selecting the interface mechanisms and appearance and (b) what is the role of the developer in the same regards?

I have stressed throughout this paper that user testing in HCI is absolutely vital. I have also suggested that users should have some freedom to decide how they wish to use their computer systems. The problem with these principles that is that they lead to conflict, particularly when the developer's own values are in the mix as well. This conflict arises out of "skins" or "themes." These are interface modification templates for programs, which can apply globally (all programs on one operating system or interface, e.g. all Aqua programs under MacOS X) or to one specific program designed to be skinnable. The former are the most dangerous, though opening up one's application to be skinnable leaves the developer in something of the same position as allowing the OS to be such.

Testing an application on a skinned OS is complicated for several reasons, and thus allowing the user unlimited freedom to do so is problematic.

For one thing, it makes support a nightmare. One can imagine the

[14] I am indebted to conversations with my correspondent and fellow MacHacker, Barry Semo (2002) for a few ideas on this theme.

life of a technical support staffperson trying to tell a user to do such-and-such with such-and-such a widget and it not appearing the same because the UI has changed. Second, unless care is taken to restrict the properties of items carefully, the skinning can create subtle instabilities or bugs in a UI. This was reported to be a grave problem with the old MacOS interface tweaker Kaleidoscope. By allowing interface modifications to change the number of pixels for interface elements (including text of dialogue boxes, for it allowed font changes), one-off bugs in programs could be provoked.

The response to this seems to be to fix interfaces. Allow users (in testing) to play a great role in the design of interfaces to computer systems, but not allow unlimited freedom once the system is "finished." This is something of a messy issue - it requires further investigation into what is a finished product in this regard.

The sixth objection I will consider comes from a management-concerned sort; in particular, someone concerned with budget. She asks: "By emphasizing user testing, aren't you running the risk of making software more expensive, longer to develop, and so on? And what if our competitors cut corners and we don't, and our product takes longer to come to market and thus loses out because we did all of this?"

I should not be read as saying that user testing is the be-all and end all value in computer systems design, and does have to be traded off with other values such as profitability (in the case of commercial systems, etc.). But I also should stress that it is my guess (studies would have to bear this out) that spending more on UI development up front would pay for itself in savings on support staff[15]. I should also mention that this paper is largely value-

[15] This was apparently born out many years ago (1995) when a Gartner Group study showed that the total cost of ownership of MacOS machines vs. Windows machines was lower on average for the former, despite larger startup costs. Apple attributed its success in this study to its greater emphasis on UI. As far as the present author knows, this sort of study has not been repeated recently.

theoretic in focus; as such it is an attempt to make things better over all, not merely maximize income of shareholders in a corporation. But this latter concern is part of another debate for another time.

A final objection to consider comes from some research in HCI. Some research in HCI is about just deciding which colours would be good for web designs, which ranges of audio to use and so forth.

This objection misses the discussion about technology vs. applied science. I admit there is an applied science of HCI as well as a technology. Many branches of computing are divided this way, and certainly there is no reason to suppose the same person cannot wear both hats. But one has to be aware that one is wearing one hat or the other at a given time[16].

Future Directions

The most important future directions for this work are two fold. One theoretical, one practical. Practically, one would have to try the interview-and-test technique I have suggested and employ it in the context of designing a computing system. Some of this is of course already well known and employed in the field. But I suggest that treating user preferences aesthetically rather than just functionally lends new insight into how they are understood by users. This in itself requires more psychological research to bear out.

Theoretically, investigating the roots and origins of the values that come up in users may allow developers and HCI people, etc. to develop better frameworks, maybe even theories to better help develop computing systems for the future. Note that this somewhat makes use of applied scientists in HCI, as noted previously. It is my hope that these reflections will help in some small way with these projects as well.

[16] I am indebted to some brief discussion of this concern with Professor Jeannette Wing (of Carnegie Mellon University's Computer Science department) on this theme. Professor Wing's research is on programming languages, and this area of computing also has an applied vs. technological split.

Works Cited

Billington, David. 1979. Robert Maillart's Bridges: The Art of Engineering. Princeton: Princeton University Press.

Bunge, Mario. 1998. Social Science Under Debate: A Philosophical Perspective. Toronto: University of Toronto Press.

Bunge, Mario. 1999. Philosophy of Science (2 vols.). New Brunswick: Transaction.

Campbell, Neil. 1993. Biology (3e.). New York: Addison-Wesley.

Douglas, Keith. 2000. "The Science/Technology Distinction, Or Who Gets to Play With Toys?" Unpublished paper for University of British Columbia Course PHIL 536A, Spring 2000 semester.

Heidegger, Martin. 1949. "The Question Concerning Technology." Lecture course, University of Freiburg. Translated and reported in Rockmore 1992.

Laurel, Brenda. 1993. Computers as Theatre. New York: Addison-Wesley.

Mumford, Lewis. 1952. Art and Technics. New York: Columbia University Press.

Rockmore, Tom. 1992. On Heidegger's Nazism and Philosophy. Berkeley: University of California Press.

Semo, Barry. 2002. Private communication.

Wing, Jeannette. 2002. Private communication.