5

A Matter of Digital Materiality

Tone Bratteteig

Design is about imagining future possibilities and making things that enable us to live some of these possibilities. ‘Maybe the most fascinating thing about design is that it is a process that starts with a thought and ends with the world looking different’ says Stolterman (2007: 13). Design starts with the making of ideas – of possibilities and of problems and solutions (Schön 1983; Lanzara 1983). The ideas get clearer as they are formulated and communicated, concretized and tried out in detail (Bjerknes and Bratteteig 1987; Henderson 1999). The imagining of the design result drives the process forward.

An essential part of design is giving form to some material so that it embodies the idea(s). Designers thus think both abstractly and very concretely about materials, making an effort to choose the right one. Design is ‘thinking with materials’, and designers need deep knowledge about their materials (Stolterman 2007: 16). The future possibilities – the ideas – are grounded upon how well the designer understands the materials: the material opens possibilities but also creates limits and conditions for the design. Some even say that the material ‘tells’ the craftsperson what it ‘wants to be’ – a particular piece of wood ‘wants to be’ a particular form in a chair. Similarly, the craftsperson must have a feeling for how a particular idea ‘wants to be’ manifest in a material or be expressed in different materials. Design thinking is thus very closely connected with the physical world, with the material and with the complex reality – with the hand (Stolterman 2007: 18).

What about digital design? Löwgren and Stolterman (1998) claim that the computer is a ‘material without qualities’, referring to Robert Musil’s (1996) novel ‘*The man without qualities*’. Computers are extremely malleable, and everything that can be described can be represented on a computer. Vallgårda and Redström (2007) criticize Löwgren and Stolterman’s view by commenting that a material without qualities or properties can ‘hardly qualify as a material’ (p. 514). The fact that the material is ‘so flexible it almost can take on any form we want’ misleads us to see it as ‘immaterial’.

This chapter sets out to discuss whether it makes sense to talk about computers as material in digital design. After a brief introduction to computers and the digital, I move on to talk more generally about materials in design, and discuss how the vocabulary for describing materials can be used to talk about digital material.

I. Wagner et al. (eds.), *Exploring Digital Design: Multi-Disciplinary Design Practices*, Computer Supported Cooperative Work, DOI 10.1007/978-1-84996-223-0_5,
© Springer-Verlag London Limited 2010

Because of the ‘immaterial’ nature of digital material, I also consider other ephemeral and less physical (‘immaterial’) creative processes, and how they might help us understand digital design and digital materials. The last section looks at relations between materials and the design process, and points to the work and the knowledges concerned with materials needed in design. The conclusion summarizes my view on whether it makes sense to talk about digital material and whether it matters that it is digital.

Characteristics of the Digital

It is well known that the term digital comes from ‘digit’, which means number – originally ‘finger’ referring to counting on the fingers. ‘Digital’ means represented as digit(s), using calculation by numerical methods that involve the Arabic numbers 1–9 and the symbol 0, or by discrete units. According to this definition, anything represented by numbers is digital: my old thermometer is digital because I measure the temperature according to a scale and read it as a digit.

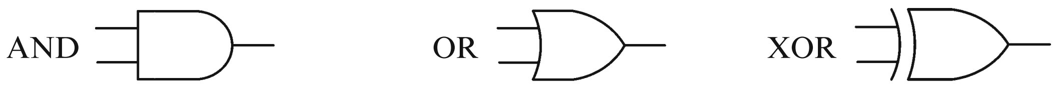
However, we normally use the term ‘digital’ about digital representations implemented on, or by means of, a computer: the digital is also electronic. In an electronic digital system – a computer – the digital representation is binary, as zeroes and ones. Everything represented – the system’s ‘content’ or information – is converted to binary form. Moreover, an electronic digital system (a ‘digital system’ for short) is a system that uses discrete values represented as binary numbers or non-numerical symbols like letters, signs, icons for input, processing, transmission, storage, or display of information, rather than a continuous spectrum of values as in an analogue system.

Abstractions

The basis for digitization is differences in voltages in electric current defined binary as 0 or 1. The data signals in a digital system carry one of two electronic (or optical) pulses: logical 1 when there is a pulse, logical 0 when the pulse is lacking. The binary representation is an abstraction from the fact that current is continuous; the abstraction is a construct, a choice, like all abstractions. The computer is built up by digital logic, by combinations of zeroes and ones into logical gates: AND if both inputs are high or 1, output is 1; OR if one input is 1, output is 1. The gates are further combined into increasingly complex logical units (see Fig. 5.1).

A digital electronic system is one of abstractions, all the way from the voltages to the surfaces that meet the user (Dourish 2001). When we click on the printer icon on our computer screen, the computer performs a number of operations at many levels of abstraction in order to connect to the physical device and make it print the signs we want it to print. The services offered at the user interface (such as printing, copying, searching) are abstractions that represent programs that are sets of abstractions themselves (like instruction sets, database architectures, communication protocols) materialized in transistors and electrical pulses. Remember: even the binary signals are abstractions imposed on continuous voltages.

The abstraction levels concern types of machine behaviour, and in computing it is common to refer to the physical machine (wires, disks, and integrated circuits); the logical machine (the collection of logical elements made from physical components like and/or gates); and the abstract machine ('a collection of abstract symbolic processors designed to resemble aspects of the world modelled' (Winograd and Flores 1986)). The abstract machine is described in modelling languages constructed to communicate to the programmer as well as to support the translations from the abstract logic into a physical machine (Winograd 1979). It is the idea of an 'abstract machine' that makes us think that the computer is immaterial.



A	B	output
0	0	0
0	1	0
1	0	0
1	1	1

A	B	output
0	0	0
0	1	1
1	0	1
1	1	1

A	B	output
0	0	0
0	1	1
1	0	1
1	1	0

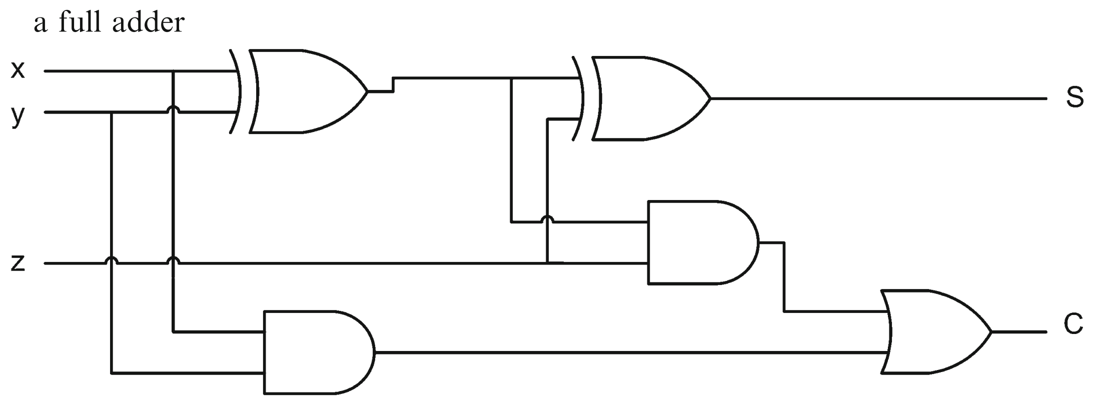


Fig. 5.1 AND, OR and XOR gates and a full adder made up of binary logic (Glette 2009)

Abstraction is the most fundamental characteristic of digital system design. The term abstract originates from Latin *abstrahere*: to pull away. Abstraction means to omit details and represent selected qualities of a phenomenon. Kramer (2007) emphasizes two aspects of abstraction: (1) 'the act of withdrawing or removing something' (p. 38) which means to leave out selected properties of the object in question, and (2) 'the process of formulating general concepts by … extracting common features from specific examples' (p. 38). Abstraction in this sense is used in other disciplines: in art, where details are omitted in Munch's painting 'The Scream' for example, or in music, with Bach's use of counterpoint. A famous design example is the map of the London Underground, in which the directions and distances do not match the physical, geographical underground tracks – and in this way makes the map easier to navigate from.

In digital design, abstraction allows systems to be considered at different levels of detail, to be broken down into individual components, and to be reassembled. Thus, the activity of systems design is to create and manipulate abstractions. Abstractions help us manage the complexity of a system by allowing us to hide it selectively in logical 'black boxes'[1] that we relate to through the characteristics of their interface. The 'we' above can be human users or pieces of code (i.e. other black boxes), and all we need to know about the black-boxed abstraction is what input they need and what output they produce (the functionality, procedure call conventions, and return values). The system's internal mechanisms, which describe and control how it goes about doing the work, are intentionally not available for inspection (Dourish and Button 1998: 414). Hiding the (internal) complexity behind simple interfaces enables us to build very complex systems and address them at different and logically appropriate levels of interaction. The fact that we design with abstractions leads us into finding ways to reduce the complexities of the phenomenon we design for (and to), and we make use of systems thinking with its support for modularization and black-boxing of system parts and layers. It is easier to design one part at a time, and to work with layers of emerging properties.

The notion of 'abstract' also connotes impersonal and detached. The act of abstraction is an act of translation that involves the creation of generalized logical units and categories that are used as building blocks in a system. Abstractions span classes of objects that are concrete: found in the real world, particular and specific, tangible and made of solid mass. Systems thinking[2] helps in making general structures and processes where the particularities of a single instance are represented with variable values in a class of similar phenomena. When applied to the world, systems thinking makes us see the relationships between the whole and its parts. The danger is, however, that systems thinking easily seduces us to think that the world *is* a system (Bjerknes and Bratteteig 1987; Bowker and Star 1999).

[1] Nygaard (2002) defines object-oriented systems as composed of interrelated components, where each component has some properties and some action connected to it. The objects are instances of object classes. Classes can be divided into subclasses that inherit the properties of their superclass. The modularization in object-oriented programming is a method of simplification. The black-boxing achieved with modularization is also discussed by Latour (1999).

[2] Systems thinking applies a systems perspective on the world (for some time and for a purpose, see (Nygaard 2002, 1986)), seeing a part of the world as a whole, built up of interdependent components, where the properties of the system are more than the sum of the properties of its parts; emergent properties appear at different hierarchical levels of the system (Checkland 1981): A bike as a whole has different properties than the sum of each of its parts.

Representations

Abstractions represent concrete objects, structures, and processes that exist in the real world and, as the abstractions are concretized come to exist in the real world themselves. The representations aim to model actual, physical phenomena happening in a real-world context, and represent what are considered to be the important properties of the phenomenon (which means that they embody choices and trade-offs). However, the modeller may over-simplify or mis-represent the phenomenon if that serves the purpose of the modelling.

Representations are not uniquely related to digital systems; they are an integrated aspect of how human beings deal with complex phenomena (e.g. medical diagnosis systems, see Bowker and Star 1999). In the hospital I am represented as a patient and I 'become' my blood test results or heartbeat or dysfunctional body part. Laboratory tests of the enzymes in a blood sample are interpreted as a representation of the size and seriousness of a heart attack.[3] Measurements of the heartbeat on a scope screen need to be interpreted because different monitors display the same rhythm differently (Bjerknes and Bratteteig 1987, 1988). Fortunately, such measurements are interpreted, taking into account the particularities of the individual real patient: the body, the sickness history, the medication, and the physical observations of the phenomena represented. Making and interpreting such representations are part of the professional work that health workers do. The representation is about their work object: the patient, and is a work object itself (Berg 1997).

The making of representations is a basic craft in system design – at many abstraction levels. Literature on system design indicates many different ways to go about representing the system-in-the-making; focusing on objects, data structures, functions etc. Software engineering methods describe how to plan and build a robust system, where the aims and requirements of the system are given. System development methodologies are often built upon a set of methods and tools for making system representations, supporting a particular view on systems and solutions (Andersen et al. 1990; Bjerknes and Bratteteig 1987). At lower levels, the representations detail the system parts, and the representations can be unambiguous, as the context of use is a fully specified computer system rather than a more or less unpredictable human use setting.

Another essential characteristic of the computer as an abstract machine is that it is 'symbolic' (Winograd and Flores 1986). It is made up of symbols taken to be tokens or signs that stand in for something else: they represent something else. A focus on symbols suggests that interpretation and meaning-making is necessary for design and use of the symbolic machine, in fact the machine must be given meaning

[3] The enzymes ALT (*alanine transaminase*) and AST (*aspartate transaminase*) leak out of damaged muscles into the blood and can be found after a heart infarct, but can be caused by other muscles as well. The same holds for the enzyme CK (*creatine kinase*). The combination of these should identify the source of the muscle damage. Blood tests for enzyme values are taken until the level is decreasing, signalling that the damage on the muscle has stopped (Bratteteig 2004).

by its users. The users in this sense co-construct the symbolic machine by relating to it in accordance with the meaning s/he gives to it by using it. The challenging ambition for designers is thus to communicate their (intentional) interpretation well enough for the users to share their interpretation and utilize the intended potential of the machinery in question. Symbolic machines therefore depend on successful communication between designer and user (Andersen 1986), and, hence, on how the symbols are materialized and given form in the artefact. This adds to the general challenge in design to communicate functionality of an artefact (Bratteteig 2002). The communication of the functionality of a symbolic machine can make use of both symbolic and physical forms, but need to speak the language of the contemporary culture.

Symbols are culturally and socially defined. Context and genre constitute conventions that shape our interpretations of symbols[4]: the symbol '@' reads differently after you have opened your first email account. Symbols require interpretation and meaning-making: linguists would say that the 'reader' needs to understand the meaning (the signified) that is communicated through the symbol/sign (the signifier). Symbols are concrete forms – matter – that signify meaning.

> [The] representation is in the mind of the beholder. There is nothing in the design of the machine or the operation of the program that depends in any way on the fact that the symbol structures are viewed as representing anything at all (Winograd and Flores 1986: 86).

Representation is an act of signification, which includes creating a concrete expression that can be made sense of by the people in its context. All representations influence the way we act and understand the world, and get embedded in our way of living; a classic example is the clock representing time – deeply embedded in Western society, culture and organizations, even our identity (Weizenbaum 1976).

Process

A basic characteristic of digital electronic systems is that they are program executions: they do things. A program execution is a process characterized by aspects like input (start condition), output (end result), and properties such as speed, duration, rhythm etc. A machine that performs processes automatically is an automaton; the automaton processes input and transforms it to output, it produces responses to stimulus, and it changes its state(s). Simple automatons, like the thermostat, are set to turn the heat on and off for you. More complex automatons, such as the calculator or the bank's account system, do the mathematical calculation for you.

[4] cf. Andersen 1986; Andersen & Bratteteig 1989. In this sense, language is a system of symbols, but the meaning-making in our culture also includes iconic symbols and symbols that speak to other senses: hearing the Nokia phone signal three short, one long, three short beeps is easily interpreted as 'sms' in Morse code ▪▪▪– ▪▪▪ by telegraphers.

'A washer is a washer, whatever clothes you put inside, but when you put a new program in a computer, it becomes a new machine' (Gelernter 1998: 24). A program is a description of a 'virtual machine' that becomes real when activated in a computer. The concept of the virtual machine is 'a way of understanding software that frees us to think of software design as machine design' (Gelernter 1998: 24).

The virtual machine combines two aspects of the digital (and electronic) that makes it into a general machine: (1) the fact that the representation is an abstraction and can refer to anything: a number could be a temperature, an amount of money, a time (hour or date), a measurement of length or weight – it depends on the context in which we set the number; and (2) the fact that the representation is an abstraction of a process in a machine that can change its state based on input, and that contains a specification of operations to produce output.[5] It is the second aspect that I shall discuss in this section.

The abstract machine includes structures for action, both automatic action and responses on input from internal as well as external sources (like humans). The structure for action is called a procedure: a series of operations in a particular order that, when performed, will transform a particular input to a specified output. A procedure specifies the preconditions and frames for action. A simple thermostat measures the temperature and when a certain condition occurs, a certain action is taken; the temperature is below a preset value and so a heater is turned on. The washing machine washes by moving its interior at a certain speed (presented by labels as 'careful' or 'normal' washing), at certain temperatures and for a certain time (not independent of the temperature).[6] The ATM does not give me any money if I input a number greater than the number that the bank computer has registered as the deposit in my bank account.

In an electronic system, the procedure is called an algorithm; a concept traditionally used to denote the solving of mathematical problems. 'Algorithms are abstract descriptions of the solution to a problem, which may be solved by a machine' (Knuth 1973). Algorithms express structures for processes, and can be characterized by properties that refer to the way they are structured – finiteness, definiteness, input, output, effectiveness (Knuth 1973).

Algorithms work with symbols that refer to classes of concrete instances and thus represent abstractions from the specific values of the instances. 'The concept of "a variable" represents an abstraction from its current value' (Dijkstra 1976: 11). The concept of a variable captures the 'the quintessence of programming'.

A well-known example of an algorithm is Quicksort, invented in 1960 by C.A.R. Hoare. Quicksort sorts a set of cards (or whatever needs to be sorted) in an elegant way. It makes use of some basic abstraction mechanisms: recursion (referring to itself), and calling a procedure (a repeated set of operations). It partitions an array into small and big elements, and continues to do the same in each of the two new arrays

[5] All digital electronic systems are Turing machines: universal devices that manipulate abstract symbols and can simulate the logic of any computer (Minsky 1967; Knuth 1973).

[6] A curious fact is that the temperature scale on the machine fits the categories of the washing instructions attached to the clothes: the making of wool-washing programs is intertwined with developing ways of preparing wool so that it can stand this kind of machine washing.

(recursively) until there are no arrays left to be sorted. Here follows a short way of specifying this algorithm, below is a program that does the same:

```
pick one element of the array (the "pivot").
partition the other elements into two groups:
  "little ones" that are less than the pivot value, and
  "big ones" that are greater than the pivot value.
recursively sort each group. (Kernighan and Pike 1999: 32)
```

The representation of the Quicksort algorithm – and the algorithm itself – illustrates the way that real phenomena can be translated and represented in an electronic system. The skills and knowledges about forming digital (electronic) materials are just like this: forming abstract structures and algorithms into representations that can be read by humans (like the program below) and translated to electronic signals visible in machine behaviour.

```
/* quicksort: sort v[0] .. v[n-1] into increasing order */
void quicksort(int v[], int n)
{
   int i, last;
   if (n <= 1)                      /* nothing to do */
   return;
   swap(v, 0, rand() % n);          /* move pivotlem to v[0] */
   last = 0;
    for (i = 1; i < n; i++)         /* partition */
    if (v[i] < v[0])
    swap(v, ++last, i);
      swap(v, 0, last);             /* restore pivot */
    quicksort(v, last);             /* recursive sort */
    quicksort(v+last+1, n-last-1);  /* each part */
}
```
The swap operation, which interchanges two elements, appears three times in quicksort, so it is best made into a separate function:
```
/* swap: interchange v[i] and v[j] */
void swap(int v[], int i, int j)
{
   int temp;
   temp = v[i]
   v[i] = v[j]
   v[j] = temp
}
```
(Kernighan and Pike 1999: 32–33)

Processes controlled by a machine need to be correct, predictable, controllable, reliable etc.; they must behave according to a set of engineering qualities. We need to trust that the calculation is correct or else the calculator is useless. In particular, processes that are non-transparent and incomprehensible processes must be correct. We accept that we cannot make a call if we have no connection to a provider, or if the battery is flat, but if the telephone cannot be used under normal conditions, we throw it away. Predictability and human control of automatons is crucial.

As the automaton is always right, a certain level of knowledge is required to question its output. This also holds for the automation and digitization of manual processes:

it makes us question the value of the knowledge involved in the manual processes. In the 1970s, Norwegian dairies were automated and knowledge concerned with tasting, smelling, feeling, looking at the milk as it travelled through the factory became obsolete – and eventually disappeared. Instead came knowledges concerned with the representation of temperature and chemical composition, which constitutes a different set of skills and knowledges (cf. Zuboff 1989). The delegation of knowledge work to machinery made it uninteresting to maintain the knowledge about the physical processes.

Automatons are machines that process things and perform operations by themselves. An automaton has been delegated a symbolic process, e.g. calculation (Säljö 2000) and its calculations may be part of a larger human activity system. We can see the automaton as a 'prosthesis' that enhances human capacities (Weizenbaum 1976). As a lever enhances the human capacity to lift, a calculator enhances human capacities concerned with calculations. We can say that the calculator is delegated some calculation work – or even intelligence and memory – and that calculation is performed by an assemblage of humans and machinery.[7]

The level of abstraction of knowledge in society increases when many physical processes get transformed and translated to representations and measurements. The ubiquity of representations influences how we relate to both signifiers and the signified.

Chapters 1 and 2 describe aspects of contemporary ICTs that deeply influence our experiences with computers, both as users and designers: the developments in size-power-price relations, the miniaturization and the distribution of computing on ubiquitously present digital networks (be it gsm, gps, or the Internet). Nano technologies and extremely small computing devices that act as sensors and actuators can be distributed in the environment and embedded in physical materials (even woven into textiles). Wireless and mobile computing enable us to let go of the desk top as the place where we work or gather information. Ubiquitous computing (Weiser and Brown 1997) and 'everyware' (Greenfield 2006) open up possibilities for processing power in virtually all everyday artefacts. Digital design can range from global communication systems to digital dust. Many digital electronic systems are distributed over several devices and parts, and with increasing convergence to other systems. These developments give new possibilities for the digital material to be mixed with other materials, or take different shapes from previous generations of ICTs.

Materials in Design

A material is 'a physical substance that shows specific properties for its kind' (Vallgårda and Redström 2007: 514). Material is the stuff of which things are made. Material – referring to matter – is physical; it has a mass and occupies space, but it does not normally have a specific form and can be shaped. Matter can exist in

[7] Like distributed cognition. See Hutchins 1995, Säljö 2000, Latour 1999, and Suchman 2007 for different accounts of distribution of cognition over humans and artefacts.

different phases: solid, liquid, gas or plasma. Material sciences operate with categories of materials referring to their properties or their origin (artificial, natural). We can perceive materials by one or more of our senses.

While contemporary architecture and product design use digital tools to construct their expressive forms (see e.g. Sevaldson 2005; McCullough 1998), the material used is still mainly non-digital: wood, stone, brick, glass, metal, plastic, concrete etc.

The close relation to the material is easy to see in the crafts, for example in traditional boat building (like the Viking ships, see Fig. 5.2 left) where the builder tries to find pieces for the arched ribs by looking for trees with ‘knees’, as such naturally grown crooks are more rigid and flexible than wood with fabricated bends (Juul-Nielsen 1984).

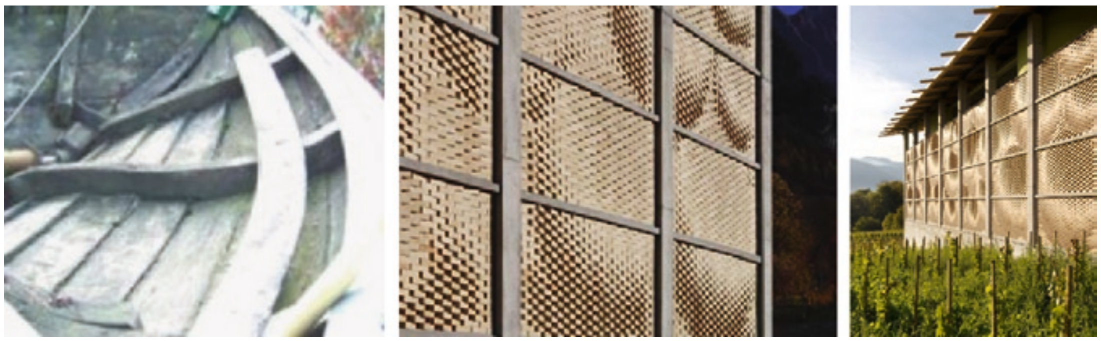


Fig. 5.2 From *left to right*: Ribs of Viking boat, and Gramazio and Kohler’s computer-designed brick wall and corridor, see www.gramaziokohler.com

Architects are also close to their materials, and spend much time getting to know, explore and experiment with materials as part of the inspirational phases of their design processes (Jacucci and Wagner 2007). Contemporary architects’ works include experimental use of materials; for example, the architects Gramazio and Kohler (2007) digitally construct and automatically build brick walls that express their design idea – and challenge our conception of a brick wall (see Fig. 5.2 right and middle). The composition of bricks so that they express shapes (grapes) and allow light into the room while avoiding direct sunlight, is impossible to construct without a computer. The automatic production of the brick walls required the bricks to be glued – which gave the wall elements different properties than bricks put together with mortar; the wall elements, for example, can be lifted and moved (Gramazio and Kohler 2007).

The material properties can be characterized on many levels, from the chemical basis to the use value of (compositions of) materials (e.g. timber shifts its properties when glued in layers (laminate)). Vallgårda and Redström (2007) characterize materials according to their:

- *Substance* The substance is the physical stuff that the material is made of. Definitions of materials refer to the atoms and the chemical and physical properties of the stuff.
- *Structure* Materials have structures – we can even say that materials at a molecular level are structures. Some material properties have their origin in chemical properties at the molecular scale.

- *Surface* All materials have surfaces, acting as the interface to the surroundings. Surfaces can be characterized by their texture and colour, but the surface often depends on other characteristics of the material, e.g. temperature, special treatment etc.
- *Properties* The chemistry of materials is important for understanding their properties at higher levels. However, characterizing the properties of a material depends on the perspective, what the material is evaluated in relation to; wood is, for example, seen differently by a chemist or an architect.

Vallgårda and Redström (2007) introduce the term 'composite materials' that are made in order to create a new property or change the properties of a material by combining it with another. They particularly mention the alloy aluminium, made from naturally-occurring bauxite refined into pig-aluminium – which is light-weight but weak – and then combined with other materials to make it strong and flexible – what we normally refer to as aluminium (p. 516).

This leads Vallgårda and Redström to point to the difficulties of distinguishing between materials and products: timber, the product of the sawmill, is a material for the carpenter. The blurring is even more present in composite materials, especially when the composition is fabricated to allow new forms. It comes down to the perspective or purpose of the activity in which the material becomes a part. We can say that it is a material if it is used to create something new that expresses a new idea. A 'bricoleur', who uses products and product parts as materials, can illustrate this point (Harper 1987).

Computers as Material

When discussing computers as material in design we can use the same categories as for characterizing other design materials: substance, structure, surface and properties.

> Perceiving computers as a material is … more than a metaphorical maneuver. It is a question of accepting their similar characteristics as significant enough to hereafter work with the computer in the same manner we work with materials like aluminium or glass (Vallgårda and Redström 2007: 516).

Substance Computers can be characterized at many abstraction levels, ranging from the 'immaterial' information, signs and meaning to the very concrete level of how the electronic mechanisms work: the voltages that 'do' the processing of input to output. At this level there is no difference between software and hardware; all levels we make up to handle the complexities of a computer are, in the end, voltages and manipulations of voltages. The size of the computer refers to the number of instructions processed per clock cycle – which also points to the fact that the computer needs to be whole in order to work, and that a smaller computer is not a big computer cut in two. The substance of a computer is thus the physical workings of an electronic artefact.

Structure the structural aspects of computers can also be discussed at several abstraction levels. At the level of voltages we deal with binary logic, whereas we deal with components as cpu,[8] memory and input/output devices at the physical composition of machinery for the desk top. Like other materials, the abstraction levels refer to particular levels of granularity that at lower abstraction levels are detailed even more. The structure prescribes particular processes in the computers. It is these processes that characterize the computer as material. 'This is analogous to how energy in other materials holds the molecules together as a structure and thereby constitutes them as materials.' (Vallgårda and Redström 2007: p. 517).

Properties We again need to distinguish between levels of abstraction since the lower levels consist of the processes that handle sequences of voltages that are translated into binary logic, while properties of the higher levels are concerned with the quality of the higher order processes. The many layers with emergent properties make it useful to apply a systems perspective on the computer – just as a bicycle as an assembly has different properties than each of its parts. At some level the computer as material is combined with other materials (silicon, metal, plastic, glass) but the 'raw' material of a computer is the processes; the computations. Vallgårda and Redström (2007) therefore compare the computer with aluminium: the raw aluminium is useless unless prepared and combined (in an alloy) with other materials. Raw aluminium is interesting because its properties are potentially useful, but it needs to be treated and prepared in particular ways in order to make use of its potential. They conclude with characterizing the computer as a composite material.

A view on computers as composite materials emphasises that the properties of the 'raw' computing is maintained or realized through its combination with other materials, and that additional or changed properties can develop in such combinations. The combination involves other materials that have particular properties, and it involves the preparing of the composite as one composite material.

Vallgårda and Redström (2007) use the concept of the computational composite to discuss computational textiles, computational concrete and computational 'tensegrity' (tensegrity referring to 'a skeleton structure that consists of members in continuous tension and members in discontinuous compression.' (p. 519). They maintain that the properties of computational composites are concerned with the computational processes, and connect the composite properties to the states that the composite goes through, the transitions between these states, and the control of this process. They therefore connect the properties to the algorithms and data sets in the computation and to whether the control of the process is distributed (an all predetermined, dynamically controlled data set or a set of dynamically changing computing conditions depending on dynamically collected data sets (p. 517)).

Concrete Abstractions

From my walkthrough of digital electronic systems above, it seems that the two properties characterizing digital design results are processes and abstractions. It is

[8] CPU central processing unit.

a basic property of computers that computational processes play out in time and also enable the computer to present time-consuming information (e.g., film, music). It is also a basic property that the computer is constructed by means of abstraction: abstraction of processes as well as of the structure and content of the processes. The computer, however, is very concrete.

> At first glance, the title of this book [Concrete Abstractions] is an oxymoron. After all, the term *abstraction* refers to an idea or general description, divorced from physical objects. On the other hand, something is concrete when it is a particular object, perhaps something that you can manipulate with your hands and look at with your eyes. Yet you often deal with concrete abstractions. Consider, for example, a word processor. When you use a word processor, you probably think that you have really entered a document into the computer and that the computer is a machine which physically manipulates the words in the document. But in actuality, when you "enter" the document, there is nothing new inside the computer – there are just different patterns of activity of electrical charges bouncing back and forth. Moreover, when the word processor "manipulates" the words in the document, those manipulations are really just more patterns of electrical activity. Even the program that you call "word processor" is an abstraction – it's the way we humans choose to talk about what is, in reality, yet more electrical charges. Still, although these abstractions such as "word processors" and "documents" are merely convenient ways of describing patterns of electrical activity, they are also *things* that we can buy, sell, copy, and use. (Hailperin et al. 1999: ix)

Hailperin et al. (1999) distinguish between three basic types of (concrete) abstractions: procedural abstraction, data abstractions, and abstractions of state. Procedural abstractions are abstractions of processes, seen as a 'dynamic succession of events' (p. ix) – which leads us to abstraction of states: the changes made by the program that affect the further execution of the program (or other programs). Abstractions of data concern how information is represented and structured so as to fit the computational processes. Their description of computing is a more specific account of the two characteristics addressed above: processes and abstractions. Procedures are structures for processes to go through a sequence of states, and the abstraction of data is the structures limiting the processes – here it makes sense to just talk about abstractions and processes.

Designing with processual material means to create or change processes – to look for processes and how general, repeating, quantifiable processes can be delegated to machines. We look for processes to automate – and create both generalised routines and exceptions to them. However, general categories and routines are creations rather than expressions of real life facts (Star 1991; Suchman 1994; Bowker and Star 1999).

Material for Process Design

Vallgårda and Redström (2007) characterize computational technology as temporal due to its computational processes, and as spatial due to the 'spatial form given to these processes by other materials with strong spatial elements' (p. 514). The secondary property is what Vallgårda and Redström calls spatial: the space made for the process to become concrete – be it physical or virtual. As pointed out above, Vallgårda and Redström are most concerned with the physical (e.g., pillows that combine textiles and computations), and claim that the computational 'immaterial' material is dependent upon other concrete materials to present itself – digital materials are therefore best understood as elements of new, composite materials.

Mazé and Redström (2005) suggest studying the computational object from the inside: from material to form, and from the outside: from interaction to form. They claim that the form of a computational object does not communicate the fundamental characteristics of that object, unlike, for example, how the size of a mechanical object tells us something about its power. 'There is no longer any correspondence between the complexity of the surface and the complexity of the inner workings of an object.' (p. 9). Maeda (2000) claims that this has consequences for both the designer and the user: the user cannot evaluate the object by its exterior; the designer gets less space for expressing his/her ideas – but can instead use the time dimension (when there is not enough space to present all necessary information, you need to present pieces of information over time). Mazé and Redström (2005) discuss the computational form as a combination of spatial and temporal form, claiming that this makes it impossible to separate form from interaction. Temporal form 'is manifested through spatial form elements in use' (p. 10). We therefore need to understand use as the concrete process of the temporal form rather than referring to users' experiences and needs concerned with their practices: use simply means the concretization of temporal form. Through experiments with spatial and temporal form combinations, they suggest considering the interplay of spatial and temporal properties (space may change over time) to recognize how temporal form develops through mobility (users moving), and that form is not entirely determined by the designers if temporal influence on spatial form is allowed – and vice versa. The form is thus dependent upon the interaction with the environment (the users).

Processual Material

Material – or matter – refers to a substance that occupies space. What if the space occupied by digital materials is a symbolic space spanned by the activities in which the process takes attention and time? As a starting point to explore the possibility of talking about processual material, I will use other kinds of processual design results. Candidates for such analogies are design results that exist as an experiential process; music, theatre, dance or other performances. Design of such processes results in descriptions of activities at a very detailed level that are used as prescriptions for the concrete realization of the performance (see e.g. Larssen et al. 2004; Loke et al. 2007). For all performances there exist notations that can be read and interpreted as structure for the process – bearing in mind that the process is a concrete instance of the envisioned process and will be different every time and with every new performer (which may make a new artist's performance enjoyable even if you have heard the piece many times before).

Based on these analogies, it makes sense to characterize processual material as structural representations framing processes, emphasizing that it requires knowledge to 'see' the process when reading the representation. Composers and musicians hear the music when reading the scores; dancers and choreographers feel the dance when they read the choreography; actors and directors experience the story and the characters when reading the manuscript or storyboard; programmers see the computational process when reading program code and systems designers see the system behaviour when reading the system description. Working with processual material thus implies working with representations of process abstractions, recognizing that the concretization of the process will be formed by the situational circumstances (see e.g. Harper 1987; Goodwin 1997). A good abstraction works for all relevant types of concrete circumstances.

It is interesting to bring improvisation into the debate, as seemingly unruly behaviour. However, Becker (2000) has observed that most improvising is 'not quite so inventive as the language we use' and jam sessions have a 'very strict etiquette' that says that, for example, the 'number of choruses the first player played set the standard others should follow. To play more would be rude, pushy, self-aggrandizing; to play less hinted that the first player had gone too far and, worse, that the following players who played less had less to say' (Becker 2000: 171). Theatre sport (a group of actors who gets some of their role-play specifics from the audience in the moment of acting) also follows certain rules of improvisation. Improvisation is thus just another set of rules, opening up for a limited set of variations – just like some computer applications open up for a larger set of user input or include a greater variety of responses to user input – both creating more variation but within frames.

Processual design is to arrange for processes to unfold in particular ways. It seems to make sense to talk about digital design as the composition of processual structures to larger processual structures that can be realized with different symbolic values materializing different process experiences for (and with) users.

Digital Material

Digital design deals with both the electrical processes allowing you to use you phone and the processes you engage in when using your phone. 'When we call a process a *computational* process, we mean that we are ignoring the physical nature of the process and instead focusing on the information content.' (Hailperin et al. 1999: x). If you worry about 'the current carrying capacity of the copper wire' (p. x) when you use your phone, your focus is on the electrical rather than the computational process. Digital design deals with concrete abstractions of processes and their conditions (the data). Some abstractions seem to require knowledge about the concretizations of the abstraction in digitized form, e.g., in order to create an adequate sound or good musical presentation process you need to know about digital representation of sounds.

Stolterman argues that the basic material for building digital systems is bits (Stolterman 2006; Blevis et al. 2006), Vallgårda and Redström (2007) that it is the composite of electrical voltages and other materials (e.g. textiles) that constitutes the digital material. Acknowledging their physical focus, I still maintain that it is the abstractions composed into a general machine that characterises computers as products and thus constitutes the building materials in digital design. I suggest seeing the digital material as concrete abstractions of processes, addressed at different levels of concretization. This view refers back to the view represented in Hailperin et al. (1999) – and many, many other computer science books – that abstracting processes is the basic skill of the digital designer. We need, however, to maintain that digital design can be carried out addressing different levels of concretization though digital design surprisingly often requires us to traverse several levels. One example is the design of a door-opening device: key cards that use magnetic strips as the key and add sound as a feedback to the user to signal correct or faulty card use. To make the sound easy to hear, the wavelength most easily detected by the human ear is chosen (3,000 Hz), also enabling the use of the smallest loudspeaker[9]. Similar need for detailed material knowledge can be found at all levels of digital design, from interface design that chooses a particular blue background colour for ease of reading for dyslexic users (Fjuk et al. 2006), to the design of the capacity of an electronic circuit to match the battery's capacity so that the device does not get over heated when activated.

In line with this, among the challenges of designing the iPod is making the very thin battery which, while providing enough power without overheating, also solves the legal, power-related and technical (storage, interface etc) issues necessary for realising the iTunes web site (Moggridge 2007). The iPod and its properties is a re-formation and re-configuration of (some of) the actors in music practices, providing a form that gives the iPod its identity and meaning – illustrating the complexity and range of a digital design by including the service infrastructure provided by iTunes and the aesthetically pleasing entry point to that service: the iPod. The content and meaning of the iPod crosses any layered model of the digital artefact.

The meaning of the iPod includes all concretization levels; it makes no sense to distinguish between software and hardware when they both cross the iPod artefact and the iTunes service – and the combination of them. With reference to music as a practice, it also makes no sense to single out the 'content' or 'meaningware' as separate from the apparatus in which it has its existence.

Whalley and Barley (1997) confirm that 'technicians work at the empirical interface between a world of physical objects and a world of symbolic representations' (p. 47). They claim that technicians act as 'the link between a larger system of work and the materials on which the system depends' and that 'the materials of relevance may be hardware, software, micro organisms, the human body, a manufacturing process, or a variety of other physical systems' depending on the context.

[9] A 3.000 Hertz tone of 0 dB is the softest sound that a normal human ear can hear.

The properties of the iPod are a result of design practices, where the fascination for the pieces, the materials, the small parts and their solutions can drive development of the overall design result: the iPod. Design includes the practices of tedious processes of getting the technical solution right, insisting that the idea will work,[10] and processes of bricolage, utilizing existing materials to achieve what you want.[11] Gelernter (1998) reflects such practices in his emphasis on the joy of programming.

Levels of Digital Design

The concrete abstractions with which we work in digital design can be seen as belonging to different levels of concretization (Mörtberg 2001; Bratteteig 2004), or as packages of 'processed bits' made into higher level logical pieces of hardware–software. Think, for example, of the carpenter who works with boards made of wood, or the tailor, who works with yarn and cloth of wool or cotton. Carpenters and tailors know about the processes of making cloth or boards from wool and cotton – or birch and teak – and about conventions for using particular types of cloth or boards (like 2×4 for building scaffolding or tweed for a suit) even if they do not do this preparation themselves. The building up of 'packages' of digital logic into larger logical units – or abstract processes composed into higher level abstract processes – enable us to apply digital logic at the design process by black-boxing system pieces so that we do not always need to worry about their internal composition, and can focus on the whole design as well as the details. The layering also makes it more difficult to distinguish between the materials and objects.

Earlier, I referred to timber as an example of being both a product of the sawmill and a material for the carpenter, illustrating the difficulty of distinguishing between materials and products. The blurring is even more present in composite materials, especially when the composition is fabricated to allow new forms. Levels of design encourages the packaging of increasingly larger pieces of digital logic to be outsourced during design, as well as sold as pieces to be easily tailored to the use context through integration and modification of variables (Grinter 1995, 1998). It is tempting to compare the levels of concretization to the layers of a building construction suggested by Brand (1994), where he distinguishes between the layers by reference to the rate of changing the layer, ranging from the site where the building stands to the stuff that the people living in the building buy, change, rearrange and throw away.[12] Also, different types of professional expertise are involved in building and changing the different layers: carpenters, electricians, plumbers etc.

[10] Hård (1994) documents how engineers try hundreds of times to make their idea work.

[11] Harper (1987) documents the knowledges and skills of a 'bricoleur', cf. also Ciborra (2002).

[12] Brand distinguishes between site, structure, skin, services, space plan, and stuff.

The distinction between tools and materials is particularly ambiguous in digital design. In his discussion of the construction of human–computer interfaces considered as a craft, Wroblewski (1991) says that '[a]ll partially finished work acts both as a tool and material' (p. 6) and also that '[t]he software craftsman works in a virtual toolsmith's shop, where all materials can become tools, and all tools are raw materials' (p. 11).

Close to the Material

McCullough (1998) introduces the term 'digital craft' in exploring how computer-aided design can be seen as a development of craft skills. He emphasises the dematerialized and symbolic nature of computers and thus how interpretational skills become more important. 'Common sense becomes visual sense' (McCullough 1998: 46): we read images rather than feel the artefact; the hand becomes less important as the kinaesthetic and tactile sensitivity of hand skills is replaced with interpretations of representations; where the formal properties are partly in the representation and partly in the phenomenon represented. Form in the representation can be seen directly; in the same way as graphical language elements often present structure in a distinct way (graphic symbols, 'boxes and lines', indentation in texts). McCullough suggests that the activity of seeing the form in the phenomenon represented is analytical, and emphasizes representational aspects of the language (system architecture, logical structures such as class structures and hierarchies of subclasses, interface properties).

Designing with digital (electronic) abstractions makes us focus on the quantifiable aspects of a phenomenon, and makes representations that can be subject to calculations and processing. Representations stand in for something else – but after some time, the original reference may be forgotten and the representation itself gains the status as the real thing (e.g. money). Working with representations is the work of interpretation and meaning-making.

Digital representations can also, however, be processed, presenting a model of the design result (Bjerknes and Bratteteig 1987). Laumann (2005) describes a process of creating a recording of a song. He documents the states that the song goes through, including the manipulation of sounds on his pc by means of the recording studio software. He reads the visual representations of the sound and manipulates the visual representations, cutting and pasting different recordings into one in order to get the sound he wants on the final version to be printed as the record. The skill to read the sound visualisation can be compared to reading musical scores: he hears the sound from seeing the visualisation (Fig. 5.3).

The design processes result in material forms that cross the contexts of design and use – and cross the concretization and abstraction layers of a digital system. Barad (2003) discusses the relation between materiality and signification:

> materiality is discursive (i.e., material phenomena are inseparable from the apparatuses of bodily production: matter emerges out of and includes as part of its being the ongoing

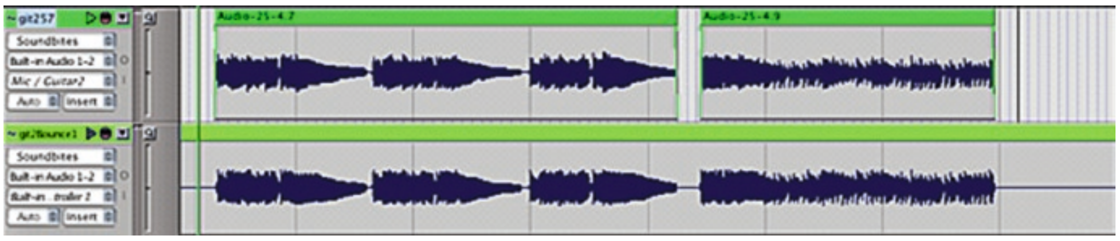


Fig. 5.3 Sound image of guitar recording: bounced recording on top, processed with Freeze Selected Tracks on bottom (Laumann 2005: 89, Fig. 10.21)

> reconfiguring of boundaries), just as discursive practices are always already material (i.e., they are ongoing material (re)configurings of the world) (Barad 2003: 822).

Barad argues against the representationalism present in software engineering and other representational crafts. She bases her argumentation on Butler's concept of 'becoming' and insists that action and speaking are inseparable, that language is an act, and that we cannot *not* communicate (see also Winograd and Flores 1986). The meaning of an artefact includes both the conceptual and the material – what Barad (2007) calls material-discursive. Fujimura (1996) similarly discusses how scientific knowledge is translated into methods and tools in scientific practices. Digital material is discursive-material composites, and new digital materials expand the boundaries of symbolic, representations and processes – we can dress in digital textiles, take digital medication, make 3D prints (Capjon 2004), get weather reports from opening a bottle or see the traffic density displayed as a shift in colour on our desk lamp (e.g. Ishii et al. 2001; Ishii and Ullmer 1997), and earn money in digital (virtual) worlds. The symbolic representations become more haptic and the haptic more symbolic.

Digital Matters in Design

The concept 'material' comes from Latin *materia*: matter, and refers to the 'elements, constituents, or substances of which something is composed or can be made' (Webster 2008). 'Matter' means physical substance: 'material substance that occupies space, has mass, and is composed predominantly of atoms consisting of protons, neutrons, and electrons, that constitutes the observable universe, and that is interconvertible with energy'. Matter, however, has a double meaning and refers also to facts. 'Materiality' refers to 'the quality or state of being material' (Webster 2008).

As a design material, the digital characterizes digital design. I have argued that digital material can be seen as concrete abstractions of processes, addressed at different levels of concretization. This view builds on seeing abstraction of processes as the basis in digital design (Hailperin et al. 1999). A levelled view also addresses the view that digital materials at the lowest level are electric voltages (Vallgårda and

Redström 2007), which at this level can be combined with other physical materials (like textiles) as computational composites. I also appreciate the view that a levelled view introduces a problem of distinguishing between materials and objects (Mazé and Redström 2005; Hallnäs and Redström 2006). Linking concretization levels with types of design enables us to acknowledge different kinds of design work ranging from nano-electronics to tailoring of systems to a specific organizational context (cf. Brand 1994).

However, we also should recognize that digital design often addresses different levels of concretization, and that a good design requires that we combine innovation at several levels (cf. the iPod/iTunes). The work of digital design is concerned with the building of working systems by imagining its use – not to be confused with the use perspective of the user in the use experience (see Mazé and Redström 2005). Digital design utilizes the properties of digital materials – building concrete abstractions of processes that fit use activities at the physical as well as on the symbolic level. The discursive-material nature of digital design changes the world in a material as well as a discursive sense. ‘Computer programs are unlike any other material, and the form of craftsmanship in software will surely be unique’ says Wroblewski (1991: 17). I agree with him that ‘[f]undamentally, the materials shape the craft’ (p. 17): digital design is profoundly shaped by the characteristics of the digital (Bratteteig 2004).

Digital design opens up for new possibilities and for things that embody these possibilities. The materials and tools we use in design influence which possibilities we see and choose to realize. Design is thinking with materials, and the discursive-material digital material brings the head and hand even closer to each other. Seeing digital design as thinking with concrete abstractions of processes, at different levels of concretizations as well as across them, suggests that digital designers should understand their material in a way that enable them to move between levels of concretization and choose the right abstraction for the actual design process as it evolves in time. The many levels of digital design open up for many different competencies being involved in imagining and building possible futures.

Acknowledgments
Thanks to Christina Mörtberg and Erik Stolterman for long-lasting discussions about the material side of computing. Thanks also to Gisle Hannemyr for pointing out the distinction between digital and electronic.

References

Andersen, N.E., Kensing, F., Lassen, M., Lundin, J., Mathiassen, L., Munk-Madsen, A., & Sørgaard, P. (1990). *Professional systems development – Experiences, ideas, and action.* Upper Saddle River: Prentice-Hall.

Andersen, P.B. (1986). Semiotics and informatics: computers as media. In P. Ingwersen et al (Eds.), *Information technology and information use. Towards a unified view of information and information technology* (pp. 64–97). London: Taylor Graham.

Andersen, P.B., & Bratteteig, T. (Eds.). (1989). *Computers and language at work. The relevance of language and language use in development and use of computer systems.* The SYDPOL Programme, Department of Informatics, University of Oslo.

Barad, K. (2003). Posthumanist performativity. Toward an understanding of how matter comes to matter. *Signs: Journal of Women in Culture and Society*, 28(2), 801–831.

Barad, K. (2007). *Meeting the universe halfway: Quantum physics and the entanglement of matter and meaning*. Durhamn & London: Duke University Press.

Becker, H. (2000). The etiquette of improvisation. *Mind, Culture, and Activity*, 7(3), 171–176.

Berg, M. (1997). On distribution, drift and the electronic medical record: some tools for a sociology of the formal. In J. A. Hughes, W. Prinz, T. Rodden & K. Schmidt (Eds.), *Proceedings of the fifth conference on European Conference on Computer-Supported Cooperative Work (ECSCW'97)* Lancaster, UK (pp. 141–156). Dordrecht: Kluwer Academic.

Bjerknes, G., & Bratteteig, T. (1987b). Perspectives on description tools and techniques in system development. In P. Docherty, K. Fuchs-Kittowski, P. Kolm & L. Mathiassen (Eds.), *System design for human development and productivity: Participation and beyond* (pp. 319–330). Amsterdam: North-Holland.

Bjerknes, G., & Bratteteig, T. (1988a). The memoirs of two survivors – or evaluation of a computer system for cooperative work. In I. Greif (Ed.), *Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, Portland, Oregon, USA (pp. 167–177). New York: ACM.

Blevis, E., Lim, Y. K., & Stolterman, E. (2006.). Regarding software as a material of design. Design research society In *Proceedings of Wonderground – the 2006 Design Research Society International Conference,* Lisbon, Portugal.

Bowker, G., & Star, S.L. (1999). *Sorting things out: Classification and its consequences.* Cambridge: MIT Press.

Brand, S. (1994). *How buildings learn: What happens after they're built.* New York: Penguin.

Bratteteig, T. (2002). Bringing Gender Issues to Technology Design. In Floyd, C., Kelkar, G., Kramarae, C., Limpangog, C. & Klein-Franke, S. (Eds.), *Feminist challenges in the information age.* Opladen: Verlag Leske + Budrich.

Bratteteig, T. (2004). *Making change. Dealing with relations between design and use*. Diss. Oslo: Department of Informatics, Faculty of Mathematics and Natural Sciences, University of Oslo.

Capjon, J. (2004). *Trial-and-error-based innovation: Catalysing shared engagement in design,* Diss.Oslo: Oslo School of Architecture and Design.

Checkland, P. (1981). *Systems thinking, systems practice.* Chichester: John Wiley & Sons.

Ciborra, C. (2002). *The labyrinths of information: Challenging the wisdom of systems.* Oxford: Oxford University Press.

Dijkstra, E.W. (1976). *A discipline of programming.* Englewood Cliffs: Prentice Hall.

Dourish, P. (2001). *Where the action is: The foundations of embodied interaction.* Cambridge: MIT Press.

Dourish, P., & Button, G. (1998). Technomethodology: Paradoxes and Possibilities. In M. J. Tauber (Ed.), *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground (CHI '96)*, Vancouver, British Columbia, Canada (pp. 19–26). New York: ACM press.

Fjuk, A., Kaasbøll, J., & Groven, A.-K. (2006). Improvements of teaching and tools for learning object-orientation. In A. Fjuk, A. Karahasanovic, & J. J. Kaasbøll (Eds.), *Comprehensive object-oriented learning: The learner's perspective* (pp. 205–220). Santa Rosa: Informing Science Press.

Fujimura, J. (1996). *Crafting science. A sociohistory of the quest for the genetics of cancer*. Cambridge: Harvard University Press.

Glette, K. (2009): personal communication, Dept. of Informatics, University of Oslo.

Gelernter, D. (1998). *Machine beauty. Elegance and the heart of technology.* New York: Basic Books.

Goodwin, C. (1997). The blackness of black: Color categories as situated practice, resnick. In L. B. Resnick, R. Säljö, C. Pontecorvo & B. Burge (Eds.), *Discourse, tools and reasoning; essays on situated cognition* (pp. 111–140). Berlin: Springer.

Gramazio, F., & Kohler, M. (2007). *Digital materiality*, talk at Oslo Arkitektforening. October 18. 2007.

Greenfield, A. (2006). *Everyware: The dawning age of ubiquitous computing.* Berkeley: New Riders Publ.

Grinter, R.E. (1995). Using a configuration management tool to coordinate software development. In N. Comstock & C. Ellis (Eds.), *Proceedings of conference on organizational computing systems (COCS '95)* (pp. 168–177). New York: ACM press.

Grinter, R. E. (1998). Recomposition: putting it all back together again. In I. Greif (Ed.), *Proceedings of the 1998 ACM conference on computer supported cooperative work (CSCW '98)* Seattle Washington, USA (pp. 393–403). New York: ACM.

Hailperin, M., Kaiser, B., & Knight, K. (1999). *Concrete abstractions.* Pacific Grove, CA: Brooks/Cole Publ. Co.

Hallnäs, L., & Redström, J. (2006). *Interaction design: foundations, experiments*. Borås: The Interactive Institute.

Harper, D. (1987). *Working knowledge: Skill and community in a small shop.* Berkeley: University of California Press.

Henderson, K. (1999). *On line and on paper.* Cambridge: MIT Press.

Hutchins, E. (1995). *Cognition in the wild.* Cambridge, MA: MIT Press.

Hård, M. (1994). Technology as practice: local and global closure processes in diesel-engine design. *Social Studies of Science*, 24(2), 549–85.

Ishii, H., Mazalek, A., & Lee, J. (2001). Bottles as a minimal interface to access digital information. In J. Jacko & A Sears (Eds.), *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '01),* Seattle, Washington (pp. 187–188). New York: ACM pres.

Ishii, H., & Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In S. Pemberton (Ed.), *Proceedings of the SIGCHI conference on Human factors in computing systems* (*CHI '97),* Georgia, United States (pp. 234–241). New York: ACM press.

Jacucci, G., & Wagner, I. (2007). Performative roles of materiality for collective creativity. In B. Schneiderman, G. Fischer, E. Giaccardi & M. Eisenberg (Eds.), *Proceedings of the 6th ACM SIGCHI conference on Creativity & Cognition (C&C '07),* Washington, DC, USA (pp. 73–82). New York: ACM press.

Juul-Nielsen, J. (1984). Personal communication at Risør Trebåtbyggeri Wooden Boat Building, Norway.

Kernighan, B. W. & Pike, R. (1999). *The practice of programming. Simplicity, clarity, generality*, Reading. Massachuesetts: Addison-Wesley.

Knuth, D. (1973). *The art of computer programming vol. 3: Sorting and searching*, Reading. Massachusetts: Addison-Wesley.

Kramer, J. (2007). Is abstraction the key to computing? *Communications of ACM,* 50(4), 37–42.

Lanzara, G.F. (1983). The design process: Frames, metaphors and games. In U. Briefs, C. Ciborra & L. Schneider (Eds.), *Systems design for, with and by the user* (pp. 29–40) Amsterdam: North-Holland.

Larssen, A.T., Loke, L., Robertson, T., & Edwards, J. (2004). Understanding movement as input for interaction – A study of two Eyetoy™ games. In *Proeedings of OZCHI 2004* (1–10). Available at: http://www.ozchi.org/proceedings/2004/index.html.

Latour, B. (1999). *Pandora's hope: Essays on the reality of science studies.* Cambridge, MA: Harvard University Press.

Laumann, K. (2005). *Men er det kreativt? Digitale verktøy i kreative prosesser* (In Norwegian: But is it creative? Digital tools in creative processes) MA thesis. Oslo:Department of Informatics, University of Oslo.

Loke, L., Larssen, A. T., Robertson, T., & Edwards, J. (2007). Understanding movement for interaction design: Frameworks and approaches. *Personal and Ubiquitous Computing*, 11(8), 691–701.

Löwgren, J., & Stolterman, E. (1998). *Design av informationsteknik – materialet utan egenskaper* [In Swedish]. Lund: Studentlitteratur.

Maeda, J. (2000). *Maeda@media.* London: Thomas & Hudson.

Mazé, R., & Redström, J. (2005). Form and the computational object. *Digital Creativity*, 16(1), 7–18.

McCullough, M. (1998). *Abstracting craft. The practiced digital hand.* Cambridge: MIT Press.

Minsky, M. (1967). *Computation: Finite and infinite machines.* Englewood Cliffs: Prentice-Hall.

Moggridge, B. (2007). *Designing interactions.* Cambridge: MIT Press.

Musil, R. (1996). *The man without qualities.* New York: Vintage books.

Mörtberg, C. (2001). Abstracting, quantifying, classifying, simplyfying, standardising, building hierachies: What are the systems designers sorting out? The conference *Information Technology, Transnational Democracy and Gender.* Ronneby, Sweden.

Nygaard, K. (1986). Program Development as a Social Activity. In *Information processing 86: Proceedings of the IFIP 10th World Computer Congress*, Dublin, Ireland (pp. 189–198). Amsterdam: North-Holland.

Nygaard, K. (2002). Foreword. In C. Ciborra, *The labyrinths of information: Challenging the wisdom of systems.* Oxford: Oxford University Press.

Schön, D. (1983): *The reflective practitioner. How professionals think in action*. New York: Basic Books

Sevaldson, B. (2005). *Developing digital design techniques: Investigations on creative design computing*. Diss. Oslo: Oslo School of Architecture and Design.

Star, S.L. (1991). Invisible work and silenced dialogues in representing knowledge. In I. Eriksson, B.A. Kitchenham, & K. Tijdens (Eds.), *Women, work and computerization: understanding and overcoming bias in work and education* (pp. 81–92). Amsterdam: North Holland.

Stolterman, E. (2006). Personal communication.

Stolterman, E. (2007). Designtänkande (In Swedish: Design Thinking). In Harvard, Åsa & Ilstedt, Sara. (Eds.), *Under ytan: en antologi om designforskning [In Swedish: Under the surface: An anthology on design research]* (pp. 12–19). Stockholm: Raster Förlag.

Suchman, L.A. (1994). Do categories have politics? The language/action perspective reconsidered. *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, 2(3), 177–190.

Suchman, L. A. (2007). *Human-machine reconfigurations. Plans and situated actions.* Cambridge: Cambridge University Press.

Säljö, R. (2000). *Lärande i praktiken. Ett sociokulturellt perspektiv* [In Swedish: Learning in practice. A socio-cultural perspective]. Stockholm: Prisma Studentlitteratur.

Vallgårda, A., & Redström, J. (2007). Computational composites. In M. B. Rosson & D: Gillmore (Eds.), *Proceedings of the SIGCHI conference on human factors in computing systems (CHI '07),* San Jose CA, USA (pp. 513–522). New York: ACM press

Webster (2008). *Webster dictionary and thesaurus online*: www.webster.com. Accessed June 2008.

Weiser, M., & Brown, J. S. (1997). The coming age of calm technology, In P.J. Denning, & R. M. Metcalfe (Eds.), *Beyond calculation: The next fifty years of computing* (pp. 75–86). New York: Springer-Verlag.

Weizenbaum, J. (1976). *Computer power and human reason. From judgement to calculation.* San Francisco, CA: W. H. Freeman.

Whalley, P., & Barley, S. R. (1997). Technical work in the division of labor: Stalking the Wily anomaly. In S. R. Barley, & J. Orr (Eds.), *Between craft and science: Technical work in the U.S. settings* (pp. 23–52). Ithaca: Cornell University Press.

Winograd, T. (1979). Beyond programming languages. *Communications of the ACM*, 22(7), 391–401.

Winograd, T., & Flores, F. (1986). *Understanding computers and cognition.* Norwood: Ablex.

Wroblewski, D.A. (1991). The construction of human–computer interfaces considered as a craft. In J. Karat (Ed.), *Taking software design seriously* (pp. 1–17). Boston: Academic Press.

Zuboff, S. (1989). *In the age of the smart machine: The future of work and power.* New York: Basic Books.