KNOSYS1336

Knowledge-Based Systems 00 (2001) 000–000

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

Human–computer interaction viewed as pseudo-communication

Frieder Nake*, Susanne Grabowski

FB 3, Universität Bremen, Postfach 330440, D-28334 Bremen, Germany

Abstract

Semiotics is considered fundamental to an understanding of human–computer interaction, and of all computer artifacts. Informatics should therefore be viewed as *technical semiotics* (or semiotics engineering). In particular, interaction between human and computer is characterized by features of communication, a sort of communication, however, that lacks decisive communicative features. It must be identified as a process of pseudo-communication. Interaction is viewed as the coupling of two autonomous processes: a sign process (carried out by the human user) and a signal process (carried out by the computer). Software appears as a semiotic entity in a duplicate way: *calculated and calculating*, i.e. both result and agent of calculations. This dialectics characterizes the class of signs on the computer medium. Problems of software design (functionality and usability design) are specific problems of the coupling of sign and signal processes. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Theory of informatics; Semiotics engineering (technical semiotics); Foundations of human–computer interaction

Only when a phenomenon has become common and ubiquitous practice, we can hope to understand it. The interactive use of software is almost exclusively the mode of using a computer. It seems as if we were now at the edge of understanding the phenomenon of interactivity. In his recent doctoral dissertation, Dag Svanæs [1] raises this issue and discusses seven paradigms of trying to understand it. One of these is the semiotic paradigm, most convincingly propagated by Refs. [2,3]. One should, however, not forget the works in Refs. [4–8].

In a 1994 paper, one of us has suggested a view of human–computer interaction that may be considered an answer to viewing humans and computers as partners of a more or less symmetrical kind. Such a view is based on the notorious 'physical symbol systems hypothesis', which was put forward so vigorously by Allen Newell and Herbert A. Simon in the 1960s and later, and which was favored by some participants of the artificial intelligence movement because it claimed that humans and computers were merely two cases of information processing systems. Even if it seems ridiculous to many of us to draw a parallel between computers and humans, there must be something in their interactive exchange that made artificial intelligence enthusiasts believe in their strong claims. We will try to identify that 'something'. It must possess the contradicting properties of being equal, and at the same time different, to humans and computers. What could that be?

We will formulate a general and, we believe, powerful position from which we try to understand software in function and use. It relies on the concept of a sign, and allows us to treat functionality and usability within one framework. To this end, we study the dual nature of algorithmic signs and identify the interaction of humans and computers as pseudo-communication, i.e. as a relation that shares certain aspects with communicative action, but in essence is no communication. In the conclusion, we briefly consider how such an understanding of interaction may influence software design.

1. Fundamental propositions

Over a period of 30 years, the work of our research group has evolved out of the assumption that the study of informatics[1] should be based on the concept of a sign, and that therefore semiotics should be equally important as mathematics for an informatics fundamentum. The following propositions identify our position. Today, some of them may appear as almost self-evident. This has not always been the case.[2]

* Corresponding author. Tel.: +49-421-218-3525/2576; fax: +49-421-218-4867.

E-mail addresses: nake@informatik.uni-bremen.de (F. Nake), susi@informatik.uni-bremen.de (S. Grabowski).

[1] The term 'informatics' should be taken as synonymous to 'computer science'.

[2] Despite a language problem with publications in German, we include references that treat these topics [15–23].

0950-7051/01/$ - see front matter
PII: S0950-7051(01)00140-X

- The ultimate goal of any software development is the *mechanization*[3] of some specific kind *of mental labor*.
- In order to achieve this in a given case, a threefold reduction[4] of entities and processes of the outside world is required in order to get them 'into' the computer:[5]
  first, a *semiotic reduction* transforms the chosen aspect of our environment into the semiotic universe (descriptions replace things),
  second, the *syntactic reduction* strips this description of all its connotations and denotations (only form aspects remain),
  third, the *algorithmic reduction* turns into computable form the syntactically reduced description of the environmental aspect under consideration (only data and algorithms count).
- Software possesses an *intrinsically semiotic nature*. It has often been said that software differs from other artifacts by its *immaterial* nature. Interesting as this observation is, it only scratches the surface. Software is semiotic in so far as it is *relational*. All by itself, it is of little interest only. But taken in relation to the activity processes we are involved in as humans, software gets into the focus of our interest. Software is largely relational. Since signs are also not determined as things but as relations, the relational nature of software is the reason for its semiotic essence.
- Signs on the surface, or in the memory, of the computer take on a peculiar state: we call it a state of the calculated *and* calculating sign. A sign is calculated in so far as it is the result of an algorithmic process. It is calculating in so far as it stands for an algorithmic process that can be started and run. Since the substance of a sign is often taken for the sign itself, the surface use of a sign on the computer periphery creates our belief the sign itself was calculating.

The algorithmic sign is the sign that can be manipulated by a computer. It is double-faced: it remains a sign in the full meaning of the word (as given by Ref. [9]); but at the same time, it is a maximally reduced sign, which we call a *signal*. This reduction leaves nothing of the relational character but the mere physical substance of the sign. The algorithmic sign thus is an object of computer manipulation and human interpretation alike. Open interpretation by humans (the sign as sign) and fixed determination by a computer (the sign as signal) together characterize the algorithmic sign in its dual nature. This nature reveals the algorithmic sign as a new category of signs. It becomes (or, rather, should be) the most important object of study in computer semiotics.[6]

- Informatics turns out to be an academic discipline of a new kind: it belongs to both the engineering disciplines and the humanities. Clarisse Sieckenius de Souza and René Jorna independently gave it the name *Semiotic Engineering*. Informatics viewed this way emerges as a first positive postmodern science: positive because it is constructive, postmodern because it deals with media.
- The *three main goals* of informatics — correctness of algorithms, efficiency of programs, and usability of software systems — turn out to be nicely related to the three *semiotic dimensions*: correctness is a matter of syntactics to be answered by considering form aspects only; efficiency is a matter of semantics related to the object world; usability, taking the user's interest and motivation into account, is a matter of pragmatics.
- The semiotic dimensions of syntactics, semantics, and pragmatics are also related to *data*, *information*, and *knowledge*, in turn. Thus we gain a useful differentiation in dealing with the algorithmic sign. In view of societal impact, it becomes clear that we are not manipulating knowledge, nor information, but merely data on the computer. Knowledge itself is, and remains, inaccessible to the computer. But in its reduced form, called data and residing in the syntactic dimension only, knowledge may become the subject matter of software. The issue of Artificial Intelligence turns out to be a non-issue. It vaporizes before we can even define it thoroughly.[7]
- *Human–computer interaction* may semiotically be characterized as the *coupling of two independent*, *yet related*, *processes*: one of these is a full-fledged sign process that humans are involved in. It takes place in concurrency with a restricted signal process inside the computer. (We will talk about these in more detail in a moment.) These two independent processes are coupled. Cultural and interpersonal aspects influence the sign process, which is a process of open, unlimited interpretation. Technical and algorithmic aspects influence the signal process, which is a process of a prescribed determination of meaning without any leeway. The computer as a programmed machine precisely follows predetermined steps in order to establish the described sequence of operations.

Viewed against this background, it becomes clear that semiotics provides new insights to HCI, if only on a

[3] In German, we call this development, 'Maschinisierung von Kopfarbeit'. There does not appear to be an appropriate translation of 'Maschinisierung'. Literally, it would be something like 'machinization' which sounds awkward. Therefore, we prefer 'mechanization' although mechanization is only one historic form of machinization.

[4] More generally, it is a transformation. Only from some perspectives it appears as a reduction.

[5] Think, for a moment, of that formulation: "to get something into the computer". Whatever the corporeal nature of that thing or process may be, taken as such it could not possibly be put into the computer. The thing or process must first be transformed into something different, something that stands for the first and refers to it, but is distinct from it. Only signs have such a property and are therefore entities that may get 'into' the computer.

[6] For pioneering work, see Ref. [2].

[7] This may explain why artificial intelligence research so often ended in good software products or practice despite the exaggerated claims by its activists.

descriptive level. We are convinced, however, that HCI — considered as the task of coupling those two processes — opens up to more than a new kind of *description* of the usability problem. Semiotics, in its current state, does not directly offer constructive solutions to problems. Its methodology must be developed further in order to allow for such constructive work. We expect a cross-fertilization: if semiotics offers to informatics powerful means of description, informatics in turn offers to semiotics methods of construction.

Semiotics in any case offers a promising *approach* to deal with issues in a unifying way: the approach of dialectics. In semiotics, we are not dealing with formally defined models that actually *replace* things and processes of the observed world before those things and processes as models become subject matter of scientific research. We are rather concerned here with hermeneutic interpretations of processes by which we influence those processes themselves (feedback). Therefore, dialectics is needed more than logics as a method for drawing conclusions. Such conclusions are not of the type "if *p* then *q*". They are rather of the type "if *a* and *b* are two contradictory driving forces then *c* is likely to emerge".

2. Signs between humans and computers

After having stated a principal position, we move on to considering in more detail the type of exchange that takes place between a human and a computer. Our usual, first idea when thinking of some exchange is the assumption that there must be some *thing* that gets exchanged. Such an idea calls for transport and transfer of some entity from place A to B, with or without effect on the entity itself.

A different picture suggests that there are two separate and autonomous agents. They act more or less independent of each other. Each agent is part of the environment of the other agent. Their independent activity may call for adaptation to the environment in an attempt to *fit*. An influence of one agent on the other, or an exchange between them, is just that: an adaptation to the environment.

Computers obviously function as if they were thinking. On first sight, they appear as if they were interpreting and manipulating signs. This is why they appear as if they were similar in some respect to humans.[8] But it is equally obvious that computers cannot, and will never be able to, *think* in any comprehensive meaning of that the word. In fact, there does not seem to be any good reason to assume that computers could *ever* be able to *interpret* signs except for the most trivial kind of interpretation.[9] The reason is simply that computers are human-made machines whereas we are naturally evolved living creatures. We have bodies and minds, and an innate interest to stay alive [10].

A lot has been written about that fundamental difference, and this is not the place to enter that debate. However, for our view of human–computer interaction, it is decisive to start out from that simple observation of an insurmountable (and trivial) difference of human and computer. We take off from here for a remark that we hope will shed some light on the reason why intelligent people could come to believe that the most precious and distinguishing capability of humans was something a machine could possess, too.

The reason for such a misconception lies in the nature of computational entities. They are semiotic by origin and remain in that state no matter how we perceive of them and how we treat them. As signs they are the entities the brain is working on. But since the computer is also working on them, and very successfully so, it seems justified to assume some kind of equality between brain and computer.

But observe what happens to the sign when it is operated on by the computer. On the computer, the complex sign relation[10] is reduced to the material substrate of the sign. Some would say: the sign is reduced to its carrier. It is not the *meaning* that we type into our file when we produce a text; the meaning is rather what we have in mind, what we try to express in words. Of all that only that series of coded signals (as in ASCII, e.g.) enters the computer that correspond to the typed-in characters.

The reduction of a sign to ist material substrate is essential in two ways: it is responsible for the fundamental difference between brain work and computer operation; at the same time it is necessary for the computer, as a machine, to do anything meaningful at all. In the social and individual contexts of human activity, a sign constitutes a pragmatic relation between some absent entity (the *object*), ist interpreted meaning for us (the *interpretant*), and the perceivable, corporeal entity (the *representamen*). This relation is established only when a living human is present and through his or her activity, consciously or not, creates it here and now.

It is decisive for the interaction of human and computer that, of the components and subrelations of the sign, only the syntactical component (the representamen) can be the subject matter of a computer program. Hitting a key on the keyboard results in precisely this: whereas the human typist is occupied with all the intentions and interests he or she is pursuing with the emerging text, the computer receives only signal chains that get processed under control of the software. The goal of software development is, of

[8] The mathematician Felix Hausdorff has once characterized the human as the *semiotic animal*. Mihai Nadin has called the computer the *semiotic machine*.

[9] It is unfortunate that in computer science the term 'interpreter' is used in a different way: an interpreter there is a program that reads another program and makes the computer carry out precisely those operations that are supposed to be done according to the definition of the programming language.

[10] We take the sign, in Charles S. Peirce's terms, as a triadic relation. A representamen stands for an object by virtue of an interpretant. The representamen is the material substrate, the object is what the sign signifies, the interpretant — itself a sign — catches ist meaning.

course, to efficiently and effectively create meaningful and correct output sequences from the input. This transformation is possible only if the program is a description of meaningful, correct, and unambiguous transformations. Each such transformation has to be a computable function, or otherwise the trick would not work. The computable function, however, is no device to evoke free interpretations of the kind a living being could do. It is rather a mechanical description with a precise and unique meaning not allowing any uncertainty when obeying its steps.

The remaining material reductions function deterministically. Upon input, they have lost virtually all of their semiotic (relational) character. A bit of that is kept only in so far as the material substrate[11] causes certain events to take place during the emerging computational process. In fact, what we called a reduction of the original sign, is the emergence of a *new* sign of s special kind. This new sign usually keeps the representamen (there is usually a physical, one-to-one transformation of it). But it now signifies some operation on the machine, and this operation — the sign's *object* in Peirce's terms — tends to coincide with its interpretant.[12] We call that interpretant the *causal interpretant*, in order to distinguish it from the original interpretant of human origin, the *intentional interpretant*.[13]

The new sign type, characterized by a causal interpretant, we call a *signal*. The meaning, i.e. the interpretant, of a signal cannot be debated. It is the same as its object. In signals, object and interpretant coincide. Therefore, the process of creating meaning (which is the process of interpretation) is almost trivial. It does not allow for interpretation of an open, insecure, debatable, changing kind: it is determined and fixed. But it is the experience of freedom that accounts for the essence of interpretation.

Our main hypothesis now becomes

> software, as an increasingly important element of many cultural processes, can only be understood, and therefore well designed, if we view it as a process of complex signs; software constitutes a new type of sign, the algorithmic sign, which is characterized by two interpretants, the intentional and the causal interpretant.

This hypothesis nicely connects to the characterization of the computer as a semiotic machine. Mihai Nadin is to be credited for first using this attribution in his paper on a question of interaction design.[14] So an analysis of the *use* situation helps to bring out some peculiarities of the computer. Studying the computer's *functionality* alone leaves it in the state of a machine as any other means that we use to transform something. But the computer can no longer be taken as an isolated instrument. We have learned to analyze and design software as subsystems of an encompassing human–computer system. The situation software artifacts get developed for is one of function-in-use and of reflection-in-action (Donald Schön's term). Therefore interaction between human users and their machines is to the heart of complex software design. The interactive use is not a component that could be added after all other components have been designed and constructed. This integrated view of software design leads to the detection of the semiotic machine. Functionality and use of software find a common ground in their sign character.

There is another characterization of the computer that connects to the semiotic tradition even if, at first sight, it appears as somewhat more distant. In a number of papers we have outlined the notion of the computer as an *instrumental medium*. This research culminated in a doctoral dissertation [11]. Both aspects, the instrumental use and the medial effect of software and computers, have existed for long or always. They have governed the technical and theoretical development of computer artifacts. But the media aspect has recently gained much attention, and is becoming even more prominent.

Questions to be raised include the following:

- What happens to a sign when it is manipulated by the semiotic machine?
- How should the study of informatics, and of interaction with digital media, be based on semiotics?
- How is it possible that we can rely on the nonsense computer in sense-making processes?

Semiotics offers a powerful foundation to all software activities (design, usage, maintenance, impact). This foundation is, however, descriptive by nature. It is helpful because it unifies and organizes disparate subareas of software practice and theory.

Interaction is process. At first sight, it appears as a sequence of interwoven simple (or not quite so simple) operations taken in turn by two systems, the human and the computer. Signs leave the human and enter the computer, instantaneously becoming signals. Inside the machine, signals get processed algorithmically until the results of the processing re-appear on the surface of the computer. Immediately, the human transforms those signals back into signs. He or she cannot but read those signals as signs. An immediate and unavoidable embedding into contexts and purposes takes place.

Given current technological standards, the human user does not have much time to forget his or her involvement in thinking about something that finds an expression in the signs he or she produces. And even if there is some delay on behalf of the computer, the human thinks and interprets the processes in terms of subject matter, interest, purpose, and

[11] The substrate on the computer is of the type of a electro-magnetic field.

[12] The sign, in Peircean semiotics, is a triple of a representamen standing for an object by virtue of an interpretant. The pair (representamen, object) denotes the signifying function of the sign; the triple (representamen, object, interpretant) denotes the meaning.

[13] Peter Bøgh Andersen takes the credit for developing these terms together with one of the authors.

[14] Wolfgang Coy has again introduced the term.

Table 1
Signs and signals compared

Signal	Sign
Engineering	Semiotics
Hard	Soft
Cause	Sense
Computation	Interpretation
Quantity	Quality
Syntactics	Pragmatics
Well-defined	Wicked
One-sided	Many-faceted

activity. He or she is embedded in a permanent sign process, the web of semiosis.

Designing processes of interaction is designing special kinds of semioses. There can be no prospective theory of the type that mathematical theory uses to be for the natural and engineering sciences. The new approach centers around model building. Semiotic models have one new property: they can be run. The sign on the computer is no longer a passive entity that gets produced and shown around. It is a sign in a state of flux, in self-application, seemingly autonomous and producing signs itself. De Souza has beautifully worked out this aspect.

But people still think, when they design, in terms of products. It seems to be much harder to design for fluid processes with little control than for finished products with a definite appearance. The theory that is needed is a *theory of design*; it will be *design as theory* at the same time. This amounts to designing contexts — a task that is possible in semiotic terms only. Table 1 compares the signal and the sign proper.

3. Pseudo-communication

Peter Wegner, in an exciting paper [12], has told us what the essence of the paradigm shift is that computer science authors so often mention, but hardly ever really deal with. The shift is from the Turing concept of computability to a media concept of interaction.

The title of Wegner's paper, *Why interaction is more powerful than algorithms*, characterizes the situation. Not that computation, algorithms, or computable functions could ever disappear from the computer. They are here to stay as long as we use computers. Whenever a lasting shift in paradigm occurs in a science, the old paradigm stays on keeping some of its explanatory power at least for a while. But context and environment of embedding systems change. The system in its new context is, of course, a new system. In constructivist terms there is nothing like a fixed system. Whenever we think of a system, we also think of its environment. System-and-environment is the concept we observe and define.[15] It appears to us as if a new context produces something totally new. In reality, a dialectical jump happens when we switch to a broader context. Algorithms and programs remain what they were as physical entities. But in the semiotic dimension they change when we take their interactive use into account.

In our case of human use of computer signals, it no longer suffices for the successful design of useful and usable computer artifacts to merely construct computing systems. The notion of system itself has to be revised, and a bold step from the closed technical system of a piece of software to the open socio-technical system of human-and-software has to be done: from computer-as-artifact to human-and-computer-as-system.

The openness of such systems rests with the human's choice, at any point in time, to do something unexpected. It is the human condition to be able (and forced) to unceasingly interpret observations, and to draw conclusions from observations. Signals in the environment get picked up, get turned into signs which then, by interpretation, get transformed into actions.

Considered in relative isolation within a general environment, a human and a software system ('computer') may be observed as if two autonomous and independent systems behaved in some concurrent and coupled way. The coupling is established by signs-turned-into-signals. The coupling is more or less tight depending on further circumstances. Since the coupling is semiotic by nature, it does not come as a surprise that people speak of 'communication' between the two systems. They are right and wrong at the same time.

They are right because we observe a behavior that is quite similar to the behavior of humans in communicative situations. They are wrong because the human's actions in pursuing his or her interest, and the computer's operations in following its routines are so fundamentally different that it appears silly to think of any kind of similarity of the two subsystems in that *pseudo-communicative* situation.

It would make sense to describe the interactive use of software by a term like *pseudo-communicative exchange*. Human users of software exchange something with the computer. What they exchange is not matter nor energy but signs. Therefore, semiotics enters the scene. The semiotic analysis, however, shows that something strange is happening during the act of exchange: the exchanged object changes its character twice, from sign to signal and back to sign. Observed from an external display surface, we cannot notice this switch. But internally, exactly this is happening.

At second sight, we observe even more radically that nothing is really *exchanged*. The sign aspect totally belongs to the human whereas the signal aspect belongs to the computer. The two processes take place separately. Their coupling becomes manifest when we consider the input activity of the human. He or she hits keys, moves the mouse, presses buttons. Immediately, these primitive operations are transformed into codes. The codes are the signals we talked about. The input sequence is the closest we can

[15] The definition is needed when we try to express our observations.

get to some *exchange*. Not so with the output sequence. It is wide open to interpretation.

The secret of human–computer communication is that it is an exchange, and it touches upon signs, but it is no communication. It is happening anyways. On the surface, it looks as if it was communication; underneath, it lacks essential features of communication.

Such a miracle is only possible within the semiotic dimension. We never leave the realm of signs and we always stay grounded in the material world. But through acts of interpretation (human) and determination (computer), the interpretants attributed to a sign change. The causal interpretant a software system injects into a signal is, under most circumstances, totally different from the open intentional interpretant the human brings into the situation.

Changing the interpretant of a triadic sign relation is the typical operation in the course of a sign process. It takes place within no time. This creates the feeling in (some of) us of the computer being similar, equivalent, pseudo-human, or the like. The similarity is one in function at best, one that, semiotically, seems quite clear to explain.

4. Conclusion

We have given the reason for the bewildering habit of humans to attribute human behavior to a computer. The reason is deeply rooted in semiotics. We are now ready to study special software situations. Examples could be the role of digital media in learning environments, or the contradiction of computability and beauty in algorithmic aesthetic processes.

In a learning environment, persons meet in different roles. Some are primarily interested in teaching, others in learning. They meet in a situation that is further characterized by artifacts brought here on purpose and in hope of supporting the learning processes.

If the learning environment contains digital media, learning processes will call for the type of interaction we have studied here. Those who have created the digital medium must have had in mind, as a scenario, learning situations as partially interactive experience for learners. Programmers and designers will most likely work hard to arrange the medium such that some *predictable* and prescribed, controlled learning effects will occur.

We now know that this is virtually impossible. All the programmers can provide is the constant signal aspect of a large (infinite) set of future sign processes loosely coupled with those controlled signal processes. This makes prediction of learning outcomes impossible. But what can be done is the arrangement of likely events.

A designer of a digital medium for a learning environment could decide to create a surprise event somewhere. It would be possible, to a large extent, to make sure that the particular medium behavior would indeed be interpreted as surprise. Nobody could predict, however, what that surprise would mean in terms of learning.

The analysis of human–computer interaction as an (indeterminate) sign process coupled to a (determined) signal process shows that designing for human–computer interaction requires a kind of skepticism. Designers should resist the idea of direct influence, or of predictable behavior. They should instead look upon that open situation as a promising situation, one that helps to bring forward typical human capabilities instead of forcing people to comply with machines. Without entering a difficult and loaded ethical argument, the semiotic analysis shows that designing for interaction should become an activity of humbleness.

5. Uncited references

[13]. [14].

References

[1] D. Svanæs, Understanding interactivity, Steps to a phenomenology of human–computer interaction, Doctor philos avhandling, NTNU Trondheim, 2000.

[2] P.B. Andersen, A Theory of Computer Semiotics, Cambridge University Press, Cambridge, 1990.

[3] C.S. de Souza, The semiotic engineering of user interface languages, Int. J. Man-Mach. Stud. 30 (1993) 753–773.

[4] P. Maranda, Semiotics and computers: the advent of semiotronics?, in: T.A. Sebeok, J. Umiker-Sebeok (Eds.), The Semiotic Web, Mouton, New York, 1988, pp. 507–533.

[5] U.L. Figge, Computersemiotik, Zeit. Semiotik 13 (3/4) (1991) 321–330.

[6] R.J. Jorna, Knowledge Representation and Symbols in the Mind, Stauffenburg, Tübingen, 1990.

[7] M. Nadin, Interface design and evaluation — semiotic implications, in: H.R. Harrison, D. Hix (Eds.), Advances in Human–Computer Interaction, vol. 2, Ablex, Norwood, NJ, 1988, pp. 45–100.

[8] M. Nadin, Interface design: a semiotic paradigm, Semiotica 69 (1988) 269–302.

[9] C.S. Peirce, A syllabus of certain topics of logic. 1903, Contained, in part, in Charles S. Peirce, Collected Papers, 1.180-202, 2.219-225 and other paragraphs, Translation to the German by Helmut Pape under the title, Phänomen und Logik der Zeichen, Frankfurt, Suhrkamp, 1993.

[10] H.L. Dreyfus, S.E. Dreyfus, Mind over Machine, The Free Press, New York, 1986.

[11] H. Schelhowe, Das Medium aus der Maschine, Campus, New York, 1997.

[12] P. Wegner, Why interaction is more powerful than algorithms, Commun. ACM 40 (4) (1997).

[13] P.B. Andersen, M. Nadin, F. Nake, (Eds.), Informatics and Semiotics, Dagstuhl-Seminar-Report 135, IBFI Schloß Dagstuhl, Wadern, 1996.

[14] W. Coy, Brauchen wir eine Theorie der Informatik?, Inf. Spektr. 12 (1989) 256–266.

[15] F. Nake, Ästhetik als Informationsverarbeitung, Springer, New York, 1974.

[16] F. Nake, Informationssysteme als Mittel zur Maschinisierung von Kopfarbeit, in: K. Brunnstein (Ed.), Gesellschaftliche Auswirkungen großer Informationssysteme aus der Sicht verschiedener Disziplinen, Mitteilungen des Inst. f. Informatik, Universität Hamburg, Nr., 46, 1977, pp. 4.3.1–4.3.11.

[17] F. Nake, Informatik und die Maschinisierung von Kopfarbeit, in: W.

Coy, F. Nake, J. Pflüger, A. Rolf, J. Seetzen, D. Siefkes, R. Stransfeld (Eds.), Sichtweisen der Informatik, Vieweg, Braunschweig, Wiesbaden, 1992, pp. 181–201.
[18] F. Nake, Von der Interaktion Über den instrumentalen und medialen Charakter des Computers, in: F. Nake (Ed.), Die erträgliche Leichtigkeit der Zeichen. Ästhetik, Semiotik, Informatik, Agis, Baden-Baden, 1993, pp. 165–189.
[19] Frieder Nake, (Ed.), Zeichen und Gebrauchswert. Beiträge zur Maschinisierung von Kopfarbeit, Universität Bremen, FB Mathematik/Informatik, Bericht Nr. 6/94, 1994a.
[20] F. Nake, Human–computer interaction: signs and signals interfacing, Lang. Des. 2 (1994) 193–205.
[21] F. Nake, Der semiotische Charakter der informatischen Gegenstände, Semiosis Heft 85–90 (1997) 24–35.
[22] F. Nake, Was heißt und zu welchem Ende studiert man Informatik? Ein akademischer Diskursbeitrag nebst Anwendung, in: V. Claus (Ed.), Informatik und Ausbildung, Springer, Berlin, 1998, pp. 1–13.
[23] F. Nake, Work.Computers.Design, Scand. J. Inf. Syst. 10 (1–2) (1998) 53–59.