In C. Floyd, H. Züllighoven, R. Budde, R. Keil-Slawik (Hrsg.): Software Development and Reality Construction. Springer Verlag, Berlin, Heidelberg, etc. 1992, S. 15-27.

Human Questions in Computer Science

Christiane Floyd

In this introduction to our book, my aim is to provide a common platform for the contributions that follow. I will outline the main issues at stake, as I see them, in order to motivate the variety of themes taken up later on, and to show how they are connected. In contrast to subsequent contributions, I will not give comprehensive references to background literature here, but confine myself to bringing out a few seminal publications, which were inspirational to many authors of this book. I will start by commenting on the motto "Software Development and Reality Construction" which was coined as a suggestive phrase to indicate the range of questions relevant to us.

1 Software Development and Reality Construction

We focus on *software*, since we consider it to be pivotal in the intertwining of computer technology and the human world. Through software we tailor computers to meet specific purposes, through software we model mental processes to be simulated on the computer, through software we establish the conditions and constraints for people working with computer based systems. Software is a product with unique attributes, its development calls for new ways of working together that we do not yet fully understand.

Software is tied up with our thinking in a particularly intimate way. We meet fascinating challenges in building formal models and setting up artificial worlds. We struggle to find sophisticated ways for delegating some of our mental faculties to the computer. We come up against our limits in dealing with complexity. We are faced with our own proneness to errors. We see our assumptions, values, and relations to others mirrored in our technical work. We model and make rules for ourselves and for others to follow. Through software we control the computer and, indirectly, strive to control the human context where the computer is used.

Software development is meant here in a very general sense, with no restriction to any particular class of programs or development setting. Some contributions are based on the experiences of researchers working as individuals, others refer to routine production in industry. Most contributions in this book deal with software used by people as part of their work. Such software embodies knowledge from an area of human expertise, and serves to enact information processes on the computer, thereby replacing traditional ways of handling information, and allowing more elaborate processes to be carried out by people with the help of the software.

Thus, even though our focus is on software development, we are also concerned with *software use*, where the computer appears as an artifact in various human contexts. In fact, we consider software development and use to be inherently related, so that one domain cannot be adequately considered without taking the other into account. The term software *development* suggests the *evolutionary nature* of this process, which typically involves cycles of design, implementation, evaluation and revision.

The phrase *reality construction* has been chosen so as to evoke the spirit of recent discussions in the humanities, where it has become a vehicle for focussing attention on *our active role* in constituting what we hold for real. Thus, in applied epistemology, reality construction refers to cognitive processes, in which we bring forth what we perceive and know. In developmental psychology, it relates to the gradual formation of conceptual schemata shaping the cognitive faculties of the growing child. In sociology, this phrase refers to social processes instrumental in shaping and transforming social reality. In the philosophy of science, it concerns processes of intellectual inquiry leading to scientific insight.

The phrase seems provocative to many, as it takes issue with the notion of *reality* dominant in European thinking. The latin root for this term, "res", means "things" or "affairs". It suggests that what is "real" is *given* in terms of things or affairs, which exist "out there" independently from us. This is compatible with the rationalistic tradition in epistemology, which views cognition as *matching* the things or affairs

constituting reality as faithfully as possible in the mind. It is also connected to the basic postulate of modern science stating that the properties of the observer should not enter into what is observed.

For example, a software developer analyzing an organization with a view to proposing a software system to support its information processes, is often encouraged to start from the "real world", conceived in terms of the entities and actions constitutive of the information flow in the existing organization. These are supposed to be "given", while the software developer's task is to analyze, to abstract and to elaborate a correct model that can be manipulated by the computer. While this may be difficult to do, the task itself - *discovering* the correct description - is supposed to be clearly defined and independent of the software developer as an individual. Also, his or her responsibility in carrying out this task is restricted to matching the real world in the model with the greatest possible care.

This picture changes drastically, when we acknowledge our active role in bringing about what we hold for real, which is the key to constructivist thinking. The emphasis now is on the *observer* constituting the way he or she sees reality and *inventing* a suitable description. Thus, the software developer is portrayed as making *choices* in an open situation, where there is more than one possibility. When developing the product software, we make choices in selecting the aspects we consider relevant for modeling, in making available modes of interaction with the computer, in determining the software system's architecture, in the way we use the technical resources for implementing the system. Moreover, we make choices in anticipating how the computer will be embedded in its use context and in creating facilities and constraints for users and other concerned parties. And lastly, we make choices in how we conduct the development process itself.

Only a small part of these choices do we make explicitly, more often they are implied by the course of actions we take and, perhaps, even come about by our lack of awareness for potential alternatives or our unwillingness for coping with them and making conscious choices. Also, our choices are constrained by our interaction with others. When seen in these terms, the task of the software developer clearly involves reference to the individual software developer. Through our choices, we constitute the process of development, the product, and the possibilities for its use. In paying attention to our making choices, there is, at the same time, also an emphasis on our responsibility for seeing possibilities and making choices. Thus, the ethical dimension of our activities is always present and included in the discussion.

In constructivist thinking, the ontological question of what *is* is placed in relation to the epistemological question of what *we can know* in a poignant way. Only what we can know is accessible to us, and it is accessible in those terms in which we know it. The seemingly safe ground of the given reality is seen as built up in processes of our own making.

Constructivist authors vary in what concretely they mean by reality construction and in the degree to which they regard reality construction as primordial. The use of the term *construction*, though established, is also misleading. It seems to suggest an unwanted arbitrariness for individual experience and action, and to deny our embedding in the world around us. However, our individual reality construction is interacting with that of others, building up on those before us and grounded in the endless recursion of human (co-)evolution.

Cognition, then, may be viewed as bringing forth concepts and insights *fitting* our experience and *viable* for obtaining our aims in open situations where we interpret our needs. It is shaped by our perspective and unfolds against a background or meaning horizon coloured by our tradition, our interests and our life experience. The main points of current controversial discussion concern the relation of my own reality construction to yours and that of others, and the interleavement between our scope for reality construction and the so-called objective world of nature shaped by socio-cultural evolution.

In this book, we do not aim to contribute to the ongoing discussion of constructivist thinking. Neither do we wish to single out a particular constructivist position as the proper one. Individual authors clarify how they relate to constructivist thinking, and which shades of meaning associated with this term they value or reject. In fact, several of them are explicitly rooted in other schools of thought ranging from Hermeneutics to Marxism, and accept constructivist thinking only to the extent as it corroborates or enriches their own world-view. However, constructivist thinking is applied here in one way or another to several interleaved domains of interest:

- to the process of *software development* which lends itself to being understood as *design* in constructivist terms,

- to the technical result of software development, the *execution of programs* which may be characterized as *constructed reality*,

- to the social outcome of software development, the *human reality of the use situation* resulting from the application of computer programs in a given context,

- to the *emergence of scientific insight* in computer science and other disciplines dealing with questions of design,

- to various *epistemological approaches* providing insights for understanding software development and use,

- to *social reality* in general, shaped and transformed increasingly by the development and use of computer based systems.

Moreover, the way the whole book is made reflects important elements of constructivist thinking, exemplifying, as it does, the use of key notions such as perspectivity, process-/ product-complementarity and self-organization.

The remaining sections of this introduction serve to elaborate the set of questions concerning software development and computer science that provided the motivation for working on this book.

2 Reality and Human Cognition

Whether or not an "objective reality" exists is unanswerable and, according to some, uninteresting. There is, however, increasing evidence that "objective cognition" of the world is impossible, since human cognition is inherently selective and embedded in the processes of biological and social evolution.

Current arguments in biology, neurophysiology, epistemology and the social sciences suggest a view of human cognition, according to which some of its most important facets are:

- It is profoundly affected by human perception as developed in biological evolution.

- It is geared to human needs arising in situations, and therefore action-oriented and interest-governed.

- It is shaped by the history and experience of the individual, the community and the species.

- It is mediated by the language, methods, procedures and tools we use.

- It leads to deeper insights by merging different perspectives.

Understanding human cognition affects computer science in various ways: it helps us draw the line between aspects of intelligent behaviour that can be modelled in the form of computer programs and the full human cognitive experience; it provides a basis for understanding the cognitive processes arising both in the development and in the use of computer programs; it urges us to think about the potential impact of the computer as a thinking tool in human cognition; and lastly, it leads us to an increasing awareness of what it means to pursue computer science as a scientific endeavour.

3 Cognitive Interest underlying Scientific Endeavour

In recent years, there has been a growing concern about the assumptions and the world-view underlying, in particular, the natural sciences as we know them, and extending from there to a considerable extent into the humanities, the social sciences and into everyday thinking.

The traditional way of thinking in science rests on dichotomies contrasting, for example, man and nature, mind and matter, facts and values. It assumes the existence of an objective reality, which can be studied by an observer without the observer affecting the result of the observation. Its primary concern is to discover truth, all questions related to values and human needs being regarded as outside the realm of scientific inquiry. It emphasizes analytical thinking, experiments and proofs as basic elements of scientific methodology. Scientific interest serves to further the domination of man over nature and over fellow human beings.

In contrast, a new understanding of science is currently gaining ground, which is sometimes characterized as a new paradigm. Motivated by recent developments both in the physical and biological sciences, it suggests new ways of overcoming the traditional dichotomies, and emphasizes the unity of human beings and nature. It embodies an awareness of how the observer constructs reality by the act of observation, how the questions we ask influence the answers we get and how we interpret them. It transcends the reductionistic view of the established paradigm by offering systemic ways of practice and extends the ethos of establishing truth by that of promoting life. And it replaces the quest for domination and control by that for preservation and nurture.

If this new understanding of science becomes accepted as a basis for technological development in our society, it may contribute to facilitating changes which seem urgently needed. Computer science is firmly rooted in the established scientific paradigm, as is evidenced by its theoretical teachings as well as its professional practice. In view of the shortcomings of this approach, it is faced, like many other sciences, with the demand for richer ways of thinking.

4 Computer Science as a Scientific Discipline

To date, computer science has failed to make explicit its underlying assumptions. They can, however, to some extent be inferred from the emergence of computer science in its historical context and from the way in which theory formation is interleaved with practical experience in technological development.

Computer science originated, on the one hand, from the need to carry out complex computations during the Second World War; and, on the other hand, from the invention of a machine, the digital computer, capable of carrying out such computations. As a consequence, it has emerged from the beginning as both "computing" science and "computer" science. That means, it views itself as a formal and an engineering science, relying strongly on the traditional scientific paradigm as outlined above.

Moreover, the computer has quickly become a widely used *metaphor for understanding human cognition* both within and outside of computer science. Equating human beings with computers in important ways is explicit in the claims raised to date by researchers in artificial intelligence. It is also implied by traditional approaches in fields like software engineering, requirements engineering or human-computer interaction, where methods tend to assume a machine-like behaviour on the part of both software developers and users. In view of the increasingly subtle interrelation between people and computer programs, these assumptions need to be re-examined.

An important aspect of computer science is that it deals with *creating reality*: the technical reality of the programs executed on the computer, and the conditions for the human reality which unfolds around the computer in use. Therefore, the conceptual categories "true" and "false" it relies on are not sufficient in themselves. We have to go beyond them by finding categories for expressing *the felicity of our choices*, for distinguishing "more or less suitable" as we proceed in making distinctions and decisions in communicative design processes. This is essential for dealing with quality in software development and use.

The need to relate the technical reality of computing to the human reality of our own thinking and interacting is also reflected in the basic concepts used in computer science. We find specific patterns of conceptual confusion here, which can be traced to pervasive problems:

- The need to clarify both similarities and differences in phenomena pertaining to human beings and computers, as in "intelligence", "information", "communication", or "dialogue"ö

- the need to reflect the complementarity of ongoing processes and their outcome or products as in "design", "error" or "quality";

- the need to differentiate between entities emerging in evolutionary processes and formal artifacts created to meet specific purposes as in "language" or "system".

Common usage, however, tends to equate the different shades of meaning contrasted here. Since our choice of basic concepts strongly affects the claims we make about computer science and its possible achievements, our conceptual confusion has already spread far beyond computer science into everyday thinking with unpredictable effects.

Lastly, it must be noted that, to a considerable extent, computer science creates and modifies its own object of investigation. Computer scientists themselves develop formal models and description techniques for technical systems developed by computer scientists; they also develop ways of thinking about and evaluating the systems thus derived, which in turn lead to new developments.

Thus, while the fundamental assumptions underlying scientific work are questioned only to limited extent by computer scientists, it may be argued that computer science brings about such questions in a particularly challenging manner.

5 Human beings versus computers

The driving force behind computer science was the rapid advance of information technology, accompanied by a public willingness to attribute to the computer far-reaching powers. From the beginning, this development has given rise to questions about the relationship between human beings and computers in terms of their capabilities and their desirable interaction. These questions remain unsettled to this day and have a strong bearing on our thinking and public decision making. They come up in different ways:

- Intellectually: Are human beings in their cognitive faculties similar to computers?

- Technologically: Can computers, in principle, be likened to human beings?

- Morally: How should computers be allowed to interfere with human affairs?

There is no scope here for treating these questions in depth. Yet, I find it indispensable to bring them in the open, since the stand we take on them profoundly affects the issues raised in this book. Also, we need to see them as intertwined, whereever decisions concerning the development and introduction of information technology are made. Our understanding of the relationship between human beings and computers necessarily influences what we think of as desirable ways for the use of computers. Thus, while these questions may remain the topic of interesting academic speculations for some time to come, they are of basic importance in social reality construction here and now, and have a decisive influence in shaping tomorrow's computerized world.

Equating human beings and computers is in line with recurrent attempts in the history of European thinking to use machine models for *understanding human beings*, and is reflected even in the colloquial use of language today. At the same time, it has emerged in a cultural background, where machine metaphors are applied at different levels to *prescribe the desired behaviour* for individuals, groups, or social bodies such as large organizations, and portray predictable routine performance as a mould for individuals to cast themselves onto. Computability has almost become a modern moral category, a vehicle for discussing the viability of decisions for action in human terms.

Equating human beings and computers rests on singling out the human faculties for *rational thinking* and *functional behaviour*, considering them on their own, and abstracting from their connection with other modes of experience. The fundamental assumption here is that human cognitive faculties can be

meaningfully discussed without taking account of our embodied and social nature constituted in the process of co-evolution of all living beings. There are important socio-cultural roots for this idea in the whole of Western civilization, leading up to a historical situation in the past decades, which made the discussion about the relationship between human beings and the newly invented computers, urgent and significant.

The craving for rationality as a basis for conducting human affairs, inherited from Greek philosophy, was formulated into a programme for human progress in the Age of Reason. However, the hope for the fulfilment of this programme was profoundly shaken in the twentieth century. On the personal level, the discoveries of psychoanalysis have confronted us with irrevocable limitations in controlling our own rational behaviour. On the political level, the violent social upheavals and the horrors of warfare and totalitarian regimes have infringed upon the lives of vast numbers of people and shaped the thinking of a whole age. Thus, faith in human rationality has declined, and the computer appeared as a desperately-needed rational authority beyond human passion and error.

Moreover, in an age of fear and international confrontation, the computer provided the basis for a key technology enabling the policy of deterrement. It helped accomplish prestigious space missions and promised the illusion of global protection. While the strive for control is intrinsic in modern science and technology, the computer comes in as a quasi-intelligent and quasi-autonomous agent allowing to carry out control on an un-precedented scale. It can be programmed to handle formerly unimaginable complexity and to function in settings where humans could not survive. Thus, in an age, where the possibility for international understanding seemed forever blocked, the computer was taken as a technological guarantee for safety.

The computer has even acquired a mythical significance for many people lost in a disenchanted world. Beliefs in a future governed by machine-implemented rationality, in beings, originally created by us, but later developing on their own, thrive on myths taken from antique sagas and the ancient religions. They are reflected, in particular, in the roles cast out for mankind in connection with computers. Here, Man no longer sees himself in the image of God, but, on one hand substitutes for God as the creator of intelligent living machines, on the other hand likens himself to Machines taken as models for desirable behaviour. According to some, intelligent machines would even eventually take over, set up constraints for humans and treat them as the inferior beings they supposedly are, forever unable to reach their own perfection.

We might be tempted to relegate such ideas to the realm of science fiction. But we find their trace in scientific papers and official research programmes discussing progress in information technology as desirable in its own right with no reference to human concerns. Thus, while upholding the unshaken belief in scientific and technological progress, the established stand on the relation between people and computers tends to be coloured by a pessimistic view of human affairs, and lends itself to the development of a technology, that is potentially destructive to the future of human life on earth.

Unlimited beliefs in the computer have found their critics early on, but it takes time for their voice to gain ground in scientific discussion and in public decision making. Their seminal work has helped many of us shape our thinking and formulate our own positions. The most articulate critics were Hubert Dreyfus who, in his book "What Computers Can't Do"[1], furnished a profound philosophical critique of the claims raised by Artificial Intelligence, and Joseph Weizenbaum who, in "Computer Power and Human Reason"[2], voiced an urgent moral appeal to the scientific community about its role in developing a potentially destructive technology.

Dreyfus drew on the whole history of philosophy and on arguments taken from biology, psychology and linguistics for pointing out what he saw as unalterable differences between human thinking and the rule-governed symbol manipulation carried out by computer programs. As a result of his detailed analysis he outlined the limits for the potential capabilities of computer programs based on the symbol-manipulation paradigm of traditional Artificial Intelligence. While his argumentation was extremely rich and subtle, his conclusions were met with a mixture of applause and skepticism. In particular, his statement that fundamental limitations of computer-implemented intelligence could, in principle, not be overcome, was rejected by some as pertaining only to the technology we know at present, and being

[1]Dreyfus 79

[2]Weizenbaum 77 ?

speculative with respect to future developments. Later on, Dreyfus, in collaboration with his brother elaborated a practice-oriented account of his view of human expertise geared towards supporting public decision-making about the application of current computers in support of human skills in our society[1].

In contrast, Weizenbaum took no issue with the claims purported by leading researchers about what computers could do, but addressed the moral issues inherent in an unlimited development and use of computing technology. Speaking like a prophet, he outlined in no uncertain terms the possibility of technology-induced doom and drew the attention of scientists and programmers to their role in contributing to the dehumanization of our social lives brought about by computer technology. Starting from his own experience with the program Eliza he had developed, he showed how easy it was for claims equating human beings and computers to gain social reality by people being willing to attribute human-like qualities to computers. On a large scale this would enable mechanisms of surveillance and control to be computer-implemented and even induce us to entrust vital decisions such as the use of weapon systems to computers with unprecedented destructive potential for our whole civilization.

I see these two books as sharing a common perspective on the world of computing, focussing on advanced research in Artificial Intelligence, and expressing messages complementary to one another. While they provide important inspirations for the issues at stake in the present book, their focus is far away from the everyday-world of developing and using computer programs in ordinaty settings.

6 Programming as a Human Activity

Our understanding of the activity of programming influences the way in which we carry it out, the priorities we set and the methodological support we seek. Thus, it also shapes the results we obtain. In computer science, however, the nature of programming as a human activity has so far received little attention.

In traditional programming methodology, the activity of programming is portrayed as solving given problems. Programs are studied as mathematical objects with intricate formal properties, divorced from the human context of their development and use. Methods are seen as rule systems for finding a solution, starting from an abstract specification and matching it by a correct program derived in steps of refinement and transformation. Large scale software development is treated as the production of a set of programs designed to meet fixed requirements, proceeding in a sequence of separable stages.

These views are based on several important assumptions. One is that of an objective reality providing us with well-defined problems to start from. Another is our ability to understand these problems completely in advance, at least in principle, in order to write the specification. A third one is that we can abstract from the specific properties of the technology we use while deriving the correct program. A fourth one is that the cognitive processes involved can be broken up into predefined stages. And lastly that we need to take no notice during development of the human context, where the program is to be used.

Many software developers, however, educated in traditional programming methodology, experience a painful clash between trying to adhere to their teachings and what actually seems sensible to do. Even less are they prepared for the social role they find themselves in. Computer programs emerge as the outcome of complex human processes of cognition, communication and negotiation, which serve to establish the problem to be dealt with and to anticipate the meaningful embedding of the computer system in its intended use context.

Programming as a human activity takes place in diverse settings. Distinct professional traditions have evolved around problem classes typical for certain application areas. Skill in programming is defined in terms of mastering specific languages, methods and technical environments. Programming involves dealing with people in a variety of roles giving information, making demands and setting constraints. It rests on the software developers' ability for inventing relevant ways of using computer technology in the actual situation. It implies constructing formal artifacts to be embedded in the unique application context at hand.

---

[1]Dreyfus, Dreyfus ?

In discussing programming as a human activity, the starting point for several contributions in this book is Peter Naur's view of programming as *theory building*.[1] Theory building here refers to ongoing human processes of increasing our understanding for an area of concern. Theory is what enables us to cope intelligently with questions and problems as they arise. In software development, theory buiding pertains to the global task of finding ways for how computer technology can meaningfully be applied to meet the customer´s demands. Theory building, in this sense, happens as a continuous process. It is enfolded in the totality of activities involved in communicating with the customer, in establishing the requirements, in selecting the technology to be used, and in designing, implementing, using and evaluating the software. Thus, theory building is inherently tied up with the people carrying out software devlopment, be they individuals or teams. Naur draws some radical conclusions. There can be no right method for theory building, as each process unfolds in a unique way. The "life" and "death" of a program depends on the availability of its developers, who alone possess the theory enabling them to make meaningful modifications and enhancements. The role of a software developer is primarily that of a consultant advising the customer.

While Naur provides a stimulating and provocative view of the conceptual and technical aspects of our work as software developers, he says little about its social quality. He leaves open, how we can cooperatively build a shared theory, even less does he account for how we become instrumental in bringing about a transformed social reality for all people affected by computer based systems.

Kristen Nygaard, by contrast, has studied *programming as a social activity*[2]. He sees software development as a cognitive activity shaped by *perspectives*. Perspectives provide view-points, from which we structure the cognitive processes we are involved in. Perspectives, in Nygaard´s sense, make us understand the development situation in social terms (as harmony or conflict between the participants); they stand for conceptual repertoires used as a basis for software development (for example object-oriented programming); and for anticipating the use situation (for example the "systems perspective" vs. the "tool perspective"). The concept of perspective, as elaborated concretely by Nygaard, remains open to criticism. On one hand it subsumes many different phenomena under the notion of perspectivity, on the other hand it takes no account of the implicit perspectivity always present in subjective authenticity. However, the idea of perspectivity, illustrated by him in its many forms, is undoubtedly basic for illuminating design with others and for others.

The views of Naur and Nygaard are related to many efforts reported in this book. They imply a shift of emphasis from regarding software development as problem solving and production to viewing it as *design*. Design relates the human reality where the computer is to be embedded to the technical reality of the emerging computer program. It is a constructive process, carried out by the people involved in a unique way. It is constituted by their perspectivity and their interaction, by their understanding of the development situation and their anticipation of the use reality. In design we make choices. We create worlds for ourselves or for others to inhabit.

7 Computer Programs in the Human World

Traditional computer science does not concern itself with the application of information technology in the human world. This is revealed both in how it views its own scope, and in the approaches admitted into scientific discussion. In particular, the treatment of software development concentrates on development divorced from use. This provides no scienific platform for considering how programs as artifacts can meaningfully be embedded in human activity - although practising computer professionals act as designers for such artifacts.

Thus, the design of computer artifacts tends to be techno-centered: the computer as an artifact is perceived from the developer´s perspective and the users are controlled by computer-implemented rules set up by the developers. Human competence on the part of the users is reduced to their ability for operating programs correctly, everything else being an error, and outside the scope of consideration for designers.

---

[1] Naur 85, see also .?

[2] Nygaard.86 ?

Software developers get little guidance for understanding the *use-situation*, where people are carrying out their work with the help of the compute. Therefore, they have no basis for evaluating whether the results of their design appear felicitous there. An adequate consideration of the embedding of computer programs in the human world does indeed require us to go scientifically beyond the formal and mathematical methods provided for in traditional computer science, and to open ourselves to approaches from the humanities.

The approaches developed there for understanding human learning and creativity, individual and cooperative work, and the interrelation between technology and organizations, provide a starting point for dealing with the problems at stake here. However, these approaches mostly have been developed with no specific concern for computing. Therefore, we face the task of selecting suitable approaches and tailoring them to the needs of our discipline. As the intertwinement between computer technology and the human world takes place in a variety of contexts, elaborating an adequate understanding for it becomes an extremely challenging task.

In studying the connection between development and use of computer artifacts, we need to take the following aspects into account:

- the *social granularity*, i.e. are we interested in individuals working with the computer, in groups communicating or cooperating via the computer, in organizations being transformed by the introduction of computer technoloy or in its effect on even larger social agglomorates?

- the complementarity between *theoretical understanding* and *methodical support* for practice, i.e. are we looking for descriptive or prescriptive approaches, and how can these two be combined to infer a suitable orientation for design from a deeper understanding of the use-situation?

- the *type of human activity to be supported* by computer technology, i.e. how can we understand the interleavement of computerized and other work-steps, what kind of pre-understanding is relevant, what metaphors for the computer can best express human meaning for its use in the context at hand?

These problems can be understood and evaluated quite differently in terms of various philosophical schools. For example, the widely discussed book on "Understanding Computers and Cognition" by Terry Winograd and Fernando Flores[1] focusses on the embedding of computers in organizations which are viewed as networks of conversations. Winograd and Flores analyze the assumptions underlying conventional computer science in terms of the rationalistic tradition in philosophy, whose fundamental influence they demonstrate in how we think of language, of decision making, of human learning and cognition. In order to transcend the limitations of the rational tradition, they incorporate elements from three schools of thought:

- Hermeneutics, in a somewhat adapted version of Heidegger´s philosophy,

- Constructivism, as expressed specifically in the new biology of Maturana,

- Language Philosophy, as found in the speech act theory of Austin and Searle.

By combining these and tailoring them to the problem of design, they elaborate a platform for understanding language processes and the role of the computer artifact in supporting them.

Many authors in the present book take off from Winograd and Flores in one way or another. We share with them the concern for providing an adequate foundation and orientation for design. However, our scope and approach differ from theirs in many ways. The most important difference, in my opinion, is that we do not aim at building up one coherent platform for treating all questions relating to design. Instead, individual contributions deal with different levels of social granularity, focus on certain types of human activity to be supported and bring out the relevance of distinct philosophical schools for illuminating specific aspects. This is a conscious choice on our part, as it serves to show the variety of relevant perspectives, whose elements can be combined in manifold ways for coordinating our understanding of design as needed in different contexts.

---

[1]Winograd, Flores 86

8 Towards a foundation for practice

In closing this introduction, I would like to point out that there *is* no computer science independent from us. Computer science is *what we make it*. Every professional is instrumental in bringing forth computer science. While we are constrained in whatever situation we work, we also have the scope and the option for making choices. Through our choices we shape our own understanding, we set priorities in our scientific work, we produce the technology that reflects our design, and we create conditions for ourselves and others who inhabit a computerized world.

The computer science we know to a large extent still sees itself as a formal and engineering science *only*, and disregards the fundamental human questions raised here. Though the term "Informatics" is widely used in Europe, it does not yet imply a conscious strive towards a more encompassing approach. While the traditional self-understanding may have been appropriate at the time when computer science originated, it does not provide a sufficient basis for viable decisions on developing and using computer technology today. The authors of this book would like to contribute to providing more adequate foundations for practice in science and design.

One key to such deeper understanding is the willingness to reflect our own practice. Therefore, some of us open up and show how we see our own work, our professional role and our personal motivation. We acknowledge the full human reality of our lives as the basis for scientific work.

However, understanding does not take place in individuals alone. It is shaped by our involvement with others, by the mutuality of complementary view-points allowed into discussion and by reaching insights through dealing with differences and conflicts. Also, we tailor our insights so as to meet our needs and express our values. For this reason, this book offers a variety of view-points in contributions by different authors, some of whom make their values explicit, and not all of whom agree. Thus, the reader is invited to form his or her own opinion.

We should not expect definitive answers to our questions here, but learn to raise them together in more relevant ways, as they apply to individual situations. We need this conversational foundation for working together towards designing computer technology with a view to promoting human development.

Acknowledgements

This paper is based to some extent on an earlier version, entitled "Software Development and Reality Construction: An Introduction" and used as a thematic introduction for the conference at Schloß Eringerfeld . I would like to thank my former co-authors Werner Langenheder and Dirk Siefkes for their contribution to the original paper and my co-editors Reinhard Budde, Reinhard Keil-Slawik and Heinz Züllighoven for numerous helpful comments and discussions.