

# 19

## Systems Development Philosophy\*

**Bo Dahlbom**  
**Lars Mathiassen**

**Abstract.** A philosophy book for systems developers is outlined, and it is argued that philosophy can play an important role in the education of professional practitioners. All systems developers acquire a philosophy in their education and practice. This philosophy is expressed in their views on the problems of their profession. But to what extent is this philosophy confronted with their personal practice? To what extent is it open to discussion and subject to change?

We argue that philosophy can play an important role in the education of reflective practitioners and we outline a philosophy book for systems developers. We do not want to make philosophers out of systems developers, but rather to encourage them to philosophize. Practitioners and students are invited to engage in a rational conversation about matters which they in their daily activities take for granted. The goal is to take a few steps in the direction of a richer, deeper, clearer, different, more truthful understanding of the activity of systems development.

### 1. Introduction

Systems developers must master a wide spectrum of methods and technologies related to their profession and they must have the energy and skill to frequently evaluate new trends and to modify and extend their repertoire for action. They must be able to cope with unstructured situations, understanding and appreciating the unique

and specific characteristics of the problematic situations involved in their daily work. They have to work under more or less formal, contractual arrangements with limited resources but at the same time they must be able to interact with imaginative and demanding users and managers in quite turbulent environments.<sup>1</sup>

With this in mind, how can we effectively educate and support students and inexperienced practitioners in becoming reflective and competent practitioners? How do we provide them with the energy and skill to modify and extend their personal tool kits? And how do we develop their attitudes and skills so that they can cope with unstructured situations with both effectiveness and sympathetic insight?

In the following we will outline a philosophy book for systems developers, and we will argue that philosophy can play an important role in the education of reflective practitioners. To philosophize in systems development is to search for knowledge about the nature and meaning of the profession. It is to participate in increasing our knowledge of the inherent challenges of this specific kind of human activity. Philosophical-minded systems developers reflect on their practice and in doing so they develop a personal view on their profession. They do not accept problems at face value and they challenge traditions, concepts and methods. Philosophical-minded systems developers do not become uneasy or nervous when confronted with uncertainty. They develop a calm and open attitude, even in the face of frustration, difficulty and danger.<sup>2</sup>

In section 2 we discuss the relation between philosophy and systems development and we argue in favor of a specific kind of philosophy book for our field. In section 3 we discuss alternative ways to design a philosophical textbook for systems developers, and in section 4 we describe the contents, structure and style of a specific proposal for such a book. Finally, section 5 contains our concluding remarks.

## **2. Philosophy and systems development**

### **2.1. What kind of philosophy**

The kind of philosophy book we have in mind is not primarily concerned with philosophers. People like Plato, Aristotle, Hegel, Marx, Heidegger, or Wittgenstein will play only minor roles. We are not writing a book on philosophy as such but on systems development.

Terms like realism, epistemology, Dasein, phenomenology or intentionality will not intimidate our readers. Instead, we want to invite them to reflect upon the activity of systems development, to participate in a discussion of the question “What is systems development?” Such a discussion can, to a large extent, be pursued in the terminology of the field itself.

Philosophy, like any other social activity, has its theories, concepts, and methods. But if your interest lies in philosophizing rather than in philosophy, you need surprisingly little of the technical jargon philosophers use to express their theories, concepts and methods. You can use those methods, learn from those theories, make the relevant distinctions, without having to introduce the jargon itself. Moreover, it makes good sense to avoid too much philosophical jargon. For one thing, it saves the reader from having to make an unnecessary detour through overcoming, comprehending and memorizing a strange and unfamiliar terminology that never will get a clear meaning. The use of the philosophical jargon is a practice that needs time to sink in—years rather than weeks. More important is the fact that the philosophical jargon is loaded with connotations that draw the attention away from the issue here at stake—the profession of systems development—to deep troubles about the nature of man, his situation in the universe, the possibilities of knowledge, the nature of language, the good and right, etc. Our intention is not to make philosophers out of systems developers, but rather to support inexperienced practitioners and students in becoming more reflective.

A book in systems development philosophy is an invitation to thinking. It is an invitation to a rational conversation with arguments, counter-arguments, definitions, analyses, conclusions; an invitation to a discussion of presuppositions and assumptions that we in our day to day work take for granted. The goal of such a discussion is always to take a few steps in the direction of a richer, deeper, clearer, different, more truthful understanding of the activity of systems development.

The model for such a book is a combination of Plato's dialogues and Aristotle's lectures. In Plato's dialogues we are thrown head first into conversations about the nature of the good, knowledge, justice, virtue, courage, writing, work, the soul, the state, etc. We never get any final answers. Instead what we learn from the dia-

logues we learn by participating in the unfolding of the arguments, by entering into the discussion.

Aristotle's lectures are more systematic or differently systematized. Still, everything he says he is ready to question. He interrupts himself to consider imagined objections from the reader, apologizes for weaknesses in the arguments, for lack of knowledge, etc. In spite of being scientific dissertations, Aristotle's books, therefore retain much of the dialogical nature of Plato's works.<sup>3</sup>

Already in Plato's time there were rivaling opinions about what a philosophy book should be like. Plato's most successful opponents were the sophists, who did not believe in truth, reason, argument, who viewed the dialogue as a method of persuasion, reality as a social construct, truth as relative to the purpose, justice as power, etc. For the sophists there were no such thing as progress in philosophy, no process of digging deeper, of revealing the true nature of things. Philosophy was just a conversation, for the fun of it, or for the sake of gaining power and influence, of getting what you wanted.<sup>4</sup>

Now, if we accept the books by Plato and Aristotle as models for our philosophy book, then what is a philosophy book for systems developers? It could be a book that teaches you to philosophize by reflecting on issues specific for this profession. Or it could be a book that teaches you to philosophize by showing you how great philosophers have argued, and exemplifying this in relation to issues relevant to systems development. And it could be both: a book teaching you a method and provoking you to take a stance by showing you the routes<sup>5</sup> already traveled by great philosophers of the past but all the time related to relevant issues in systems development.

## **2.2. Philosophical methods**

A philosophical inquiry typically starts by asking "What is this?" with a very special tone of thoughtfulness. The inquiry can address a particular thing, event or situation, but more often it addresses a general issue. The philosophy of systems development can begin by simply asking "What is systems development?" In order to answer this question, to perform the inquiry, we need a method.

There are a number of such philosophical methods we can use in getting a clearer and deeper understanding of the constituting elements of systems development. These methods are analytical in nature and among the most commonly used are:

- (a) linguistic analysis

- (b) phenomenological analysis
- (c) contextual analysis
- (d) analysis by analogy
- (e) historical analysis

Using method (a) we look at the way we describe systems development, the words and concepts we use, rather than the activity itself. Questions come easy: Does “development” mean that there already is something to be developed? Does it mean development of something that is already in use? What is it then that is already in use? Since we call it “systems development” it must be a system? But what is a system? Obviously, there are lots of systems involved here. There are computer systems, information systems, communication systems, organization systems, etc. Which one is the fundamental system, or is there more than one? When we develop a computer system we are really interested in developing an information system and eventually an organization. But at the same time a major concern is to construct a computer system. How do we differentiate between the different types of systems, and how do we perceive their relations? What does it mean to “develop” a system? To change it. What then is “change”? Instead of developing systems we often speak of “designing systems”. What does it mean to “design a system”? And so on. A linguistic analysis naturally turns into a closer look at the concepts system and development, and an analysis of those concepts brings up other concepts such as structure, process, design, analysis, management, project, product, etc.

Method (b) focuses on systems development as it is performed and experienced as a kind of organizational activity. Here we are concerned with ways in which systems development can be divided into functions, activities or sub-tasks and how the individual tasks relate to each other. We are also concerned with understanding the characteristic differences between sub-tasks and the variety of skills and technologies needed to perform them. Is the activity of systems development well formed in the sense of having clear demarcating lines against other kinds of activity?

In method (c) we look at the larger whole of which systems development is a part. What is the role of systems development in organizational change? What is the role of computers and other kinds of information technology in designing effective organizations? Here we are concerned with the context of which systems development is

a part. We look at the economic, technological and organizational conditions for doing systems development and we inquire into the relations between systems development on the one hand, and decision making, organizational learning, power games, and organizational change on the other. We are also concerned with the relation between development, maintenance and use of computer systems, or, more generally, between the traditions, transformations and transcendences related to the use of computers in organizations.

In method (d) we don't look down into the parts nor into the context of systems development. Instead, we analyze systems development by comparing it to different but similar kinds of human activities. Systems development can, for instance, be viewed as technical construction, as human learning and problem solving, as communication and negotiation, or simply as a political process. What can we learn about systems development by comparing it to these forms of organizational behavior, and what can such comparisons teach us about the different kinds of roles that systems developers engage in?

In method (e) we attempt to understand a practice by studying its history. In this way we come to know how it came to look this way, and we can see clearer what it is today, in all its complexity, by seeing how it has grown out of more simple, less sophisticated, practices. But we won't necessarily arrive at a rationale for its current character unless we believe in history as always striving towards more rational ways of doing things.<sup>6</sup>

Philosophers have always dreamed of a method, the method, which, like the philosophers' stone, would turn our leaden confusion into golden truth. Now we all know that this dream is forlorn. Firstly, one and the same method will produce very different results depending on the metaphysical, ontological and epistemological pre-conceptions of its user. Fundamental notions about change and stability, substance and properties, the way we carve up the world, and ideas of what is knowable and how, will determine the outcome of our philosophical analyses even if we apply the same methods. Secondly, we cannot arrive at a concrete and operational understanding of systems development as part of our practice by merely combining analytical methods. The analytical methods help us to identify and separate the constituting elements of systems development. But to understand systems development in its totality we need to develop a personal synthesis.

There are in the literature on systems development a rich variety of frameworks or perspectives that can support us in our synthetic efforts and with the aid of which we can deepen our reflections. Of particular importance are of course the different versions of the so-called systems approach. No philosophical inquiry on systems development can avoid bringing in, unwittingly or not, the perspectives developed by C. W. Churchman (1968) in *The Systems Approach* and P. Checkland (1981) in *Systems Thinking, Systems Practice*. The systems approach concentrates on the matter of systems. In order to focus on the aspect of development we need to emphasize changes, transformations and transcendences. This can be done with a dialectical approach, using contradictions for instance in the manner of J. Israel (1979) in *The Dialectics of Language and the Language of Dialectics*.

There is no way getting around a discussion of perspectives or frameworks, ontological and epistemological preconceptions, if we want to seriously reflect on the activity of systems development. But there is always the danger that we will end up in an abstract ontological or epistemological discussion with little or no relevance to the business at hand. Practitioners do well to turn their back on such discussion. If the aim is to become more professional in one's practice, questions of framework and choice of perspective will have to be grounded in and subordinated to concrete problems of practice.

One way of ensuring that our reflection does not stray too far from its subject matter is to postpone all worries about frameworks and begin with concrete examples, real worries, and attack them with a naive, ad-hoc approach. We can all do this, without explicitly having to think about methods or perspectives. In a more general fashion, every systems developer ought now and then to stop and consider such questions as: What language do I use? How do I speak about my tasks, my colleagues, my clients, my job? What is said and what is not said? What secrets do I keep? Do I use a jargon to shut out clients, to establish a professional image, or as a genuine means for better communication? How does theory relate to practice? What do I say I do and what do I really do? How do I spend my working day? What is the architecture like? What kind of building, office, do I work in? What does my material environment tell about my work? What does the organization look like? How do we describe it to our clients, and how does it really look? Who is my employer, my client? What are the most important concepts I use? What are my ideals,

dreams? What would I really like to do? Why do I not do it? And so on.

In summary, there is no single way in which to reflect on the practice of systems development. On the contrary, many different approaches can be applied. For a discipline-oriented book in philosophy it is equally important to stress relevant methods of how to philosophize as it is to present specific reflections and arguments related to the discipline. A philosophy book for systems developers should therefore use and present a variety of relevant philosophical methods.

### **2.3. A philosophy book for systems developers**

When philosophy is understood the way we have described it here—as an activity of reflection on a practice—it is not difficult to argue that practitioners generally need to philosophize.

Systems development requires expert skills: it is a practice relying on extensive use of ever changing technologies and its very idea is to participate in and facilitate complex changes in organizations. If systems development is perceived in this way there is no doubt that it is a practice, or set of practices, that needs reflection as an important integrated element. And even if the role of the changing technology is played down, systems development is definitely a practice concerned with people in organizations, and our understanding of people and organizations is still so rudimentary that systems development warrants reflection.

A book that invites the reader to reflect on the practice of systems development takes that practice seriously as an important activity. It aims at reaching fundamental questions, reflecting on general truths, without losing its concrete relevance. It believes that systems developers should practice critical and constructive thinking, that they should see alternatives, argue, and reason. In short, such a book believes that systems developers should be practical philosophers.

Existing books on philosophical issues of systems development typically address researchers rather than practitioners. There are, however, important differences between a philosophy book for researchers and one for practitioners. Simply put, a philosophical examination of an activity will be different depending on what activity it is. Researchers do research on systems development, systems developers develop systems. To do research and to do systems devel-

opment are two very different kinds of activities, even if there are strong similarities. There is, of course, overlap since what researchers do involves philosophizing on the nature of systems development and what systems developers do involves research, i.e. reflection on this very same matter. A philosophy book for researchers will, however, have as its first aim to reach a deeper understanding of the nature of research on systems development rather than of the nature of systems development itself. And even if a philosophy book for practitioners will invite them to philosophize and therefore to develop a research attitude towards their practice, there are lots of things you need to know as a researcher that you don't need in order to do, and to develop a philosophical attitude to, systems development.

We write for present as well as future practitioners. We want the book to be read as an inspiration to present practitioners, and we want it to be used as part of the education of computer and information science students. A book for practitioners must feel right to them. It must have the right tone, identify the practice accurately, and use the right terminology in order to be convincing. Such a book should speak directly to practitioners. It should address the practice in all its concreteness and invite the reader to reflect on that practice. If we cannot describe that practice in such a way that the practitioners can recognize themselves in it, the philosophizing will have little or no effect.

So far we have spoken as if we can take an activity that already has a form, and then go on to philosophize upon it. But the practice of systems development is still so young and our perceptions are so vague and undecided, that reflecting on it will include defining it in new ways. What we want to do is to some extent to define a practice related to something that already exists. Our goal is partly to change a practice by criticizing it, by showing what it really is, by discussing what it could be by philosophizing upon it.

### **3. A discussion of alternatives**

#### **3.1. Structural considerations**

In designing a philosophy book for systems developers there are several alternative options. The philosophical methods outlined above provide us with different ways of structuring the book, depending on what combination of methods we choose as the funda-

mental ones. C. W. Churchman's (1971) *The Design of Inquiring Systems* is designed by combining what we have called analysis by analogy and historical analysis. A similar design of a philosophy book for systems developers is certainly possible. There are in the literature a number of alternative ways of conceiving systems development by analogy. There is Yourdon's idea of systems development as automation of work processes, Jackson's idea of systems development as scientific modeling, Herbert Simon's idea of systems development as problem solving, the Habermasian idea of systems development as rational discourse, the Wittgensteinian idea of systems development as a language game, etc. It should not be too difficult to order these approaches like Chinese boxes, thus arriving at a Churchmanian structure. If, at the same time, we managed to give our structure a historical order we would really have made it.

We must admit that this was our very first idea. We used this as an alternative to, what we conceived as a more standard method of presentation namely, relying on linguistic analysis to give us a good thematic structure. Such an analysis would identify fundamental concepts used by systems developers—data, information, system, development, project, etc.—and built a structure around an exposition of these concepts.

Posing these two approaches against one another as options for structuring a philosophy book for systems developers now seems a little arbitrary against the background of the variety of methods we have distinguished earlier. What is wrong with the other methods? Could they not, just as well, be used for giving structure to our presentation? Certainly they could, and realizing this we have decided not to structure our book in terms of one or a few of these methods. We intend to use them all in the book, and as they are put into play they give local structure to our discussion. The book as a whole is structured by quite different principles.

When philosophers ask their standard question "What is this?", they generally ask about some sort of object. Were our philosophical analysis of systems development to follow the example set by, e.g., epistemology (analyzing knowledge) or aesthetics (analyzing art) we would concentrate on the idea of a system. The activity of systems development would be understood as an activity resulting in systems. But there are good reasons to change this priority, especially in a book for practitioners, emphasizing instead the analysis of the

activity, and then understanding systems as the results of the activity of systems development.

There are systems everywhere in the world of systems developers, so some sort of general as well as computer specific systems thinking has to be part of their competence. But systems are being developed and to understand what development is and what the inherent problems are is a prerequisite for being professional. In addition, systems developers want to do a good job. They aim for quality in their products. But what is quality? What is a good system and what is a good process? How do we measure that, and how do we make sure that our hard work results in high quality systems? These are questions that any reflecting systems developer will worry about—and so do we. Finally, when starting to philosophize one will experience it as an intellectual game involving many perspectives, and if we study the philosophical literature on systems development we also see many perspectives. In the very act of philosophizing we are encouraged to reflect upon and rethink our perspectives on systems and systems development. Where do those perspectives come from? How many are they and how are they constructed?

The result of these considerations is a book in four parts, each part identifying a vital aspect of the practice of systems development, and together adding up to an understanding of the basics of that practice. Not a book telling what every systems developer needs to know, but rather a book presenting some fundamental issues every systems developer needs to think about, reflect upon.

Asking ourselves what are the most vital aspects of systems development we came up with four elements: systems, development, quality, and perspectives. These four elements make our book superficially similar to standard computer and information science text books. The choice of basic structure is, however, a consequence of our considerations on the nature of a discipline-oriented philosophy book, and we hope to have convinced you that the content is far from traditional. Unless the vital aspects of systems development are given a serious treatment the book would not be aptly titled.

Within each part separate chapters deal with a few outstanding issues of developing computer-based systems. Ideally each chapter should begin with something concrete related to the experience and background of the reader: a situation, a method, a case, and then from there begin to reflect and argue. But what situations

should one choose? And how could one make sure that such a book would add up to some sort of comprehensive view of the field of systems development? Only by giving the book a basic structure as mentioned above.

### 3.2. Educational considerations

To knowingly reflect on the practice of systems development you need to know that practice. Students cannot be expected to have personal experience of systems development, so how can they participate in a philosophical discussion of that practice? This is a general problem for all theoretical teaching of practices. But there are ways of partly overcoming that problem, and we will use four of them: telling about that practice will be part of the book, we will prefer readers with some experience of programming, systems development exercises and case studies, we will aim at encouraging attitudes and ways of philosophizing rather than teaching facts, and, finally, our aim is for the book to become part of the reader's library rather than be read and then put away.

More generally one can question whether it is possible to learn anything worthwhile about a practice like systems development from reading a book about it? And if so, how should that book be composed? An attempt to answer these questions can serve as an illustration of what it means to work with different perspectives.

Traditional theories of education like to compare human beings to plants which, under suitable conditions, grow and develop abilities with which they are innately endowed. Such growth is not viewed as a matter of acquiring new information. And consequently, in this traditional view, growth is not particularly helped by the reading of books. Listen only to what Plato has to say about writing in the Phaedrus:

*“Then anyone who leaves behind him a written manual, and likewise anyone who takes it over from him, on the supposition that such writing will provide something reliable and permanent, must be exceedingly simpleminded; he must really be ignorant of Ammon's utterance, if he imagines that written words can do anything more than remind one who knows that which the writing is concerned with.”*

To modern theories of education human beings are not plants but rather containers to be filled with knowledge by an education in

which books play an important role as rich information carriers. The human soul is metaphorized as a wax table (John Locke) waiting to be written on, or as an attic (Sherlock Holmes) in which things are stored. Holmes is particularly worried about crowding his attic with information he does not need taking space from such information he really needs. Memory as a storage area is the dominating metaphor in modern cognitive science with its view of the thinking mind as an automatic book (a program).

With a traditional perspective it makes little sense to write and read books on systems development. Only practice will tell if one has the abilities to become a good systems developer and if one has those abilities, books won't matter much. With a modern perspective the prescription for a good book on systems development is straightforward: cram it with useful facts, leave out irrelevant information. Obviously, these recommendations give us too little to go on and we need to look closer at the phenomenon of education.

Other perspectives can easily be found. The easiest way is to borrow from other areas using the method of analogy. Science, for example, is viewed from different perspectives by different theories of science such as positivism, hermeneutics, structuralism, critical theory, etc. But these theories can easily be generalized into perspectives on almost any type of human activity. As different perspectives on education they bring out nice contrasts. Positivism is a modern theory stressing control and standards, suitable for an industrialized system of mass education.<sup>7</sup> Hermeneutics is a traditional theory viewing education as a conversation between a master and a disciple, a mentor and a protégé, with the disciple mostly listening.<sup>8</sup> For a critical theorist education is a group process where the group itself is the most important resource, while structuralism wants to stress the importance of the material embedding for the educational process.

Depending on what perspectives on education you favor, your idea of how to compose a book on systems development should differ, since books play very different roles in these different perspectives. It is our intention to create a book based on the assumptions and limitations of all of these perspectives without falling into the trap of exclusively relying on a single one. The book is going to contain facts, at the same time making clear that it presents our interpretations. The book is going to invite its readers to engage in a dialogue, at the same time being authoritative. And even if the book

can only remind its readers about what they already know, it is based on the belief that texts can influence the ways we organize our thinking.

Finally, in the process of creating the book it is crucial to be aware that we are producing a text that is to be used as only one amongst other means for teaching practitioners and students to philosophize in relation to their profession. We should be humble as to what kind of impact on the attitudes and practices of the reader the book can have in itself. But at the same time we should attempt to create a strong and convincing argument with a high degree of relevance in relation to the inherent problems and challenges of systems development.

#### **4. A specific proposal**

We have arrived at a specific proposal for a philosophy book for systems developers, and we are now going to describe the structure, content, style and form of this proposal. The working title of the book is *Systems Development Philosophy*, and its overall structure is presented in figure 1 below.

The first part of the book is about the kinds of systems involved in systems development and about different ways to think and learn about them. Chapter 1 addresses the key technology involved and the general solution suggested in systems development, i.e. the computer. The potentials and limitations of this technology is discussed in terms of four metaphors: the computer, the tool, the automaton, and the medium. Computer systems are artifacts constituting a reality in which systems developers are involved. Chapter 2 addresses the complementary reality, i.e. organizational situations that have to be analyzed, interpreted and eventually changed. The challenge in systems development is to bridge these two realities by identifying relevant problems and implementing useful computer systems. Different kinds of systems will be discussed and compared including the levels of: data, information, communication, competence, and organization.

The first two chapters discuss the kinds of phenomena that systems developers work with, whereas chapter 3 addresses the ways we think and learn about these phenomena. Three different kinds of epistemology are compared, starting with hard systems thinking, further on to soft systems thinking, and finally arriving at

---

*Systems Development Philosophy*

- I. Systems
    - 1. Facing computers
    - 2. Facing situations
    - 3. Thinking about systems
  - II. Development
    - 4. Constructing systems
    - 5. Solving problems
    - 6. Changing organizations
  - III. Quality
    - 7. Striving for quality
    - 8. Struggling with quality
    - 9. Using computers
  - IV. Perspectives
    - 10. Computers and people
    - 11. Structures and processes
    - 12. Thinking with perspectives
- 

*Figure 1. The title and overall structure of our proposal for a philosophy book for systems developers.*

dialectical thinking. All three represent valid approaches to systems development each with their strengths and weaknesses.

Part I tells three parallel stories. We attempt to understand and fit together two realities, i.e. computer systems and social organizations, and to understand the consequences of different conceptual approaches for doing so. What are the differences and similarities between the two worlds? In what sense are they related? Do they relate like body and mind? Does it make sense to talk about social construction of technology? Does it make sense to talk about construction of social situations? What is it that always make us think of problems and solutions like we think of illness and treatment? Are there other options?

The second part of the book is about development; about the different kinds of problems, challenges and roles that are involved

in the development of computer-based systems. Each of the three chapters present a specific view on what are the constituting elements in systems development. In chapter 4 systems development is seen as technical construction with the systems developer in the role of economic man behaving rationally by optimal usage of standard engineering techniques. In chapter 5 systems development is seen as problem solving with the systems developer in the role of information processor and decision maker. The problem is assumed to be given, but there is no obvious solution. Too little relevant knowledge is available and the systems developer has to search for new insights by exercising satisficing rather than optimizing behavior. This view stresses the learning aspects of systems development. In chapter 6 systems development is seen as organizational intervention with the systems developer in the role of organizational actor. The problem is no longer given, instead the systems developer is facing unstructured situations with many actors and different interests and interpretations. The systems developer has to engage in organizational games in attempting to set the problem and identify possible actions. This view stresses development as organizational change.

Part II presents three perspectives on systems development. We attempt to understand and fit together three complementary views on systems development with the purpose of providing the reader with a rich and differentiated view on this kind of human enterprise, and also with the intention of covering the defining characteristics of the activity. What kinds of competencies and methods are needed? In what areas of systems development can we hope to develop useful methods, and to what extent do we have to rely purely on the competence and intuition of the actors? What is the relation between systems developers and bystanders in a systems development project? What is the role of communication, cooperation and management?

The third part of the book is about the crucial issue of quality. Chapter 7 addresses quality on the intentional level by philosophizing on the basic question "What is quality?". We discuss the possibilities and limits related to defining, describing and measuring quality in general. More specifically, we present and compare two complementary views on quality in systems development: quality as related to computer systems and quality as related to the practical use of these systems. Chapter 8 addresses quality on the

practical level by concentrating on the question “How can we deal with quality?”. We discuss the contradiction between quality and resources and compare and evaluate different approaches to quality control and quality assurance in systems development. We inquire into the social dimension of quality by discussing expectations, interpretations and interests. We also emphasize the relation between quality of the development process and quality of the product, i.e. a new computer system and related organizational changes. Finally, chapter 9 addresses quality issues related to the use of computers in organizations. Why and when should computers be used, and what kind of impact do they have on the quality of work and on the kinds of services and products that organizations provide.

Part III discusses one of the most important and at the same time difficult issues related to systems development and the use of computers in general. Professional systems developers want to do a good job, but according to what standards? What are the goals and who is the customer? What are the underlying values and assumptions? What kinds of conflicts and dilemmas do we face when striving for quality? To what extent can quality be planned and result from purposeful human intervention? What is the relation between planning, participation and incentives in systems development?

The fourth part of the book is about perspectives, one of the dominating academic concepts in contemporary attempts to philosophize on different approaches to systems development, and at the same time a key concept in understanding how to interpret, describe and design organizational situations involving computer usage. Chapter 10 addresses people and computers as two fundamental perspectives related to systems development. We discuss specific methods, problems and practices related to social and technical perspectives. On a more general level we expand and deepen these perspectives into a discussion of positivism and hermeneutics.

Chapter 11 addresses structure and process as two other fundamental perspectives related to systems development. The structure-process dimension is considered to be orthogonal to the people-computer dimension. Structures and processes or stability and change are again discussed as concrete perspectives on systems development. On a more general level they are expanded and deepened into a discussion of structuralism and critical theory.

Finally, in chapter 12 we discuss perspectives in theory and practice, as viewpoints in systems development, as means to sup-

port critical and constructive thinking. We discuss the use of metaphors and analogies to intentionally think with different perspectives, and we emphasize perspectives as implicit personal background and institutional framework. Finally, we address the role of theories in systems development and argue that perspectives can be viewed as the basis for such theories.

Part IV is different from the other three parts in not only addressing systems development as a kind of practice, but as an academic discipline as well. In a way, part IV is introducing a forth issue in its own right, at the same time reviewing the issues already treated. We explore the relation between theory and practice—in theory as well as in practice—and we attempt to formulate a holistic view based on the various discussions in the book.

Concerning form and style, we want the book to be short and still have substance. We want the book to contain good examples and fundamental theorems and principles for easy memorizing. The book will include suggestions for smaller and larger exercises for the benefit of both teachers and students.

We want the book to be a useful part of any kind of systems development curriculum in computer science departments as well as in business schools. We presuppose no philosophical background and we write for undergraduate students as well as practitioners. We consider it important to present our discussions in a style that will simulate further thinking, but still be easy to remember and make examinations possible. We will attempt to write a book that can be used in the next twenty-five years, that is about the development of computer technology applications without running the risk of becoming outdated by the rapid development of that technology, changes in work organizations, and a changing *Zeitgeist*. The book should be written in a clear and sober style, using and clarifying concepts, but at the same time demonstrating and encouraging an attitude of openness.

## **5. Concluding remarks**

We have outlined a philosophy book for systems developers and argued that such a book can play an important role in educating professional practitioners. Our purpose has been to engage others in a discussion of the role of philosophy in systems development educa-

tion and training, and to prepare ourselves to write a philosophy book for systems developers.

Philosophy has played an important and active role throughout the history of science and scientific education. A couple of decades ago, many universities left an old tradition of having a general philosophy course as part of all curricula, and in the remaining period philosophy has played a minor role in most scientific education. We believe that philosophy again should play an active role, but this time not (only) as a general discipline in its own right. Instead, philosophy should be integrated into various scientific fields. We see philosophy as an important approach to mature and improve many professions and especially systems development. Philosophizing in systems development can support our students in acquiring skills that are independent of specific technologies and support them in becoming systematic and critical thinkers.

Our own proposal for a philosophy book for systems developers will be written and tested in the period to come. We hope that the present discussions—together with the list of references indicating what we consider to be the theoretical platform on which we will write the book—have provided a clear picture of what we want to achieve. Any objections, criticism or support to our undertaking is more than welcome.

### Notes

- 1 This view on professional practice in general and systems development in particular is discussed in detail in (Schön 1983; Lanzara 1983; Andersen 1990).
- 2 See the definition of “philosophy” in the Oxford English Dictionary.
- 3 Tradition tells us that Aristotle wrote a great number of dialogues in his youth, all of which are lost, that were equal to or even more beautiful than Plato's.
- 4 The ancient drama between Plato and the sophists is replayed in the late eighties with “modernists” like Jürgen Habermas playing the role of Plato and “postmodernists” like Francois Lyotard advocating sophistry.
- 5 The Greek word “methodos” means “route”.
- 6 Method (a), often called “conceptual analysis”, is the philosophical method par preference, in our century the defining method of “analytic philosophy”. Method (e) is the preferred method in the philosophical tradition established by Hegel.

- 7 B. F. Skinner's very influential theory is a typical example. A somewhat frightening, popularized version of his theory can be found in his *Walden Two* (Skinner 1948).
- 8 See Herrigel (1953) for a good example of an hermeneutic education.

### References

- Andersen, N. E., F. Kensing, M. Lassen, J. Lundin, L. Mathiassen, A. Munk-Madsen & P. Sørgaard (1990): *Professional Systems Development. Experiences, Possibilities and Action*. Englewood Cliffs, New Jersey: Prentice-Hall.
- Andersen, P. B. & L. Mathiassen (1987): Development and Use of Computer-based Systems. A Science of Truths or a Theory of Lies. In G. Bjercknes *et al.* (Eds.): *Computers and Democracy*. Aldershot: Gower.
- Bansler, J. (1989): Systems Development Research in Scandinavia: Three Theoretical Schools. *Scandinavian Journal of Information Systems*, Vol. 1 (3–20).
- Bjercknes, G., B. Dahlbom *et al.* (Eds.) (1990): *Organizational Competence in System Development. A Scandinavian Contribution*. Lund: Studentlitteratur.
- Bermann, T. (1990): Between Micro and Megalomania: (inter-) Organizational Competence in Systems Development. An Essay. In G. Bjercknes *et al.* (1990).
- Boehm, B. W. (1981): *Software Engineering Economics*. Prentice-Hall.
- Boguslaw, R. (1965): *The New Utopians. A Study of System Design and Social Change*. Prentice-Hall.
- Boland, R. J., Jr. (1985): Phenomenology a Preferred Approach to Research on Information Systems. In E. Mumford *et al.* (1985).
- Boland, R. J. & R. A. Hirschheim (1987): *Critical Issues in Information Systems Research*. Chichester: Wiley.
- Checkland, P. (1981): *Systems Thinking, Systems Practice*. Chichester: Wiley.
- Churchman, C. W. (1968): *The Systems Approach*. Delta Books.
- Churchman, C. W. (1971): *The Design of Inquiring Systems*. New York: Basic Books.
- Dahlbom, B. (1990): Using Technology to Understand Organizations. In G. Bjercknes *et al.* (1990).
- Dahlbom, B. (1991): The Idea that Reality is Socially Constructed. In Floyd *et al.* (Eds.): *Software Development and Reality Construction*. Berlin: Springer-Verlag.

- Dahlbom, B. & L.-E. Janlert (1990): An Artificial World: An Invitation to Creative Conversations on Future Use and Design of Computer Technology. *Scandinavian Journal of Information Systems*, Vol. 2.
- Ehn, P. (1988): *Work-Oriented Design of Computer Artifacts*. Stockholm: Center for Working Life.
- Feldman, M. S. & J. G. March. (1981): Information in Organizations as Signals and Symbol. *Administrative Science Quarterly*, Vol. 26, No. 2 (171–186).
- Floyd, C. (1987): Outline of a Paradigm Change in Software Engineering. In G. Bjerknes *et al.* (Eds.): *Computers and Democracy*. Gower.
- Franz, C. R. & D. Robey (1984): An investigation of User-Led System Design: Rational and Political Perspectives. *Communications of the ACM*, Vol. 27, No. 12 (1202–1209).
- Herrigel, E. (1953): *Zen in the Art of Archery*. Pantheon Books.
- Hirschheim, R. & H. K. Klein (1989): Four Paradigms of Information Systems Development. *Communications of the ACM*, Vol. 32, No. 10 (1199–1216).
- Israel, J. (1979): *The Dialectics of Language and the Language of Dialectics*. The Humanities Press.
- Klein, H. K. (1984): Which Epistemologies for Future Information Systems Research? In M. Sääksjärvi (Ed.): *Report of the Seventh Scandinavian Research Seminar on Systemeering, Part 2*. Finland: Helsinki School of Economics.
- Klein, H. K. & K. Lyytinen (1985): The Poverty of Scientism in Information Systems. In E. Mumford *et al.* (1985).
- Kling, R. (1980): Social Analysis of Computing: Theoretical Perspectives in Recent Empirical Research. *ACM Computing Survey*, Vol. 12, No. 1 (61–110).
- Kling, R. & W. Scacchi (1980): Computing as Social Action: The Social Dynamics of Computing in Complex Organizations. In M. C. Yovits (Ed.): *Advances in Computers*, No. 19. Academic Press.
- Kling, R. & W. Scacchi (1982): The Web of Computing: Computer Technology as Social Organization. In M. C. Yovits (Ed.): *Advances in Computers*, No. 21. Academic Press.
- Lanzara, G. F. (1983): The Design Process: Frames, Metaphors and Games. In U. Briefs *et al.* (Eds.): *Systems Design For, With and By the Users*. Amsterdam: North-Holland.
- Lyytinen, K. & H. K. Klein (1985): The Critical Theory of Jürgen Habermas as a Basis for a Theory of Information Systems. In E. Mumford *et al.* (1985).

- Lyytinen, K. (1987): Different Perspectives on Information Systems: Problems and Solutions. *ACM Computing Surveys*, Vol. 19, No. 1 (5–46).
- Lyytinen, K. & E. Lehtinen (1987): Seven Mortal Sins of Systems Work. In P. Docherty *et al.* (Eds.): *Systems Design for Human Development and Productivity: Participation and Beyond*. North-Holland.
- Lyytinen, K. (1986): *Information Systems Development as Social Action: Framework and Critical Implications*. Ph.D. thesis, Department of Computer Science, University of Jyväskylä, Finland.
- Madsen, K. H. (1988): Breakthrough by Breakdown. *Proceedings from International Conference on Information Systems for Human Progress*. North-Holland.
- Mathiassen, L. (1981): *Systems Development and Systems Development Methods*. Ph.D. thesis, Oslo University. (In Danish)
- Mathiassen, L. & A. Munk-Madsen (1985): Formalization in Systems Development. *Proceedings of the Joint International Conference on Theory and Practice of Software Development*. Springer-Verlag.
- Mathiassen, L. (1987): Systems, Processes and Structures—A Contribution to the Theoretical Foundation of Systems Development. In P. Docherty *et al.* (Eds.): *Systems Design for Human Development and Productivity: Participation and Beyond*. North-Holland.
- Mathiassen, L. & P. A. Nielsen (1989): Soft Systems and Hard Contradictions. *Journal of Applied Systems Analysis*, Vol. 16 (75–88).
- Mathiassen, L. & J. Stage (1990): Complexity and Uncertainty in Software Design. *Proceedings from COMPEURO 90, IEEE*.
- Mintzberg, H. (1984): *Structure in Fives: Designing Effective Organizations*. Prentice-Hall.
- Morgan, G. (1986): *Images of Organization*. Sage Publications.
- Mumford, E. *et al.* (Eds.) (1985): *Research Methods in Information Systems*. Elsevier.
- Mumford, L. (1934): *Technics and Civilization*. New York: Harcourt Brace Jovanovich.
- Naur, P. (1984): Programming as Theory Building. *Microprocessing and Microprogramming*, Vol. 15. North-Holland.
- Naur, P. (1985): Intuition in Software Development. *Proceedings of the Joint International Conference on Theory and Practice of Software Development*. Springer-Verlag.
- Nurminen, M. (1988): *People or Computers: Three Ways of Looking at Information Systems*. Lund: Studentlitteratur.
- Nygaard, K. & P. Sørgaard (1987): The Perspective Concept in Informatics. In G. Bjeknes *et al.* (Eds.): *Computers and Democracy*. Gower.

SYSTEMS DEVELOPMENT PHILOSOPHY

- Parnas, D. L. & P. C. Clements (1985): A Rational Design Process: How and Why to Fake it. *Proceedings of the Joint International Conference on Theory and Practice of Software Development*. Springer-Verlag.
- Pirsig, R. M. (1974): *Zen and the Art of Motorcycle Maintenance*. The Bodley Head.
- Robey, D. & M. L. Marcus (1984): Rituals in Information System Design. *MIS Quarterly*, Vol. 8, No. 1 (5–15).
- Schön, D. A. (1983): *The Reflective Practitioner*. New York: Basic Books.
- Simon, H. (1969): *The Science of the Artificial*. Cambridge, Massachusetts: MIT Press.
- Simon, H. (1982): *Models of Bounded Rationality: Behavioral Economics and Business Organization*. Cambridge, Massachusetts: MIT Press.
- Skinner, B. F. (1948): *Walden Two*. MacMillan.
- Suchman, L. (1987): *Plans and Situated Action*. Cambridge University Press.
- Weston, A. (1987): *A Rulebook for Arguments*. Hackett Publishing Company.
- Winograd, T. & F. Flores (1986): *Understanding Computers and Cognition, A New Foundation for Design*. Ablex.