A Reappraisal of Structured Analysis: Design in an Organizational Context

JØRGEN P. BANSLER
University of Copenhagen
and
KELD BØDKER
Roskilde University

We review Structured Analysis as presented by Yourdon and DeMarco. First, we examine the implicit assumptions embodied in the method about the nature of organizations, work processes, and design. Following this we present the results of an exploratory study, conducted to find out how the method is applied in practice. This study reveals that while some of the tools of Structured Analysis—notably the data flow diagrams—are used and combined with other tools, the designers do not follow the analysis and design procedures prescribed by the method. Our findings suggest that there is a gap between the way systems development is portrayed in the normative technical literature and the way in which it is carried out.

Categories and Subject Descriptors: D.2.1 [**Software Engineering**]: Requirements/Specifications—*methodologies, tools*; D.2.9 [**Software Engineering**]: Management—*life cycle*; D.2.10 [**Software Engineering**]: Design—*methodologies, representation*; H.1.0 [**Models and Principles**]: General; J.1 [**Computer Applications**]: Administrative Data Processing—*business, financial*

General Terms: Design, Documentation, Management

Additional Key Words and Phrases: Design process, qualitative empirical studies, structured analysis

1. INTRODUCTION

The term 'structured' was first introduced in connection with programming. Structured programming and underlying principles like 'top-down development,' 'stepwise refinement,' 'hierarchical decomposition,' and 'modularization' were introduced in the late sixties. From the start these concepts were

Authors' addresses: J. P. Bansler, Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen, Denmark, email: bansler@diku.dk; K. Bødker, Department of Computer Science, Roskilde University, DK-4000 Roskilde, Denmark.

subject to criticism, by for instance Naur [32]. Later the term structured was applied to technical design and implementation: Structured Design [41, 51]; to systems analysis and design: Structured Analysis [9] and Structured Systems Analysis [19]; as well as to system development in general: Structured Systems Development [37], SADT—Structured Analysis and Design Technique [40], and ISAC—Information Systems Analysis and Construction [29].

We are concerned with the Structured Analysis method only, originally developed by Yourdon and DeMarco and refined several times. Throughout the eighties this method has been subject to much criticism from researchers within the informations systems field [5, 6, 7, 15, 24, 45]. In spite of this criticism, Structured Analysis is widely used. In Denmark this method is by far the most well-known and most widely used method for development of business applications for banking, insurance companies, and so forth.

Empirical studies of system development in practice are limited. We have not found empirical evidence in the literature to questions like *why* Structured Analysis is introduced—and *how* it is applied in real projects. To obtain answers to these questions we conducted a small scale exploratory study interviewing nine designers[1] in three Danish DP-departments on their application of Structured Analysis.

The paper has two main parts: In the first part (Sections 2 and 3), we review Structured Analysis from a theoretical perspective. The major principles of Structured Analysis are presented in Section 2. In Section 3, we examine the underlying assumptions about the nature of organizations, work, and systems design. In the second part of the article (Sections 4 and 5), we describe the procedure and discuss the results from our exploratory study in three Danish companies making extensive use of Structured Analysis.

2. THE PRINCIPLES OF STRUCTURED ANALYSIS

Since its introduction in the late seventies, the Structured Analysis method has been refined and modified several times. The underlying principles and aims have, however, more-or-less remained the same. In this section we describe these fundamental principles of Structured Analysis as they are presented in the main textbooks [9, 38, 50]. Yourdon [49] describes a *project model* covering the activities from the survey to the final implementation and installation. The project model, called 'the structured project life cycle,' integrates a number of structured methods/techniques like Structured Analysis and Structured Design into an overall framework. The discussion of different project models is beyond the scope of this paper.

The basic *tools* provided by Structured Analysis are the same as presented in *Structured Analysis and System Specification* [9]. These include various

[1] Throughout the text we use *designer* as a generic term for system analysts and system designers. Similarly, *design* refers to a number of activities such as systems analysis, systems specification, and systems design. This use differs from Yourdon and DeMarco, who use the terms analyst and analysis, respectively.

types of diagrams used to model (part of) an organization as an information system, the most important being the data flow diagrams (DFD). A data flow diagram is "a network representation of a system. The system may be automated, manual, or mixed. The Data Flow Diagram portrays the system in terms of its component pieces, with all interfaces among the components indicated." [9, p. 47]. The data flow diagram models the system by describing the flow between data sources and sinks, processes, and data stores.

The data flow diagram is structured in levels with the top level diagram describing the overall functions of the system. Each process (function) is then decomposed and modeled by a hierarchy of diagrams showing more and more details. The processes at the bottom level are described by mini-specifications (mini-specs), which are decision tables or 'structured English' texts describing the rules governing the transformation of data flows. The composition of individual data elements into data flows is described in the data dictionary. The use of data flow diagrams to describe systems is seen as a way of bridging the communication gap between users and designers. Because the diagrams are graphic and nontechnical—as opposed to for instance data format descriptions and flow charts—they supposedly provide a common language between users and designers.

Palmer and McMenamin [38] supplement Structured Analysis with tools to describe data relationships at a higher level in entity-relationship diagrams (E/R diagrams). Recently, Yourdon [50] has added state transition diagrams (ST diagrams) to describe time-dependent behavior.

The *techniques*—or procedures as they are called in Structured Analysis—prescribing how to use the tools have to some extent evolved over the years. According to [9] the analysis should include four steps modeling the current physical, the current logical, the new logical, and the new physical system, respectively (see Figure 1). Each step should consist of a complete description of the system by DFDs, the data dictionary and mini-specs.

This procedure rests upon two distinctions. Firstly, DeMarco distinguishes between models describing the *current* manual or partly automated system, and models describing the *new*, partly or fully automated, system. The second distinction is between *physical* models, describing a particular implementation, and *logical* models, describing the functionality in its pure form. DeMarco [9], however, offers only vague guidelines for the transformation of a physical model into a logical equivalent (and vice versa), and he offers no guidelines at all for the transition from a current logical model to a new logical model.

The basic distinction between the physical and logical aspects of a system has been elaborated by Palmer and McMenamin [38]. They stress that the analysis should result in only *true requirements*, i.e., features or capabilities that the system must possess to fulfill its purpose, regardless of how the system is implemented. To this end they introduce the concept of 'essential model,' which is a kind of 'logical' model. However, they consider it to be more well-defined than the original concept, developed by DeMarco.

An essential model is independent of technology, i.e., it does not describe how, when, and by whom a process is carried out, but only *what* is done.

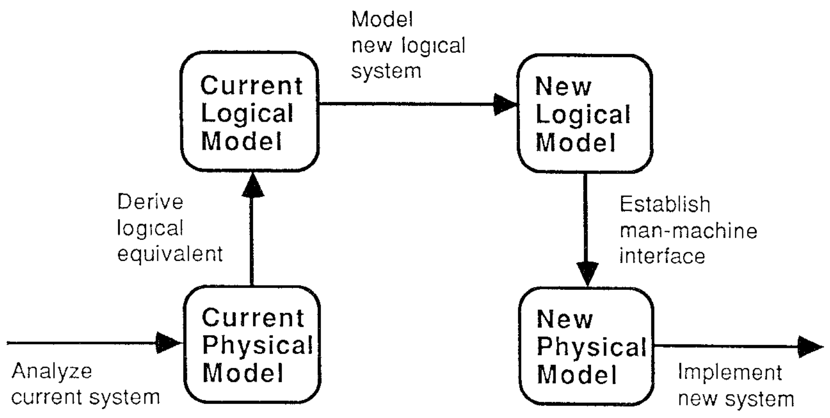

Fig. 1. Relationship among the four types of models in Structured Analysis.

Palmer and McMenamin propose two principles of essential modeling:

—An essential model contains the *essence* of the system only. Essence is defined as "all characteristics... that would exist if the system were implemented with perfect technology" [38, p. 16]. 'Perfect technology' is able to do anything and everything instantly, it has infinite capabilities and infinite work load capacity. It costs nothing, consumes no energy, takes no space, generates no heat, never makes a mistake, and never breaks down.

—The processes in an essential model must be grouped by *events*, that is, changes in the system's environment which the system must respond to. There are two types of events that systems respond to: 'external events,' which are initiated by entities in the environment; and 'temporal events,' which are initiated by the passing of time.

The emphasis on the essential model is also reflected in the principle of 'blitzing' or high-level analysis. Palmer and McMenamin suggest that the designer starts by constructing (blitzing) an incomplete, high-level essential model before modeling the physical aspects of the current system. This enables the designer to perform a detailed analysis of the relevant parts of the current system only. The designer cannot, however, skip the physical modeling altogether because the physical model is needed to verify and complete the essential model.

Yourdon [50] proposes minimizing or even skipping the model of the current physical system if at all possible. The reason given is 'political': from experience it has become apparent that the process of making a model of the current system may require so much time and effort that the user will become frustrated and impatient, and ultimately cancel the project. Yourdon maintains, however, that in some cases "the systems analyst *must* build a model of the user's current system; that is true, for example, if the systems analyst needs to model the current physical system in order to discover what the essential processes really are." [50, p. 323].

So, despite changes in terminology and recommendations, the basic ideas are still the same: The first principle is to model organizations and work processes as information processing systems, emphasizing the *flow of data*. The second principle is to distinguish between 'physical' aspects of the system on the one hand and 'logical' or 'essential' aspects on the other hand. The third, less consistently applied, principle is to use an analysis of the current 'physical' system as the starting point when deriving logical/essential aspects.

3. CRITICAL ASSUMPTIONS UNDERLYING STRUCTURED ANALYSIS

3.1 Organization and Work from a Structured Analysis Perspective

Structured Analysis envisions an organization as a kind of *information processing system*, that is, as a network of processes exchanging information according to certain rules. The construction of a set of formal models of the current work processes as well as of the new computer based work processes is the core of Structured Analysis. As described above, the aim is to produce detailed functional or 'logical' descriptions of tasks and operations while focussing on the flow and processing of information. Functional or 'logical' indicates an abstraction from 'accidental' or 'historical' peculiarities. No distinction is made between manual and automated procedures. A process can be carried out by a human being, an electromechanical device like an old-fashioned cash register, a fully electronic device like a digital computer, or any other 'information processing' device.

The most important and immediate consequence of this approach to systems analysis is that *people are made into objects*, simply perceived as 'system components,' comparable with tools, machines, and raw materials. In Structured Analysis one does not distinguish between the way people act and the way machines function, they are all alike from the method's point of view.

This is representative for a 'functionalistic' or 'system-structural' approach to the study of organizations, see the categorization of organization theories [1]. A system-structural model of an organization focuses on the structural properties of the context within which the organizational activities unfold, and individual behavior is considered to be determined by a reaction to structural constraints. Thus the organization is seen as a 'machine' designed to perform a given function optimally (in some sense); and work is essentially treated as procedural in nature, involving the workers' execution of a prescribed sequence of steps.

The machine metaphor has dominated engineering and management science, and dates back to the development of the classical management theories at the turn of the century. The basic ideas and procedures of Structured Analysis are in many respects identical with the ideas expressed by Taylor in his *Principles of Scientific Management* from 1911 [44]. DeMarco and Yourdon, like Taylor, suppose that most work processes can be reduced to 'rules, laws, and formulae.' They seem to believe that it is possible to find 'the one best way,' and to control the process by prescribing the work procedures step by step.

DeMarco's technique for analyzing information flows is based upon the process flow chart, an industrial engineering method developed by Taylor and the Gilbreths [8]. Process flow charts displayed how materials moved and indicated points where they were transformed. Later flow charts were used to show the flow of forms in office work. Eventually, with the development of data processing machines these form flow charts included transformations of data using electromechanical devices such as card punchers, tabulators, and sorters.

The analysis aims at 'rationalizing' information processing by identifying and removing the "procedural, historical, political or tool-related" peculiarities. According to [9] the designer must first build a 'physical model' of the current work processes and then "logicalize" it:

> "Working closely with the users, you learn and document the way things currently work. Rather than do this from the point of view of any one user or set of users, you attempt to assess operations *from the viewpoint of the data*.... The next step is to 'logicalize' our model of the current environment.... A particular implementation of policy is replaced by a representation of the policy itself. The underlying objectives of the current operation are divorced from the methods of carrying out those objectives." [9, p. 27–28]

While 'logicalizing' the model, the designer should focus on the regular, main-line processing and ignore the handling of error situations:

> "Most systems and most systems people dedicate the majority of their resources to dealing with error situations. Still, error processing usually does not have a strong effect on the *philosophy* of a system... Most systems derive their philosophy from the main-line processing. Internal structure must reflect this philosophy." [9, p. 68]

The designer is thus required to 'distill' the current work processes to extract the essence of the work, the pure functionality, in term of a 'logical' model. This model is then subject to systematic interpretations by the designer with the purpose of constructing a new system:

> "...the analyst builds a new model, one that documents operations of the future environment (with the new system in place), just as the current model documents the present. The new model presents the system-to-be as a partitioned set of elemental processes. The details of these processes are now specified, one mini-spec per process." [10, p. 416]

The new system is first described in terms of a 'logical' model without considering which procedures are to be performed by human beings and which are to be carried out by machines. "We do not even distinguish between those procedures that will be automated and those which will be manual. We simply declare the work that must be done, the rules that govern its doing and interfaces among component parts of the whole." [9, p. 30]. Only after the logical model has been constructed in detail, it is possible to decide, which of the procedures should be carried out by people and which should not. This decision is based on a cost-benefit analysis and jobs and tasks are simply treated as *residuals* of the automated procedures:

> "The last transformation calls for incorporating a minimum of physical information, principally the man-machine boundary. (...) Only when you have established which of the primitives of your model will be performed by a machine and which by humans will you have completed the specification of your project targets. The scope of automation is determined by a cost-benefit study." [9, p. 265]

Structured Analysis—like "scientific management"—is, however, contradictory in nature. Ultimately Structured Analysis treats workers as all-purpose human information processors, programmed and manipulated by the systems department. Still, the problem is that people can not simply be reduced to machines. People have physical and psychological needs as well as abilities which no machine possesses. By treating workers as machines and thereby removing any intrinsic interest they may have in the tasks they perform, dissatisfaction is increased and motivation is decreased. At the same time the positive aspects of human labor are forgone, i.e., their intelligence and flexibility [17]. This inherent contradiction leads to a number of deficiencies and shortcomings with regard to the practical application of the method:

(1) Structured Analysis underrates the *skills* and ingenuity of the workers. Most work processes—and not just the so-called skilled work—involve a fair amount of 'problem solving,' interpretation, and judgement, which cannot be reduced to rule following, see, e.g., [12, 27, 42].

(2) Structured Analysis ignores the significance of casual meetings and informal conversations among the workers. In many cases, however, *informal communication* plays a crucial role in the work process by allowing the workers to coordinate their tasks and to cooperate in solving difficult or unusual problems, see, e.g., [18, 36, 48].

(3) Structured Analysis underestimates the frequency and significance of *errors and exceptions* from the 'norm,' which inevitably crop up in the work process. Rarely, if ever, do actual work processes proceed as smoothly and regularly as described by a data flow diagram, and the so called 'exceptions' are not trivial [27, 42, 43, 46, 48].

(4) Structured Analysis disregards the problems posed by differing *interests* and by *power*. Consequently, Structured Analysis does not recognize the need for negotiating and thereby establishing a certain degree of consensus among users about the purpose of the new system. Opportunistic behavior or the resistance from individuals or groups of users may, however, prevent the effective implementation of a solution which—in principle— is 'technically feasible' and 'rational' [13, 23].

(5) Structured Analysis offers no help in describing and analyzing *work organization*. Even if the method has many implicit assumptions about work organization, it does not deal explicitly with questions regarding the design of jobs and modes of cooperation. The method contains no concepts for modeling organizational units or resources, and there is no way of

specifying relations of authority and responsibility, etc. [5]. As noted above, jobs and tasks are simply treated as residuals of the automated procedures [6, 12, 16, 20].

3.2 The Design Process from a Structured Analysis Perspective

Embedded in Structured Analysis is a strong belief in instrumental rationality. The design process is seen as a *problem solving* activity. Starting with a well defined problem and explicit objectives (documented in the so-called Charter for Change), the designer then searches for the best way to solve the problem and achieve the objectives, proceeding along the steps of a prescribed procedure.

Structured Analysis belongs to a broad family of design methods based on hierarchical decomposition of functions, often referred to as *functional analysis* [28]. Functional analysis has been widely applied—not only to the design of programs and information systems—but also within industrial production, architectural design, and management science. Within the computing field, Dijkstra [11] and Wirth [47] first advocated using the principles of functional analysis in their work on 'structured programming.' Structured programming, as we noted in the introduction, is the antecedent of methods for 'structured design' and 'structured analysis.'

The basic idea is, that a complex problem is best understood—and solved—when it is subdivided into smaller pieces, each of which can be tackled and solved separately. According to the principles of functional analysis a product or a system is designed by decomposing it into its constituent parts, i.e., elementary functions which can be precisely defined. By a process of hierarchical decomposition the designer builds a functional tree: going out along the various branches one describes more and more simple functions, ending with the most elementary ones. At the root of the tree the functions are coordinated and the components assembled.

According to Structured Analysis the designer must build not one, but two functional trees. First, the designer analyzes the structure of the current system in order to determine its basic functionality, which is to be documented in the current logical model. The designer then reconstructs this functional tree in order to accommodate the objectives stated in the charter for change. Generating this new tree (documented in the new logical model) entails a logical *top-down partitioning* of the new system's functions:

"This partitioning differs from past efforts in a number of respects:

—It is truly top-down. Since the top is known . . . , there is no need to start in the middle as is sometimes required in studying the current physical environment.

—The analyst is boss. Instead of being guided by the user's political and procedural view, the analyst partitions according to his own standard: minimization of interfaces.

—There is a new set of rules in force. I refer to the Charter for Change. It may call for added function, new processes and data flows, increased accountability, etc." [9, pp. 260–262]

The designer is not supposed to consider the system's *implementation* until after the functional tree has been worked out in detail. In other words, the transformation from the current logical to the new logical model is a transformation from one *abstract* specification to another. Exactly how the designer is going to carry out this top-down partitioning of the new systems functions remains an open question.

The role of the users in this process is passive. They may act as sources of information and as reviewers of the designer's proposals, but they are not asked to participate in the actual design work. DeMarco almost leaves one with the impression that the prime reason for involving users in the design of the new system is to "co-opt" them:

> "Urge the user to help you 'debug' the model [of the new system] while it is still on paper. (. . .) This is not the time to ask the user to be lenient. On the contrary, he ought to be allowed some whims while the system is still made only of paper. Now is the time to co-opt him, to make him feel that, by imposing his modifications on the model, he is establishing the eventual shape of the system. This will give him a sizable measure of responsibility for the system when it is delivered, and will make him feel every deficiency is at least partly his fault. That frame of mind makes him doubly helpful during analysis and more than normally docile at acceptance time." [9, pp. 263–264].

Structured analysis is based on three basic assumptions about the nature of the design process:

(1) The problem to be solved is well-defined and clearly stated at the outset, and the given ends will not be modified during the process. Objectives as well as criteria for evaluating proposed solutions are unambiguous and consistent; and it is possible to verify whether or not a given solution meets the stated evaluation criteria.

(2) The designer is assumed to be completely rational—an ideal type with perfect information about design variables as well as about environmental parameters and constraints.

(3) It is possible and desirable to separate function from implementation, and to design and describe the function of the system without any reference to its actual implementation. Decisions about so-called 'physical aspects' must be postponed until after the so-called 'logical aspects' have been determined.

These assumptions have, however, very little to do with real-life design situations. Problems are ill-defined more often than not. Objectives and goals are vague, changing, and often in conflict with one another. In most cases the design process is one of collective inquiry and search where several actors, in cooperation or conflict, define relevant problems and possible solutions—doing so more or less simultaneously. Problem and ends can not be taken as givens, they are *negotiated and clarified* during the design process.

Malhotra et al. [31] describe an empirical study of designer-client dialogues. Their work clearly demonstrates the unstable and vague goals typical in design situations. Malhotra et al. conclude that in "real-world design situations, the goals are, typically, fuzzy and poorly articulated and cannot be mapped directly into properties of the design. Thus, the exact configuration of the final state is not prescribed. A part of the design process consists of formalizing and refining the design goals into functional requirements that can be matched by properties of the design. Even so, it is usually difficult to tell how well a design meets a particular functional requirement. In addition, the functional requirements often cover different dimensions and the trade-off between them are rarely well specified." [31, p.120].

Based upon their empirical findings Malhotra et al. propose an alternative, descriptive model of the design process. We do not want to promote this model as the only true description of this process, but we find that it highlights some aspects that are very important in our context.

Malhotra et al. decompose the design process into three fundamental, underlying processes. The first process is called *goal elaboration* and entails stating and discussing the goals of the design. During the second process, the *design generation*, attempts are made to come up with a design which meets the stated goals. After a (partial) design has been generated, the third process, that of *design evaluation*, begins. This starts with the introduction of the new (partial) design and ends with either its acceptance or rejection. The most important feature of this process is that it may uncover new requirements, especially those which are fuzzy and difficult to formalize. These three processes are usually inextricably interlaced, reinforcing each other. Results from a recent study of software designers by Guindon support this view [21]. She reports how separate operations intertwined and how opportunistic behavior played a crucial role, even among designers who subscribe to top-down design.

Because Structured Analysis focuses so strongly upon functional analysis it offers little or no help to the designer in carrying out the three fundamental processes of design described above. In particular, Structured Analysis overlooks the fact that some form of active user participation is indispensable when objectives and evaluation criteria are not explicit and fixed at the outset [12, 16, 20]. This has a number of practical implications:

(1) By taking for granted that the problem to be solved is well-defined and that means and ends are given, Structured Analysis consequently ignores the need for *goal elaboration* as part of the design activity.

(2) DeMarco and Yourdon recognize the importance of *design generation*, but do not have any suggestions on how to support it. Instead, they propose that it is "an act of creativity for which there are no mechanical rules" [49, p. 84] We agree that no 'mechanical rules' for creative acts can be formulated. However, guidelines for establishing creative settings and for stimulating creativity might be given. Prototyping techniques have been proposed to support design generation (and design evaluation) by emphasizing experiments and iterations in the systems development process [3, 4, 12, 14, 20]. Madsen and Kensing use Future Workshops and metaphorical design to stimulate generation of new ideas for design of computer systems [25, 30].

(3) According to Structured Analysis *evaluation* of the new systems design is a rather straightforward matter. The designer simply asks the user to 'verify the meaningfulness' of the new logical model [9, p. 263]. It is, however, less than likely that the user will be able to imagine how the new system will function by looking at formal descriptions such as data flow diagrams. This has been pointed out by authors discussing the value of 'hands-on' experience in the design process, see, e.g., [3, 12, 20, 22].

(4) In general, Structured Analysis underestimates the need for and the difficulties involved in communicating with users. Users play a very passive role in Structured Analysis. Their job is to supply information and to 'verify' diagrams and specifications made by the designer. The data flow diagrams and the various types of formal notations (e.g., decision tables and structured English) are, however, not designed with the user in mind; on the contrary, they reflect the world-view of the computer society. As a result, we doubt that these formalisms are suited for communication outside the group of systems designers and computer programmers.

4. THE EXPLORATORY STUDY

A number of quantitative empirical studies on the use of methods, techniques and tools in system development have been carried out, e.g., [34]. Such studies provide insight into the approach DP-departments use to develop information systems. These studies are, however, of limited value in providing answers to questions like why and how a particular method is applied in practice. The reader is often left wondering, what it *really* means, when "62% of the DP-departments say that they use a structured approach with data flow diagrams and data dictionary descriptions as the most widely used tools" [34, Tables 4 and 6].

We have therefore carried out a small exploratory study based on qualitative interviews, and thus complement our theoretical criticism with evidence on *how* Structured Analysis is actually used in practice. The study also develops hypotheses for further inquiry.

We have conducted open-ended interviews with nine designers in three Danish companies with internal DP-departments. We use the term 'designer' to designate the informants. However, only one was actually entitled designer. Of the rest, three were project leaders, two were data base designers, one was a head of a development department, one was a programmer and one was a consultant. Regardless of position, they all had extensive practical experience in using Structured Analysis. The companies were selected on the basis of their experience with and positive evaluation of Structured Analysis. We actively searched for 'success stories' when looking for candidate companies. A Danish consultant firm specializing in Structured Analysis assisted in selecting appropriate cases and providing contacts with companies.

In an international context, these companies and their DP-departments are relatively small, although relative to other Danish companies, they are relatively large. Furthermore, within software work in Denmark the division of labor is less pronounced and jobs less fragmented, when compared to the situation in the U.S. (see [26]).

Some of the interviews were conducted individually (at The Bank and at The Utility Company), whereas the interview at The Financial Institution was conducted as a group interview. The interviews were structured using a simple interview guide as reference.[2] The guide covers four major topics: Why was Structured Analysis introduced? How was Structured Analysis introduced? How is Structured Analysis applied? Experience with the use of Structured Analysis?

The interviews were tape-recorded, and the results as they appear in the following are our interpretation from listening to these tapes. The designers interviewed have read and approved the following description of their use of Structured Analysis. To illustrate main points in the description, we have cut out parts of the interviews and placed these in boxes.

The interviews should, of course, be considered verbal reports only, and as such are subject to the problems of bias, poor recall, and poor or inaccurate articulation. When designers are asked to tell about how they carry out their work, and why they do it the way they do, one must be aware that they may —in retrospect—'rationalize' their activities and their reasons for doing them, see e.g., [43]. We have therefore found it necessary to corroborate our interview data with studies of relevant design documents and demonstrations of applied CASE tools.

4.1 Study 1: The Bank

The Bank is one of the largest in Denmark, employing several thousand people. The Bank has an internal DP-department with a few hundred employees. The department is responsible for the development and maintenance as well as the daily operation of all computer systems in The Bank.

Structured Analysis was introduced in The Bank in the early 1980's primarily due to its reported success in other large companies within the financial sector, but also due to an intensive marketing effort. As one of the designers noted, "the method is not exceptional in any way, but the timing was perfect." When Structured Analysis was first introduced, all development projects had to use the method. Now, it is up to the individual project group to decide what methods and tools they will use, but the use of DFDs is strongly recommended and most projects use them. The DP-department has no special section responsible for defining standards, teaching methods, and supervising development activities. It is the responsibility of the project leaders. New employees learn Structured Analysis by attending a one week course, offered by an external consulting firm, and by working closely with more experienced colleagues.

Most of the projects use DFDs and the Data Dictionary for internal communication among designers and programmers in the project group. The diagrams are used to describe the functionality and the structure of the new

[2] The interview guide can be obtained from the authors upon request.

computer system. The diagrams do not describe manual procedures. The description is a mixture of a logical and a physical model. As one of the designers said, "we don't need a pretty theoretical model—what you call a 'logical' model— we need a model describing the programs that are going to run on the machine."

While drawing and discussing the diagrams, the project group breaks down the system into a hierarchy of subsystems or program modules with well defined interfaces. At the same time, the group analyses the system's data, using E/R modeling and standard procedures for data base normalization. In large projects, the detailed analysis, design, and programming of the individual modules may then be carried out, more or less independently, by smaller work groups. Structured Design is not applied, except in very complicated cases. Typically, the lowest level of DFDs together with mini-specifications constitute the basis for programming. As a general rule, one low-level process corresponds to one program module.

DFDs are not used for describing the current system. Designers say it takes too much time and involves too much effort to build a model of the current system, compared to the information you obtain by doing so. Instead they rely upon informal discussions with users, verbal descriptions, and their own experience with the business area. See Figure 2.

The user interface is designed by programming the screen layouts in a fourth generation language and then validating the design by showing them to future users of the system. DFDs are not used for communication with users. The designers told us that they had tried to present users with the DFDs, but the users had been unable and unwilling to read the diagrams. So, now, they communicate with the future users of the system by means of ordinary text, form and screen layouts as well as by building 'prototypes' of the user interface. See Figure 3.

In general, the DP-department does not use DFDs and mini-specs for maintenance purposes, and the diagrams are not kept up to date when programs are modified.

4.2 Study 2: The Utility Company

The Utility Company is a regulated monopoly (concessionary company). It has an internal DP-department with approximately 50 employees responsible for development, maintenance, and daily operation of the company's information systems.

In connection with plans for replacing most of the existing information systems from the sixties and early seventies, top management decided in 1985 to reorganize the DP-department and introduce new standards for systems development. A project group was established to examine the DP-department's current practice and recommended, among other things, the introduction of Structured Analysis. The group emphasized three main advantages of Structured Analysis. It would (a) alleviate communication problems between users and DP-staff, (b) provide a sound basis for estimating and controlling costs, and (c) make systems maintenance easier by providing better technical documentation.

Q: Do you follow the four steps recommended by Yourdon...
A: You mean current physical model...
Q: Yes, current physical, current logical, new logical, new physical...
A: No, I don't work that way, because it seems to me that it's too cumbersome. It takes too long to get any results. That's not the way we do things...
Q: Results? Do you mean computer systems or descriptions?
A: Yes. We have some very active users, who want to see results—so, I don't follow the specified procedures. Actually I start with the new logical, because I know the system which we're supposed to alter. If I didn't know it, then I'd have to start with the current physical model..
Q: Does that mean, you start with designing a new system—by outlining a new logical model ?
A: Yes, that's what I do And, like I said, it's because I have the existing system in the back of my mind—with all the faults this may entail If I didn't know anything about it, then I'd have to go about it the other way around.
Q: Do you continue and construct a new physical model or do you only construct the new logical model?
A: Well, its always a little bit of both. While I'm designing the logical model, I can't help looking at what files exist—that means if I can see two processes at the lowest level reading the same file, then I change things, so it's only one process reading—that's of course because I also consider how the system is to be implemented.

Fig. 2. Extract from interview with Designer A in The Bank.

A: ..and then in a drawing I show how the screen layout is going to be...
Q: Then, what you in fact do is primarily to ask the user to decide about the screen layout? You don't ask them to go in and read the data flow diagrams?
A: No, but I'd be very happy if that's what they did—but it's hopeless, it really is. So that's why, when we—and remember it's only possible because I know the system so well, that things don't go all wrong—when we had coded everything, we asked the user to test it—on the basis of his experience, where he just imagined that he was working away on a mortgage application and came up afterwards with his comments.

Fig. 3. Extract from interview with Designer A in The Bank.

No alternatives to Structured Analysis were considered. As one of the designers pointed out, it would take too much time to examine and test other methods, and "management was eager to see some results." In 1986 management decided to adopt Structured Analysis as a company standard, compulsory for all new development projects. The DP-department has no special section supporting the use of methods and tools. Instead the company relies upon external consultants, who, among other things, teach new employees about Structured Analysis.

The tools of Structured Analysis—in particular the data flow diagrams—are used in the initial survey study as well as in the subsequent analysis phase.[3] The designers and the user representatives who are members of the project group use these diagrams when they communicate about the scope of the development effort and about the functional requirements.[4] In the survey study, the group makes a very rough sketch (about two to three levels) of both the existing and the new system, but in the analysis activity the group only builds a model of the new system. At the same time the group conducts a data analysis, using E/R modeling. The detailed model of the new system, worked out in the analysis phase, includes manual as well as automated processes, but it concentrates on the structure of the *computer* system. As a supplement to the data flow diagrams, processes at all levels are described by narrative text. It is hard to classify the model as either logical or physical. See Figures 4 and 5.

In addition, designers and programmers use the DFDs to estimate the size of the new computer system (in terms of the lines of code and in terms of programming effort) and to divide the system into program modules with clearly defined interfaces.

Structured Design is not used. Programming takes as its starting point the lowest level of data flow diagrams along with the corresponding mini-specifications. In projects where programmers are brought in during the programming phase, problems have occurred, however. The explanation given, was that the diagrams and the mini-specifications did not contain the necessary information for the programmer to implement the low-level processes. The user interface is designed using a fourth generation language. The programmer produces a screen layout, shows it to the future users of the system, and quickly make changes until the users are satisfied.

The designers agree that the use of DFDs has improved communication between DP-staff and (some of) the users. Users who are members of the project groups read—and sometimes even draw—DFDs without difficulty, but the diagrams cannot be used as a means of communication with users outside the project groups. The majority of the users are engineers and electricians who are accustomed to reading different types of technical diagrams. See Figure 6.

[3] The development activities are divided into six phases: survey, analysis (including functional analysis and data analysis), systems modeling, design, programming and implementation.

[4] Users participate in the project group right from the start. The head of the project group is always a high-ranking user.

Q: Do you start by constructing a current physical or current logical model?

B: No, no, no We've already described that [in the survey]—so there's no reason to do it again, is there?

Q: No, not unless you want to do it more thoroughly. You've only outlined the current system [in the survey].

B: Yes, and that's enough. Outlining the existing system means that everyone, who is at the highest level in the group, knows what we're working with.

...

Q: So, you start with the new... you start by modeling the new...

B. Yes, we start by modeling the new—and by continuing to work on this And we do it while taking into consideration how things are done today—how their work is organized and so on. We still haven't talked about whether this is to be on-line or whether it's paper-based manual procedures. We've had some problems with this in the beginning—in telling the users: "Well, I don't care if you think it should be computerized—I don't care if you think it's going to be in a screen layout—that's not necessarily so, and I'm not interested in it—it's not what we're working on"

Fig. 4. Extract from interview with Designer B from The Utility Company.

One designer also stressed that Structured Analysis had proved to be useful for estimating and controlling development costs. Since the introduction of Structured Analysis, estimates had improved considerably and this in turn had improved the image of the DP-department. Expectations about improved systems documentation, however, have not been fully met. Changes to programs made during or after implementation have not been recorded due to time pressure. Integrated computer-based tools are considered to be necessary if the diagrams and the mini-specs are to be kept up to data. We also noted that among designers having experience from different projects there is a difference of opinion on how Structured Analysis should be used and how it is actually used, see Figures 4 and 5.

4.3 Study 3: The Financial Institution

The Financial Institution is a mixed-sector computer service center established in 1980. The institution has a little more than 100 employees, most of them DP-people. Right from the beginning, top management decided to apply structured system development techniques—including structured design and structured programming—in order to increase productivity and to minimize maintenance. Their basic philosophy was to divide the system development task into small, manageable projects will well-defined interfaces. However, they lacked adequate tools for the analysis phase. At about the same time, [9] appeared, and in many ways this book provided the missing tools.

Q: When you started analyzing the various functions, did you start by modeling the existing system?
C: No.
Q: You started directly with the new one?
C: Yes.
Q: Did you distinguish between a new logical and a new physical model—or did you construct just one model?
C: No, we—and I'm not at all satisfied with what we did... in fact I think we did ourselves disservice by doing things the way we did—which is something that has been discussed a number of times. But our time schedule didn't leave us enough time to do anything about it, and at the same time people weren't really interested in doing anything about it.
Q: What do you think should have been done instead?
C: Yeah, well, at any rate, I think it's wrong to boast about using a model when you haven't really used it... In my opinion, if you want to get results from a certain method, you have to use the method in accordance with its own premises. You can't expect to get some results if you only use half of the prescribed procedures... And I don't think that much has been accomplished by using structured analysis in this project...

Fig. 5. Extract from interview with Designer C, who works on another project than Designer B in The Utility Company.

D: From our experience—especially in cooperating with the users—it's [the use of SA] been a success—at least this project has. Among some of the comments which we have received—we had a big meeting a couple of months ago with all the participants in the project—was that it was a good way of describing things. They understood it. . But it's worth noting that we work a lot with technicians, who are used to looking at diagrams and drawings.
Q. So, a data flow diagram doesn't scare them?
D. No, on the contrary—"finally here is something, which we can understand"—in stead of having to sit and read a lot of words which are difficult to understand.
Q. What do you mean, when you say that the users are other technicians?
B: Engineers, among others.
D: And electricians—people who are used to looking at diagrams... It's a way of describing things, that they are used to.

Fig. 6. Extract from interview with Designers B and D from The Utility Company.

The institution has no special section for enforcing standards and supporting the use of methods and tools. The use of structured techniques is mandatory, however, and all senior designers have considerable experience with their use. New employees learn the techniques by attending external one week courses and by working together with more experienced designers and programmers.

The tools of Structured Analysis are used for development as well as maintenance purposes. With respect to the former, designers use DFDs to model the new computer system and to break it down into a hierarchy of well-defined modules. This, in turn, enables them to divide the total development effort into more manageable sub tasks, which to a certain extent can be carried out independently of each other. In this way, they are also able to estimate rather precisely the time and effort needed for the subsequent programming of the system.

In general, the designers do not build a model of the current system; and they do not distinguish between logical and physical models of the new system. The models they build consist of mainly the automated procedures, and the models are used for internal communication only. Communication with users depends on screen layouts, informal drawings (e.g., of the structure of the system and the organizational context), and plain text.

To a large extent, this use of DFDs corresponds to the practice at The Bank, described in Section 4.1. Unlike The Bank, however, The Financial Institution in addition uses the DFDs, mini-specifications and DDs to produce technical systems documentation. The institution has committed itself to maintain the diagrams and specifications, even after implementation of the computer system. To ensure that they are correct and up to date, the organization makes extensive use of formal walkthroughs [49, 50], also during maintenance, see Figure 7. To keep the documentation "alive," a computer based documentation tool was considered necessary. Structured Design is only applied to the most complicated tasks, but the designers we talked to contemplated using it more often. The structure charts are not maintained, however.

4.4 Overview of Study Findings

On the basis of the study we can draw the following general conclusions regarding the introduction and use of Structured Analysis in the three development departments.

Introduction of Structured Analysis. None of the organizations have chosen to use Structured Analysis, in the sense that they have deliberately selected Structured Analysis out of a variety of available methods for systems design. No alternatives have been examined and tested in order to establish their relative merits. The organizations chose Structured Analysis because they had contact with other large organizations using the method, or because they were not aware of alternatives. More surprising, Structured Analysis is even today perceived as the only feasible method for systems analysis and

Q: Did you consider other methods? Were there other possibilities?

E: Everyone knew Jackson's method and the RAS model from Sweden and the SYSKON-model—they're all 'Victorian novels', which we all wanted to get rid off, because we're from the finance sector, where the mammoth works never were up-dated. And we weren't too good at it ourselves, because so much of the documentation had to be re-written. So we had to get rid of these 'Victorian novels' and secure just enough documentation so that it was usable—and not more than people would up-date... What we also wanted at this point was a vivid and handy documentation—where, once we documented something we also used it actively—all the way down into the programs—this means that if the documentation hasn't been updated, then the programs won't work. We draw on our documentation to write the user manuals—so, if there's a mistake in the documentation, there'll be something wrong in the user-manual. That's why we try to make it into something active, so it's always up-dated, because that's an end in itself...

Fig. 7. Extract from interview with Designer E in The Financial Institution.

design in a large business environment. When asked, no designer in the study could think of any real alternatives to Structured Analysis.

We attribute the widespread use of Structured Analysis in large Danish DP-departments to two factors: (a) The promises made by Structured Analysis of bringing 'system, control, and order' to the otherwise chaotic system development process appeal strongly to management. This is particularly prominent in the case of The Financial Institution, but also in the case of The Utility Company. (b) The tools of Structured Analysis—notably the DFDs—are clearly conceived of by the designers as being useful as a means of communication and as system documentation. Many of the designers in our study underline the conceptual simplicity of the method and the nice graphical lay-out of the DFDs, making them easy to comprehend and construct.

The Use of Structured Analysis. In general the designers and their organizations have a very pragmatic attitude towards using the method. They make use of the *tools* of Structured Analysis, primarily the DFDs, but also the Data Dictionary and in some cases decision tables or structured English, and they combine them freely with other tools as they see fit; but they do not comply with the *rules and procedures* of Structured Analysis.

According to Structured Analysis the designer must build four different models of the system during the design activity: the current physical model, the current logical (essential) model, the new logical (essential) model, and the new physical model. In general, however, designers in the study only produce *one* model of the system—namely, a model of the new (computer)

Table I. Models Developed

	The Bank	The Utility Company	The Financial Institution
current physical model	no	yes, but only in the survey activity	no
current logical model	no	no	no
new logical model	no	no	no
new physical model	yes, but manual procedures are not described by the model	yes	yes, but manual procedures are not described by the model

system—and they do not distinguish between logical and physical data flow diagrams:

—The designers do not analyze the current system by first constructing a current physical model and then deriving a logical equivalent. In fact, in most cases, they do not build a model of the current system at all. Instead, they rely upon their own intuitive understanding of the system and upon verbal descriptions and informal discussions with users.

—The designers do not design the new system by first building a new logical model and then deriving a physical model from the logical model by establishing the man-machine boundary. Instead, they construct one model of the new system, which may be described as something between a logical and a physical model. The model focuses on the structure of the computer system. See Table I.

The designers clearly view themselves as technical designers of *computer* systems, not as designers of work organization, jobs, and work tasks. As a result they concentrate on specifying input and output data while leaving decisions about the redesign of the work system in general to others. This also explains why their descriptions of the manual procedures and actual working practices are rather limited.

How the *tools* of Structured Analysis are actually used and combined with other tools, and for what purpose, vary a great deal from one organization to another, and even from project to project. In Table II we have summarized how the DFDs are used in the three companies.

These difference regarding the use of Structured Analysis can be explained by referring to the *context of application*. At The Financial Institution senior management introduced Structured Analysis to increase *managerial control* and obtain a high level of *standardization* with regard to systems documentation. At The Bank, however, the individual project groups decide whether or not to use the tools of Structured Analysis, and the group members use them for *internal communication and coordination* purposes only. The Utility Company uses the tools primarily to facilitate *user participation* by improving communication between users and DP-staff. This is possible because most

Table II. Use of Data Flow Diagrams and Other Tools

	The Bank	The Utility Company	The Financial Institution
DFDs used when communicating with users	no	yes, within the project group	no
DFDs used for communication among designers and programmers	yes	yes	yes
DFDs used for breaking down software into modules	yes	yes	yes
DFDs kept up-to-date for maintenance purposes	no	yes, but changes during or after implementation are not recorded	yes
The use of DFDs is mandatory	no	yes, on new projects	yes
Structure Charts (Structured Design) are applied	only in very complicated cases (the Structure Charts are not kept up-to-date)	no	only in the most complicated cases (the Structure Charts are not kept up-to-date)
Other tools used	• E/R diagrams • Screen layouts • Prototypes of user interface • Informal descriptions	• E/R diagrams • Screen layouts • Prototypes of user interface • Informal descriptions	• E/R diagrams • Screen layouts • Informal descriptions • Informal drawings (e.g. organization charts)

of the users are engineers and technicians who are used to interpreting and constructing technical diagrams.

We find it more correct to say that the designers in our study use data flow diagrams, the Data Dictionary and mini-specifications as tools—for various purposes and in combination with other tools such as E/R diagrams, screen layouts, prototypes, organization charts, and plain text—rather than to say they employ Structured Analysis as the method for analyzing and designing information systems. In other words, when a company states that it uses Structured Analysis, it can mean many different things.

5. DISCUSSION

In conclusion, we would like to elaborate on the use of system development methods in practice as opposed to how they are presented in textbooks and scientific literature. We are especially interested in the reasons *why* the use of Structured Analysis differs from the method as theorized. To what extent do circumstances relating to the introduction of the method and the management of the development activities explain this difference, and to what extent is it caused by innate methodological problems?

To answer these questions requires more extensive, longitudinal investigations, not only of the development process, itself, but also of the implementation and use of the produced systems. From our studies we expect that some methodological problems will show up during the development phase and manifest themselves as designers' problems, i.e., how to communicate with users? (See Section 3.2). Other types of problems, will probably not show up until the new system is implemented in the organization—and they may primarily present themselves as users' problems, i.e., how to deal with the system in situations not anticipated by the designers? (See Section 3.1.) These phases of implementation and use are not included in our present study.

In the following discussion we, therefore, do not pretend to provide a definitive answer to the question why Structured Analysis is not used 'by the book.' Rather, we limit ourselves to discussing some possible explanations, to be tested in later studies.

5.1 The Use of Methods

Our study of the application of the Structured Analysis method shows that the designers only use the method partially, and that the parts chosen differ from organization to organization and even from project to project.

One obvious reason might be that the projects studied were too small to justify the complete implementation of this rather elaborate method. The projects in the study fall, however, well within the size limits recommended for the use of structured techniques, including Structured Analysis. Yourdon strongly recommends the use of structured techniques in projects ranging from ten thousand to one million lines of code. He also finds the techniques useful for smaller projects but he admits that their implementation may represent 'unnecessarily heavy artillery,' and that their use may not speed up development time at all. [49, pp. 3–5].

Another reason might be inadequate training and weak management support. There may be some truth to this. With regard to the training and building up of in-house competence, none of the companies in the study had internal departments or specialists responsible for the implementation of Structured Analysis. Instead, they rely on external consultants to teach new employees and provide advice in projects where the method is adopted. On the other hand, DP-management—at least in The Utility Company and The Financial Institution—fully supports the use of Structured Analysis.

A third reason may have to do with organizational politics. If the data processing staff constitutes an interest group within the organization that is eager to preserve and enhance the importance and power of the DP-department, then practices likely to increase the size, the budget, and the independence of the DP-staff will be viewed positively. Whereas any attempts to decrease the staffing or the equipment, or to increase the control of the DP-departments' activities are clearly considered to be disadvantageous.

The failure to construct models of both the new and the old systems and the reluctance to distinguish clearly between physical and logical aspects may thus be explained in terms of differing interests and power relationships between the user and the data processing department. By not analyzing the existing system and by conflating the physical and logical models of the new system, it may be easier for the DP-department to justify the acquisition of new resources. For example, they can make arguments of the sort, "if you want such and such new feature, then we will have to buy such and such new equipment or software," even though a careful analysis of either the old or new systems might reveal that such a new acquisition was not really necessary. We do not have enough empirical evidence to either support or reject this explanation. However we find it is an interesting hypothesis worthy of future investigation.

Finally, we would like to suggest a fourth reason, having to do—not so much with the specific circumstances of the implementation—as with the substance of the method itself: The limited use of Structured Analysis may reflect inherent methodological weaknesses. The results of the explorative study confirm the validity and relevance of at least some of the theoretical criticism, as discussed in Section 3.

When we look at the way the designers carry out their work, we observe four major deviations from the prescribed procedures: (1) the designers do not use data flow diagrams when communicating with users, (2) their diagrams do not comprise manual procedures, (3) the designers limit the number of models they construct to one (a model of the new system), and (4) they always supplement this model with other types of descriptions, diagrams, or prototypes. This seems to suggest that the tools and procedures of Structured Analysis are inadequate and cumbersome to apply in practice.

Communicating with users. Although DeMarco and Yourdon claim that one of the major advantages of Structured Analysis is that it improves communication between designers and users, the designers at The Bank and at The Financial Institution consistently told us that data flow diagrams were not suitable for communicating with users. We suspect that this is due to the fact that the diagrams reflect an information processing view, which most users presumably are unfamiliar with (see Section 3.2). Only the technicians and engineers at The Utility Company were willing and able to try and grasp the meaning of the diagrams, see Table II.

Modeling work processes. Structured Analysis is presented as a general method for describing and analyzing 'information processing systems,' whether manual or automated. However, at The Bank and at The Financial Institution the designers excluded manual procedures from their system models, and although the models developed at The Utility Company included some manual procedures, their focus was clearly on the *computer* system, see Table I. By restricting the use of DFDs to specify the computerized procedures, the designers avoid the problematic issue of describing work as data processing. While it makes sense to understand a computer system in terms of data processes and data flows, it is doubtful whether it makes much sense to understand work in these same terms (see Section 3.1). The practice of focussing on the computer system may create new problems because the organizational and behavioral aspects of work are ignored.

Limiting the number of models. Regarding the third observation, it is clear that the construction of a detailed and comprehensive system model is enormously time-consuming. At the same time most DP-departments and systems designers are under heavy pressure to speed up delivery of new applications. Therefore, designers and DP-managers are inclined to skip any activity which they believe 'returns too little on investment.'

The designers find it useful to build a model of the *new* (computer) system in order to document its overall structure, see Table II. They particularly like the data flow diagrams because they are concise graphical representations and give a good general view of the system, compared to traditional functional specifications. However, they do not think that modeling the *current* system is worth the time and effort. It is not at all clear to them what they should do with such a model. Those designers who have tried it, have missed criteria for deciding on the scope of the model and on the level of detail. In their opinion the model merely contains a lot of irrelevant information, which is of very little help in designing the new system. One of the few designers (from The Bank) who has tried to construct a current physical model told us, for instance, that the project group gave up after three months of drawing diagrams. His opinion was that "the improvement in our general understanding [of the system] wasn't worth a dime."

Both Palmer and McMenamin [38] and Yourdon [50] acknowledge this problem, which Palmer and McMenamin refer to as the 'the current physical tarpit':

> "The single most common and most deadly result of failing to understand the method or the system is what we call the current physical tarpit. It is the reason that unsuccessful structured analysis projects usually fail sometime during the study of the existing system: Members of these projects nearly always take an unacceptable length of time to study the system, because they put far more physical detail into the current physical model than is needed to develop a new system." [38, p. 351].

Palmer and McMenamin try to circumvent the problem by introducing a more iterative way of working (called 'blitzing'), while Yourdon simply proposes skipping the model of the current system "if at all possible" because it is "a politically dangerous activity" (see Section 2). We agree with Palmer and McMenamin. One has to study the existing work process to understand the need for changes, no matter how scarce project time is. However, the question is whether Structured Analysis is the right way to address this problem (see Section 3.1).

In addition to this, the designers are very sceptic about distinguishing between logical and physical models. They stress that what they need is not 'a pretty theoretical model,' but a model of the system to be implemented, containing the necessary information about modules, files, man-machine boundary, etc. They obviously consider the whole idea to be a theoretical and academic discussion without practical relevance. The reason may be that DP-departments today rarely develop new systems from scratch. The designers know, for instance, right from the start that their programs have to run on an IBM mainframe under CICS, that they have to be written in PL/1 and that they must interface with a specific older system. In general, the designers are well aware of the multiple constraints in a particular organizational context on their design choices, i.e., values and norms, power relations, work demarcations, and people. Under such circumstances, the idea of first making a 'logical model' and postponing considerations about 'implementation details,' including the man-machine boundary, until the final step of physical modeling seems odd and hard to justify (see Section 3.1).

Using additional tools. Concerning the fourth observation, the designers give two reasons for the use of supplementary tools. First, the DFDs, the Data Dictionary, and the mini-specs are unable to represent all relevant aspects, technical as well as organizational. Among the many aspects, which the designers have mentioned are data structure, timing of events, access rights, user interfaces, business procedures, and organization structure. Second, the users are in general unable and even reluctant to try and understand the DFDs. This creates serious problems with regard to design evaluation, because the designers in most cases need feedback from users (see Section 3.1).

To describe data structure the designers use E/R-diagrams (now included in the method [38, 50]), to describe user interfaces they draw screen layouts and implement small 'prototypes,' to describe organization structure they draw traditional organizational charts, and to describe timing, access rights, business procedures, etc., they use plain text, some times annotated to the DFD's. In addition they rely heavily on close personal contact and informal discussions with users. By doing so, they build a much richer picture of the new system, and may thus be able to avoid some of the pitfalls of Structured Analysis. We were unable to determine to what degree the designers succeed in this.

In conclusion, our interpretation of what happens in practice is that experienced designers—instead of following the rules and procedures of Structured Analysis—pick and choose among the various formalisms given in the method, adapt them for their own purposes, and integrate them into their own design processes. By using the tools in their own way and very often together with other, supplementary tools, the designers avoid or compensate for some of the method's deficiencies and limitations, especially with respect to describing the user-interface and the human-computer interaction.

This interpretation of what actually happens when real-life designers apply Structured Analysis is consistent with 'The Theory Building View' on program development put forward by Naur [33]. Naur suggests that programming (encompassing both the design process and implementation of programmed solutions) must be regarded as a 'theory building activity' as defined by Ryle [39]. As an essential part of the activity, the programmer achieves a deep intuitive insight into the problem at hand and develops a mental image, a 'theory,' of the programmed solution. A main claim of the 'theory building view' of programming is that an essential part of any program, the theory of it, is a body of knowledge that could not conceivably be fully expressed, but is inextricably bound to human beings. This means that the knowledge possessed by the programmer in an essential manner transcends what is recorded in the program text, user documentation, and additional documentation such as specifications. The programmer having the theory of a program will be able to (1) explain how the solution relates to the real-world problems that it helps to handle, (2) explain why each part of the program is what it is, in other words is able to support the actual program text with a justification of some sort, and (3) respond constructively to any demand for a modification of the program so as to support 'the affairs of the world' in a new manner [33]. In this perspective the programmer's skill and intuition is the pivot upon which the program development activity turns, and the use of specific principles, rules, and tools plays a secondary, auxiliary role.

Seen from this perspective the core of systems design is that the designers achieve a certain kind of insight, a 'theory' which encompasses technical as well as social aspects of the situation at hand. This knowledge is the designers' immediate possession. Any specification or 'system model' must be considered as subordinate products. Consequently, the quality of a given method—its various techniques and tools—must be judged on their ability to support the designers in building up their 'theory.'

5.2 Implications for Future Research

Our findings suggest that there is a gap between the way systems development is portrayed in the mainstream of scientific and technical literature and the way it is carried out in real life. This is not only the case for textbooks on proprietary methods such as Structured Analysis, but also pertains to much of the academic critique of these methods. One reason for this is that researchers attach too much importance to tools, methods and principles, and pay too little attention to the behavioral and social aspects associated with systems development, including the designer's skill and know how and the cooperation between designers and users.

Much of the research in system development is essentially normative and speculative. Researchers seek to develop new methods and tools in order to improve practice. In many cases, however, their research rests on a rather simple—and often misleading—understanding of the nature of system development, and it suffers from a lack of firm empirical foundation. As a result their recommendations may well prove to be infeasible from a practical point of view.

How is system development organized in real life, how do designers and programmers actually use methods and tools, and how do they cope with ambiguity, uncertainty, and conflicting interests? We know very little about the answers to these questions. A number of surveys concerning the use of methods and tools have been carried out. However, due to their quantitative nature they only provide very restricted or superficial answers to the questions mentioned above. Furthermore, the few empirical studies comparing and evaluating the use of different system development methods are primarily based upon, what you might call, 'laboratory experiments'. One way of conducting such an experiment is to ask a number of students to solve a given well-defined problem using different methods, 'measure' their performance, and compare the results (e.g., [2, 15]). Another way is to ask a number of researchers to solve a common, invented problem using their own method (e.g., [35]). Although this research yields valuable insights into the virtues and shortcomings of the tested methods, it tells us very little about how system development is practiced.

If we want to improve practice is some way we must, first of all, acquire a more comprehensive understanding of the work processes of real-life design and programming, including not only technical aspects, but also behavioral and sociological aspects. To obtain such detailed insight we would have to study the work processes in question more directly, e.g., through participant observation, video analysis, and 'document analysis' of system descriptions as well as the final systems. In short, we need qualitative 'field studies' highlighting such issues as the role of skill and experience, the use of formalisms and other kinds of tools, alternative forms of cooperation and communication, and division of labor among designers, programmers, and various groups of users. These methods are in widespread use and have played a dominant role in anthropology and in the sociology of industry and work. It seems obvious to use these methods as inspiration for research in the development of information systems. These studies should investigate not only how systems are developed, but also how they function after being installed. Only in this way, is it possible to link the question of systems quality with the use of specific development methods.

ACKNOWLEDGMENTS

We are indebted to the Danish companies and their employees who candidly participated in the study, and we thank Niveau Management ApS for providing contacts to users of Structured Analysis. We also thank Hasse Clausen, Erik Frøkjaer, Erling Havn, Finn Kensing, Philip Kraft, and the referees for comments on earlier drafts of this paper.

REFERENCES

1. ASTLEY, W. G. AND VAN DE VEN, A. H. Central perspectives and debates in organization theory. *Adm. Sci. Quart.*, *28*, 2 (June 1983), 245–273.
2. BOEHM, B. W., TERENCE, E. G., AND SEEWALDT, T. Prototyping versus specifying: a multiproject experiment. *IEEE Trans. Softw. Eng. 10*, 3 (May 1984), 290–303.
3. BØDKER, S. AND GRØNBAEK, K. Cooperative prototyping: users and designers in mutual activity. *Int. J. Man-Mach. Stud. 34*, 3 (Mar. 1991) 453–478.
4. BØDKER, S., EHN, P., KYNG, M., KAMMERSGAARD J., AND SUNDBLAD, Y. A utopian experience. In *Computers and Democracy. A Scandinavian Alternative*, G. Bjerknes, P. Ehn, and M. Kyng, (Eds.), Avebury, Aldershot, 1987.
5. BUBENKO, J. Problems and unclear issues with hierarchical business activity and data flow modelling. SYSLAB Working Paper, No. 134, Stockholm, 1988.
6. BØDKER, K. Analysis and design of computer systems supporting complex administrative work processes. *Off. Technol. People 4*, 1 (Jan. 1989), 75–89.
7. CHAPIN, N. Structured analysis and structured design: an overview. In *Systems Analysis and Design: A Foundation for the 1980's*, W. W. Cotterman, J. D. Couger, N. L. Enger, and F. Harold, Eds., North Holland, New York, 1981.
8. COUGER, J. D. Evolution of business system analysis techniques. *Comput. Surv. 5*, 3 (Sept. 1973), 167–198.

9. DEMARCO, T. *Structured Analysis and System Specification.* Yourdon Press, New York, 1978.
10. DEMARCO, T. Structured analysis and system specification. In *Classics in Software Engineering*, E. Yourdon, Ed., Yourdon Press, New York, 1979.
11. DIJKSTRA, E. W *Notes on Structured Progamming.* Technical Univ. Eindhoven, 1970.
12. EHN, P. *Work-Oriented Design of Computer Artifacts* Arbetslivcentrum, Stockholm, 1988.
13. FISCHER, P., STRATMANN, W., LUNDSGAARDE, H. AND STEELE, D. User reaction to PROMIS. *Comput. Appl. Medical Care, 3* (1980), 1722–1739.
14. FLOYD, C. A systematic look at prototyping. In *Approaches to Prototyping*, R. Budde, K. Kuhlen Kamp, L. Mathiassen, and H. Züllighoven, Eds., Springer Verlag, Berlin, 1984
15. FLOYD, C. A comparative evaluation of system development methods. In *Information Systems Design Methodologies: Improving the Practice*, T. W. Olle, H. G. Sol, and A A. Verrijn-Stuart, Eds., North-Holland, Amsterdam, 1986.
16. FLOYD, C. Outline of a paradigm change in software engineering, In *Computers and Democracy*, G. Bjerknes, P Ehn, and M. Kyng, Eds., Avebury, Aldershot, 1987.
17 FRIEDMAN, A. *Industry & Labour.* MacMillan Press, London, 1977.
18. GALLAGHER, J., KRAUT, R., AND EGIDO, C., EDS. *Intellectual Teamwork.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1990.
19. GANE, C. AND SARSON, T. *Structured System Analysis. Tools and Techniques.* Prentice-Hall, Englewood Cliffs, NJ, 1979.
20. GREENBAUM, J. AND KYNG, M., EDS. *Design at Work: Cooperative Design of Computer Systems.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.
21 GUINDON, R. Designing the design process: exploiting opportunistic thoughts. *Hum. Comput. Interaction 5*, 2/3 (1990), 305–344.
22. JØRGENSEN, A. H. On the psychology of prototyping. In *Approaches to Prototyping*, R. Budde, K. Kuhlenkamp, L. Matthiassen, and H. Zülighoven, Eds., Springer Verlag, Berlin, 1984.
23. KEEN, P. G W. Information systems and organizational change *Commun. ACM 24*, 1 (Jan. 1981), 24–33
24. KENSING, F. Towards evaluation of methods for property determination. In *Beyond Productivity: Information Systems Development for Organizational Effectiveness*, Th. M. A. Bemelmans, Ed., North-Holland, Amsterdam, 1984.
25. KENSING, F. AND HALSKOV M. K Generating visions: Future Workshops and metaphorical design. In *Design at Work: Cooperative Design of Computer Systems*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.
26. KRAFT, P. AND DUBNOFF, S. Job content, fragmentation, and control in computer software work. *Ind. Rel. 25*, 2 (Spring 1986), 184–196.
27. KUSTERER, K. *Know How on the Job.* Westview Press, Boulder, Col., 1978.
28. LANZARA, G. F. The design process: frames, metaphors, and games In *Systems Design For, With, and By the Users*, V. Briefs, C. Ciborra, and L. Schneider, Eds., North-Holland, Amsterdam, 1983.
29. LUNDEBERG, M., GOLDKUHL, G., AND NILSSON, A. *Systemering. Studentlitteratur.* Lund, 1978. (English version: *Information Systems Development—A Systematic Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1981.)
30. MADSEN, K. H. Breakthrough by breakdown: metaphors and structured domains In *Systems Development for Human Progress*, H. Klein and K. Kumar, Eds , North-Holland, Amsterdam, 1989.
31. MALHOTRA, A , THOMAS, J. C., CARROLL, J. M., AND MILLER, L. A. Cognitive processes in design. *Int. J. Man-Mach. Stud. 12* 2 (Feb. 1980), 119–140.
32. NAUR, P. An experiment on program development. *BIT 12*, 3 (1972), 347–365.
33. NAUR, P. Programming as theory building. *Microprocess. Microprogram. 15* (1985), 253–261.
34. NECCO, C. R., GORDON, C. L., AND TSAI, N. W. Systems analysis and design· current practices. *MIS Q.* (Dec. 1987), 461–476.
35. OLLE, T W., SOL, H. G., AND VERRIJN-STUART, A. A., EDS. *Information Systems Design Methodologies: A Comparative Review.* North-Holland, Amsterdam, 1982.

36. ORR, J. Narratives at work. Story telling as cooperative diagnostic activity. In *Proceedings of the Conference on Computer Supported Cooperative Work* (Austin, Texas, Dec. 1986), pp. 62–72.
37. ORR, K. T. *Structured Systems Development*. Yourdon Press, New York, 1977.
38. PALMER, J. AND MCMENAMIN, S. *Essential Systems Analysis*. Yourdon Press/Prentice-Hall, New York, 1984.
39. RYLE, G. *The Concept of Mind*. Penguin Books, 1963.
40. ROSS, D. T. Structured analysis (SA): A language for communicating ideas. *IEEE Trans. Softw. Eng. 3*, 1 (Jan. 1977), 16–34.
41. STEVENS, W. P., MYERS, G. J. AND CONSTANTINE, L. L. Structured design. *IBM Syst. J. 13*, 2 (1974), 115–139.
42. SUCHMAN, L. Office procedures as practical action: models of work and system design. *ACM Trans. Office Inf. Syst. 1*, 4 (Oct. 1983), 320–328.
43. SUCHMAN, L. *Plans and Situated Action: The Problem of Human-Machine Communication*. Cambridge University Press, Cambridge, Mass., 1987.
44. TAYLOR, F. W. *Scientific Management*. New York and London, 1947. This is a single-volume edition of *Shop Management* (1903), *Principles of Scientific Managements* (1911), and *Hearings Before Special Committee of the House of Representatives to Investigate the Taylor and Other Systems of Shop Management* (1912).
45. VITALARI, N. P. A critical assessment of structured analysis methods: a psychological perspective. In *Beyond Productivity: Informations Systems Development for Organizational Effectiveness*, Bemelmans, Ed., North-Holland, Amsterdam, 1984.
46. WINOGRAD, T. AND FLORES, F. *Understanding Computers and Cognition. A New Foundation for Design*. Ablex, Norwood, NJ, 1986.
47. WIRTH, N. Program development by stepwise refinement, *Commun. ACM 14*, 4 (Apr. 1971), 221–227.
48. WYNN, E. *Office Conversations as an Information Medium*. University of California, Berkeley, 1979.
49. YOURDON, E. *Managing the Systems Life Cycle*. Yourdon Press, New York, 1982.
50. YOURDON, E. *Modern Structured Analysis*. Yourdon Press/Prentice Hall, New York, 1989.
51. YOURDON, E. AND CONSTANTINE, L. L.. *Structured Design*. Yourdon Press/Prentice-Hall, Englewood Cliffs, NJ, 1979.

Received December 1989; revised October 1990 and October 1991; accepted July 1992