Serie Research Memoranda

The Software crisis in the Netherlands: a Survey Report

Martin Boogaard
Edu R.K. Spoor

Research Memorandum 1994-21

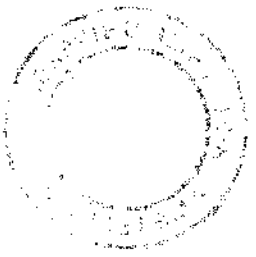
July 1994

vrije Universiteit *amsterdam*

Faculty of Economics, Business Administration and Econometrics

Department of Information Systems

THE SOFTWARE CRISIS IN THE NETHERLANDS: A SURVEY REPORT



Martin Boogaard
Edu R.K. Spoor

July 1994

Research Memorandum 1994-21

The Software Crisis in The Netherlands: A Survey Report

Martin Boogaard and Edu R.K. Spoor

Vrije Universiteit, Faculty of Economics and Econometrics, Department of Information Systems, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands

Abstract

In IS literature, the Software Crisis is often mentioned. Since the Software Crisis was openly recognized in 1968, many solutions have been proposed to solve the problems of the Software Crisis. As can be concluded from the attention devoted to the Software Crisis nowadays in both theory and practice, the problems still exist and no comprehensive solution have been found. Notwithstanding the concern for the phenomenon, it is remarkable that an in-depth analysis of the Software Crisis is not at hand. Generally, the Software Crisis is described by some introductory explication and subsequently used as an incentive for research or "solution." Therefore, insight in the problems, causes, extent, existence, and persistence of the Software Crisis is lacking. This paper provides an in-depth analysis of the Software Crisis by means of an extensive content analysis reflecting the advent of the problems and by a description and the results of a sample survey research carried out in The Netherlands. In doing so, the paper contributes to knowledge pertaining to the Software Crisis and enables to assess the contribution and the restrictions of solutions proposed. Furthermore, based on the results of the study it is argued that the concept of flexibility seems fruitful to apply to the Software Crisis.

1. INTRODUCTION

In today's literature concerning ISs, the Software Crisis is frequently referred to. Many authors take the Software Crisis as the inducement for their research (see, e.g., Vinig 1993; Brinkkemper 1990; McClure 1989). Usually the explication of the Software Crisis remains confined to some introductory comments. The remainder of the discussion is devoted to a description of a certain solution. For instance, in the preface of her book *CASE is Software Automation* McClure (1989) argues that CASE technology "... is a very practical answer to the 25-year-old Software Crisis." Furthermore, at the end of Chapter 1 she briefly discusses several other answers to the Software Crisis, i.e., structured techniques, and productivity. The remaining chapters address the components of a CASE system, the use of CASE, and the future of software developments. A similar scheme can also be found in many articles and books concerning Software Engineering (see, e.g., Sommerville 1989; Macro and Buxton 1987; Steward 1987). To conclude, the Software Crisis as such remains largely undiscussed in IS literature and its existence is never called in question.

On the other hand, many researchers focus on only one of the problems that together constitute the Software Crisis. The outcomes of these studies are often used to indicate the

significance of the Software Crisis. For instance, the -- often cited but rather dated -- result of the study of Lientz *et al.* (1978) and Lientz and Swanson (1980) that IS maintenance consumes about 50 percent of the total EDP resources is often quoted to signify the Software Crisis. However, focusing on one problem does not reveal overall insight in the extent and the underlying cause(s) of the problems of the Software Crisis.

In addition to the substantial attention in IS literature, the Software Crisis also turns out to be a commercially interesting phenomenon. In a few advertisements, the Software Crisis is mentioned in order to market products or services. The following passages -- translated from Dutch -- are recent examples of such practice. The first passage belongs to an advertisement aimed at selling a CASE-tool (the real name of the tool is replaced by 'X') while the second passage is taken from an advertisement concerning a book.

- "Do you leave the Software Crisis free play or ... do you choose for 'X' too? [...] In spite off all best intentions, many IS developers suffer from a Software Crisis. Applications backlogs, superseded techniques, maintenance problems, and lack of communication between users and IS developers form the principal part. 'X' provides the most modern tool to tackle or to prevent the Software Crisis. [...] What do you think, stick to the Software Crisis or ... ?"

- "Did you know that only 5% of all development projects directly results in working systems? The rest is sent back for reconstruction, will never be used after delivery (already obsolete), or will never be finished! Object oriented technology is currently recommended as the answer to the existing Software Crisis."

It is interesting to note that the term Software Crisis is used credulously. The existence and current extent of the Software Crisis is not confirmed with empirical evidence. Therefore, legitimate questions like the following remain for the greater part unheeded:

- Does the Software Crisis really exist or is the term used melodramatically?
- What is the current extent of the problems that constitute the Software Crisis?
- What is the cause or are the causes of the Software Crisis?
- What was the contribution of proposed solutions to the extent of the Software Crisis?
- How can the persistence of the Software Crisis be explained?

Another interesting observation that can be derived from the utilization of the term in both literature and commerce, is that the target group of these publications usually consists of IS developers. The question whether this group experiences the problems as presumed by the Software Crisis is never addressed. It is conceivable that this audience perceive the Software Crisis quite differently than stated in literature. For example, a frequently heard response of IS developers to the signalized IS maintenance problem is "... it is not our problem, it is our job." Furthermore, the perception of the problems and causes of the Software Crisis of other involved parties such as users and general management could differ substantially. Therefore, it seems relevant to distinguish between the different participants and their points of view.

The above observations form the starting-point of an in-depth analysis of the Software Crisis. The current paper provides such an analysis of the Software Crisis in order to retrieve answers to the questions stated above. Section 2 contains a historical overview of literature concerning the Software Crisis. Based on an extensive content analysis, the

evolution of the Software Crisis is presented reflecting the emergence of the problems, the signalized causes and the proposed solutions. The remainder of the paper primarily consists of a description and the results of a sample survey research carried out in The Netherlands. Section 3 describes the process of data collection and the characteristics of the respondents. In the next section, the results of this empirical research are presented and analyzed. Finally, Section 5 supplies some explanations of the persistence of the Software Crisis that proceed from the sample survey research and several additional interviews.

2. THE SOFTWARE CRISIS REVISITED

Although it seems quite normal to talk about the Software Crisis today, the admission of the existence of something as a Software Crisis was not without any reluctance. In the early days of IT, each possible caveat was seen as a new technical challenge that would be solved in a short time by means of new hardware technology. During the fifties, programming -- the term software was not in use at that time -- was considered a lacklustre activity because its only rationale was the existence of the computer. Moreover, the ingenuous persuasion was that when more powerful computers were on hand, the programming problems that primarily cohered with the optimization of processing as a result of the slowness and the limited memory capacities of the computer, would be eliminated. In fact, at that time the IT field was quite averse concerning issues that might agitate the IT idyll. To quote Dijkstra (1972) "... to talk about a Software Crisis was blasphemy."

However, the developments in hardware technology during the next decades showed that the above expectation was totally refuted. Instead of decreasing or even eliminating the programming problems by several orders of more powerful computers, the problems concerning programming increased exponentially. The existing idealism concerning IT resulting in rapid but uncontrollable utilization of the technology, the emergence of an enormous amount of hardware-specific programming languages -- sometimes referred to as the Tower of Babel --, and the overwhelming amount of rhapsodic accounts in literature turned out to be rather one-sided. Hence a more objective and open view was required.

The rapid developments in hardware technology and the resulting dramatic decrease of cost had the following three consequences:

1) The cutbacks in the absolute cost of computers implied that smaller organizations or smaller divisions within organizations could ponder the purchase of a computer.
2) The cost/performance ratio reductions affected the economic viability of more complicated applications as well as relatively simple applications with lower envisioned profits.
3) The cost reductions of hardware on the one hand with an increase of the cost of software on the other hand resulted in a significant shift in the total cost structure of a computer application.

These cost tendencies resulted in an enormous growth in the number of feasible computer applications. This in turn aroused a tremendous rise in demand for IS development personnel. Consequently, serious shortages in both IS programmers and managers appeared (see, e.g., Bylinski 1967; Brandon 1968). In recruiting personnel logical and reasoning capabilities were all important to enter the field -- like in IBM's

programmer aptitude test -- while other selection criteria were neglected. The underlying assumption was that anyone with these basic abilities could be trained to become a programmer. However, the lack of formal programming rules and procedures resulted in a long learning curve and a rather prolonged, unproductive period of on-the-job training which was considered essential. People who had acquired competence and practice with respect to the development of ISs gained extraordinary market power and independence. Programmers were soon internally or externally promoted to managerial positions just because they were good programmers and not because of their managerial capabilities. Consequently, many IS development processes suffered from poor management. This vexatious state was strengthened by the lack of software management theory, resulting in a lack of both software management instruction and specialized IS development managers. In addition, immediate promotion of a person who is remarkably competent at a particular level to a higher level with additional responsibility again, and again will finally lead to promotion to his level of incompetence. This is also known as the Peter Principle (see, e.g., Freeman 1987; Naur *et al.* 1976). By the late 1960s, the management of IS development had become an important problem area, primarily as a result of the immaturity of the IS field.

Due to fewer cost and performance restrictions of hardware, more complex and larger ISs could be aspired. ISs were no longer the simple representations of the existing, mostly administrative manual activities. Instead the complexity increased because many of these ISs were aimed at integrating systems from various divisions of the organization and provided new functions to support other more complex business processes like planning, decision making, and strategy formulation. The social impact of ISs on organizations grew simultaneously with the growth of the application of IT and also became one of the acknowledged problem realms.

The combination of managerial problems, the rapid growing demand for ISs, and the increasing complexity and size of IS brought about severe problems concerning the development, management and use of ISs such as excessive cost overruns, delayed releases, and miscarried projects (see, e.g., Freeman 1987; Brooks 1975). This resulted in the yearn to enhance control over and insight in IS development in particular and IT in general. These kind of problems came to be known as the Software Crisis. Hence by the late sixties and the early seventies, software became gradually the leading component but also the limiting factor of computerization instead of hardware technology.

Generally, the Conference on Software Engineering in Garmisch, October 1968, is considered the turning point because this conference "... created a sensation as there occurred the first open admission of the Software Crisis" (see Dijkstra 1972). However, before 1968 several authors in the IS field were already aware of future problems other than technological or hardware oriented. At the 1961 MIT conference for instance, Sayre stated that it was becoming noticeable that the limitation of the utilization of IT was beginning to be the software instead of the processing power of the computer (see Greenberger 1962). At the same conference, Brown addressed this problem as the 'programming bottleneck.' The programming problems they discussed were mainly a result of the hardware dependence of the software developed. Furthermore, the problems with respect to IS programmers and managers were already addressed by several authors, for instance, Bylinski (1967) and Brandon (1968). In an article in Data Systems of July 1967, Brandon indicated several additional problems, e.g., top management ignorance and human communication problems. According to Brandon these problems caused the perceived 40 percent IS failure rate at that time. He tagged this situation as "the dark side

of data processing" and indicated the need for an examination of the reasons for the failures.

Although there were several signals of the existence of the Software Crisis before 1968, the main outcome of the Garmisch Conference was clear; *the realization of the full magnitude of the Software Crisis* (see Naur *et al.* 1976, p. 145)[1]. During the conference, considerable time was spent on the debate on what some participants labelled the "Software Crisis" or the "Software Gap." The invited conference participants -- about 50 international experts in the field -- had ample different perspectives on the significance and the extent of the problems discussed. In general, the participants determined the tendency of a widening gap between what was hoped for from complex ISs and what was achieved. The notice of the Software Crisis was provoked by several malfunctioning ISs which brought about some public catastrophes. For example, several references were made to serious air crashes as a result of IS failures. Furthermore, the conference participants experienced three disappointing project features while developing large ISs. First, the projects overrun their prognosticated deadlines largely. Second, the development cost of ISs overgrew projects' budgets. Third, the resulting ISs did not match the (performance) requirements stated on beforehand. However, several participants believed that the extent of the problems was highly exaggerated and that the conference was too pessimistic because numerous ISs were functioning satisfiable. But generally, the participants admitted that there was indeed a Software Crisis (p. 78).

The discussion continued with the determination of the underlying reasons for the existing productivity problems. David and Fraser (p. 79) summarized the problems of the Software Crisis as "... slipped schedules, extensive rewriting, much lost effort, large numbers of bugs, and an inflexible and unwieldily product. It is unlikely that such a product can ever brought to a satisfactory state of reliability or that it can be maintained and modified." Without suggesting any priority, the participants discussed the following possible fundamental or contributing causes of the Software Crisis which were not totally independent:

a. The increasing size of the ISs to be developed. A lot of references in this respect were made to the problems pertaining to the development of the operating system of the IBM 360 family of computers (OS/360);
b. The large research content of the development of ISs provoked by the continuously use of new hardware and software technology;
c. The inability to measure progress and thus to estimate the ultimate money and time consumption;
d. The management of large numbers of IS developers;
e. The lack of management talents;
f. The employment of unqualified and unexperienced programmers as a result of labour shortages;
g. The sheer incompetence in software design;
h. The refusal to re-engineer preceding ISs;
i. The rapidly changing hardware and software technology;
j. Inadequate documentation;

---

[1] The next references in this section that are represented by their page numbers only, all refer to *Software Engineering: Concepts and Techniques* by Naur, Randell and Buxton, 1976.

k. The problematic allocation of time; too much to coding and too little to design, and to testing and debugging; and
l. The sometimes extraordinary economic pressures on manufacturers.

As there was disagreement about the causes of the Software Crisis, the suggested solutions varied and were often contradictory. Beyond the generally accepted 'solutions' like more documentation, more standardized documentation, software engineering eduction, and more emphasis on testing and design, other suggested solutions resulted in passionated discussions. Some participants claimed that the use of high-level languages would be a remedy for the problems because they would reduce machine dependence, increase the productivity of the IS developers, and make the applications easier to change. Criticisms of the use of high-level language were among others the resulting hardware inefficiency and the limited problem domain covered by each high-level language because of its specific design features. Another solution of the Software Crisis was introduced by the conference organizers, i.e., the utilization of traditional engineering tools to manage the IS development process. It was believed that this approach would solve the existing problems by: (1) the standardization of application program structures via modularization, (2) a specific sequence of stages in the IS development process, and (3) the standardization of design and programming procedures. The effects of the application of these engineering principles would be an increase of management control over the IS development process. The experiences of the conference participants which applied similar principles or heard of such principles being applied were very divergent. Other suggested solutions were the application of an iterative process between design and coding, and the evolution of the successful ISs in the field.

Since the early 1970s more and more signals of the Software Crisis have emerged and the extent and priority of the problems have been shifted. Problems with respect to maintenance, communication, and quality of ISs became frequently associated with the Software Crisis. Hence, while initially technical and managerial aspects were addressed by the Software Crisis, nowadays it concerns a cluster of aspects including social and political aspects. As was argued in the introduction of this chapter, the clarification of the Software Crisis remains rather restrained to some introductory comments like in the Software Engineering area (see, e.g, Sommerville 1989; Macro and Buxton 1987; Steward 1987), or restricted to a specific problem, for example, maintenance problems (see, e.g., Doke and Swanson 1991; Martin and McClure 1983; Lientz and Swanson 1980), quality problems (see, e.g., Macro 1990; Manns and Coleman 1988), or time and budget overruns (see, e.g., Genuchten 1991). Only a few authors provide a more in-depth description of the phenomenon. The following discussion summarizes the most noteworthy of these contributions. Other references can be found in, e.g., Nelson (1989), and Orr (1986).

In 1982 two Dutch organizations organized a conference on the Software Crisis (see ASI 1982). The speakers, whose backgrounds varied, were asked to present their experiences and solutions to the Software Crisis which resulted in several perspectives on the phenomenon. The first speaker argued that the strict distinction between EDP department on the one hand, and users and management on the other hand was the reason for the Software Crisis. The solution of the Software Crisis was to be found in eliminating this distinction. The second speaker was Dijkstra. In his presentation, the advent and the persistence of the Software Crisis were addressed. He discussed the

following explanations of the persistence of the Software Crisis that reinforce each other to some extent:

1. The negligence of computer science by mathematicians which resulted in a self-fulfilling prophecy: the field has been occupied by second- and third-rate people, which made the field even more repulsive for mathematicians.
2. The incompetence of IS developers who cannot be replaced but must be hired due to economical reasons and must be managed in a Taylor-like manner which discouraged competent personnel even more.

The resulting pessimism brought about the conviction that the Software Crisis is some kind of an incurable disease which provided a fertile soil for quackery. Next, Dijkstra discussed 'solutions' like all kinds of programming languages, PCs, and intelligent terminals. He saw an important role for the academic world in order to develop and introduce real solutions. A critical restriction is that scientists leave the path of societal relevance, i.e., when makeshifts are requested, makeshifts will be supplied. The other speakers of the conference found a solution of the Software Crisis in quality software machines, software support in hardware, standard operating systems, reusable software, computer aided software development, and software science metrics.

Pressman (1987) argues that the Software Crisis is characterized by many problems like: (1) the inaccuracy of schedule and cost estimates, (2) the productivity of IS developers has not kept pace with the demand for their services, (3) the problems during maintenance of existing ISs, (4) the problematic communication between customer and IS developers, and (5) the quality of ISs is sometimes less than adequate. According to Pressman, the Software Crisis has been caused by the character of the ISs itself (as an intellectual challenge) and by the failings of the people charged with IS development; managers with no background in IS field, and IS developers with little formal training in new techniques and development methods. Pressman argues that the real cause of the Software Crisis is perhaps the fact that while computing potential faces enormous and rapid change, the IS people responsible for employing that potential often oppose change when it is discussed and resist change when it is introduced. Furthermore, he recognizes that "... although reference to a Software Crisis can be criticized for being melodramatic, the phrase does serve a useful purpose by highlighting real problems found in all areas of software development."

Nalven and Tate (1989) describe an alternative approach to reduce the Software Crisis for Western Companies. They identify an accelerating offshore shift which is closely related to "the world's worsening Software Crisis." The decades-old applications backlog is not going away despite the introduction of new productivity tools; in most countries there is a severe shortage of IS developers; the demand for new ISs, particularly competitive edge applications where there are few packages available, is increasing; and corporate managements worldwide are trimming EDP budgets. In this environment, nations like India, and Singapore where labour, resources, and overhead costs average about half that of Western countries, see an opportunity. In addition to the low costs, these nations offer a work force with people who work harder and are more enthusiastic about technology than their Western counterparts. However, expertise on a par with Western nations is harder to come by. Western companies facing an in-house Software Crisis are taking this offshore option to keep costs and backlogs low.

Dambrot (1989) discusses the "Japan's Software Skills Crisis." The Japanese Ministry of International Trade and Industry (MITI) expects that by the year 2000 Japan will be confronted by a million too few software engineers. In an attempt to overcome problems

in the area of productivity, development, and incompatibility, MITI initiated a five-year project called SIGMA (Software Industrial Generalization & Maintenance Aids) which is a $194 million effort to automate software development and create standardized Unix-based data structures, communications interfaces, and applications platforms. The Japanese companies, which are facing a looming Software Crisis that threatens the future of their internal information processing systems, user services, and overall productivity, suspect that the following tactics will grow in importance: retraining of existing employees, contracting out of software projects, implementing structured programming aids, and automating programs. Some say, however, that by the turn of the century, new computer architectures and software engineering tools may reduce the need for these measures.

In an interesting article, entitled "Solution in Software Crisis" Auer *et al.* (1990) discuss a solution to the Software Crisis. They analyze the typical problems of the software process and arrive at the conclusion that the problems are not caused by external factors alone (e.g., ISs have become larger and more complex, must be released earlier, and require interdisciplinary skills), but also by poor understanding of the nature of the software production process. They argue that programming is problem solving which is a complex activity even for humans because problems are initially incompletely known, require human creativity and a human being is not a perfect problem solver. Already in the 1960s the proposed solution was to divide the work into stages which result in a linear or waterfall model, and later on in a spiral model. A common feature of the specification, design, and coding stages is that a person writes a solution to a problem using some representation. The stages reflect a change of representation from a higher level language to a lower level language. The division into stages is artificial and intended to make the human work easier. A better approach is to consider the whole design work as a general problem solving sequence. Then the division must reflect the nature of the problem and the aspects of the problem that should be solved in each stage. In this case, all solutions in each stage could be represented with the same language. Based on their analysis of the software production process Auer *et al.* compose seven golden rules to escape from the Software Crisis:

1. Natural division of a problem solving into stages.
2. Formal representation of all solutions.
3. Visual and textual formal representation.
4. Automatic checking of solutions.
5. Automatic formulation and transformation of a problem.
6. More knowledge in automata.
7. Libraries of solutions to known problems.

Furthermore, the authors discuss their SOKRATES method, language and compiler that fulfils most of the seven rules.

To finish the review of the Software Crisis, a recent debate on the phenomenon in the Dutch weekly journal *Computable* is recapitulated. The discussion was started by the statement that rumbling through with programming languages like Cobol, Fortran, and C is an important cause of the Software Crisis. It is curious that these languages remain in use despite their numerous shortages. The initiator of the debate argues that the most complete programming language (but not perfect), Ada, is hardly used nowadays. The above statement resulted in several reactions. The reactions can roughly be subdivided into three groups:

1. The Software Crisis is not a programming language problem. The problem is the

control and development of ISs and the way developers apply development tools.
2. The core of the problem is the quest of one universal solution to all the problems and all application areas which is impossible and results in a conflict of opinions.
3. Ada is not the most complete programming language, other programming languages, for instance, C++, and Fortran 90, provide similar or even more functionality.

As can be concluded from the above review of literature, there is no agreement on the existence, extent, and causes of the Software Crisis. As a result, many different solutions to the Software Crisis are or have been proposed. Furthermore, the Software Crisis is frequently addressed from different points of view like a technical, a managerial, a social or a political perspective or a combination of these perspectives. Although some authors solely refer to the applications backlog while discussing the Software Crisis, it can be argued that generally the Software Crisis designates the following problems: (1) time and budget overruns, (2) maintenance problems, (3) applications backlog, (4) quality problems, and (5) communication problems. These problems are sometimes addressed simultaneously by the uncontrollability of ISs and IS development processes. These observations and the questions stated in the introduction of this paper form the starting-point of a sample survey research on the phenomenon Software Crisis in The Netherlands.

3. DATA COLLECTION PROCESS

There are several research strategies available to collect and analyze empirical data about a certain phenomenon. And each strategy has its own advantages and disadvantages. The five major research strategies in the social sciences, which can be selected, are (quasi) experiment, survey, archival analysis, history, and case study. The question is when and why a specific research strategy should be employed. Common misconceptions are that case studies are appropriate for exploratory research only, survey and histories are only appropriate for descriptive research, and that experiments are appropriate for explanatory research only (see Yin 1989). However, each research strategy can be used for the three types of research; the type of research does not provide a discriminated criterion. Yin (1989) distinguishes three conditions to determine the appropriate research strategy. However, the boundaries between the research strategies remain fuzzy. The application of the three conditions protects the investigator against gross mistakes while selecting a research strategy.

The first condition that is important to select the appropriate research strategy concerns the type of research question posed. The type of research questions can be categorized using the basic scheme: "who", "what", "where", "how", and "why." "What" questions can be either exploratory or descriptive. In the case of an exploratory study, any of the research strategies can be employed. The second "what" question is more likely to favour survey or archival strategies than other strategies. "Who" and "where" questions are generally studied using survey or archival analysis strategies, whereas "how" and "why" questions are normally investigated using case studies, history and experiment strategies. The second condition involves the extent of control the investigator has over actual behaviourial events. When there is virtually no control over behaviourial events any research strategy can be used except for the (quasi) experiment strategy. The last condition to choose an appropriate research strategy pertains to the degree of focus on

contemporary events. Whenever a research focuses on contemporary events experiment, case study, and survey strategies can be used. History and archival analysis strategies are more appropriate when the research is dealing with the past.

In order to provide empirical data about the existence and the current extent of the Software Crisis, sample survey research method was selected to carry out the research. Based on the above conditions, sample survey turns out to be an adequate method to answer research questions of descriptive nature when the focus of the study is on contemporary events (see also, e.g., Baarda and Goede 1990; Babbie 1973). The main purpose of this sample survey was to yield a broad description of the current situation of the Software Crisis in The Netherlands based on several observations and the undiscussed questions concerning the Software Crisis that were already stated in the introduction. The technique used to collect the data for analysis was self-administered questionnaires by mail (see Babbie 1973). One of the main disadvantages of such questionnaires is that the degree of detail is lower than other research methods (see Schampheleire 1986). However, the aim of the research was to provide a broad description of the Software Crisis in The Netherlands rather than an in-depth analysis of the phenomenon within specific organizations.

Before the actual data collection process was started, all universities and research institutes in The Netherlands that are actively engaged in IS research were contacted in order to ensure that no similar research was conducted recently or currently. Subsequently, the data collection process began with the construction of an initial questionnaire based on a method to draft a questionnaire (see Bartelds *et al.* 1989). The topics of the questions were based on literature concerning the problems of the Software Crisis as reflected partly in the preceding section. In order to verify the relevance, unambiguousness, clarity, resulting information, and the time needed to complete the questions, thirteen interviews with managers of the IS departments of different organizations were carried out. These interviews were also used to strengthen the validity and reliability of the research (see, e.g., Staub 1989; Cook and Campbell 1979). The results of these interviews brought about several refinements of the initial questionnaire.

The intended size of the responses was fixed at one hundred IS managers of large organizations in The Netherlands which was judged to be of probable sufficiency for analytical purposes (see Lientz and Swanson 1980). Based on experience and some preliminary interviews, the expected response rate was likely to be in the range of 10% to 40%, with 20% as most likely. Therefore about 500 IS managers/organizations were selected. Only organizations with more than 500 employees, were included because these organizations were expected to have a specific IS department. The organizations were selected randomly, primarily using a CD-ROM disk with some general and financial features of 5000 organizations in The Netherlands. Furthermore, the manager of the IS department(s) of each organization was identified by telephone. The particular functionary of IS manager was chosen because, generally, literature and commerce focus on IS developers while addressing the Software Crisis (albeit rather implicitly), and because the questionnaire required respondents that had comprehensive insight in IS development within the organization. This sample will result in a rather one-sided perspective on the phenomenon. Furthermore, there is a possibility of window-dressing in the answers because the questions implicitly inquires the operation of the respondents. These two aspects were taken into account while analyzing the results of the questionnaire.

In the case that the sample is adequately chosen and all members of the sample complete and return their questionnaire, there will be no response bias. Since this almost

never occurs, response bias becomes a concern. Hence, the respondents should essentially be a random sample of the total initial sample. Response rate is one guide to the representativeness of the sample respondents. A review of papers addressing the results of survey research uncovers a wide range of response rates. Each of these are accompanied by a statement like "This is a substantial percentage considering the length and depth of the questionnaire" (see Lientz *et al.* 1978). There is, however, no statistical basis for an assessment whether the response rate is high or not. A demonstration of the lack of response bias is far more important than a high response rate. From the 485 questionnaires that were mailed out, 98 questionnaires were completed and returned which yields a response rate of 20.2% which approximated the intended sample size of one hundred IS managers. The issue of nonresponse concerns the possibility that the likely effect of nonresponse is to bias the sample by making the sample systematically different from the population from which they were drawn (see Fowler 1993; Donald 1960). During the pre-tests, thirteen IS managers were asked whether they would complete and return the questionnaire and, if not, why they would not complete the questionnaire. Eight of the thirteen IS managers stated that they would not complete the questionnaire because of lack of time, the size of the questionnaire and the current overload of all kinds of questionnaires by mail. After the deadline to return the completed questionnaires was expired, ten IS managers were phoned to ask for their reasons for nonresponse. They all indicate that did not have the time to complete the questionnaire. Therefore, there was no reason to assume that the sample is biased in one way or another.

The questionnaire was composed of eight parts that in addition to some general questions reflected the problems that are normally addressed by the term Software Crisis. The first part of the questionnaire consisted of some general questions with respect to the respondent, i.e., experience and education, and the organization, i.e., branch. The next part included questions concerning IS development within the organization and how the respondents experienced the problems that are normally associated with the Software Crisis both from their own point of view, and from an end-user and top management point of view as conceived by the respondents. The subsequent five parts consisted of questions pertaining to the problems that are commonly designated by the term Software Crisis, i.e., time and budget overruns, maintenance problems, applications backlog, quality problems, and communication problems. The last part of the questionnaire was devoted to questions concerning the familiarity with the term Software Crisis, the appropriateness to use the term Software Crisis, and the current problems with respect to ISs within the organization.

4. ANALYSIS RESULTS

The summary statistics and analysis resulting from the survey described above, are presented below for each of the problem categories identified. The data is analyzed by means of SPSS/PC+, version 5.0.1 (see Norusis 1987). Because of the nature of the study, only descriptive statistics are applied and hardly any effort is spent on the analysis of relationships between variables. The findings from the questions of the first two parts of the questionnaire are summarized in the subsection concerning the profile of the respondents.

Profile of Respondents

The respondents were asked to indicate their experience in the area of IS and whether they had received a specific IS education and, if so, the level of that education. The analysis of the responses indicated that 45.9% of the respondents had more than 15 years experience in the field of IS, 20.4% had 10 to 15 years, 25.5% had 5 to 10 years, and 8.2% had less than 5 years experience. Almost half of the respondents (49.0%) had received a specific IS education. The level of education of that half of the respondents was: 35.4% at university level, 62.5% at high level, and 2.1% at middle level. (It should be noted that the Dutch classification of the level of education was used.)

The sample represented a variety of branches (see Figure 1). The types of business were based on a classification of CBS (i.e., the Central Statistics Office in The Netherlands).

Type of Business

	Frequency	Valid Percentage
Manufacturing	21	21.4%
Utilities	5	5.1%
Building and installation	5	5.1%
Trading, hotel, and restaurant	19	19.4%
Transportation, communication and storage	9	9.2%
Finance, banking, and insurance	11	11.2%
Other services	28	28.6%
Total	98	100.0%

Figure 1. Respondents Profile

Next, the respondents were asked to indicate the distribution of effort devoted to the following six categories of IS activities: new systems development, reconstruction of existing systems, expansion of existing systems, correction of faults in existing systems, integration of systems, and other activities. The respondents were also asked to assess the accurateness of their answers. Figure 2 summarizes the responses to these questions.

The results indicate that the activities 'new systems development' and 'expansion of existing systems' absorbed most of the IS effort within the organizations. The category 'other activities,' as specified by the respondents, mainly comprised activities such as implementation of standard packages and end-user support. Given the high values for the standard deviations and the large differences between minimum and maximum values, it can be concluded that the distribution of effort devoted to the specific IS activities among organizations varied broadly.

The last questions of this part of the questionnaire concerned how the respondents experience the problems that are normally addressed by the term Software Crisis: time and budget overruns, maintenance problems, applications backlog, quality problems, and communication problems. Two other problems that are sometimes mentioned in the literature with respect to the Software Crisis, i.e., uncontrollability and malfunctioning of ISs, were also included. The question was whether these 'problems' constitute actual experienced problems in practice. In addition to their own opinions, the respondents were asked to indicate the perception of these problems from two imagined points of view, i.e., end-users' perspective and top management's perspective. The purpose of these different-

IS Effort Distribution

	Mean	Standard Deviation	Minimum	Maximum
New systems development	29.73	20.88	0	100
Reconstruction of existing systems	12.50	14.07	0	65
Expansion of existing systems	27.90	18.86	0	80
Correction of faults in existing systems	15.30	15.77	0	70
Integration of systems	9.07	9.37	0	40
Other activities	5.53	13.47	0	69

Accuracy of Answers

	Frequency	Valid Percentage
Very certain	18	18.9%
Reasonably certain	71	74.7%
Uncertain	6	6.3%
	3	Missing
Total	98	100.0%

Figure 2. IS Effort Distribution

perspective-questions was to abate the one-sided perspective of the questionnaire to some extent. IS managers are supposed to have an idea of the experiences of the above problems of end-users and top management because they communicate with both groups. However, no conclusions may be drawn from these findings. The findings are only included to obtain an impression of the differences in experiences of the problems between the three groups.

To mark the extent of the problems, the respondents could select one of the following answers for each problem and for each point of view: (1) does not exist, (2) no problem, (3) moderate problem or (4) big problem. The following figure shows the valid percentages for each problem and each point of view (Figure 3). The four answer options are clustered into two categories, i.e., (I) does not exist/no problem and (II) moderate/big problem. The respondents hardly selected the option 'does not exist' (1 to 5%), therefore the percentages of category I primarily reflect 'no problem' answers.

The findings of Figure 3 indicate the following sequence from most to least frequently experienced as a problem by the respondents. The problem of time and budget overruns was by far most frequently experienced as a problem by the respondents (approximately 80%). The second most frequently experienced as problems were uncontrollability of ISs and applications backlog (around 65% of the respondents). Of the respondents, about 60% experienced communication problems, maintenance problems, and quality problems as a problem. Finally, malfunctioning of ISs was experienced as a problem by only 37.1% of the respondents.

When the respondents took the end-user perspective, the order was rather different. The sequence from most to least frequently experienced as a problem seen from an end-user perspective was:

1. Quality problems, applications backlog, and communication problems (around 70% of the respondents);
2. Time and budget overruns (around 60% of the respondents);
3. Malfunctioning of ISs (around 50% of the respondents);

	IS Managers Opinion		End-user Perspective		Top Management Perspective	
	I	II	I	II	I	II
Time and Budget Overruns	19.6	80.5	38.5	61.4	16.7	83.3
Maintenance Problems	41.2	58.8	65.6	34.4	56.3	43.7
Applications Backlog	35.1	65.0	31.3	68.7	46.9	53.1
Uncontrollability of ISs	34.0	66.0	63.5	36.4	46.9	53.1
Malfunctioning of ISs	62.9	37.1	47.9	52.1	70.8	29.2
Quality Problems	41.2	58.8	28.1	71.9	45.8	54.2
Communication Problems	39.2	60.8	32.6	67.4	45.3	54.7

Figure 3. Perception of Problems from Three Points of View

4. Uncontrollability of ISs and maintenance problems (about 35% of the respondents).

Conceived from a top management point of view, the sequence indicated by the respondents was as follows. Again, the problem of time and budget overruns was by far most frequently experienced as a problem (about 85% of the respondents). Next, the problems communication problems, quality problems, applications backlog, and uncontrollability of ISs were experienced as actual problems by approximately 55% of the respondents. Of the respondents, around 45% experienced maintenance as problematic. Finally, the malfunctioning of ISs was experienced as a problem by only 29.2% of the respondents.

These findings indicate that there are differences in the perception of the problems seen from the different points of view. The differences can be explained considering the tasks of the different functionaries and their involvement with ISs. It should be noted that the responses to these point-of-view questions are not totally unrelated, that is, if the IS Manager perceived a certain problem as moderate or big then the responses of that respondent to the same problem seen from a End-User or Top Management point of view tended to be moderate or big as well (the cross-tables between the variables which show these relationships are not included). Therefore, caution is in order drawing conclusions from the findings. Figure 4 summarizes the results of the three points of view questions and illustrates the differences in both rating and percentages between the problems.

Time and Budget Overruns

Martin and McClure (1983) state that "... instances of actual schedules running 200% over estimates and actual cost running 300% over budgeted costs are rather rule than the exception in software projects." More recent empirical studies report less spectacular figures (see, e.g., Genuchten 1991; Phan *et al.* 1988; and Jenkins *et al.* 1984). The studies arrive at an average cost overruns of approximately 35% and the average schedule overruns of about 25%. The surveys also address the reasons for cost overruns and late

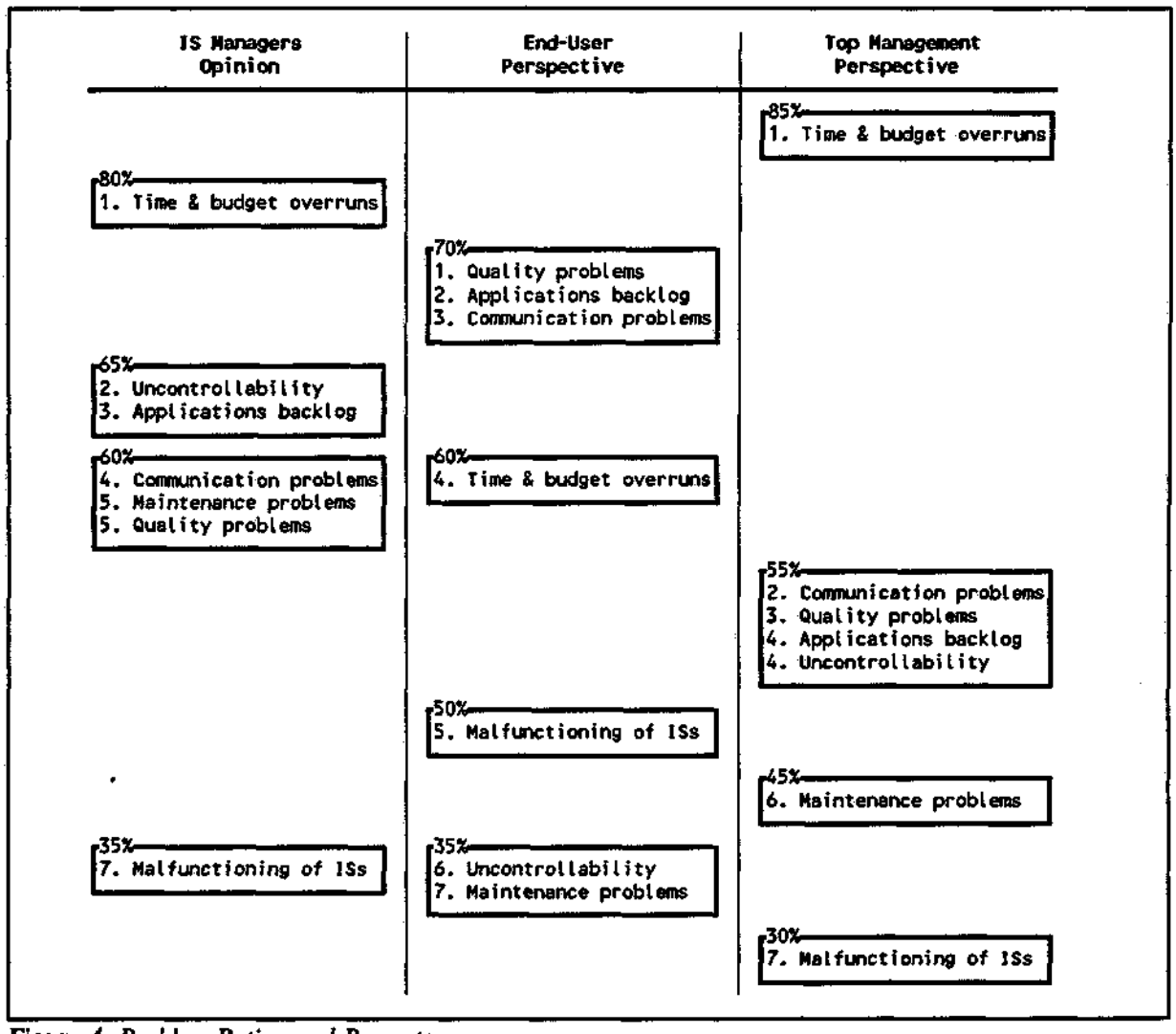


Figure 4. Problem Rating and Percentages

deliveries. Most respondents stated that over-optimistic planning, frequent changes in design and implementation, and the lack of IS development tools were usually reasons for a cost overrun and/or late delivery. According to these studies, the actions most frequently taken to regain control over delayed projects are: (1) upgrading the priority of the project, (2) shifting part of the responsibility and obligations to other groups, (3) renegotiating the plan and schedule, and (4) postponing features and upgrades to the next version. The questionnaire contained several questions with respect to time and budget overruns of IS development projects.

The first question concerned the percentage of IS development projects that suffered from time and budget overruns. The responses were:

no overruns	6.1 percent
1 to 25% of the projects	34.7 percent
25 to 50% of the projects	28.6 percent
50 to 75% of the projects	20.4 percent
75 to 100% of the projects	10.2 percent

These findings indicate that 59.2% of the respondents answered that 25% to 100% of their IS development projects overran their schedules and budgets. It seems that IS developers are unable to apply normal management practices to IS development projects.

The respondents were also questioned to assess the average extent of their project overruns both in time and costs. Obviously, there is a relationship between those two variables. Therefore, Figure 5 summarizes the responses by means of a cross-table between the variables time and cost overruns.

Time Overruns \ Budget Overruns	1 to 20%	20 to 50%	50 to 100%	100 to 200%	>200%	Total
1 to 20%	29	6	1			36 38.7%
20 to 50%	10	27	1	1		39 41.9%
50 to 100%	1	8	6		1	16 17.2%
100 to 200%	1	1				2 2.2%
Total	41 44.1%	42 45.2%	8 8.6%	1 1.1%	1 1.1%	93 100.0%

Figure 5. Cross-table: Time Overruns and Budget Overruns

The findings of Figure 5 indicate that the average time overruns of IS development projects of approximately 40% of the respondents amounted to 1 to 20% of the planning and about 40% of the respondents measured 20 to 50% of the planning. The average budget overruns of IS development projects came to 1 to 20% of the budget for around 45% of the respondents and amounted to 20 to 50% of the budget for 45% of the respondents. The cross-table supports to some extent the obvious relationship between the two variables, i.e., the longer the time overrun is the higher the budget overrun tends to be. Some respondents stated that this relationship is not that obvious because of intermediate activities such as ad hoc information requirements and correction of disturbances which will result in time overruns but will not bring about budget overruns. However, no solid conclusions can be drawn from this cross-table because the frequencies within the cells of the table are too low.

Next, the respondents were asked to indicate the measures they had applied to overcome the problem of time and budget overruns. They were also asked to estimate the effect of each measure applied by selecting one of the following options: more than expected, just as expected, less than expected, no effect at all. Figure 6 shows the percentages of the respondents that had applied certain measures and the estimated effect of these measures.

The respondents marked the application of other tools such as code generators and function point analysis. Furthermore, project management methods and tools, the application of standard packages, different development methods, and the replacement of project management were mentioned more than once as other measures to overcome time

Measure Applied	Valid Percentage	Effect			
		More than expected	Just as expected	Less than expected	No effect at all
4GL	44.9%	6.8%	50.0%	34.1%	9.1%
CASE tool	35.7%	5.7%	42.9%	42.9%	8.6%
Other tool	14.3%	7.1%	92.9%	-	-
Increase budget	27.6%	-	69.2 %	26.9%	3.8%
More personnel	49.0%	2.1%	58.3%	35.4%	4.2%
Other method of planning	33.7%	8.8%	50.0%	32.4%	8.8%
Outsourcing	40.8%	10.8%	51.4%	32.4%	5.4%
Other measures	27.6%	33.3%	54.2%	12.5%	-

Figure 6. Time and Budget Overruns: Measures and Their Effect

and budget overruns. The findings of Figure 6 suggest that the employment of more personnel, the application of 4GL, and outsourcing were the three most frequently applied measures to overcome time and budget overruns. The estimated effects of the measures indicate that the application of CASE tools in about 50% of the cases had a less than expected effect or had no effect at all. The measures, 4GL, more personnel, other method of planning, and outsourcing had a less than expected effect or no effect at all in about 40% of the cases. Several respondents explicitly stated that there are still no satisfying measures available to overcome time and budget overruns of IS development projects. Their allegation is supported by the results of the questionnaire.

The last question pertaining to time and budget overruns, concerned the reasons for overruns according to the respondents. It was an open question and the respondents could indicate more than one reason. Table 1 summarizes the ten most frequently encountered reasons for time and budget overruns. It should be noted that many other reasons were mentioned by the respondents.

Table 1. Reasons for Time and Budget Overruns

Reasons	Frequency
Changing requirements during development project	18
Immaturity of IS development (knowledge, experience, education)	16
Underestimation of complexity	12
Over-optimistic planning	10
Insufficient specifications	10
Little user involvement	10
Poor project management	9
Difficulties with planning (method)	8
Negligence of organizational aspects of IS implementation (culture, change)	8
Project size	5

The responses of Table 1 indicate that changing requirements during IS development

processes was one of the most important reasons for time and budget overruns. It turned out to be very difficult to freeze the specifications during a development project. Another often encountered reason for project overruns was the immaturity of the IS field. This included the insufficient quality, experience, and expertise of both suppliers and IS developers with respect to, for example, tools, techniques and IS development methods. In addition, other frequently encountered reasons were:

- the underestimation of the complexity (e.g., the influence of existing ISs and integration aspects);
- drawing up too optimistic plans (for political reasons, e.g., to obtain approval to start the project);
- insufficient specifications (e.g., due to the inability of end-users to specify their requirements);
- little user involvement during the development process;
- poor project management;
- the deficiency of appropriate planning methods and techniques;
- the negligence of organizational aspects while introducing the IS (e.g., culture, and organizational change); and
- the size of the development projects which makes projects uncontrollable.

Maintenance Problems

Since the early 1970s, maintenance of ISs has been recognized as a critical and important activity of the IS development life cycle. Maintenance is the general name given to the set of activities on an IS after delivery to correct faults, to improve performance or other attributes, or adapt the IS to a changed environment (see ANSI/IEEE Standard 1983). Maintenance is really a misnomer because no worn out parts are replaced but the design of the IS is actually changed (see Forte 1992; Martin and McClure 1983). Although it might be better to use a different word like 'system re-engineering,' 'maintenance' will be used as it is now firmly embedded in common usage. An article in *EDP Analyzer*, entitled "The Maintenance Iceberg," is often considered one of the first signals of the importance of IS maintenance (see Canning 1972). Since then, many empirical surveys have appeared in literature. In the surveys of Lientz *et al.* (1978) and Lientz and Swanson (1980), which are often cited, maintenance consumed about 50 percent of EDP resources. More recently, an article stated that maintenance consumed approximately 65 percent of EDP resources (see Doke and Swanson 1991). Despite the relatively large cost associated with maintenance, most tools and methodologies are directed towards IS design (see Hager 1989). Furthermore, notwithstanding the importance of maintenance, it has acquired the reputation of being a second-class area to work in because the myth exists that there is no challenge for creative people in maintenance (see, e.g., Bennett 1991; Schneidewind 1987). The recognition that maintenance consumes about half of the EDP resources does not illuminate the problems pertaining to maintenance properly. The major problems of maintaining existing ISs are (see, e.g., Ketler and Turban 1992; Bennett 1991; Schneidewind 1987; Martin and McClure 1983):

- Most existing ISs are unstructured because they were produced before the introduction of structured programming methods or they were not designed for maintenance, and the original structure has probably been lost in the face of years of modification and upgrade. Therefore, existing ISs are difficult to understand.
- The documentation of ISs is often nonexistent, or incomplete or out of date. Even if it

is available, the documentation may not actually support the maintenance staff in an effective way.

- Maintenance programmers have not been involved in the development of the IS and find it difficult to map system actions to system source codes.
- The ripple effect of changes to source codes is difficult to predict. A major concern of maintenance programmers is the avoidance of introducing more problems than are solved by the modification.
- Multiple concurrent changes are difficult to manage.
- The development costs of an IS are the only visible aspect and form the basis for decision-making and control. After introduction, management expects to be confronted with low maintenance costs. However, it appears that maintenance costs in the long run exceed implementation costs but remain often ignored. Hence, the metaphor "maintenance iceberg" of Canning (1972).

Based on the above observations, Martin and McClure (1983) among others arrive at the conclusion that maintenance is one of the most important problems of the Software Crisis. The questionnaire contained several questions pertaining to the maintenance of existing ISs. The first question concerned the ratio between new systems development and maintenance of existing ISs. Figure 7 displays the responses by means of a histogram where the vertical axis represents the ratio between new systems development and maintenance, and the horizontal axis depicts the number of respondents.

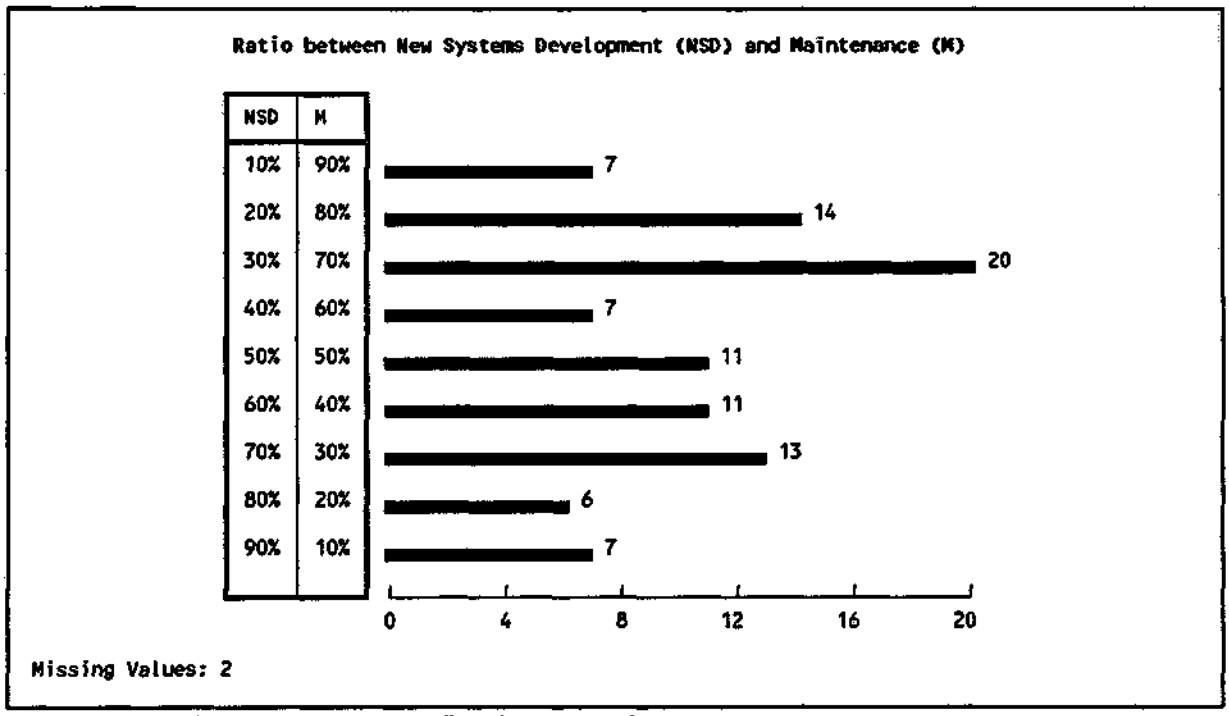


Figure 7. Ratio between New Systems Development and Maintenance

These findings correspond with the results of the surveys of, for example, Lientz *et al.* (1978), and Doke and Swanson (1991). It is noticeable that these figures are all about the same, although caution is in order because the samples of these surveys vary. Therefore, it appears that the total maintenance effort remains constant on the average throughout the years, and is not increasing, as often predicted in literature (see, e.g., Ketler and Turban 1992), nor is it decreasing as predicted by many tool vendors. The

respondents were questioned if there was a tendency perceptible towards less or more maintenance compared to new systems development or if the proportion remains constant. The responses were:

more maintenance	27.1 percent
less maintenance	34.4 percent
constant	38.5 percent

The results support the observation that the total maintenance effort remains constant in time. Subsequently, two questions were asked on: (1) what the respondent's opinions were with regard to the current ratio between new systems development and maintenance within their organization, and (2) how new systems development and maintenance effort would be redistributed if the EDP budget were to be increased. The answers to these questions are depicted in Figure 8.

Opinion about ratio between new systems development and maintenance

	Frequency	Valid Percentage
Realistic	62	64.6%
Too much maintenance	29	30.2%
Too much new systems development	5	5.2%
	2	Missing
	98	100.0%

Redistribution of the ratio if the EDP budget was increased

	Frequency	Valid Percentage
More to new systems development	54	57.4%
More to maintenance	11	11.7%
Equally	29	30.9%
	4	Missing
	98	100.0%

Figure 8. Opinion and Redistribution of Ratio

In literature, it is generally recognized that the proportion of maintenance is much too high and is therefore considered a problem. The findings suggest that around 65% of the respondents considered the current ratio between new systems development and maintenance to be realistic. Hence, the responses did not support the view described in literature that this ratio is one-sided. A plausible explanation would be that maintenance tends to be seen as a necessary evil, and the IS managers hold the conviction that nothing can be done to reduce this workload (see Canning 1981). If the EDP budgets were to be increased, 57.4% of the respondents would devote more to new systems development than to maintenance. The results of the survey of Lientz *et al.* (1978) also indicate that most additional resources would go to new development. This appears to be in contrast with the long-run historical trend of increasing EDP budgets and the constant ratio between new systems development and maintenance. The budget increases are probably absorbed by the rising burden of ISs to be maintained (see Lientz *et al.* 1978).

The total maintenance effort seems to remain about constant in time. However, it

would be going too far to conclude that the characteristics of maintenance have not changed for the better. Swanson (1976) divides maintenance into three categories, i.e., corrective, adaptive, and perfective maintenance.

1. Corrective maintenance: consists of dealing with failures in the IS, so that its behaviour does not conform to its specification. The failure may contradict the specification, or it may demonstrate that the specification is incomplete, and manifest itself in the form of an error. It is often also called "bug-fixing." Lientz and Swanson (1980) found that this type of maintenance generally consumes about 20% of the maintenance efforts.
2. Adaptive maintenance: includes responding to changes in the environment in which the IS operates. The changes can occur in the data or processing environment. For example, it includes converting to a new operating system, responding to changes in government legislations, restructuring a database, and so on. This type of maintenance consumes about 25% of the maintenance effort according to the survey of Lientz and Swanson (1980).
3. Enhancement or Perfective maintenance: consists of dealing with changes in user requirements of the ISs. These changes include (1) enhancing processing efficiency, performance and IS maintainability, (2) improving IS documentation, and (3) making enhancements for users. This form of maintenance consumes about 55% of the maintenance effort, and almost two-thirds of that went toward making enhancements requested by users (see Lientz and Swanson 1980).

This subdivision is commonly accepted in the literature with respect to IS maintenance. However, several authors have introduced additions, like Reutter (1981), who recognizes supportive maintenance as a separate category to emphasize the importance of communication between the user and the maintainer and the planning for system support. Another category that is sometimes distinguished, is preventive maintenance which is undertaken on an IS to anticipate to future changes and make subsequent maintenance easier (see, e.g., Bennett 1991). Several authors, for instance Canning (1981), argue that the developments in the IS maintenance area on the one hand reduces the amount of corrective maintenance but on the other hand make ISs easier to adapt which encourages end-users to ask for more changes.

The respondents were asked to indicate their current maintenance effort distribution. The results of this question may be compared with the results of the survey of Lientz and Swanson (1980), and Doke and Swanson (1991), although caution is in order because the samples of these surveys vary. This comparison may provide an impression of the shift within the total maintenance effort. Furthermore, the respondents were questioned which activities they usually considered as maintenance. This question was included in the questionnaire because there is some discussion whether an activity is maintenance or new systems development (see, e.g., Doke and Swanson 1991). It is often argued that perfective maintenance, and sometimes adaptive maintenance too, should be considered new systems development instead of maintenance. Figure 9 summarizes the results of both questions. The figure also depicts the respondents' assessments of the accurateness of their answers on maintenance effort distribution.

Five respondents indicated other maintenance activities. Four of them marked the adjustments of existing ISs due to organizational change as other maintenance activities. Therefore, the other maintenance activities may be added to the activity 'Adapt to changes in the external environment.' When the maintenance activities of Figure 9 are clustered

Maintenance Effort Distribution

	Mean	Standard Deviation	Considered Maintenance
Correction of faults	14.20	12.69	86.9%
Adapt to technological advancements	13.15	13.01	84.0%
Adapt to changes in external environment	14.30	13.95	76.8%
Functional enhancement of the systems	40.10	20.32	52.1%
Enhance systems maintainability	6.33	6.25	73.9%
Enhance systems performance	8.88	9.24	81.1%
Other maintenance activities	3.15	10.79	-

Accurateness of Answers

	Frequency	Valid Percentage
Very certain	13	14.6%
Reasonably certain	60	67.4%
Uncertain	16	18.0%
	9	Missing
	98	100.0%

Figure 9. Maintenance Effort Distribution

into the three maintenance categories described above, the results were:

1. Corrective maintenance consumed about 15% of the total maintenance effort (correction of faults);
2. Adaptive maintenance consumed about 30% of the total maintenance effort (adapt to technological advancements, adapt to changes in external environment, and other maintenance activities), and
3. Perfective maintenance consumed about 55% of the total maintenance effort (functional enhancement of the systems, enhance systems maintainability, and enhance systems performance).

These findings indicate a small shift in the maintenance effort distribution compared to the results of the survey of Lientz and Swanson (1980) discussed above. However, the results indicate a change from corrective maintenance to adaptive maintenance rather than to perfective maintenance as argued by Canning (1981).

Of the respondents, 64.5% treated maintenance of existing ISs as a separate activity from new systems development projects. Most respondents had IS developers assigned to both maintenance as well as the development of new systems. Around 35% of the respondents used a different method for maintenance then they were using for new systems development. Most respondents agreed that there is no sharp distinction between maintenance activities and new systems development. Often the distinction was made rather instinctively and depended on the extent of the activities. In that case, an arbitrary criterion like less or greater than 400 man-hour distinguished maintenance from new systems development respectively. Other indicated criteria were, for instance, whether the activities involved new functionality or an existing (operational) IS.

The respondents were asked to give their opinion about six assertions that are encountered in the literature with respect to IS maintenance. The assertions are listed in Figure 10. The respondents could select one of the following answers to express their opinion for each assertion: strongly agree, agree, no opinion, disagree, and strongly disagree. Figure 10 summarizes the respondents' opinions about the six assertions.

Assertion	Strongly Agree	Agree	No Opinion	Disagree	Strongly Disagree
a. Maintenance is at the expense of new systems development	5.2%	21.6%	4.1%	58.8%	10.3%
b. Maintenance is an investment in the life span of an IS	20.6%	73.2%	-	6.2%	-
c. Maintenance differs from new systems development	12.5%	58.3%	2.1%	26.0%	1.0%
d. Maintenance is more important than new systems development	2.1%	25.0%	19.8%	45.8%	7.3%
e. The increase of productivity did not result in a decrease of maintenance effort because of the increasing complexity and integration of ISs	11.5%	69.8%	6.3%	12.5%	-
f. The increase of productivity primarily results in an increase of the quality of ISs	1.0%	41.2%	15.5%	40.2%	2.1%

Figure 10. Opinions about Assertions

As can be concluded from the findings, almost 70% of the respondents disagreed with the assertion that maintenance is at the expense of new systems development. Of the respondents, around 94% considered maintenance an investment in the value and life span of the ISs. In addition, 80% of the respondents indicated to perform some kind of cost-benefit analysis. Most of the respondents (70%) agreed with the assertion that maintenance differs essentially from new systems development. The findings described above indicated that the difference did not result in wide application of specific methods for maintenance and maintenance teams. Only a small majority of the respondents disagreed with the assertion that maintenance is more important than new systems development, but 20% of the respondents did not provide their opinion about this assertion. About 80% of the respondents agreed with the assertion that the increase of productivity did not result in a decrease of maintenance effort because of the increasing complexity and integration of ISs. The last assertion, which stated that the increase of productivity primarily results in an increase of the quality of ISs, could not agreed upon as it received as much opponents as supporters.

Next, the respondents were questioned to indicate the measures they had applied to overcome the problems with respect to maintaining ISs. They were also asked to estimate the effect of each measure applied by selecting one of the following options: more than expected, just as expected, less than expected, no effect at all. Figure 11 shows the percentages of the respondents that had applied certain measures and the estimated effect of these measures.

The respondents indicated the application of other tools such as documentation tools and test tools. Furthermore, the application of standard packages, budget cutbacks, increase control over maintenance, and analysis of the necessity of maintenance requests were mentioned more than once as other measures to overcome the maintenance problems. The findings of Figure 11 suggest that the application of 4GL, structured

Measure Applied	Valid Percentage	Effect			
		More than expected	Just as expected	Less than expected	No effect at all
4GL	49.0%	11.1%	46.7%	33.3%	8.9%
CASE tool	33.7%	-	59.4%	34.4%	6.3%
Other tool	11.2%	30.0%	70.0%	-	-
Increase budget	17.3%	-	58.8 %	29.4%	11.8%
More personnel	20.4%	10.0%	65.0%	20.0%	5.0%
Outsourcing	32.7%	6.9%	51.7%	31.0%	10.3%
Structured programming	48.0%	4.3%	76.6%	19.1%	-
Reusable software	21.4%	5.0%	80.0%	15.0%	-
Standardization of documentation	38.8%	10.8%	59.5%	27.0%	2.7%
Other measures	22.4%	38.1%	57.1%	4.8%	-

Figure 11. Maintenance Problems: Measures and Their Effect

programming, and standardization of documentation were the three most frequently applied measures to overcome the maintenance problems. Furthermore, the estimated effects of the measures indicate that the application of 4GL, outsourcing, increase budget, and the application of CASE tools had a less than expected effect or had no effect at all in about 40% of the cases.

The last question with respect to maintenance problems, concerned the reasons for maintenance problems according to the respondents. It was an open question and the respondents could indicate more than one reason. Table 2 summarizes the ten most frequently encountered reasons for maintenance problems. It should be noted that many other reasons were mentioned by the respondents.

Table II. Reasons for Maintenance Problems

Reasons	Frequency
Changes in external environment (organization, market, legislation)	23
Changes in functionality	8
Continuity of ISs is essential	5
Changing user specifications	5
Large amount of existing ISs	5
Inheritance from former times (obsolete technology, architecture, etc.)	4
Insufficient communication between end-user and developer	4
Complexity	4
Meeting change requests too easily	4
Natural evolution of ISs	4

The responses of Table 2 indicate that changes in the external environment of ISs was by far the most important reason for maintenance problems. Organizational change, the

turbulence of the environment/market, and changing legislation necessitated ISs to adapt which bring about (adaptive) maintenance. In addition, other more or less frequently encountered reasons were:

- changing and expanding IS functionality (e.g., because standard packages do not cover all the information requirements);
- continuity of existing critical ISs is essential;
- end-users keep changing their information requirements and therefore keep changing their specifications;
- the large amount of existing ISs that need to be maintained;
- the inheritance from former times (e.g., ISs are developed using obsolete methods, technologies and architectures);
- insufficient communication between end-users and developers;
- the increasing complexity of ISs which make them more difficult to maintain and more error-prone;
- all change requests are accepted without analysis whether the change is necessary and economically viable; and
- ISs evolve rather naturally which suggests that maintenance is an inevitable process (compare the 'program evolution dynamics laws' of Lehman and Belady [1985], for instance the law of continuing change: "a program that is used in a real-world environment must necessarily change or become less and less useful in that environment").

Applications backlog

It appears that the demand for ISs remains continuously higher than the supply of ISs (see Genuchten *et al.* 1993). The gap between the demand for and the supply of ISs is known as the 'applications backlog.' Some authors perceive the high proportion of EDP resources consumed by maintenance as one of the main reasons for the existence of the applications backlog (see, e.g., Bennett 1991). Although many respondents did not agree with the assertion that maintenance is at the expense of new systems development, the applications backlog was seen as one of the most frequently encountered problems of the Software Crisis by the respondents both in their own opinion and from an end-user perspective (see Figure 4). Martin and McClure (1983) argue that most organizations have a three- to four-year backlog of ISs waiting to be developed. For example, the U.S. Air Force Data Systems Design Office has identified a four-year applications backlog because of the limited supply of personnel and funding, much of which must be devoted to maintenance of existing ISs (see Boehm 1986). Campbell (1985) states that 90 to 95 percent of all organizations have an applications backlog of 18 months or more. Each year this backlog grows as users demand more productivity (see Campbell 1985; Schoman 1984; Martin and McClure 1983). The hidden backlog or invisible backlog of user needs, which have not formally entered the queue of pending applications because of the large number of already outstanding requests, compounds the problem (see Campbell 1985; Martin and McClure 1983).

According to Boehm (1986), the applications backlog intensifies two serious problems:

1. The backlog has a restraining influence on our ability to achieve productivity gains in other sectors of the economy. The applications backlog implies that many people's job still have a great deal of tedious, repetitive, and unsatisfying content, because the software to eliminate or support those parts of the job cannot be developed.
2. The applications backlog creates a situation which yields a great deal of ISs with bad

quality. It creates a personnel market in which "just about anybody can get a job to work off this backlog, whether they are capable or not."

The questionnaire contained several questions with respect to the applications backlog. The first questions concerned an assessment of the amount of development projects that were started later than originally planned or desired and an assessment of the average delay of the development projects due to the existence of an applications backlog within the organization. Figure 12 summarizes the responses to these questions.

Amount of Development Projects that Started Later than Planned

	Frequency	Valid Percentages
None of the projects	2	2.2%
1 to 10% of the projects	21	23.6%
10 to 25% of the projects	30	33.7%
25 to 50% of the projects	26	29.2%
More than 50% of the projects	10	11.2%
	9	Missing
	98	100.0%

Average Delay of Development Projects Due to the Applications Backlog

	Frequency	Valid Percentages
1 to 3 months	31	40.2%
3 to 6 months	21	27.2%
6 to 12 months	21	27.2%
More than 12 months	4	5.1%
	21	Missing
	98	100.0%

Figure 12. Delay of Projects

The findings of Figure 12 that 75% of the respondents indicated that 10 to more than 50 percent of their development projects started later than planned. The responses pertaining to the average delay of development projects did not support the existence of a three- to four year backlog as argued by Martin and McClure (1983) nor the estimation of 18 months asserted by Campbell (1985). The results of the questionnaire suggested that the applications backlog was frequently seen as a moderate or big problem (see Figure 4). However, it was not perceived as dramatically as often stated in literature. A plausible explanation for this could be the difference in interpretation of "demand" which affects the moment of entering the queue of pending applications. Furthermore, several respondents argued that the applications backlog was a question of priority setting and organizational change-over, and thus could be considered a friction problem.

Next, the respondents were asked to indicate the measures they had applied to overcome the applications backlog. They were also asked to estimate the effect of each measure applied by selecting one of the following options: more than expected, just as expected, less than expected, no effect at all. Figure 13 shows the percentages of the respondents that had applied certain measures and the estimated effect of these measures.

The respondents indicated the application of three other tools to overcome the

Measure Applied	Valid Percentage	Effect			
		More than expected	Just as expected	Less than expected	No effect at all
4GL	33.7%	12.5%	50.0%	34.4%	3.1%
CASE tool	22.4%	4.5%	50.0%	40.9%	4.5%
Other tool	7.1%	28.6%	57.1%	-	14.3%
Increase budget	21.4%	-	61.9%	28.6%	9.5%
More personnel	39.8%	2.6%	63.2%	28.9%	5.3%
Outsourcing	39.8%	7.7%	59.0%	20.5%	12.8%
End-user computing	33.7%	15.6%	46.9%	28.1%	9.4%
Reusable software	17.3%	12.5%	43.8%	37.5%	6.3%
Other measures	20.4%	31.6%	52.6%	10.5%	5.3%

Figure 13. Applications Backlog: Measures and Their Effect

applications backlog, i.e., OODBMS, productivity tools, and function point analysis. Furthermore, the application of standard packages, the solid determination of priorities, and mobilization of end-users were mentioned more than once as other measures. The findings of Figure 12 suggest that the measures more personnel and outsourcing were the two most frequently applied (40% of the respondents). The measures 4GL and end-user computing were applied by approximately 35% of the respondents. In addition, the estimated effects of the measures showed that the application of CASE tools and the implementation of reusable software had a less than expected or no effect at all in 45% of the cases.

The last question of this part of the questionnaire concerned the reasons for the existence of the applications backlog according to the respondents. It was an open question and the respondents could indicate more than one reason. Table 3 summarizes the seven most frequently encountered reasons for the existence of the applications backlog. It should be noted that many other reasons were mentioned by the respondents.

Table III. Reasons for the Existence of the Applications Backlog

Reasons	Frequency
Insufficient budget/capacity	22
Highly increasing demand for applications	15
Changes in the external environment	7
Opinion and commitment of top management	6
High proportion of maintenance	5
Technological change	5
Communication between end-users and developers	4

The responses of Table 3 indicate that the respondents most frequently viewed the insufficient size of their budgets and capacity as the reason for the existence of an

applications backlog. The high proportion of maintenance which is often suggested in literature as the prime cause for the existence of the applications backlog was only mentioned occasionally (i.e., five times). However, the high proportion maintenance consumed of EDP budgets and the insufficient size of the budgets implicitly supported the line of reasoning in literature. Today, many organizations enforced EDP budget cutbacks due to the declining economical situation of organizations. On the other hand, the demand for new applications remained high which was another frequently indicated reason for the existence of the applications backlog within organizations. Furthermore, other more or less frequently encountered reasons were:

- changes in the external environment that results in new information requirements (e.g., organizational change or changing legislation);
- little involvement and commitment of top management who do not perceive IT as high priority;
- high proportion of maintenance at the expense of new systems development;
- rapid technological advancements; and
- communication problems between end-users and IS developers.

Quality Problems

The word quality is used in everyday speech to describe the degree of excellence of a product or service. Within the area of IS, quality appears to be a hot topic nowadays (see, e.g., Khalil 1994; King 1993). Many authors (see, e.g., Manns and Coleman 1988; Boehm *et al.* 1978) pay attention to the definition, measurement, quality attributes, and implementation of quality. The quality of ISs is defined as the extent to which ISs works according to its functional specifications and the ease and safety of its modifiability without depleting either its compliance or future modifiability (see Macro 1990). Some authors argue that quality not only means "compliance to requirements" but should be extended to embrace a much broader definition: "fitness for customer use" (see, e.g., Fournier 1991). It is generally acknowledged that many ISs lack an appropriate level of quality (compare Figure 4). IS quality is one of software engineering's Cinderella subjects. It seems to be nobody's favourite. Too often it is considered an activity at the very bottom end of the development process (see Macro 1990). Lack of quality in the development process may, for example, result in ISs that are not used or that have a high level of maintenance repairs or enhancements (see Davis and Olson 1985).

The questionnaire contained several questions pertaining to the quality problems of ISs. The first question concerned an assessment of the amount of ISs that suffer from quality problems. Figure 14 summarizes the responses to this question.

The findings of Figure 14 indicate that a moderate to fairly high percentage of the ISs suffered from quality problems. About half of the respondents applied an explicit quality assurance policy for their ISs development and maintenance activities. Of the respondents that applied an explicit quality assurance policy:

- about 80% used quality manuals;
- around 45% grounded their IS development processes on the ISO 9000 series (see ISO 9000 and 9001, 1987); and
- approximately 70% set their quality specifications on beforehand.

Next, the respondents were asked to indicate the measures they had applied to overcome the quality problems. They were also asked to estimate the effect of each measure applied by selecting one of the following options: more than expected, just as

Amount of ISs that Suffer from Quality Problems

	Frequency	Valid Percentage
None of the ISs	4	4.1%
1 to 10% of the ISs	28	28.9%
10 to 30% of the ISs	37	38.1%
30 to 50% of the ISs	14	14.4%
More than 50% of the ISs	14	14.4%
	1	Missing
	98	100.0%

Figure 14. Amount of ISs with Quality Problems

expected, less than expected, no effect at all. Figure 15 shows the percentages of the respondents that had applied certain measures and the estimated effect of these measures.

Measure Applied	Valid Percentage	Effect			
		More than expected	Just as expected	Less than expected	No effect at all
Quality manuals	37.8%	14.7%	55.9%	26.5%	2.9%
4GL	45.9%	7.1%	52.4%	35.7%	4.8%
CASE tool	32.7%	3.6%	57.1%	39.3%	-
Quality metrics	11.2%	9.1%	72.7%	9.1%	9.1%
Structured programming	64.3%	6.8%	78.0%	15.3%	-
Reusable software	22.4%	15.0%	60.0%	25.0%	-
Other measures	20.4%	30.0%	55.0%	15.0%	-

Figure 15. Quality Problems: Measures and Their Effect

The respondents marked the application of standard packages, the application of standard IS development methods, and training more than once as other measures to overcome quality problems. The findings of Figure 15 indicate that structured programming is by far the most frequently applied measure to overcome the quality problems. 4GL was applied by 45% of the respondents. The estimated effect of the measures suggest that the measures 4GL and CASE tools had a less than expected or no effect at all in about 40% of the cases.

The last question with respect to quality problems, concerned the reasons for quality problems according to the respondents. It was an open question and the respondents could indicate more than one reason. Table 4 summarizes the seven most frequently encountered reasons for quality problems. It should be noted that many other reasons were mentioned by the respondents.

The responses of Table 4 indicate that time and budget pressures were among the most important reasons for quality problems. IS developers were forced by time and budget pressures to deliver and maintain ISs quickly (and dirty). Hardly any attention was

Table IV. Reasons for Quality Problems

Reasons	Frequency
Time and budget pressures (quick development and maintenance)	17
Insufficient quality of IS developers and methods	13
Little attention to requirements analysis and design	8
Communication between end-users and developers	8
Lack of quality system, criteria, and policy	5
Long life-span of existing ISs	4
Insufficient documentation	4

payed to the provision of extra quality like modifiability because of the existing short-term thinking culture. Another frequently encountered reason for quality problems according to the respondents was the insufficient quality of IS developers and the IS development methods that were used. In addition, other often indicated reasons were:

- little attention to requirements analysis and systems design;
- communication problems between end-users and IS developers;
- lack of a quality system, criteria, and policy (the importance of IS quality is often underestimated);
- long life-span of existing ISs (the old-software-never-dies problem [see, e.g., Freeman 1987]); and
- insufficient documentation.

Communication Problems

There is a rather general agreement that user involvement is a key to the success of ISs (see, e.g., Fournier 1991; Brabander and Edström 1977). The underlying axiom is that the more effective the communication between end-users and IS developers is, the higher the quality of the outcome will be. However, the benefits of user-involvement have not been convincingly demonstrated. It is also commonly acknowledged that this very communication between users and IS developers is extremely problematic (see Newman and Noble 1990). Among others, the following reasons for miscommunication are stated in literature (see, e.g., Martin and McClure 1983; Brabander and Edström 1977):

- IS developers have little knowledge of the problem domain;
- IS developers are not able to produce their conceptual designs in an understandable form for end-users;
- End-users and IS developers bring different conceptual frameworks to the situation;
- End-users are not willing to participate actively or sufficiently; and
- End-users are not able to articulate their needs.

Miscommunication between end-users and IS developers creates problems of implementation, quality and acceptance of the IS.

The questionnaire contained several question with respect to the communication problems between end-users and IS developers. The first question concerned the usual user involvement during IS development projects. Almost half of the respondents (48.5%) indicated that the end-users are involved during the analysis stage of the IS development process. Of the respondents, 46.4% answered that end-users were involved during most of the stages of the IS development process. Figure 16 summarizes the responses to this question.

End-user Involvement during IS Development

	Frequency	Valid Percentage
Hardly or not involved	4	4.1%
Involved in analysis stages	47	48.5%
Involved in most of the stages	45	46.4%
End-user computing	1	1.0%
	1	Missing
	98	100.0%

Figure 16. End-Users Participation

Next, the respondents were asked to indicate the measures they had applied to overcome the communication problems. They were also asked to estimate the effect of each measure applied by selecting one of the following options: more than expected, just as expected, less than expected, no effect at all. Figure 17 shows the percentages of the respondents that had applied certain measures and the estimated effect of these measures.

Measure Applied	Valid Percentage	Effect			
		More than expected	Just as expected	Less than expected	No effect at all
CASE tool	23.5%	4.8%	61.9%	28.6%	4.8%
Prototyping	52.0%	18.4%	51.0%	30.6%	-
Structured programming	41.8%	7.9%	76.3%	15.8%	-
Other measures	37.8%	37.5%	53.1%	9.4%	-

Figure 17. Communication Problems: Measures and Their Effect

The respondents marked the structure of project organization with end-users involvement, the training of end-users, and the transfer of responsibility for the development project to end-users more than once as other measures to overcome communication problems. The findings of Figure 17 also indicate that prototyping is the most frequently applied measure to overcome the communication problems. Structured programming was applied by about 40% of the respondents. The estimated effect of the measures suggest that the measures prototyping and CASE tools had a less than expected or no effect at all in about 30 to 35% of the cases.

The last question with respect to communication problems, concerned the reasons for communication problems according to the respondents. The respondents could select more than one of the pre-printed reasons that are listed in the following table (Table 5).

The other reasons include among others the difference in conceptual frameworks between end-users and IS developers, and lack of knowledge of IS developers about the problem domain. These findings conform to a large extent to the reasons given in literature. However, the reasons for communication problems were primarily designated to end-users. This may be explained by the sample of the survey, i.e., IS managers,

Table V. Reasons for Communication Problems

Reasons	Frequency
Little availability of end-users during development process	57
Insufficient cooperation of end-users during development process	56
Expectations of end-users on ISs are too high	50
Developers are not able to project themselves into the end-users situation	49
Fast changing end-users requirements	47
End-users cannot formulate their requirements clearly	45
Difference in knowledge level between end-users and developers	30
Other reasons	18

which results in a rather one-sided perspective on the problem.

Final Questions

The questionnaire ended with three questions. The first question concerned the respondent's familiarity with the term Software Crisis before the questionnaire was received. Of the respondents, 23.5% was not familiar with the term on beforehand. The answers to the question whether it is appropriate to use the term Software Crisis nowadays revealed that 44.9% of the respondents considered it appropriate to use the term. Some of the respondents indicated that the problems of the Software Crisis indeed exist but that the use of the term 'crisis' to indicate these problems was exaggerated and gave a wrong picture of ISs for outsiders. It is of interest to determine if there exists a relationship between the answers given to these two questions. Figure 18 shows the cross-table between the two variables: familiarity with the term on beforehand and appropriateness to use the term.

		Appropriateness		
		No	Yes	Total
Familiarity	No	15	8	23 23.5
	Yes	39	36	75 76.5
	Total	54 55.1	44 44.9	98 100.0

Chi-Square	Value	Significance
Pearson	1.24297	.26490
Continuity Correction	.76612	.38142
Likelihood Ratio	1.26246	.26119

Minimum Expected Frequency - 10.327

Figure 18. Cross-table: Familiarity and Appropriateness

Both variables were measured on a nominal scale (Yes or No). In that case, the chi-square test can be applied to determine whether two variables are related. The null hypothesis is that variables are statistically independent, i.e., unrelated. The minimum expected frequency is greater than 5, hence the observed significance level based on the

chi-square is correct. The value of the chi-square statistics is small, and the observed significance level is rather high (.26490). The null hypothesis cannot be rejected at the $\alpha = 0.05$ level. Therefore, it is unlikely that the two variables are related.

The final question of the questionnaire concerned the problems that the respondents are currently confronted with in their organization (different from but possibly overlapping with the problems discussed above). It was an open question and the respondents could indicate more than one problem. The following table summarizes the six most frequently encountered problems. Many other problems were reflected just once or twice.

Table VI. Current Problems

Problem Category	Frequency
Organizational change	10
Budget/personnel cutbacks	10
Technological change	7
Integration and complexity problems	7
Control problems due to, e.g., decentralization, downsizing, end-user computing	6
Changing role of IS organization and end-users	4

Most of the respondents referred to the problems discussed in the questionnaire as their current IS problems. The findings of Table 7 indicate that change both in an organizational and technological sense, and the EDP budget cutbacks were the most frequently mentioned problems that the organizations were currently confronted with. In addition, other problems that were more or less frequently reported, were:

- opinion, ignorance, and non-commitment of top management on IS issues;
- integration of ISs and increasing complexity;
- control problems due to developments such as decentralization, down/rightsizing, end-user computing, and outsourcing; and
- changing role of both the IS organization (e.g., the postponement from IS development to advice and consultancy) and end-users (e.g., more critical attitude to IS issues).

2.5 Clarification of Persistence

The results of the preceding section may be summarized by using them to answer the five questions stated in the introduction of this chapter. The first question concerned the existence of the Software Crisis. This question can be addressed directly and indirectly using the results of the questionnaire. When the respondents were asked if they consider it appropriate to use the term Software Crisis, about 45% of the respondents answered that it is appropriate. In addition, some of the respondents stated that the problems exist but that "crisis" is overstated and may give a wrong impression of the field to outsiders. Therefore, the use of the term is rather disputable. Furthermore, when the answers to the questionnaire are considered, it may be concluded that many of the respondents indeed did perceive the problems that are normally associated with the Software Crisis (see

Figure 4). Hence, this corresponds to the opinion of Pressman (1987) that reference to a Software Crisis can be criticized for being melodramatic, but that the term does serve a useful purpose by emphasizing existing problems found in all areas of IS development and maintenance.

The second question concerned the current extent of the Software Crisis. As can be concluded from Figure 4 (as seen from a IS Manager's perspective), most of the problems that are frequently addressed in literature formed a moderate or big problem in practice. Time and budget overruns were by far most frequently experienced as a problem by the respondents (approximately 80%). The second most frequently experienced were uncontrollability of ISs and applications backlog (around 65% of the respondents). Of the respondents, about 60% experienced communication, maintenance, and quality issues as a problem. Finally, malfunctioning of ISs was experienced as a problem by 37.1% of the respondents. The more in-depth analyses of each problem suggested that they were not as dramatic as often stated in literature. For instance, Martin and McClure (1983) argue that most organizations have a three- to four-year applications backlog. The results of the questionnaire indicated that the applications backlog exists and that it was frequently perceived as a moderate or big problem but the average extent of the backlog was only a few months.

The next question involved the cause or causes of the Software Crisis. The last question of each part of the questionnaire that discussed a specific problem, concerned the reason(s) or cause(s) for each problem. These questions were open questions and the respondents could indicate more than one reason. The following reasons for the problems were the most frequently mentioned (in random order): changing requirements, over-optimistic planning, changes in external/organizational environment, technological change, immaturity of IS development, insufficient specifications, little user and top management involvement, poor project management, high demand for new applications, insufficient budget and budget cutbacks, time pressure, and underestimation of the complexity. Hence, a number of reasons was mentioned for the existence of the problems of the Software Crisis. It is interesting to note that many of these reasons were also mentioned during the NATO conference (see Section 2.2).

The fourth question concerned the contribution of proposed solutions of the Software Crisis. The respondents were asked to indicate both the measures they had applied to overcome the problems and to estimate the effects of these measures. The results indicated that the following measures were applied most frequently by the respondents: more personnel, 4GL, outsourcing, structured programming, standardization of documentation, and prototyping. Furthermore, the results indicated that the effects of the measures applied, particularly of the measures more personnel, 4GL, CASE, and outsourcing, were often less than expected or had no effect at all. Another indication of the modest contribution of the proposed solutions to the Software Crisis, may be the persistence of the Software Crisis. None of the solutions have eliminated the Software Crisis. Based on the above discussion of the results of the questionnaire, it can be argued that the problems of the Software Crisis -- with a shift in priority and extent -- have continued for the last 25 years. This conclusion is shared by the original participants of the NATO conference in 1968. At the 11th International Conference on Software Engineering (1989) a panel was held entitled "Twenty-Year Retrospective: The NATO Software Engineering Conferences." The panellists, who had attended the NATO conferences, were asked to try and recapture what had happened to their ideas. The panellists agreed on two items, i.e., the conferences were exhilarating and memorable,

and the relevance of the ideas and topics even nowadays. Mary Shaw formulated this latter point as follows: "The thing that fascinates me most about the Garmisch Proceedings is how fresh they are even now, twenty years later."

This observation leads to the fifth question with respect to the Software Crisis; how can the persistence of the Software Crisis be explained? Based on the results of the questionnaire and the findings of the interviews during the pre-test of the questionnaire, the following three categories of explanation of the persistence could be distinguished, i.e., the influence of conservative factors, productivity eliminating forces, and the inadequacy of new methods and technologies (the three categories are subsequently discussed below). Another explanation of the persistence of the Software Crisis, which was not mentioned by the respondents but proceeds from the review of literature (see Section 2.2), could be the inclusion of more aspects during the period of its existence. As has been argued in Section 2.2, the Software Crisis initially included primarily technical and managerial aspects. However, nowadays social and political aspects are also part of the problems of the Software Crisis. Thus, although it is obvious that progress has been made, the effect of it has been overshadowed by the inclusion of other problems and aspects. The three categories based on the findings of the pre-test and the questionnaire are discussed subsequently.

1. Conservative Factors

The first category consists of the powerful influence of several conservative factors as a result of which new methods and tools are only applied on a small-scale in practice. A theoretical solution does not mean that the idea is simply applicable in practice. The conservative factors act as a kind of (social) inertia (see Keen 1981). In addition to the rather general diffusion period of 10 to 15 years before a proposed solution will be applied in practice, the following factors play a conservative/resistive role:

a. The massive investment organizations currently have in their existing ISs, developed by means of old methods and tools.

   Organizations appear to maintain these generally badly structured and ill-documented ISs rather than to replace them using new methods and tools. Due to economical, technological, social and psychological factors, organizations seems to be unwilling to apply these unfamiliar methods and tools.

b. The amateurish and improper use of new concepts.

   A frequently heard complaint is that the educational system is not in the position to deliver a sufficient amount of IS developers with proper skills and knowledge to apply the new methods and tools. Furthermore, it turns out to be very cumbersome to make the older IS developers familiar with the new methods and tools. The improper use of new concepts expresses itself by the utilization of the concepts without applying the underlying meaning. For instance, in the 1980s almost every vendor of database management systems (DBMS) labelled their DBMS 'relational' while many of these systems were far from relational (see Codd 1985). Another example of such practice is given by Dijkstra (see ASI 1982). He argues that he introduced terms like "structured programming" and "layers of abstraction" with a thoughtful underlying meaning. In no time, not the objective of these concepts but the terms itself became common property.

   It is beyond doubt that the area of IS is characterized by the come and go of so-

called 'buzzwords.'

c. The reservedness of organizations to invest in new methods and technologies.
   This reservedness can be explained by the following reasons:
   - The new methods and technologies are often technically so difficult and complex that managers find themselves confused by any explanation of their capabilities;
   - The general attitude towards IT has been changed because many of the promises have not been redeemed;
   - Due to the economical situation of many organizations, organizations are trimming their EDP budgets; and
   - It is still quite difficult to assess the benefits of the application of IT.

2. *Productivity Eliminating Forces*

The effects of the improvements in productivity, which is the main objective of the proposed solutions to the Software Crisis, are neutralized by opposite forces that are directly or indirectly connected with the improvements in productivity. Without pretending to be complete, the following list reflects several examples of opposite forces that were mentioned by the respondents of the questionnaire and/or during the interviews of the pre-test:

a. The increased demand for large, complex and integrated systems.
b. The external and organizational environments have become more complex and dynamic as a result of which information needs increase and change more rapidly.
c. The application area shifted from the support of mass, routine and stable activities to the support of activities in areas that are more dynamic by nature and are characterized by unstructured and non-routine activities such as Management ISs and Executive ISs.
d. The improvements of the quality of ISs.

3. *Inadequacy of New Methods and Technologies*

Some of the respondents argued that the new methods and technologies do not provide an appropriate solution to the problems of the Software Crisis because these 'solutions' solve the wrong problem. They indicated that issues like resistance, end-user involvement, top management involvement, and Information Economics, are hardly or insufficiently addressed by the new methods and technologies.

Although the above three categories suggest that one should be careful to assess the benefits of new methods and technologies, a different approach to the problems of the Software Crisis appears to be justifiable. Particularly, the second category indicates that the main objectives of current solutions, i.e., increasing productivity and controllability, did not solve the problems. However, this does not imply that these objectives are worthless. In our research, we apply the concept of flexibility to the problems of the Software Crisis. This perspective is based on the observation that *change* in the various IS environments was very often mentioned by the respondents as an important cause of the problems of the Software Crisis (see Section 4). Although the respondents did not refer to it as an explanation for the persistence of the Software Crisis explicitly, it is argued that the fact that ISs cannot cope with change adequately because of their rigidity forms one of the main reasons for the problems of the Software Crisis (see Tables 1, 2, 3, 4, and 6 in which change was frequently indicated as an important reason for the problems of the

Software Crisis). Therefore, the objective of our research is to examine how the concept of IS flexibility can be applied to solve/reduce the problems of the Software Crisis (see Veldwijk *et al.* 1994; Boogaard 1994; Veldwijk 1993).

REFERENCES

ANSI/IEEE Standard (1983), *An American National Standard IEEE Standard Glossary of Software Engineering Terminology*, ANSI/IEEE Standard 729.

ASI (1982), *Proceedings of the Conference on the Software Crisis*, October 1982, Utrecht, The Netherlands.

Auer, A., Levanto, M., Okkonen, A., and Okkonen, J. (1990), "Solution in Software Crisis," *Microprocessing and Microprogramming*, 30, pp. 273-280.

Baarda, D.B., and Goede, M.P.M. de (1990), *Basisboek methoden en technieken: Praktische handleiding voor het opzetten en uitvoeren van onderzoek* (Basic Book Methods and Techniques: Practical Guide for the Design and Implementation of Research), Leiden: Stenfert Kroese Uitgevers.

Babbie, E.R. (1973), *Survey Research Methods*, Belmont, California: Wadsworth Publishing Company, Inc.

Bartelds, J.F., Jansen, E.P.W.A., and Joostens, Th.H. (1989), *Enquêteren: Het opstellen en uitvoeren van vragenlijsten* (Survey: The Design and Implementation of Questionnaires), Groningen: Wolters-Noordhoff.

Bennett, K.H. (1991), "Automated Support of Software Maintenance," *Information and Software Technology*, 33, 1, pp. 74-85.

Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M., MacLeod, G.J., and Merritt, M.J. (1978), *Characteristics of Software Quality*, TRW Series of Software Technology 1, Amsterdam: North Holland Publishing Company.

Boehm, B.W. (1986), "Understanding and Controlling Software Costs," in *Information Processing 86*, ed. Kugler, H.-J., Elsevier Science Publishing B.V. (North-Holland), pp. 703-714.

Boogaard, M. (1994), *Defusing the Software Crisis: Information Systems Flexibility Through Data Independence*, Ph.D. thesis, Amsterdam: Thesis Publishers.

Brabander, B. de, Edström, A. (1977), "Successful Information System Development Projects," *Management Science*, 24, 2, pp. 191-199.

Brandon, D.H. (1967), "The Dark Side of Data Processing," *Data Systems*, July 1967, pp. 32-37.

Brandon, D.H. (1968), "Personnel Management: Missing Link in Data Processing," *Data Management*, June 1968, pp. 50-56.

Brinkkemper, S. (1990), *Formalisation of Information Systems Modelling*, Ph.D. thesis, Amsterdam: Thesis Publishers.

Brooks, F.P. Jr. (1975), *The Mythical Man-Month: Essays on Software Engineering*, Reading, Massachusetts: Addison-Wesley Publishing Company.

Bylinski, G. (1967), "Help Wanted: 50,000 Programmers," *Fortune*, March 1967, pp. 143-145/168/171-172/176.

Campbell, D.F. (1985), "Reducing the Applications Backlog with 4GLs," *Journal of Information Systems Management*, Fall 1985, pp. 8-13.

Canning, R.G. (1972), "The Maintenance Iceberg," *EDP Analyzer*, 10, 10, October 1972.

Canning, R.G. (1981), "Easing the Software Maintenance Burden," *EDP Analyzer*, 19, 8, August 1981.

Codd, E.F. (1985), "Is Your Relational Database Management System Really Relational? An Evaluation Scheme," *Computerworld*, October 1985, pp. ID/1-ID/9 and 49-60.

Cook, D., and Campbell, D.T. (1979), *Quasi-Experimentation: Design and Analysis Issues for Field Settings*, Chicago: Rand McNally.

Dambrot, S.M. (1989), "Japan Prepares for Software Crisis," *Datamation*, May 1, 1989, pp. 80.13-80.16.

Davis, G.B., and Olson, M.H. (1985), *Management Information Systems: Conceptual Foundations, Structure, and Development*, second edition, New York: McGraw-Hill Book Company.

Dijkstra, E.W. (1972), "The Humble Programmer," 1972 ACM Turing Award Lecture, *Communications of the ACM*, 15, 10, pp. 859-866.

Doke, E.R., and Swanson, N.E. (1991), "Software Maintenance Revisited: A Product Life Cycle Perspective," *Information Executive*, Winter 1991, pp. 8-11.

Donald, M.N. (1960), "Implications of Nonresponse for the Interpretation of Mail Questionnaire Data," *Public Opinion Quarterly*, 24, 1, pp. 99-114.

Forte, G. (1992), "Software Maintenance Under the CASE Umbrella," *CASE Outlook*, January-February 1992, pp. 7-21.

Fournier, R. (1991), *Practical Guide to Structured System Development and Maintenance*, Englewood Cliffs, New Jersey: Yourdon Press.

Fowler, F.J. (1993), *Survey Research Methods*, Newbury Park: Sage Publications.

Freeman, P. (1987), *Software Perspectives: The System is the Message*, Reading, Massachusetts: Addison-Wesley Publishing Company.

Genuchten, M. (1991), "Why is Software Late? An Empirical Study of Reasons For Delay in Software Development," *IEEE Transactions on Software Engineering*, 17, 6, pp. 582-590.

Genuchten, M. van, Bemelmans, T., and Heemstra, F.J. (1993), "De software-fabriek; beheersing van ontwikkeling, onderhoud en hergebruik" ("The Software Factory: Controlling Development, Maintenance and Reuse"), *Informatie*, 35, 4, pp. 258-267.

Greenberger, M. (1962), *Computers and the World of the Future*, Cambridge, Massachusetts: The MIT Press.

Hager, J.A. (1989), "Software Cost Reduction Methods in Practice," *IEEE Transactions on Software Engineering*, 15, 12, pp. 1638-1644.

ISO 9000 (1987), *Quality Management and Quality Assurance Standards: Guidelines for Selection and Use*, ISO.

ISO 9001 (1987), *Quality Systems: Model for Quality Assurance in Design/Development, Production, Installation, and Servicing*, ISO.

Jenkins, A.M., Naumann, J.D., and Wetherbe, J.C. (1984), "Empirical Investigation of Systems Development Practices and Results," *Information & Management*, 7, pp. 73-82.

Keen, P.G.W. (1981), "Information Systems and Organizational Change," *Communications of the ACM*, 24, 1, pp. 24-33.

Ketler, K., and Turban, E. (1992) "Productivity Improvements in Software Maintenance," *International Journal of Information Management*, 12, pp. 70-82.

Khalil, O.E.M. (1994), "Information Systems and Total Quality Management: Establishing the Link," *Proceedings of the ACM SIGCPR '94 Conference*, March 24-26 1994, Alexandria, USA, pp. 173-183.

King, J. (1993), "Quality Conscious," *Computerworld*, July 1994, pp. 89-91.

Lehman, M.M. and Belady, L.A. (1985), *Program Evolution: Processes of Software Change*, London: Academic Press.

Lientz, B.P., Swanson, E.B., and Tompkins, G.E. (1978), "Characteristics of Application Software Maintenance," *Communications of the ACM*, 21, 6, pp. 466-471.

Lientz, B.P., and Swanson, E.B. (1980), *Software Maintenance Management*, Reading, Massachusetts: Addison-Wesley Publishing, Inc.

Macro, A. (1990), *Software Engineering: Concepts and Management*, New York: Prentice Hall.

Macro, A., and Buxton, J. (1987), *The Craft of Software Engineering*, Reading, Massachusetts: Addison-Wesley Publishing Company.

Manns, T., and Coleman, M. (1988), *Software Quality Assurance*, Hampshire: MacMillan Education Ltd.

Martin, J., and McClure, C. (1983), *Software Maintenance: The Problem and Its Solutions*, Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

McClure, C. (1989), *CASE is Software Automation*, Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Nalven, C., and Tate, P. (1989), "Taking the Offshore Option," Datamation, February 1, 1989, pp. 72.5-72.6.

Naur, P., Randell, B., and Buxton, J.N. (1976), *Software Engineering: Concepts and Techniques*, Proceedings of the NATO Conferences, London: Mason/Charter Publishers, Inc.

Nelson, R. (1989), "Software's Midlife Crisis," *Electronics*, June 1989, pp. 68-72.

Newman, M., and Noble, F. (1990), "User Involvement as an Interaction Process: A Case Study," *Information Systems Research*, 1, 1, pp. 89-113.

Norusis, M.J. (1987), *The SPSS Guide to Data Analysis*, Chicago: SPSS, Inc.

Orr, K. (1986), "De Software Crisis" ("The Software Crisis"), *Computerworld*, July 1986, pp. 6-7.

Phan, D., Vogel, D., and Nunamaker, J. (1988), "The Search for Perfect Project Management," *Computerworld*, September 1988, pp. 95-100.

Pressman, R.S. (1987), *Software Engineering: A Practitioner's Approach*, second edition, New York: McGraw Hill Book Company.

Proceedings of the 11th International Conference on Software Engineering (1989), May 15-18 1989, Pittsburgh, Pennsylvania.

Reutter, J. (1981), "Maintenance Is a Management Problem and a Programmer's Opportunity," *AFIPS Conference Proceedings on 1981 National Computer Conference*, May 1981, pp. 343-347.

Schampheleire, W. de (1986), *De techniek van de enquête: Een inleiding* (The Technique of Survey: An Introduction), Leuven/Amersfoort: Uitgeverij ACCO.

Schneidewind, N.F. (1987), "The State of Software Maintenance," *IEEE Transactions on Software Engineering*, SE-13, 3, pp. 303-310.

Schoman Jr., K.E. (1984), "The Case of the Applications Backlog," *Datamation*, November 1984, pp. 107-110.

Sommerville, I. (1989), *Software Engineering*, third edition, Reading, Massachusetts: Addison-Wesley Publishing Company.

Staub, D.W. (1989), "Validating Instruments in MIS Research," *MIS Quarterly*, June 1989, pp. 147-169.

Steward, D.V. (1987), *Software Engineering with Systems Analysis and Design*, Monterey, California: Brooks/Cole Publishing Company.

Swanson, E.B. (1976), "The Dimensions of Maintenance," *Proceedings of the Second International Conference on Software Engineering*, October 1976, pp. 492-497.

Vinig, G.T. (1993), *Impact of CASE Technology on the Systems Development Life Cycle*, Ph.D. thesis, Amsterdam: VU Boekhandel/Uitgeverij B.V.

Veldwijk, R.J. (1993), *Introspective Systems Design: Exploring the Self-Referential Capabilities of the Relational Model*, Ph.D. thesis, Enschede: Copyprint 2000.

Veldwijk, R.J., Boogaard, M., and Spoor, E.R.K. (1994), "Assessing the Software Crisis: Why Information Systems are beyond Control," forthcoming in *Information Science: An International Journal.*

Yin, R.K. (1989), *Case Study Research: Design and Methods*, Applied Social Research Methods Series, Volume 5, Newbury Park: SAGE Publications.

1992-1	R.J. Boucherie N.M. van Dijk	Local Balance in Queueing Networks with Positive and Negative Customers
1992-2	R. van Zijp H. Visser	Mathematical Formalization and the Analysis of Cantillon Effects
1992-3	H.L.M. Kox	Towards International Instruments for Sustainable Development
1992-4	M. Boogaard R.J. Veldwijk	Automatic Relational Database Restructuring
1992-5	J.M. de Graaff R.J. Veldwijk M. Boogaard	Why Views Do Not Provide Logical Data Independence
1992-6	R.J. Veldwijk M. Boogaard E.R.K. Spoor	Assessing the Software Crisis: Why Information Systems are Beyond Control
1992-7	R.L.M. Peeters	Identification on a Manifold of Systems
1992-8	M. Miyazawa H.C. Tijms	Comparison of Two Approximations for the Loss Probability in Finite-Buffer Queues
1992-9	H. Houba	Non-Cooperative Bargaining in Infinitely Repeated Games with Binding Contracts
1992-10	J.C. van Ours G. Ridder	Job Competition by Educational Level
1992-11	L. Broersma P.H. Franses	A model for quarterly unemployment in Canada
1992-12	A.A.M. Boons F.A. Roozen	Symptoms of Dysfunctional Cost Information Systems
1992-13	S.J. Fischer	A Control Perspective on Information Technology
1992-14	J.A. Vijlbrief	Equity and Efficiency in Unemployment Insurance
1992-15	C.P.M. Wilderom J.B. Miner A. Pastor	Organizational Typology: Superficial Foursome of Organization Science?
1992-16	J.C. van Ours G. Ridder	Vacancy Durations: Search or Selection?
1992-17	K. Dzhaparidze P. Spreij	Spectral Characterization of the Optional Quadratic Variation Process
1992-18	J.A. Vijlbrief	Unemployment Insurance in the Netherlands, Sweden, The United Kingdom and Germany
1992-19	J.G.W. Simons	External Benefits of Transport