"...but some are more equal than others"

devlopment of an instrument for software performance prioritization

van den Berg, R.J.

Award date:
1992
"...but some are more equal than others"

Development of an instrument
for software performance prioritization

Roelof J. van den Berg
Graduate School of Industrial Engineering and Management Science
Eindhoven University of Technology
Eindhoven, the Netherlands

mentors:
ir J.J.M. Trienekens
prof dr H.G. Boddendijk
prof dr T.M.A. Bemelmans

internships with:
Twijnstra Gudde, management consultants N.V.
Amersfoort, the Netherlands
mentor: J.B. Loman

Carlson School of Management
University of Minnesota
Minneapolis, U.S.A
mentor: G.B. Davis, Ph.D.

Amersfoort, November 15th, 1992
ABSTRACT

In this master's thesis the development of an instrument for support of trade off decisions regarding the performance of information systems is discussed. With the final product a system's performance requirements can be derived from the characteristics of the supported task.
SUMMARY

This is a thesis about information system performance assurance by Roel van den Berg, a master's candidate in Industrial Engineering from Eindhoven University of Technology in the Netherlands.

It is based on a nine month research project during an internship. The internship was fulfilled at Twijnstra Gudde (TG), one of the major management consultancy firms in the Benelux, in Amersfoort, the Netherlands from February, 17th 1992 till November, 26th 1992. The stay with TG was split in two by a research trip to the Carlson School of Management, University of Minnesota, U.S.A, from May, 13th till August, 14th, 1992.

Performance assurance is one of the ways to enhance the quality of a system. Because of the increased international competition and the rise of standards for quality control, normalisation and certification quality has become an important issue in many industries. In the information systems business this increased importance is only strengthened by decreased growth of the market, the increased dependency on information systems in many organizations and the more critical standpoint of clients regarding information technology products.

A wide range of interpretations of the concept of quality of information systems can be found in literature. Some limit quality assurance to control of reliability of the system, others claim quality is synonymous with performance, few think it involves both the performance and the functionality and others state quality should be interpreted as success of the system.

The work of this research project was embedded in TG's project 'Quality of Information Systems'. In this project 'quality' was synonymous with performance. The objective of the TG project was to develop an instrument which would support consultants in the process of performance requirements definition. Three senior consultants from TG's Information and Technology sector had started the project at the end of 1991. They participated in it on a part time basis. The intern was a project member during his thesis work.

Performance requirements definition can be split in two phases. The first one is the information planning phase and involves the complete portfolio of planned projects. At this stage the IT budget is allocated to the different projects. The total budget for a system should be related to its strategic value for the organization. Information planners do not deal with the individual performance factors but the outcome of the information planning process puts constraints on the decisions about these factors in the second phase.

In that phase, during the development of individual systems, it is necessary to make trade off decisions about the performance of the system. Such decisions are necessary because costs increase exponentially with the performance level of a system. Secondly, it is very difficult to combine high performance levels for some factors, e.g. response time and maintainability.

A large investment in the flexibility of a system for 24-hours cash withdrawals is a waste of your money. But developers of a flight control system for a fighter aircraft can be sure their product will be a failure if they do not spend a lot of effort on reliability guarantees. It is necessary to introduce the concept "performance profile" here in order to make clear the exact objective of the thesis research. A performance profile is a list of all performance factors and their relative importance for the success of a particular application. The importance could be stated as, e.g. a 5-point scale ranging from a very low to a very high importance. It is not the same as the performance requirements because no exact goals are given. But of course the two are closely connected. The performance profile "drives" the performance requirements.

The performance profile is only a concept. In practice it is almost never made explicit. A profile is often just a list with priorities in the heads of the individuals involved in performance requirements definition. But because it has such a significant effect on the outcome of the definition process it is important to make it tangible.

Several theorists state that although performance profiles differ significantly from application to application it is possible to identify clusters of situations requiring the same performance profile. But how should 'situation' be defined. That became the objective of this thesis research.
After an initial month of literature research and orienteering on the situation at Twijnstra Gudde and their views on quality a survey was held among 12 consultants who provided information about 18 system development projects they had been involved in. The purpose of the survey was to find out if the consultants differentiated in the importance of performance factors, and if they did which factors were important and which were not and under what circumstances. Of course, how to define "circumstances" was not clear at all when the survey was designed.

28 organizational characteristics, mainly strategic in nature, were used to pinpoint the projects. Several theorists suggested this kind of factors influenced the importance of performance factors. The consultants who participated in the preparation of the survey mentioned strategic characteristics too. The use of strategic characteristics made it possible to check to what extent it would be possible to make statements about important factors for a particular system already in the information planning phase, because most of these characteristics are discussed there already.

It was found that the consultants did not consider all performance factors equally important. Some factors were always important, others only sometimes or even never. Although some relationships between the organizational characteristics and the importance of performance factors could be deducted the survey results made clear that other variables than the strategic dimensions provide important input for the decisions about the importance of performance factors.

During the stay in Minnesota the search for these determining factors was continued. In spite of the claim that typology development would be possible not many taxonomies have been created. McCall related performance factors to application/environment characteristics while Bemelmans used the typology of control situations to support decisions about performance priorities.

Because of the availability of both a number of research outcomes about the relationship between task characteristics and system performance and expertise from faculty members of the Carlson School in the area of task-system fit the task became a new perspective for looking at performance prioritization.

In the U.S. an instrument, that is typology plus questions for analysis, incorporating the idea of the task as lighthouse for performance differentiation, was developed via a prototyping approach. Faculty members and a few representatives from the industry participated in giving constructive criticism about several versions.

Because the typology had to be incorporated in a TG instrument the model developed in Minnesota was only an intermediate product. After revision by TG consultants it was decided to embed the typology in their "Integral workplace design" method. This method is a reaction to the many development projects which only take the technological aspect in consideration and neglect the socii-organizational consequences of implementation of a new system. According to the method the workplace components - task, personnel and system - should be in balance, i.e. fit each other.

In the last month of the thesis project the instrument was adjusted to the process described in the "integral workplace design" method. Unfortunately, no time was left for systematic testing. Although quite some people were involved in the development instrument much more testing is needed to find out its practical worth.

The instrument does not advocate a revolutionary new approach to system development or even performance assurance. It is embedded in a method which itself is not radically different from what for years has been accepted as a sound development strategy. The most important contribution of the instrument lies in its support of the discussion about performance requirements and its help in making explicit the assumptions which are used in that discussion. It is not an analysis tool which can be used by an expert to simply find out performance requirements, but a tool for communication. As such it may be valuable because for successful quality assurance communication is crucial.
PREFACE

During this thesis project I got help from many people.
First of all I would to thank my three mentors from Eindhoven University for their time and help. I would like to thank Theo Bemelmans especially for the effort he put in making the arrangements regarding both my stay in Amersfoort and Minneapolis.

At Twijnstra Gudde I got assistance from many employees.
I would like to thank them all for creating a pleasant working climate. I will only mention a few people here.

Jan Bas Loman, my mentor, left me enough room to make the thesis that I have made. I am very grateful for that.

I would like to thank Roland Bushof, Joop Peek and Luc Wijnen for sharing their expertise with me in the POIN056 meetings. Dirk Dekker stayed in touch with my work throughout the internship and made valuable contributions during the preparation of the survey and at the very end of the project.

I thank Nathalia Bullock, Annemarie Blokzijl and Marie-Christine Wormser for telling me many things I needed to know to more or less fit in with the TG way of doing things.

At the Carlson School the help from Gordon Davis was indispensable. During our first meeting he said I would have fun in Minnesota and he was right. (For the record: others said he would make me work and they were also right). Working with Gordon was a very stimulating experience.

Of the other faculty members I would like to mention Cynthia Beath, Dale Goodhue and James Wetherbe who took the time to critically review my ideas about performance prioritization.

The contact with the Ph.D. students was an important reason why I enjoyed my stay in Minnesota very much. I would like to mention Susan Misterick and Sandy Slaughter for their explanations about QFD and software maintenance respectively and Soon Ang for her company, for staying friendly although I frequently turned her room into a mess and for taking me to the best Oriental restaurants in the Twin Cities.

I would like to thank the secretarial staff from the IDS department for their help, especially Jan DeGross for making the housing arrangements in St.Paul.

Jim Johnson’s evaluation of the intermediate instrument and his explanations about quality assurance at General Mill’s MIS department were a valuable example of the use of SQA in practice for me.

Finally I would like to thank Stephen Wilkowski, from AT&T’s Bell Laboratories, for his valuable explanations about the growing impopularity of object related quality definitions in R&D environments and Charissa Jones, from the Application Consultants Group, for her critical analysis of the intermediate report.

Roel van den Berg
Amersfoort, November 15th, 1992
# TABLE OF CONTENTS

Summary  
Preface  
Chapter 1: Introduction  
  1.1 This report in its context  
  1.2 Quality of information systems  
  1.3 Initiatives within Twijnstra Gudde  
Chapter 2: An overview of the organizations  
  2.1 TG  
  2.2 TG: external developments  
  2.3 University of Minnesota  
  2.4 U of M: external developments  
Chapter 3: Research objectives  
  3.1 Start of TG's "Quality of information systems" project  
  3.2 The objectives of POIN056  
  3.3 Task assignment  
Chapter 4: Research plan  
  4.1 Software quality assurance: a misleading term  
  4.2 Need for improvement in SQA  
  4.3 Current practice in SQA and need for metrics development  
  4.4 Need for development of a typology  
  4.5 Research plan  
Chapter 5: Analysis of TG's approach to system quality  
  5.1 Analysis of perspectives on software quality within TG  
  5.2 Survey objectives  
  5.3 Survey preparation and analysis  
  5.4 Results  
  5.5 Conclusions  
Chapter 6: Typology development  
  6.1 Construction of the typology: first versions and evaluation  
  6.2 Discussion of the final version  
Chapter 7: Beyond information systems  
  7.1 Integral workplace design  
  7.2 "System follows task and user" versus "integral workplace design"  
  7.3 Process description of performance assurance in the "integral workplace design" method  
Chapter 8: Conclusions and recommendations  
  8.1 Conclusion  
  8.2 Recommendations  
References  
Appendix I: TG's method for project management  
Appendix II: Project result characteristics mentioned in the survey  
Appendix III: Survey results  
Appendix IV: Discussion of the first three versions of the instrument
SETTING
"According to the SEI five level process maturity model, current software engineering practice is largely at the initial level (level 1, or lowest level) of process maturity. There is a small number of level 2 organizations and a few level 3 projects in some organizations. No projects have been reported or assessed at level 4 or 5."

Watts S. Humphrey

The SEI (Software Engineering Institute) was established at Carnegie Mellon University by the U.S. Department of Defense. It is only one of the many initiatives related to software quality improvement that were taken in the last five years. The mission of the SEI is to transition improved software methods into general practice. As part of this mission, work started to characterize and report on the state of the practice of software engineering in the software community. Humphrey developed a model to rate the maturity of the development process in a software organization. As Humphrey’s statement makes clear serious need for improvement exists. But this has been a fact for decades.

In this first chapter it will be made clear why quality assurance has only recently started to become an important issue in the software industry.
1.1 This report in its context

Eindhoven University of Technology in the Netherlands offers a 4-year program in Industrial Engineering which leads to an 'ir' degree. The program consists of seven trimesters of required courses, three trimesters of electives and is concluded with a thesis based on a nine month research project. The research is done during an internship in an organization with relevant activities.

The project started at February 10th, 1992. The first three months were spent with Twijnstra Gudde, management consultants (TG) in Amersfoort, the Netherlands. The next quarter, the project was continued at the University of Minnesota, U.S.A, at the Carlson School of Management. The last three months were spent in Amersfoort again.

1.2 Quality of information systems

During World War II the importance of operations management was generally accepted. But in the next two decades the attention to quality dropped in the Western World and disciplines like marketing and finance became better bases for careers in management. In the '70s, though, when management theorists realized that one of the most important reasons for the strength of the Japanese industry was a continuous attention for quality, the interest in that issue started to increase again. Product quality was distinguished from process quality. It became clear that testing at the end of the production process did not provide sufficient guarantees for product quality, but that quality assurance of the complete process was necessary.

In Europe these global effects were intensified by the creation of standards for quality management, normalisation, testing and certification in preparation of the realization of the internal European market. Currently, quality management is recognized as one of the crucial pillars of a sound business policy by almost every industry. The information system field is no exception to that (van Genuchten 1991).

Three other developments, more or less typical for the information systems market, have contributed to the increased importance of quality in this area (van Dorst & Huijbregts 1992).

---

1 'ir' is an abbreviation of 'ingenieur' (engineer). International accrediting committees have established that the ir degree is equivalent to the M.Sc. degree.
Introduction

1. The scope of applications for software products has dramatically expanded.
In contrast to the initial years of information system development, nowadays the developer is often confronted with applications he no longer automatically understands. At the same time the current user is often a layman who is foreign to system development. Consequently, more effort is needed to prevent misunderstandings about the task of the user or technological limitations of the system.

2. The user has become more critical.
Management's attention for investments in information technology is increased. IT investment are more and more treated as normal investments. It is less frequently accepted that IT investments are necessary and wise per se.
Besides that clients have become more knowledgeable about IT and are better able to critically analyze the proposals from suppliers.

3. The information systems discipline in no longer a 'sellers' market.
The astronomical growth of the business is over. Suppliers experience increased competition.

In spite of the increased awareness of the importance of quality assurance still only a small minority of the companies in the IT business actually use methods for quality management or product certification.

1.3 Initiatives within Twijnstra Gudde

The consultants at the Information Management group of Twijnstra Gudde, management consultants N.V. have encountered in their daily work this increased awareness of the importance of system quality.

In the last quarter of 1991 they decided to expand their technique repertoire through development of an instrument for information system quality assurance.
The objective of this thesis research was to contribute to this project via research in the area of software performance assurance.
"It is not necessary to leave your house to get to know the whole world.
The farther you go, the less you will know."

Te-Tao Ching

The Te-Tao Ching is one of the oldest books in the world where the author tries to make clear what 'good' means. Some issues never lose their relevance.
Lao-Tzu advised his followers to do simple, monotonous work because that would stimulate the meditation. His ideas about the required mobility for wisdom are clear.
Some things do change in 4500 years. The designers of the Industrial Engineering program at Eindhoven University of Technology strongly disagree with Lao Tzu on both the preferred work content and cosmopolitan attitude. This last aspect contributes to the fact that they require that a thesis has been based on a of-campus research project.
If the intern would be on a quest for quality the Taoistic way this chapter would deal with the geometry of his living room. Because he is not, in this chapter the host organizations will be introduced to the reader.
**Figure 2.1** TG's operating revenues 1991, according to market sector

<table>
<thead>
<tr>
<th>Market Sector</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banking &amp; insurance business</td>
<td>17.7 %</td>
</tr>
<tr>
<td>Central government</td>
<td>23.3 %</td>
</tr>
<tr>
<td>Provincial, regional and local government</td>
<td>11.9 %</td>
</tr>
<tr>
<td>Industry, transport, distribution and trade</td>
<td>15.3 %</td>
</tr>
<tr>
<td>Health care</td>
<td>11.6 %</td>
</tr>
<tr>
<td>Other markets: profit</td>
<td>10.2 %</td>
</tr>
<tr>
<td>Other markets: not-for-profit</td>
<td>10.0 %</td>
</tr>
</tbody>
</table>

**Figure 2.2** TG's operating revenues 1991, according to services

<table>
<thead>
<tr>
<th>Service</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management consultancy</td>
<td>37.3 %</td>
</tr>
<tr>
<td>Facilities planning &amp; property consultancy</td>
<td>28.7 %</td>
</tr>
<tr>
<td>Information Technology &amp; Telematics</td>
<td>18.8 %</td>
</tr>
<tr>
<td>Interim Management</td>
<td>15.2 %</td>
</tr>
</tbody>
</table>
2.1 TG

In the first half of the 60s Andries Twijnstra, then a consultant with Berenschot\(^2\), thought about starting his own firm. He discussed the idea with Arie Gudde, a colleague, who reacted enthusiastically. On October, 1st, 1964 they founded their own consultancy bureau, which at that moment only got Twijnstra's name\(^3\). Initially office was held in Twijnstra's house and the number of employees was very limited. Originally, the services of the new firm were limited to the management of building projects. Currently, TG has over 200 consultants and over 100 employees in the support staff. Gross sales in 1991 were just over 75 million guilders. TG is separated into four sectors: Interim Management, Facilities Planning & Property Consultancy, Management Consultancy and Information Technology & Telematics. The last sector consists of three groups: Telematics Consultancy, Information Technology Project Management and Information Management. The internship for this master's thesis was fulfilled at the Information Management group.

2.2 TG: external developments

The management consultancy market is becoming more competitive. TG operates at the high end of the market. It tries to get a competitive advantage by giving better consultancy service rather than via low fees. Currently, management is actively seeking new opportunities to improve the quality of its firm's consultancy products.

The professional interaction between groups is limited. Consultants from one group do not work very often on a project managed in another group, in spite of the fact that additional expertise from another group may enhance the quality of a project. This is related to the fact that managing partners, the heads of a group, are held responsible for reaching a certain amount of billable hours with their group every month. The number of hours that is set as an objective is based on the available capacity in the group. The probability that the monthly target will be reached decreases when consultants partly start to work for other groups.

\(^2\)Berenschot is another Dutch management consultancy firm.

\(^3\)In 1970 the name of the firm was changed to 'consultancy bureau Twijnstra and Gudde' In 1978 the final change to 'Twijnstra Gudde nv, management consultants' was made.
Minnesota: the "10,000 lakes" state

Minnesota is one of the midwestern states in the north of the United States. Within its borders more than 13,000 lakes of varying size can be found. The state borders Wisconsin in the east, Iowa in the south, North and South Dakota in the West and Canada in the north. Its area is almost 7 times that of the Netherlands.

Total population is 4.3 million (1988). Exactly half of them live in the Twin Cities agglomeration (Minneapolis/St. Paul). St. Paul is the state’s capital, Minneapolis is the business center.

Currently industry is the main source of income, but agriculture is still very important. Both arable farming and stockbreeding represent a significant part of the state’s economy. Tourism is becoming increasingly important. The "10,000 lakes state" provides an abundance of resources for fishing, hunting and especially watersports.
In contrast to some of its biggest competitors in the Dutch management consultancy market (KPMG, Ernst & Young, McKinsey, Coopers & Lybrand) TG does not have an international network. So far this has not had a significant influence on the quality of its consultancy products. As a matter of fact, in a recent survey among clients of management consultancy agencies by the influential Dutch magazine FEM the quality of TG's products was perceived the highest. However, the lack of global sensors for what is going on in research and the international market may become a more serious disadvantage in the future especially in the case of the Information Technology & Telematics sector.

So far, TG consultants have been able to catch up with their competitors on the latest issues before their clients started asking about them. But the developments in the world of information technology follow up on each other more and more rapidly, leading to an increased number of issues that have to be addressed by consultants. Furthermore, the clients become more and more knowledgeable about the IT field, allowing less and less time for the consultants to become comfortable with the new matters.

2.3 University of Minnesota

The University of Minnesota was founded in 1851. With over 50,000 students it is one of the biggest institutions for higher education in the United States. Of the total people enrolled 35,000 are daytime undergraduate students, almost 9,000 are in daytime graduate programs and the rest takes evening classes. Just over 7,000 students are foreigners.

The total campus area of the University of Minnesota (U of M) is the largest in the United States. The U of M has five campuses. The three major ones are in the Twin Cities: one on each side of the Mississippi bank and one in St. Paul.

The Carlson School of Management at the U of M is one of the oldest schools of business and management in the United States. Founded in 1919, it was one of the first schools in the nation to be accredited by the American Assembly of Collegiate Schools of Business. Today, more than 3,000 day and evening students attend the school, both in undergraduate and graduate programs. In 1986, the school was renamed to honour Curtis L. Carlson, an alumnus, entrepreneur and chairman of one of the largest privately held companies in the country.

---

4Because quite a number of TG employees have part time jobs at universities in the Netherlands the company is not isolated from research, of course. But international developments can only be traced indirectly this way.
An overview of the host organizations

The school's faculty is divided over six departments: Accounting, Finance, Operations Research, Marketing & Logistics, Strategic Management and Information & Decision Science. The Carlson School's graduate program in management information systems (MIS) and the MIS Research Center are the oldest of their kind in the world. A recent survey in *U.S. News and World Report* rated the Carlson School IS faculty second only to MIT's. The Carlson school does not offer an undergraduate business program with MIS specialization.

2.4 U of M: external developments

Cutbacks in both state and federal education budgets have had a significant impact on higher education institutions in the United States. One of the effects is the steady rise of tuition fees, meaning less and less non-affluent people can afford higher education. This will affect the U.S. position in international competition and it will accelerate the current process of the disappearance of the middle class.

In the United States much bigger differences in quality of education and research between institutions of higher education exist than in the Netherlands. It is much more common to rank the colleges and universities. The position in the hierarchy affects the eligibility for funding programs. For this reason American universities generally put more effort in making clear, e.g. via publications, that they are doing high quality research than European institutions.

Besides that most universities have switched to more aggressive fund raising to compensate for the loss of governmental support. The U of M fund-raising organization keeps records of more than 400,000 potential donors (alumni, faculty and past donors) for direct mailing purposes and 50 student workers spend five nights per week calling potential donors.

Cuts in government funding may affect the quality of the research at American universities in the long run. But currently, in most fields the leading departments are still American and researchers from all over the world are still very eager to spend some time at one of the better American universities.

For graduate schools the situation is similar. A recent *Newsweek* survey rated the American graduate school system as the best in the world. More than half of the American graduate degrees go to foreign students.

---

In 1980 the relative frequency of people in the categories affluent, middle class and working class was 10, 60 and 30 percent of the total population in the United States. Predictions for the year 2000 are that these figures will have changed to 30, 30 and 40 percent respectively.
CHAPTER 3
RESEARCH OBJECTIVES

"The Gileadites seized the fords of the Jordan before the Ephraimites arrived. And when any Ephraimite who tried to escape said: 'Let me cross over,' the men of Gilead would say to him: 'Are you an Ephraimite?' If he said: 'No' then they would say to him: 'Then say 'Shibboleth'. And he would say 'Sibboleth,' for he could not pronounce it right. Then they would take him and kill him at the fords of the Jordan.'

Judges 12:5-6

It was easy to distinguish right from wrong for the Gileadites. Much easier than it is in discussions about definitions of quality. In the latter case it is very likely a universally applicable definition of quality will never be found.

Still, all over the world many discussion groups started to deal with the question: what is quality? Many people have participated in these discussions. The fact that making incorrect statements had less serious consequences than in the case of the test of the Gileadites may have contributed to this enthusiastic participation.

In this chapter one of those initiatives, focusing on quality of information systems, will be discussed.
Research objectives

3.1 Start of TG's "Quality of information systems" project

To ensure that the consultancy services of TG are regularly adjusted to qualitative shifts in demand and incorporate state of the art knowledge about the problem area as much as possible consultants devote part of their time to what is called product development. Product development is done in teams, on a part time basis. Most product development teams consist of consultants from only one consultancy group. In the fourth quarter of 1991 the product development project 'Quality of information systems' (coded POIN056) was initiated in the Information Technology & Telematics sector.

3.2 The objectives of POIN056

The objective of this project was to develop an instrument that would support consultants in the I&T department in making decisions regarding quality of information systems. Three senior consultants from the Information Management group became involved in it. From the beginning of the internship their meetings were attended. At the end of March the group members had decided on the strategy to reach the objective. They stated six subgoals for the project:

1. List the factors that are relevant with respect to quality of information systems.
2. Develop a better insight into the importance of the factors. Are all factors equally important? Is the importance of one particular factor unaffected by the type of application?
3. Develop scales and measurement prescriptions to measure the different factors. This makes it possible to set unambiguous quality requirements at the beginning of the development process and to standardize progress checks throughout the development process.
4. Develop scales to normalize the measurements mentioned in 3. A statement like "the mean time between failures for your system X is 23 hours" is not very insightful to people with a low level of IT knowledge. But with extra information like "a comparable system of your competitor A has a MTBF of 95 hours and the systems of your other competitors are even better in this respect", even a decision maker without a good knowledge of IT will be able to understand that the reliability of his system is quite worrisome. The scales meant here transform the measurement mentioned in 3 to a rating based on a comparison with relevant other systems, e.g. systems of competitors in the same market.
5. Relate the quality factors to development techniques to enhance the level of quality.
6. Develop a procedure for the use of a decision tool for information system quality assurance,
Neural nets: another type of information systems

In this thesis only performance aspects of traditional information systems and especially management information systems will be discussed. No attention will be given to neural nets.

Neural nets are electronic models of the mind. They are especially useful in situations requiring "maybe" statements in contrast to the "yes-no" problems which can be dealt with by using traditional algorithmic problem solving methods. A typical example of "maybe" thinking is pattern recognition. In this area algorithmic solutions only offer limited success. Neural nets are much more powerful.

Research on neural nets started more than 40 year ago. But only since it is possible to execute digital simulations of the parallel processes in the brain on relatively cheap and powerful workstations practical use of neural net technology has started\(^1\). Neural nets do not require specific computers. Systems run on IBM PC's or SUN workstations. Of course, in the case of large applications powerful computers are needed. For some applications requiring intensive processing special multiprocessor systems are used.

In the 80s the research on neural nets intensified. But still a lot of issues are unexplored. According to Wetherbe and Straub (1990) neural net technology is still far from reaching maturity. Performance is one of the untouched items. Not many systematic results about other performance aspects are available. Most of what is known about it became available as a side effect of a test. As in the case of the test results of the MINOS II, a system designed to classify pictures with military symbols, from which it can be concluded the system showed graceful degradation. An increasing number of civilian applications of neural nets is developed (e.g. from speech synthesis, decision about mortgage applications and even estimations of cost of software development projects).

But the fact that most neural net applications are still military complicates research about them.

\(^1\) At the basis of these practical applications stood a bunch of brilliant monomaniacal undergraduates at MIT. Originally members of a model railroad club on campus they soon found a greater challenge in creating hacks on the first versions of Digital minis during illegal nightly visits to MIT's PDP-lab (which was lead by John McCarthy and Marvin Minsky) than in adding electronic features to the complex model railroad. Minsky soon found out about the talent of the hackers (among them Richard Greenblatt, Stuart Nelson and Peter Deutsch) and provided them with a more official status, which soon resulted in research programs funded by the Department of Defense and an official department of Computer Science at MIT.

Because of their complete devotion to AI applications (especially to an attempt to develop a program for the control of a robot arm which would allow it to play table tennis) the study results of the hackers deteriorated and they almost all dropped out. (This did not count for Peter Deutsch who was only 12 years old at the start of the experiments. He went to Berkeley at the end of the 60s and graduated.) The high correlation between enthusiastic involvement with neural nets and a significant drop out risk from above average engineering schools did not influence the intern's decision to focus his attention on another than this fascinating field.
Research objectives

which incorporates the findings related to the five other subgoals

3.3 Task assignment

In June, 1991 the Dutch Software Engineering Research Center started a research project called Quint, on software quality assurance. The objectives of the Quint project were comparable with the joint results of subgoals 1, 3 and 5.

In March, 1992 it became clear that the Quint project would end in September, 1992 and that the results would be published as a handbook on software quality. Because of the level of expertise and manpower available in the Quint project the members of the TG group thought it would be of not much use to spend much time on similar kinds of research. They decided to focus on achieving subgoals 2, 4 and 6 instead.

By the time the tasks related to these subgoals were assigned to the members of the project group it had already become clear that a major part of the thesis research would be in the area of subgoal 2. Since the beginning of the internship most of the literature study had been focused at this issue and a survey had already been developed to find out to what extent practitioners used a contingency approach in the prioritization of performance factors. For these reasons the intern got the responsibility for subgoal 2 while the three consultants became responsible for subgoals.

When the subgoals were accepted the group had not yet exactly defined ‘information system’. However, the previous week it had already been decided during discussions with the mentors that the thesis research would emphasize software quality. Consequently, most literature discussed in the remainder of this report will be related to this issue.

---

6. The procedure the consultants had in mind was the following:
1. at the beginning of the life cycle top management would set a target quality grade for the future system.
2. during the requirements analysis stakeholders would express their quality requirements via 10-point scale ratings per quality factor.
3. the total grade of the expected result (result grade) would then be calculated using the results of subgoal 2 to attach weights to the quality factors. The result grade should be equal to the target grade. In the case of a difference adjustments of the requirements would be necessary. This way the stakeholders would not be exposed to the metrics or very technical considerations.
This procedure was a ‘wouldn’t it be nice if’ proposal. In the group much time was spent on discussions about the validity of multiplications of scores from different scales.

7. The Software Engineering Research Center (SERC) in the Netherlands was founded in 1987. Its main objective is to increase expertise on software engineering in the Netherlands via academic research with a high relevancy for the industry and to stimulate the transfer of software engineering knowledge from universities to industrial companies.
The Quint project is a joint effort of Akzo Systems, BSO, Cap Gemini Pandata, Volmac and the Institute for Applied Computer Science of TNO.
The objective of the initiators of the project was to develop a standard in software quality measurement by means of a handbook.

8. Because of discussions about the contents of the final version of the handbook it was not presented to the press until November, 5th 1992. The intern attended this presentation.
RESEARCH
"If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle."

"The Art of War"
Sun Tzu

The words of "The Art of War" are 2500 years old, but their topicality did not decrease. The opposite is true. In recent years Sun Tzu's principles have been successfully applied in many matters other than military ones.

In this chapter the current problems in the system development business regarding knowing yourself - use of metrics - and knowing the enemy - knowledge of the application - are discussed.

Based on these findings the research plan is explained.
Quality: a journey not a destination

Numerous definitions of quality exist. Some are very much related to each other, others are very distinct. Several authors have made taxonomies of quality definitions, e.g. Garvin (1984).

Because quality is much more a journey than a destination the most important criterion to judge definitions of quality should be their ability to motivate people throughout the organization to create customer satisfaction. No quality definition is perfect in this respect. Pirsig's distinction between object and subject related definitions of quality will be used to make that clear. Almost all useful quality definitions are either object or subject related.

The object related definitions share the idea that you only have to look at a product to see if it has quality or not. It allows engineers to work in isolation, using their set of development techniques to make products as they think the customers want them. The problem with the assumption that customers want products is that they do not. Customers want solutions to their problems.

All subject related definitions have this last idea as a premise and consequently share a strong customer orientation. But because the uniqueness of the customer and his perception of a good product is emphasized users of a subject related definition often get stuck in describing the problem. But it is necessary they relate the problem to a custom made mix out of a set of standard development techniques to make sure the developers are sent into the right direction.

A good quality assurance method incorporates three things: analysis, to understand the user's problem, development techniques, to make the products, and a mechanism for translation from the analysis world to the development world, to make sure the user eventually gets a solution.
4.1 Software quality assurance: a misleading term

System quality assurance should not be an end in itself. The ultimate goal of every system development project is a successful system. But it is possible that all measures to achieve that are incorporated in a system quality assurance policy. This of course depends on the content of that policy, which in turn is related to the used definition. See "Quality a mission not a definition" for a discussion of different ways of looking at quality.

Intuitively one expects a high quality product to be a 'good' product (Pirsig 1984). And thus one expects software quality assurance (SQA) processes to separate 'good' from 'bad' software. When is a software product good? It is easier to answer the opposite question: when is it not? A software product can become a failure after delivery for three major reasons:

1. Acceptance problems: potential users do not even try the software because of fear, ignorance or other organizational factors.
2. Low functionality: the users conclude they do not need the functions offered by the system.
3. Low performance

Based on the intuitive notion of what quality is one may expect that SQA deals with preventing problems in all three areas. But this is not true. In practice SQA only deals with performance assurance, often only with reliability (Chow 1985). Even theorists limit SQA to this area (McCall [1978], Boehm [1978], Salisbury [1973], Evans[1987]). Basili (1991) is one of the very few who relates functionality to quality by stating that a missing function is equivalent to a function with a mean time between failures = 0.

This does not mean that the industry has not paid attention to the other two issues. The problem of acceptance is getting more and more attention (DeLone & McLean 1992) and the impact of both user involvement during development and proper user training on system acceptance are already widely accepted ([Verhaart 1991], [Zaal 1991]). The main objective of the wide variety of system development methods is to elicit the system functionality that is needed (Olle et al. 1988). In IT-assessment procedures, measurement of functionality of existing systems gets a significant amount of attention (van der Zee 1989).

These areas are not neglected. But they are not regarded as part of quality assurance. This makes the term system quality assurance misleading. The fact that a product passed a sound SQA process, does not automatically mean it will be a success.9

---

9In the remainder of this report SQA, quality assurance, software performance assurance and performance assurance will be regarded as synonymous. The same applies to software quality and software performance.
During a development project the actual situation is somehow measured and compared with norms. Ideally it is found the project is on schedule. If not it has to be decided to either take corrective action or replan. Correction is preferred. Crucial in this whole cycle is the existence of norms. This has been the most serious problem in application of control cycles in information systems development. Developers claimed their work could not be measured.
4.2 Need for improvement in SQA

In contrast to issues of functionality and acceptance, software performance assurance has not been able to attract significant attention. But concern for the performance of software is important. Ninety percent of the system development costs are related to bringing the performance to a level that is acceptable for the users while only 10 percent is used for fulfilling functional requirements (Bemelmans 1991). But during the requirements analysis only 10 percent of the time is spent on performance requirements\textsuperscript{10}. Established methods for performance assurance do not exist.

4.3 Current practice in SQA and need for metrics development

In many development fields it has been generally accepted that control cycles are needed to manage the development process. In figure 4.1 such a control cycle is discussed. Software development was an exception (DeMarco 1983). Practitioners claimed that because of the exceptional nature of both the software development process and the resulting software product attempts to use traditional control techniques in this business would be doomed to fail\textsuperscript{11}. Rockwell (1991) uses a cultural/historical analysis to explain this scepticism. Because of their power in a sellers market the developers were not forced to reconsider this attitude. Till the end of the '80s, when the market started to become less tolerant of the lack of discipline that was associated with it.

To survive, software developers actually had to be in control, not just think they were. Basili (1991) is one of the people who claims that software quality control without using a traditional control cycle is impossible.

Currently, many software companies that take the time to formalize their processes proceed as if a single monolithic software process could meet all project needs. Current practice, where SQA does not get much attention until the acceptance test, seems to be based on the implicit assumption that the major preoccupation of quality assurance is reliability.

But software performance involves a wide range of factors. In the last twenty years

\textsuperscript{10}The high costs involved in satisfying the users about the performance of the system is related to the lack of specification. Connath (1990) found that fulfilled expectations have a strong positive effect on overall satisfaction. Clear specifications prevent rise of unrealistic expectations.

\textsuperscript{11}This only applies to the situation in the U.S. and Europe. Japan is an exception. Already at the beginning of the 70s some large Japanese firms (like Hitachi and Toshiba) started 'software factories' based on the idea that software development could be managed the same way as traditional production processes. Initially the Western world was very sceptical about these Oriental initiatives. But their success contributed to the growing acceptance in the West that maybe after all controlling software development the traditional way was possible (Cusumano 1989).
Evans' typology

In their book "Software Quality Assurance" Evans & Marciniak discuss a typology of situations and their connection to performance factors. This typology can also be found in the work of other theorists, but Evans and Marciniak provide the most elaborate description about it.

Its importance lies in the fact that it makes people aware of the need to prioritize performance factors. The connections mentioned in the typology are intuitive. They are not based on research findings. The characteristics in Evans' typology do not share a common denominator, they are not attributes of one entity.

This will make it difficult for participants in the discussion about performance requirements to see what really matters and notice when non-relevant arguments are brought into the discussion.

The biggest disadvantage of the typology is that its dimensions are more extremes on a scale than scales themselves. Human lives, e.g. could be regarded as the extreme on the "output criticality" scale and long life cycle as the extreme on the "expected life time" scale.

The problem of talking about extremes instead of scales is that a conclusion about whether or not one is dealing with an extreme does not solve the prioritization problem. For instance, it may be true that integrity, reliability etc. are important when a system can affect human lives. But when it does not, does that automatically mean those factors are not important?

Because most systems are not extreme cases but score somewhere in the middle Evans' typology can only provide limited support in discussions about performance requirements. For the cases which are not extremes people who have to decide about performance requirements need more guidance for differentiation.

Evans seems to be aware of the rigidity of his typology because he adds a discussion about how to prioritize the factors to it. He uses a technique similar to Kepner and Tregoe's decision analysis. But the arithmetics related to prioritization can only be of secondary importance. What really matters is why a certain factor is given a certain weight. This is not covered by the typology.
several people have developed structures, mainly hierarchical, incorporating those factors or some of them in slightly different ways. The best known structures are those by Boehm (1978) and McCall (1978). Most of these factors require attention in an early stage of the development process, because in the testing phase it is really too late to influence them.

Making software performance factors operative is necessary to make sure user needs are communicated to developers via unambiguous objectives, and to enable those developers to measure the progress with regard to them in the subsequent development phases.

So far, the number of metrics which can be directly related to the quality of intermediate products in the development life cycle with regard to a particular factor are very limited. Most of what is available deals with reliability (Dobbins (1987), Musa (1979). Currently most metrics which can be used before the final test phase focus on characteristics like complexity, e.g. McCabe (1976), and thus measure quality only indirectly.

4.4 Need for the development of a typology

There is another reason why SQA is to a large extent, contrary to what current practice seems to imply, an early life cycle issue. Whether or not it is possible to unambiguously specify what exactly has to be achieved with regard to each quality factor, to reach high objectives for all of them requires a tremendous amount of development resources, i.e. ultimately time and money.

This is primarily because for most factors costs are often an exponential function of the performance level.

A second reason is that frequently performance requirements are conflicting. Optimization of algorithms has a positive influence on, e.g. response time, but it has a negative effect on the portability of the software product. These interactions are determined by the current arsenal of development techniques. New techniques may only have strong effects on one factor or a particular set and no effect at all on the rest. Until that will be the case the user of the previously mentioned software quality hierarchies should realize that performance factors have much more the characteristics of network elements.

---

12 In the Netherlands the hierarchical structure developed by Delen and Rijsenhijn (1990) is very popular.

13 Because of an improvement of, e.g. response time, with a factor 2, the costs related to that factor will increase 300%. Or 500%, but not 50%.

14 Boehm mentions some influences between quality factors, but he does not include them in his structure. He only states that more research is needed on the interactions.
<table>
<thead>
<tr>
<th>Application/environment characteristics</th>
<th>Software quality factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human lives affected</td>
<td>Integrity</td>
</tr>
<tr>
<td></td>
<td>Reliability</td>
</tr>
<tr>
<td></td>
<td>Correctness</td>
</tr>
<tr>
<td></td>
<td>Verifiability</td>
</tr>
<tr>
<td>Long life cycle</td>
<td>Maintainability</td>
</tr>
<tr>
<td></td>
<td>Expandability</td>
</tr>
<tr>
<td>Experimental system or high rate of change</td>
<td>Flexibility</td>
</tr>
<tr>
<td>Experimental technology in hardware design</td>
<td>Portability</td>
</tr>
<tr>
<td>Many changes over life cycle</td>
<td>Flexibility</td>
</tr>
<tr>
<td></td>
<td>Reusability</td>
</tr>
<tr>
<td></td>
<td>Expandability</td>
</tr>
<tr>
<td>Real time application</td>
<td>Efficiency</td>
</tr>
<tr>
<td></td>
<td>Reliability</td>
</tr>
<tr>
<td></td>
<td>Correctness</td>
</tr>
<tr>
<td>On board computer application</td>
<td>Efficiency</td>
</tr>
<tr>
<td></td>
<td>Reliability</td>
</tr>
<tr>
<td></td>
<td>Correctness</td>
</tr>
<tr>
<td>Processing of classified information</td>
<td>Integrity</td>
</tr>
<tr>
<td>Interrelated systems</td>
<td>Interoperability</td>
</tr>
</tbody>
</table>

Figure 4.2 Evans' typology
For these reasons several people have stressed the necessity to differentiate in importance of quality factors. McCall (1980), Evans & Marciniack (1987) and Perry (1987) all discuss a rudimentary taxonomy which relates systems and situations to important quality factors. The fact that they all use the same taxonomy makes clear progress in this area has not been spectacular. See 'Evans typology' and figure 4.2 for a discussion of the taxonomy.

The need to prioritize quality factors is not typical for software quality. Garvin (1983 and 1984) states that it is necessary for companies to specialize in certain dimensions of quality unless they are willing to charge unreasonably high prices for their products. Saha (1989) claims that Japanese companies do not make economic trade offs with regard to quality because of the Confucian ethic which forbids selling imperfect goods. Under the same token Kishida (1987) explains why the use of contracts for software development projects is highly uncommon in Japan. But Garvin clarifies that even the Japanese have concentrated on specific dimensions while disregarding others. 'Software development in Japan' makes clear this applies to Japanese SQA as well.

The need to prioritize is not a new problem either. Since quality management emerged as a discipline in itself, people have wondered how far they should go in making their product perfect. In the 1940's, Juran already mentioned in his Quality Control Handbook that in many situations a vital few elements of an assortment account for most of the total effect, and contrawise, the bulk of the elements (the trivial many) account for very little.

Juran named the phenomenon "the Pareto principle" and used a biblical example to make his point (Juran 1979). The example clarifies that this trade-off approach is not even something typical for quality management but just another example of allocation of limited resources, a problem that man has been struggling with since the beginning of his existence.

McCabe claims he is the one who introduced the Pareto principle to software quality (McCabe 1987). He gives multiple examples of ways of using the Pareto principle in this field. Figure 4.3 gives an overview of the different approaches used by McCabe in one particular case.

---

15. Garvin suggest that companies make a strategic choice for a certain quality niche. This implies he is skeptic about organization's flexibility in covering different niches at the same time. Whether he is right or not has of course important consequences for those who accept a contingency approach.

16. Juran named the principle after Vilfredo Pareto (1848-1923), an Italian economist. He later footnoted in his Quality Control Handbook that this was a mistake. Pareto Quantified the extent of inequality or nonuniformity of the distribution of wealth, but had not generalized this concept to other fields. The curves Juran used in the first edition of his book should have been identified with M.O. Lorenz, a Swiss economist, who had used the curves to depict concentration of wealth in graphic form. In spite of all this the name "Pareto principle" has endured for the phenomenon mentioned above, which has been encountered in a large number of disciplines.

Software Development in Japan

Japanese specialized in custom-made software. About 94% of the software produced in Japan is custom made. In the West custom made software is only 35% of total production. Japanese software developers were weak in product innovation. It is not necessary for them to export software programmes or compete for big software development contracts abroad because demand for software still exceeds supply in their country. Based on these two facts many people in the West have concluded that the Japanese are weak in software development. This is a wrong conclusion.

In comparison with their Western competitors the Japanese have been more serious about applying 'software factory' ideas: introduction of good procedures and tools, refinement of techniques for project management, advanced technology for support of reuse and automated programming.

This more systematic approach to software development has resulted in the Japanese being the leaders in reliability and maintainability of custom-made software. The Japanese consider the problems related to reliability control and project management as being tackled. With the 'continuous improvement' philosophy in mind they are now preparing themselves to significantly enhance the functionality and userfriendliness of their products. In 1988 one of the conclusions of a survey by Nikkei Computer was that at that moment functionality of Japanese custom made software was already as good as that of American products. A Dutch team of researchers which travelled to Japan in 1991 to compare practices in information system quality assurance found that the Japanese take user friendliness much more seriously than their Dutch competitors (Giezeman et al. 1992).

In the area of standard software the Japanese are still behind. But Western 'Japan watchers' warn that with the new accents on development of custom made software Japanese software factories will prove an even more direct threat for Western products (Cusumano 1989).
The 'quality of the product' means of applying the principle is the most interesting with regard to the development of a typology. McCabe applies Pareto analysis in the second step (measurement of the operational quality of the system components) but he does not use it in the first step: supplying an operational definition of quality. Selecting a subset from the assortment of factors is certainly an example of distinguishing the vital few from the trivial many. Why is it not possible to apply Pareto analysis here?

In order to use Pareto analysis it is necessary to allocate fractions of the total value of a system to the individual performance factors. Currently it is already very difficult to establish the value of an existing system. Only recently some progress has been made in systematic benefits analysis (Parker & Benson (1988), Sebus (1991)). Cost estimation is still highly inaccurate, in spite of the large number of available estimation models (van Genuchten & Koolen (1991), Janssens (1990) and information economics are still too immature to be of significant help in isolating important performance factors (Heemstra (1987), van der Vos (1990)).

4.5 Research plan

Because currently the usefulness of purely quantitative cost/benefit techniques is only very limited other tools are needed to isolate important performance factors. In this research project the aim was to contribute to improved insight in the importance of performance factors via development of a typology.

One way to do this is taking an already existing typology and making adjustment to it. But because of the serious shortcomings of the taxonomies already available it was decided to try to develop a new taxonomy instead. This goal was incorporated in the research plan. At the very beginning of the project only one step in the research plan was fixed:

1. Introduction to quality and system quality
   a. via literature research and
   b. talks with experienced people, during the first weeks of the stay with TG.

At the beginning of the internship the primary goal became:

2. Determine objectives of TG’s "Quality of Information Systems" project.
   a. via participation in the project group and
   b. talks with non project group members, to find out TG’s approach towards quality.

This has been discussed in chapter 3.
<table>
<thead>
<tr>
<th>Ways to apply Pareto principle</th>
<th>Content of the application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manpower</td>
<td>Identify the different functions performed in the SDLC (e.g. planning, testing) and then analyze the personnel expenditure on each task.</td>
</tr>
<tr>
<td>Cost of contracts</td>
<td>Construct a distribution of the funds spent by contractor. The result should indicate to which contractor the program should first be applied.</td>
</tr>
<tr>
<td>By release</td>
<td>Analyze historically the various releases processed by the development team and rank their quality.</td>
</tr>
<tr>
<td>By function</td>
<td>List the functions of the development department as under manpower discussed above. Then determine which of the functions leads to the most problems.</td>
</tr>
<tr>
<td>Quality of the product</td>
<td>First the department decides how to define quality by prioritizing the software quality factors and selecting a subset of the factors as an operational definition of quality. Once the definition of quality is agreed upon, the second step is to apply it to different components or packages in a release. This results in a quality distribution through which Pareto analysis can identify the critical components.</td>
</tr>
</tbody>
</table>

Figure 4.3 Different ways of using the Pareto principle
When the objectives had become clear the plan could be elaborated.

3. Develop a typology for performance factor prioritization
   a. via literature research and
   b. a survey among TG consultants.

restrictions: the typology is to be used in practice. It should be easy to comprehend and easy to use.

At the beginning of the project it was expected that the development of the typology would take three months. Therefore additional planning was done.

4. Develop measures for some performance factors mentioned in the typology.
   a. via literature research and
   b. interviews with practitioners about the needs.

But during the project not much time could be spent on this issue.

5. Test both the typology and the metrics using TG clients
   It has not been possible to do this during the project.

The remainder of the project will be related to the development of a typology.
"There are only two qualities in the world:  
efficiency and inefficiency;  
and only two sorts of people:  
the efficient and the inefficient."

G. Bernard Shaw 1856-1950

If Shaw is right then one thing is sure: TG consultants can not afford to be classified as members of the second group.

In this chapter the introduction to the typology development will be discussed. As a first step a survey was designed which was held among TG consultants to find out if they make trade off decisions regarding system performance during development projects.
effectivity of the specified decision support
level of fulfillment of the specifications

functionality

performance

correctness
continuity
response time
availability

maintainability
portability
connectivity
reusability

user friendliness
fit with manual procedures

Figure 5.1 The schema used in the orienteering interviews
5.1 Analysis of perspectives on software quality within TG

The internship with TG started on February, 17th, 1992. The previous week was used for literature research on software quality, at Eindhoven University of Technology. The first four days of the internship were spent in the same way, except that this time general information about TG and TG methods related to quality of information systems were analyzed in preparation of the introductory meetings with consultants from the Information Management group. The purpose of these meetings was to find out the existing views about quality in the group.

In the previous chapter it already has been made clear that with regard to quality a wide range of definitions and many different viewpoints are possible. To provide a common reference in the discussions with the consultants and thus facilitate the communication a schema was made at the end of the first week. An abbreviated version of the schema is shown in figure 5.1.

The schema is based on the notion "system success" and a formula related to it.

\[
\text{system success} = \text{quality} \times \text{acceptance} \quad \text{(Bemelmans 1991)}
\]

In this formula quality refers to product characteristics only. It is restricted to functionality and performance of the system. Acceptance is a separate factor. Some theorists claim acceptance is determined by the quality of the system. Davis (1990) relates the acceptance of a new system to its usefulness and usability, which in his definitions are very similar to what is generally regarded as functionality and performance. But although quality and acceptance may be strongly interrelated a closed cause-effect relationship between the two does not exist. Thus acceptance was introduced as an additional factor to explain (lack of) system success.

Since the purpose of the schema was to initiate discussions and not killing it these assumptions were explained during the meeting, but the schema was explicitly presented as just one possible way of looking at quality.

Five consultants from the IM group were approached about the subject. Most consultants restricted quality of information systems to performance factors. As made clear in chapter 4 this is a common view in the literature about software quality, but to hear it from TG consultants was still somewhat surprising. This is related to TG's quality management philosophy. This is part of an in house developed project management method. A brief description of this method is given in appendix I.

The TG consultants assumed that no extra control is necessary for the functional requirements because they are sufficiently taken care of in the 'normal' activities of the project.
Two levels of prioritization

The requirements of an information system are determined by the application and the environment (Davis 1985). The environment incorporates the constraints set during the information planning. During the information planning the quality level which is needed for a particular system is established based on its strategic value (Wetherbe 1983), Theeuwes (1991)). Often this quality level is not explicitly mentioned. But it can be derived from the budget which is allocated to the development of a new system or the maintenance of an existing one. Information planners do not deal with the importance of the individual performance factors.

Those are taken into account at the second level. In the phases which follow up on the information planning a particular system is analyzed with an increased amount of detail. Eventually one tries to find out if a factor, e.g. response time, is important or not. The fact that a performance factor is important does not automatically mean developers will be pushed to the limits to achieve high standards. The restrictions set during the information planning may make it impossible to spend a lot of resources on any factor. Of course, every system needs a performance which exceeds a certain lower limit. Information planners who initially allocated a low budget to a particular system may be forced to reconsider their plans because designers tell them it is impossible to develop an acceptable system with that amount of money.
The obviousness of functional requirements is easily overestimated for many artefacts in our society. Even in the case of a product like a chair the designer is confronted with a high level of uncertainty about exact product utilization. In the case of an information system the application is much less clear. And still most often the outcome of a requirements analysis is already a concept of a system. The largest pitfall for analysts is to translate the problem into a product to soon ("just another marketing/warehousecontrol etc. system").

This "seeing as" phenomenon occurs in many disciplines and individuals with a high level of professionalism and experience tend to be more vulnerable to it (Schon 1982). TG consultants are aware of this. But by taking a more active stand against it they could turn this awareness, combined with their expertise in understanding organizational processes, into a competitive advantage.

Although the consultant agreed on the definition of quality they all suggested different elements from the schema as focal points for the research. This had to do with their own interest and in some cases with the fact that they considered the performance issues as already sufficiently covered by others.

In addition to the discussions in the IM group, a testing expert from the Information Technology Project Management and a quality specialist from the Profit Group (Management Consultancy sector) were interviewed18.

On March, 6th, during a meeting with both the mentor from Eindhoven Tech and the one from TG, it was decided to limit the research to performance factors. During that same meeting a survey was planned as a first step towards such a typology.

5.2 Survey objectives

The objectives of the survey were as follows:
1) To find out if in an information system development project managers consider all quality factors as equally important or differentiate between factors. Based on the initial interviews differentiation was expected.
2) To find out which factors they regard as important and which they more or less neglect, in the case of differentiation.
3) In case prioritization differs from project to project, to find out if it would be possible to predict the importance of quality factors using project result characteristics.

---

18 The first consultant had developed a method for software testing. The second consultant had a lot of experience with the ISO-norms on quality and attempts of quality improvements in industry based on these norms.
Analysis of TG's approach to system quality

In selecting these project result characteristics the idea of performance prioritization at two level was used [see left page]. The selected characteristics were the ones that could be used in the information planning phase. Some of them were derived from strategic management literature (e.g. Earl [1989] and Mintzberg [1979] and [1983]). If only this kind of characteristics was needed to successfully predict the importance of a certain quality factor in a specific information system the two levels of prioritization mentioned in the could be merged.

5.3 Survey preparation and analysis

Preparation

Two forms were developed. Respondents were supposed to give information about one of the previous system development projects they had been involved in by filling out both forms.

The first form incorporated 28 project result characteristics. Every characteristics was presented as a variable with a limited number of possible values. All values ("options") were given and the respondent had to choose one of them to characterize his project with regard to that characteristic. Although this approach imposed a level of rigidity on the respondents it was preferred to open end questions, because it limited the time needed from them. The list of 28 characteristics was created via a brainstorm session with the TG mentor, study of literature and two individual evaluations of an intermediate list by two TG-consultants.

The second form incorporated 30 quality factors. Per quality factor the respondent had to rate its importance in that project on a 5-point scale, where 1 meant a very low and 5 a very high importance.

TG consultants would act as respondents in the survey. The TG-mentor provided a list with 9 potential respondents in the survey. They were all approached with a short explanation about the survey and a request to participate. One person made clear his experience would not be useful for the survey, while the appointment with another to actually fill out the forms was repeatedly postponed till the deadline for analysis was reached. One of the respondents suggested to contact two other consultants. Both participated. Three other people from the Information Management group who were not on the original list offered to participate. Thus, the total number of respondents became 1219.

---

19Of these 12, 5 participants were consultants in the Information Technology Management group, 5 in the Information Technology Project Management group, 1 in the Profit group and 1 in the Human Talent group. The last two groups are both part of the Management Consultancy sector.
<table>
<thead>
<tr>
<th>Score</th>
<th>Relative frequency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 5.2 Relative frequency of the scores

<table>
<thead>
<tr>
<th>Difference (in points)</th>
<th>project 1</th>
<th>project 2</th>
<th>project 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 5.3 Frequency of differences in the scores for projects which were analyzed twice

- large system
- horizontal system
- TPS
- long life time
- central data collection
- data intensive
- proven technology

Figure 5.4 The options with very high frequency in the sample
Analysis of TG's approach to system quality

Data analysis

The design of the forms was such that they could be handed out and picked up or brought back later. But 17 times the filling out of the questionnaire was attended. The rest was evaluated with the respondent after the forms were returned. On average filling out one set of forms took about 20 minutes. The 12 participants provided 21 analyses. Three times information about a particular project was provided by two consultants, who had both worked on that project. Three weeks passed between the first and the last observation.

5.4 Results

The relative frequency of the scores for the total sample is given in figure 5.2. Note the high relative frequency of the scores important (4) or very important (5) for quality factors. The frequency of the differences in the scores, in figure 5.3, is a measure of the level of agreement about the importance of the performance factors in the three projects that were analyzed by two consultants.

Almost 80 percent of the analyzed projects have characteristics as shown in figure 5.4. Unintentionally the sample has been limited to a specific class of systems. The frequency of options in the sample varies strongly. This makes comparison more difficult and limits generalization of the conclusions.

More results can be found in appendix III.

5.5 Conclusions

The most important conclusion that can be drawn from the survey results is that TG consultants do not regard each performance factor as equally important in a system development project.

The similarities between two TG consultants working on the same project are bigger than those between two randomly picked project members. The TG consultants share the same background, they often look at the project from very much the same perspective and they communicate more intensively during the project. Besides that, the statistics in table 3 are

\[ H_0: \text{median} < 3 \]
\[ H_1: \text{median} > 3, \text{ with alpha}=0.05 \]

20Per characteristics a limited number of responses were possible. Each response is called an option. Relationships between options and performance factors were tested on strong significance in the following way:
Examples

Correctness of input

In February, 1991 the Register of Population in Amsterdam switched from handwritten and typed cards with information about individuals to an automated system: the Municipal Population System. After the recent disaster with the ELAL-jumbo in Amsterdam when a crisis team used the system to draw up a list of people who resided in the hit apartments, it became clear a lot of data was incorrect.

The mayor of Amsterdam claimed at least 10 percent of the data was incorrect (why more than 10 percent!). The director responsible for the Register of Population said he could guarantee (guarantee!) the percentage of incorrect data was less than 10 percent. But he presented a plan to improve the state of the MAPS. Without correct data the output from an information system will be more of a burden than a help in the decision process.

The "Closed loop" principle is generally advocated as a way to improve correctness of data. According to this principle the ones who are affected by incorrect output should be the ones responsible for the input.

Flexibility

One of the projects dealt with an automated cashing system for a supermarket chain. It made it possible for customers to use their bankpass for electronic payment instead of paying with cash. Since the use of this kind of system is extremely structured requirements regarding flexibility can be low.

Reliability and availability

Another project involved a system for florists which allowed them to contact one of their colleagues somewhere else in the country electronically to order a delivery. The experience with IT among florists was generally low. Till the introduction of the system the florists used the telephone for ordering. The system had to compete with this medium and really prove its additional value.

In the case of a competing medium and users who wait to see which way the cat jumps reliability and availability are important. Disturbances in the use of a system are devastating for its success in this case.
to set priorities regarding performance factors it is important to communicate those priorities throughout the development team. Table 3 indicates that it is possible to get the noses of the project members in the same direction with regard to the importance of the individual software performance factors.

With the current backlog in software production, there is no way software which needs changes can be economically replaced. Thus maintenance will stay a large investment for a while. Currently about 75 percent of a system's total costs during its life cycle is related to maintenance. Without significant improvement in management of software changes maintenance will be restricted to only solving the most urgent problems, thus keeping the quality of most software far below its theoretical maximum. Some organizations, e.g. Hewlett Packard, significantly improved their maintenance management by developing methods and metrics (Grady 1987) but more often than not performance factors related to maintenance are highly neglected. The survey results confirm that. Flexibility, maintainability, portability, connectivity, reusability and expandability were not regarded as important under any circumstances. The scores for flexibility were especially low. Part of all this can be related to the kind of systems that were analyzed. See left page for examples.

The position of TG consultants in a project is an additional factor to explain the lack of attention for maintenance related factors. TG employees are often hired to fulfil positions in the management of the development project, not to do the development themselves. This is often done by a software house. TG consultants are only involved in the project until right after the acceptance test and they are not the first responsible for the end product. Besides that the client often does not pay attention to maintenance factors. But here again TG could improve the quality of their services by doing so, e.g. by requiring more guarantees from the software developers in this respect.

All this makes clear that the interest the decision maker has in the development process has an important influence on his rating of the importance of the performance factors. Hoondert investigated prioritization among four groups of stakeholders concerning one particular system\textsuperscript{21}. He found significant differences in priorities among the groups. Giezeman (1992) found differences between Japanese and Dutch information system executives.

The universal lack of a high prioritization for security also reflects a common practice in the industry that cannot be explained by economic statistics (Goodhue & Straub 1991). EDP-auditors especially warn that industry is losing billions on computer fraud.

\textsuperscript{21}The four groups were: end users, functional users, project managers, functional designers
Correctness of input is always important. Completeness of input and processing, verifiability, timeliness and accuracy of input are almost equally important. The 'garbage in, garbage out' principle seems to be an important drive in quality assurance.

Reliability is not always crucial, which implies that the common preoccupation with this performance factor is not just "only part of the story" but sometimes even a waste of time and money. Note that reliability, robustness and recovery facilities often pop up as a trio. The importance of availability is less strongly but still significantly correlated with the former three factors.

Although the survey results provided sufficient information about objectives 1 and 2 their contribution to objective 3, definition of 'situation' was only limited. Differences in priorities could not be explained with characteristics like concern level, the number of users\textsuperscript{22} and the organization strategy.

Other characteristics, more technical in nature, e.g. the difference between on-line and batch systems and between system based on an innovative technique and those using proven technology did provide some insight in the importance of performance factors.

During discussions in the POIN056-group about this aspect of the survey results the following question was brought up: to what extent are characteristics like "on line system" or "innovative system" really characteristics of a situation? Are they not the effect of underlying factors related to the application?

The group members agreed that this was the fact. They concluded that these underlying factors were closely related to the application and not the environment. They decided that therefore the quest for the dimensions of a 'situation' should be directed towards a more operational level. With this knowledge the intern departed for the United States.

\textsuperscript{22}In the survey 'user' was defined as an individual who directly interacted with the system. From the reactions of the users it could be concluded that not so much the number of users as the number of people affected by the system was a decisive factor for performance factor prioritization. In a bank only a few people could be involved in data entry, but systematic errors can affect thousands and be devastating for the bank's reputation.
"Being a husband is a whole-time job. That is why so many husbands fail. They cannot give their entire attention to it."

Arnold Bennet

Bennet only sees one characteristic of a job: the number of hours one needs for it. In the discipline of organizational behaviour several other characteristics of the task have been developed. These task technology characteristics were encountered in the more detailed analysis of the application. Because it is possible to link the importance of performance factors to these task technology characteristics four of them were eventually selected to form the basis for the typology. In this chapter the development process and the outcome will be discussed.
6.1 Construction of the typology: first versions and evaluation

The first weeks at the Carlson School of Management were used for literature research again. More evidence was found that not much research had been done in this area. But what was available, e.g. the results of the research by Specht (1981) on the relationship between task technology and information system requirements, suggested that a typology based on task technology characteristics could be a useful one. Because of the available expertise in the department on the system-task fit it was possible to strengthen the initial concept via talks with various faculty members.

The set of job technology characteristics used by Specht included task variety, task knowledge and task significance. In a first discussion about these proposed variables with the U of M mentor minor changes were made to these concepts and user discipline was added to the list.

During the second week of June these characteristics were related to quality factors. In the third week this prototype was checked by the U of M mentor and a visiting associate professor from Eindhoven University of Technology. Two more dimensions were added, namely integration with other systems and expected life time. These two dimensions differed from the other four because they are related to the system instead of the workplace.

In the last week of June the director of the MIS department at the headquarters of General Mills, Inc. was interviewed about the instrument. At that moment all dimensions were introduced in the instrument as binary variables: for a particular application the score could only be high or low. Except for the question about output criticality the GM manager thought the level of ambiguity associated with the low/high approach in the responses was acceptable. But in the next week a U of M faculty member raised more fundamental objections against this approach.

A modeller always has to trade off comprehensibility and usefulness. At that moment the binary approach in the typology allowed its user to draw conclusions about performance factors in a vary straightforward way. Analysis would lead to one set of important factors out

---

23 General Mills, Inc. markets consumer goods and services in two principal business areas: consumer foods and restaurants. Total sales in 1991 was $ 7.15 billion.

24 The GM systems management uses two concept to evaluate its systems: user satisfaction and quality. User satisfaction was measured with questionnaire eliciting the users rating of the system on a 4-point scale (from no problems to almost continuously problems) and his advice about the future of the system (from 'system is healthy' to 'should be removed'). Quality was almost synonymous with reliability. Although the MIS director stated that no contingency approach was used the management had tried to establish different levels of output criticality to find out how far they should go in assuring reliability on different occasions.
Software Tools
The complexity of a system is initially determined when the system is developed. Appropriate software tools and development methodologies can aid programmers to write maintainable software.

Volutility
Some systems will undergo more frequent maintenance because they are subject to frequent regulatory changes, used in a highly dynamic and competitive industry, or must interface with a rapidly evolving system.

Functionality
One may think of the size of the entire application system as a complexity metric, but it is one which is not under control of the programmers, being largely determined by user requirements. Since more ambitious systems may require more ambitious code, the size of the application system may make it difficult for programmers to maintain good levels of software complexity.

Age
Age is not a factor which is directly controllable by management, except in the extreme sense that management may choose to replace an old system.

Operating Requirements
Another source of system complexity is the imposition of operating requirements such as rapid response time or cost efficiency. While it is not necessarily the case that technical constraints force programmers to write complex code, they do tend to cause programmers to pay less attention to such niceties.

Error rates
Just software complexity to affect error rates, error rates may, in their turn feed back upon the software complexity of an application system.

Figure 6.1 Factors affecting system complexity
The most significant one was the assumption that the importance of a performance factor is related to a dimension via a step function. In the case of a 'medium' score for a dimension the typology forces to make conclusions which are far too drastic. A more linear approach is better. Of course, to make the two any different it is necessary to create more elaborate scales for both the dimensions and the importance of the factors.

Secondly more dimensions than the six mentioned in the instrument influence the prioritization of performance factors. It is acceptable that the other factors are not explicitly mentioned in the analysis phase, but they deserve some room in the prioritization phase. This also calls for less rigidity.

In the second and third week of July the typology was significantly restructured using the criticism from the various sources. In the following weeks the development of a user interface for the instrument started (design of forms etc.). This way it would be easier to communicate the content of the instrument after return to the Netherlands. In appendix III the intermediate results are discussed further.

In the Netherlands, the instrument and the underlying assumptions were discussed in several meetings with people from Eindhoven University of Technology and TG. As a result of these discussions a new version of the instrument was developed.

6.2 Discussion of the final version

The latest version of the instrument incorporates four dimensions. They are all related to the task and will be discussed subsequently.

Volatile of the environment

Volatile of the environment replaces task variability, which was used in the previous versions. In literature volatility is regarded as a stronger determinant of the importance of performance factors than variability (Banker [1992], van Vliet [1988]).

maintenance

Banker lists six factor affecting the maintenance of a system. See figure 6.1. Only volatility of the environment cannot be influenced by the organizational members. Systems which are subject to frequent regulatory changes, are used in a highly dynamic or competitive industry or must interface with a rapidly evolving system will undergo more frequent maintenance. Other things being equal the maintenance effort related to a system will increase during its existence. Every maintenance occurrence is an opportunity for errors to sneak in. Besides that
De Leeuw’s typology of control situations

De Leeuw distinguishes six types of control measures based on two criteria:
direction of control: either internal or external
type of control: either routine, adaptive or strategic

De Leeuw calls the set of six types the control characterization. In every situation
one has to decide about the control mix, i.e. a subset of the control characterization.
The more emphasis control situations require on external (as opposed to internal) and
strategic (as opposed to routine) measures in the control mix the more complex they
tend to be. Their complexity is determined by the following characteristics.

The sensitivity and stability of the controlled system
In the short time frame of routine decisions the value of many variables does not
change or cannot be changed. Strategic decisions often cover years.

The number of variables and their interdependencies
De Leeuw distinguishes
goal variables
control variables
irregular variables, which are outside the decision maker’s scope of control

The controllability and predictability of the variables
The predictability of moments of problems

It is important to distinguish the controlled system from the model of the controlled
system. A decision maker bases his decisions on a model. This is a simplification of
the controlled system itself. The characteristics are determined by the effect of
discrepancies between reality and model due to these simplifications.
Typology development

the entropy of the system will increase due to maintenance efforts, especially if the
maintenance is done in a hurried way. With the next maintenance job the system will be more
complex, thus making maintenance more difficult (van Vliet [1988], Grady [1987]).

portability

At a certain moment it will be easier to build a new system than to continue maintenance.
This implies environmental volatility affects the expected life time of the system as well.
The probability of changes in a system's hardware and system software environment increases
with its existence. Thus portability is negatively correlated with volatility. With the rapidly
decreasing prices of hardware and its rapidly increasing performance a system's hardware
component will be replaced more and more frequently.

security

A major part of the costs related to integrity problems of a system is incurred because the
owners of the system are forced to replace it by another one earlier than they had planned. The
costs of this early replacement are correlated to the difference in years between the planned and
actual date of replacement. The importance of security is therefore affected by volatility as well.

Task analyzability

When a process is analyzable the work can often be reduced to mechanical steps and
participants can follow an objective, computational procedure to solve problems. When work
is unanalyzable participants have to spend time thinking about how to solve problems and they
may actively search beyond readily available procedures. Judgements based on intuition and
experience figure prominently in unanalyzable work decisions. Several scales to measure task
analyzability have been developed (e.g. van de Ven, Withey [1983]).

These scales mainly measure the effect of the task analyzability, i.e. the existence of
procedures. Ghani [1992] used Withey's scale to relate task analyzability to the use of
information systems and found that a decrease of analyzability was correlated with an increased
need for flexibility. But additional analysis of the causes of task analyzability increases the
insight in the relationship with performance factors. See "De Leeuw's definition of control
situations" for an example of a description of the causes of the level of task analyzability and
'Bemelmans' typology' and figure 6.2 for the connections with performance factors.
Bemelmans' typology

Bemelmans relates De Leeuw's definition of control situations to the importance of performance factors. This means Bemelmans, in contrast with Evans, uses only one concept as a basis for the performance prioritization, in this case "control situation". This will make it easier to explain to others how one arrived at the decisions about prioritization and most probably will improve the quality of the discussion.

The quality of the discussion will also improve because gliding scales are used instead of extremes as in the rigid, binary "important/not important" approach that Evans' typology seems to suggest.

Only a few performance factors are incorporated in the typology by Bemelmans. And some of the attributes in the typology are not even real factors but only related to them, e.g. complexity of the system. Other characteristics are needed to cover a broader range of performance factors.

Here again the connections between dimensions and performance factors seem to be based on merely intuition. The rather theoretical concepts may complicate communication. Many individuals will have trouble rating their situation based on number of control variables or predictability of the environment. The strong interdependency between the dimensions only intensifies this problem. It is necessary good operationalizations of the dimensions are available to compare different control situations.

<table>
<thead>
<tr>
<th>characteristics</th>
<th>important factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensitivity and stability of the controlled system</td>
<td>correctness</td>
</tr>
<tr>
<td>low</td>
<td>accuracy</td>
</tr>
<tr>
<td></td>
<td>response time</td>
</tr>
<tr>
<td>number of variables in the decision model and their</td>
<td>complexity</td>
</tr>
<tr>
<td>interactions</td>
<td>flexibility</td>
</tr>
<tr>
<td>high</td>
<td>response time</td>
</tr>
<tr>
<td></td>
<td>data storage time</td>
</tr>
<tr>
<td>controllability and predictability</td>
<td></td>
</tr>
<tr>
<td>low</td>
<td></td>
</tr>
<tr>
<td>predictability of disruptions in the controlled</td>
<td>user friendliness</td>
</tr>
<tr>
<td>system</td>
<td></td>
</tr>
<tr>
<td>low</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.2 Bemelmans’ typology
Typology development

The importance of three performance factors depends on the level of task analyzability:

**flexibility**

Bemelmans (1991) relates an increased number of variables in the decision model to increased flexibility of the system. This is supported by Ghani's findings.

**response time**

In the case of a short time scope, the system has to provide the decision maker with the necessary information fast.

**efficiency**

When the number of variables increases the amount of necessary data also increases. When the time scope increases data have to be stored for a longer time. To limit the costs of increased storage capacity more efficient use of memory and processing resources is needed.

**Task significance**

Task significance is defined as the degree to which the job affects the lives or work of other people whether in the immediate organization or in the external environment. When a system supports a task with a high level of significance the risks related to a system failure tend to be higher. Here risk is defined as:

\[ \text{risk} = \text{probability} \times \text{effect} \]

The effect of a problem is to a large extent beyond the control of the system developers. Except for recovery and graceful degradation increased importance of the following performance measures is therefore aimed at limiting the probability of occurrence of a problem.

The factors are related to three major reasons for failure in use and processing of information.

1. A user gets incorrect output from the system

Incorrect output is the result of either incorrect input or incorrect processing.

**correctness of input**

Although the problem of data entry errors has diminished in the last two decades, e.g. because of the possibility of real-time validation, it is still the most common cause of computer errors.

**correctness of processing**

Computer processing errors are becoming less frequent as improved software design techniques proliferate.

**security**

In cases where correctness of output is really critical it is not sufficient to only pay attention to preventing mistakes. Users with ill intentions have to be stopped on time as well. Baskerville (1988) mentions several studies about fraud cases. His conclusion is that most often
fraud is an inside job, mainly related to data manipulation (as opposed to program manipulation). Especially preventing fraud by insiders requires measures which go beyond the standard correctness checks, because insiders often know how to affect the integrity of a system with input which still satisfies the correctness constraints.

2. A system down
Important factors related to this problem can be split in two groups:

The ones related to preventing the problem:

reliability

Of all performance factors reliability has got the most attention from researchers. The theorists focused on what kind of, often highly complicated, models to use to predict reliability. Practitioners had to deal with other problems. Time and budget constraints forced them to accept that reliability could only be guaranteed to a certain level, sometimes significantly lower than 100%. Only in the last couple of years theorists have shifted the discussion a little more from how to why to get a certain level of reliability. The risks involved in a system down is the most important factor in reliability allocation ([Musa (1987), Evans (1988)]. Zahedi and Ashrafi (1991) developed a mathematical programming model to calculate the desired level of reliability once the criticality of the application and some budget constraints are given. Establishing a realistic reliability level is important, because the levels of all the other factors mentioned under 2 are highly correlated with it.

testability

Of all techniques used to achieve highly reliable systems intensive testing is the most widely accepted. The importance of testability has traditionally been strongly related to the importance of reliability [Musa (1987), Banker (1992)].

security

Baskerville distinguishes process risks and data risks regarding information systems. The data risks were discussed under 1). It is possible to affect the data but not the process. But damage to processing elements ultimately affects the data too. Security of processing elements can be considered as preventing that the reliability of the system is not affected by unauthorized individuals.

The ones related to limiting the effects of a system down:

recovery facilities

graceful degradation
<table>
<thead>
<tr>
<th>Dimension</th>
<th>performance factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volatility of the environment</td>
<td>maintenance, portability, security</td>
</tr>
<tr>
<td>Task analyzability</td>
<td>flexibility, response time, efficiency</td>
</tr>
<tr>
<td>Task significance</td>
<td>correctness of input, correctness of processing, security</td>
</tr>
<tr>
<td></td>
<td>reliability, testability, security, recovery facilities, graceful degradation</td>
</tr>
<tr>
<td></td>
<td>userfriendliness</td>
</tr>
<tr>
<td>Task interdependence</td>
<td>connectivity, security</td>
</tr>
</tbody>
</table>

Figure 6.3 The core of the typology
As in the case of testability the importance of these two performance factors is highly correlated to the importance of reliability [Toigo 1989].

3. a user does not notice output

What is meant here is not only the errors due to perceptual limitations of the user, e.g. an operator who does not see a flashing light on a control panel. In the case of technical information systems this is an important issue. But in the case of management information systems a broader interpretation should be used. Although screen design is important here, the fact that a user may not see an item on a screen should get less attention than if he would able to easily get the desired screen.

The meaning of 'easily' differs with the user. A unexperienced user requires menu oriented user interfaces. But an experienced user prefers a command language, because they are more efficient than menus. A system serving users with different experience levels ideally incorporates both query methods [Davis 1985]. When this is not possible the developers are faced with a dilemma. Schiffman [1992] found that unexperienced users have attitudes regarding IT which are significantly more negative than those of experienced users. Davis hypothesized that experience was highly correlated with frequency of use. This was confirmed by Schiffman. This implies an individual's initial confrontation with IT is crucial. Once he is behind the screen and likes it, less and less effort is needed to keep him there. From this point of view unexperienced users should be taken seriously.

On the other hand, when the system is too much adjusted to the unexperienced users the experienced ones may become unsatisfied with the system's performance and stop using it. Of course, user development is another way to solve thus dilemma. This kind of techniques to create an organizational fit will be discussed in the next chapter.

Task interdependence

Task interdependence is the degree to which the activities of separate individuals force them to depend on each other, thereby requiring more coordination to reach organizational goals.

connectivity

Connectivity provisions are a conditio sine qua non for system interconnection [Evans 1988].

security

Woods [1990] found the ratio of the number of system security personnel to total system personnel increased with the level of interconnectivity of an organization's systems.

A system with a lot of connections to other systems is more vulnerable to access by unauthorized individuals [Stoll 1988].
"...if they are to be operated effectively, today's systems almost always require major, sometimes radical alterations in an organization's structure, personnel, roles and business processes, sometimes even in the culture of the corporation itself. Thus the economic, behavioral and political consequences of today's information technology applications should be well thought out and the requisite change processes effectively managed."

Rockart, 1988

In the last two decades the definition of software quality has shifted from reliability, via performance and performance plus functionality to system success. The last two classes of definitions are still quite uncommon but have gained acceptance in the last five years.

System development is more and more regarded as part of a bigger change project. This also reflects on the approach to quality of software. That will be discussed in this chapter.
Of course, user development is another way to solve this dilemma. This kind of techniques to create an organizational fit will be discussed in the next chapter.

**Task interdependence**

Task interdependence is the degree to which the activities of separate individuals force them to depend on each other, thereby requiring more coordination to reach organizational goals.

**Connectivity**

Connectivity provisions are a *conditio sine qua non* for system interconnection. [Evans 1988].

**Security**

Woods (1990) found the ratio of the number of system security personnel to total system personnel increased with the level of interconnectivity of an organization's systems. A system with a lot of connections to other systems is more vulnerable to access by unauthorized individuals [Stoll 1988].
7.1 Integral workplace design

In chapter 4 it has already been made clear that even a system with a high performance could fail when other aspects, e.g. acceptance, are neglected. Therefore system quality should not be an end in itself. How can a system be a quality system when it is not a success? Likewise one should be careful interpreting system success in too narrow a way.

Successful system development is more than the fulfilment of the functional and performance requirements within time and budget. Still in most projects the technological aspect is dominant. The social and organizational consequences of a new system are hardly taken into account. This often means too much adaptation is required from the user. Not only in his use of the system, but even in the way he does his work. Many systems fail for this reason.

In 1991 TG developed the "integral workplace design" method to give their consultants systematic support in their integration of technological and socio-organizational requirements of a system developments project. See "integral workplace design" for a description of the philosophy behind this method.

7.2 "System follows task and user" versus "Integral workplace design"

Initially the instrument was designed with a "system follows task and user" approach in mind, as a reaction to common practice, which is often "task and user follow system".

The TG consultants protested that the assumption that the system will be completely adjusted to the current situation will be incorrect most of the time for two reasons:

1) customization is almost never completely possible and sometimes even not at all. For some applications, e.g. specific control systems for technical production processes, only a few suppliers are available with a limited number of standard systems. In such cases the system acts as a constraint in the workplace design process.

2) even if it would be possible to completely adjust the system to the current task and user characteristics it is not wise to do so immediately. Significant increases in productivity of employees are seldomly found in processes which hardly changed after IT implementation, but in the ones where system development was accompanied by a change process of a larger scale, involving both the users and the way they do their work. In such cases full use could be made of the potential of IT. An analyst should not accept a process and its related tasks as they are, but critically check why people do things the way they do them and find out where and how productivity gains are possible, before designing the IT support.

This leads to integral workplace design.
Tom Gilb's evolutionary development

Some theorists are sceptical about a client's ability to fully grasp the consequences of his decisions by only looking at a design. And with the current development strategies, where the complete system is developed in one big project, the client will not be confronted with the product until it is too late to make any changes.

Even if the client would fully understand what he will get in the end, the fact that most large projects take years to completion will cause problems in the requirements definition. It is likely the client changes his mind about the requirement half way the development process because the environment of the application has changed.

Tom Gilb advocates the evolutionary development strategy to deal with these and a couple of other problems. Evolutionary developers deliver the system in many small steps instead of one big project. After the first step the user can get "hands on" experience with the system and provide the developers with feedback about his experiences. This feedback can be used in the development of the other parts.

At the beginning of the very first step the infrastructure of the complete system is designed to make sure everything will fit in the end. Every time only the result of the next step is planned in detail. This allows for a high degree of flexibility in the development. Gilb advises to deliver the "juicy bits" of the system first to get people enthusiastic about IT soon and guarantee commitment from management. Before every step developers have to ask themselves:

"How little development resources can we expend, and still accomplish something useful in the direction of our ultimate objectives?". This is sharp contrast with the traditional question:

"How much can we accomplish within some critical constraint (budget, deadline, storage space)?

Although Tom Gilb claims evolutionary development is as natural as driving a car, it is certainly not as common.

TG consultants do not use the evolutionary strategy very often. According to one of the consultants from the Information Technology Project Management group the biggest problem with evolutionary design is the splitting up of the complete system in smaller products. Every user group will push to have "their" part developed in the first cycle, with the result that the first part will be a significant part of the complete system.

In spite of these, mainly political problems, evolutionary design seems to be worth more attention than it currently gets.
7.3 Process description of performance assurance in the "integral workplace design" method

Integral design starts with the definition of the organizational mission and goals. This is followed by the planning phase.

**planning phase**

In the planning phase the organizational plans are defined. These plans deal with different aspects of the organization, e.g. structure, finance, IT, human resources, and are an elaboration of the mission. It is important these plans are well atuned, because discrepancies will have consequences in the projects later on. The plans are combined in the integral organization plan.

With regard to the performance assurance the planning phase is important because here the decisions are made about priorities regarding the different development proposals. The decisions deal with both the allocated amount of money to a project and its planned date of delivery. The decisions from the planning phase act as constraints for the decisions later on.

**definition phase**

The definition phase is always related to one project only. During the definition phase the requirements of the project result have to be defined as clearly as possible.

In the definition phase the current situation with regard to the three workplace components is analyzed. Within the analysis scope the organization is analyzed in more detail than during the planning phase. In this phase the instrument can be used to find gaps in the fit between task, users and performance of the system.

In case an automated information system does not yet exist the instrument will give an indication about the performance requirements if task and user are not changed. Although this assumption will most probably not hold during the rest of the project, the analysis of the current situation provides important guidance for the designers. After the analysis of the current situation proposals for change will be made. In these proposals the performance aspects deserve more attention than they currently get. At the end of the definition phase a choice is made about the proposals and definition of the project result can start.

**design phase**

In the design phase alternative solutions are generated and the most suitable solution is selected. In this phase the relevant workplace clusters and workplaces are analyzed. Especially in discussions about this elementary level of the organization the instrument can be
1. Keep the performance requirements as low as possible.
Performance assurance represents a significant part of the total development costs. Any unnecessary target means much money is wasted.

2. If it is possible to replace a design, where the importance of the performance factors A and B is very high and very low respectively, for a design where ceteris paribus the importance of factor A and factor B is high and low respectively, than do it. Because the costs related to a factor increase exponentially with the performance level, much more is won by topping of the importance of a important factor than is lost by the increased importance of an unimportant factor.

3. Try to limit the number of clusters that incorporate important factors.
The Quint-researchers clustered the performance factors based on the underlying techniques. See figure 7.2 for the clusters. Two factors from the same cluster often reinforce each other, measures taken to enhance one factor also have a positive effect on the other factor. When the set of important factors includes factors from different clusters a technique conflict is very likely.

Figure 7.1 Three rules of thumb for design comparison.
Beyond information systems

a valuable tool for communication. Because of the strong interrelationships between system
performance and the task and user component frequent tuning sessions are necessary. This will
not always happen in a formal meeting but often in ad-hoc events, sometimes involving only
representatives of two components. By using the instrument the performance assurer could
already provide the other designers with design principles regarding performance. A good at­
mosphere between the different component specialists is necessary for successful tuning
sessions. At the same time performance assurers should not be afraid to confront other
designers with the consequences of their plans for the performance of the system. Hopefully
the other specialists have already used the instrument for their design. Otherwise the instru­
ment can be used to strengthen the position of the performance specialist in the discussion
with others. It is important the discussion has enough detail. It is easy to reach agreement
about rather abstract, general principles. But for successful completion of the project it is
necessary differences in opinion about the exact characteristics of the workplace are made
explicit. It is better to accept differences in opinion than not to discuss crucial issues.

Initially several designs will be made to provide the decision makers with alternatives
for development. During the design phase alternatives will be eliminated and the remaining
ones will become more detailed. The more detailed they become the more partial they will be.
In this phase more effort is needed from the project management to make sure the project keeps
its integral character. Because of the strong relationship between the system performance
factors and the characteristics of the other two workplace components this certainly applies to
the process of performance requirements formulation too.

The client should be the one who takes the decisions about the design alternatives, not
the designer. Because many clients have only limited knowledge about IT, making sure they
will be able to make sound judgements about the alternatives requires effort from the designers.
They should take the time to explain to the client the consequences of every alternative.
Of course, the users have to take into account many aspects about the designs. The
performance of the system is only one of them.
Only the process of making sure the user understands the implications regarding the
performance requirements of the system will be discussed here.

Especially performance requirements are typically left to the designers. Users will
already have problems analyzing the current situation in their workplace and making
statements about important performance factors. It will be even more difficult for them to see
if a workplace which only exists on paper will be an improvement to the current situation
which alternative will be the most significant improvement and if a shift in performance
priorities will occur. See 'Evolutionary design' for a remedy against these problems.
Purposiveness:
The effectivity of the software product.
completeness
reusability
connectivity

Certainty:
The logically correct working of the software product.
security
correctness
verifiability

Efficiency:
resource behaviour
response time
time behaviour

Ergonomics:
operability
self instructiveness
understandability
user friendliness
transparancy
learnability

Changeability:
The effort that is needed to understand the software product and adjust it.
exploitation effort
repairability
start up effort
portability
testability
changeability

Continuity:
The technically correct working of the software product.
reliability
availability
robustness
graceful degradation
recovery provisions

Figure 7.2 The performance clusters, according to the Quint researchers
It is important that designers explain to users how workplace characteristics and the performance requirements of systems are connected, using the theoretical framework in the instrument. For every alternative they should explain how much it differs from the current situation, how close it gets to the preferred situation, which workplace component is emphasized to create a 'performance fit', and what costs are associated with every design.

It will be difficult to provide the client with exact figures about the costs of the different alternatives and their associated performance profiles. But designers could keep the three rules of thumb discussed in figure 7.1 in mind in their discussions with the client.

These rules are based on the desire to limit the development complexity. An increase of development complexity will lead to higher development costs.

When used wisely they can facilitate the trade off problem in spite of the lack of exact costs forecasts.

Representatives from different groups of stakeholders should be involved in this process since the interests of each them significantly bias their decision making. By discussing the alternatives using the instrument it is more probable that those interests will be made explicit and constructive discussion will occur.

In the last stages of the design phase, discussions about performance profiles will be mixed with the ones about exact requirements. Then the metrics will become increasingly important.

At the end of the design phase the stakeholders should have a list of performance requirements, as exact as possible. Once the rating for the performance factors are established they can be related to development techniques.

The development itself should result in a 1-to-1 translation of the design.

Because performance priorities were set in an early stage it is more likely this last assumption will become fact.
CONCLUSIONS
"Before the electrodes were attached to his head he'd lost everything tangible: money, property, children; even his right as a citizen had been taken away from him by order of the court. All he had left was his one crazy lone dream of Quality, a map of a route across the mountain, for which he had sacrificed everything. Then, after the electrodes were attached, he lost that."

"Zen and the art of motor cycle maintenance"
Robert M. Pirsig

Pirsig's book acquired about the same status in quality management as Carrol's "Alice in Wonderland" in strategic management. They both are often used as a source for quotations. Still the text mentioned above is not found in literature about quality. Most probably because quality theorists rather not confront their readers with the probability that they lost their mind in the process of finding a route across the mountain.

This last chapter is devoted to the results of this scouting expedition.
8.1 Conclusions

This thesis focused on how to define 'situations' in system development to be able to relate performance profiles to them.

Eventually the definition of 'situation' was strongly related to task technology characteristics and a start for a typology relating situations and performance profiles has been presented in the previous chapter.

The major contribution of this instrument will not be related to the uniqueness of its procedure. It is embedded in a method which is very similar to what for years already people have been saying sound analysis should be like.

Many people already know how things should work, and almost nobody does it that way. Many do not do make any trade offs at all, because that is so tempting, until late in the development phase when it is really too late.

The final instrument is not a typology in the strict sense any more. It thus can be used by both individuals who believe typology development is possible and those who think a priori decisions about the importance of performance factors are an illusion.

The main issue regarding this instrument however, is not to what extent it can be considered a typology and to what extent uncertainty about the importance of performance factors can be taken away by simply scaling a workplace on the four dimensions.

This instrument makes it possible to approach performance prioritization at the beginning of the life cycle. Systematically. And openly. That is its major contribution.

Because in quality assurance communication is the name of the game.

Although during the nine months of this project quite a number of theorists and practitioners have contributed to the final product, it is still mainly based on evidence from literature. Much further corroboration in practice is needed to check its workability.

Issues directly related to the typology have already been discussed in the previous chapters. The recommendations will therefore mainly be related to 'side effects' of the research.

8.2 Recommendations

*Increase the awareness about the importance of performance.*

Performance is not something that can be left to the software engineers once the functional requirements are clear. It is a common cause for system failure and can be an important source for competitive system advantage.
Conclusions

*Make sure all stakeholders in a project agree on a specific quality definition.*
The fact that all stakeholders use the same definition of quality is much more important than what definition exactly they use. Invest time in finding out what their ideas are and in making clear your own. This prevents that misunderstandings sneak in with serious effects later on in the project.

*Prevent occurrence of the "seeing as" effect.*
Customers do not want systems, they want solutions to their problems.

*Make sure the performance profile is communicated throughout the development team.*
It is no use telling developers to be quality conscious if they do not know what it means. Everyone should focus his attention on the same performance factors. The survey provided evidence that it is possible to get the noses in the same direction. But the project management has to communicate the priorities.

*Try to select developers for a project who have worked with systems with similar performance profiles before and who have a good knowledge about techniques related to performance factors which are important in that project.*
It is highly improbable one person can have a state of the art knowledge of techniques related to, e.g. reliability, maintainability and userfriendliness. Especially when particular performance factors are really important at least some developers should be experts in these areas. Of course, awareness for "seeing as" is extra important in that case.

*Be honest to users*
Explain to users what they can expect. Do not promise them heaven on earth because of the new system. Make clear prioritization is a logical consequence of the limited power of IT developers. Be sure users understand prioritization is not just necessary because management does not want to spend money on ‘their’ system.
The satisfaction with a system will increase when the users get what they were promised.

*Look beyond the system*
Often the problem in a workplace goes deeper than the automated system, or lack thereof. Significant productivity gains require changes in task structure and/or personnel. One should be very critical about a priori limitation of a development project to system development, even when one lacks competence in development of the other workplace components.
Conclusions

*Take the three rules of thumb mentioned in figure 7.1 into account in comparison of design alternatives*

Early attempts to limit development complexity have a significant effect on control of the total project costs.

*Limit extensive measuring to the important performance factors*

To use valuable resources to measure factors that do not require strict control is inefficient.

*Test the instrument extensively, with different stakeholders and different applications*
REFERENCES


41


DeMarco, Tom, Controlling Software Projects, Yourdon inc., New York, 1982


Dorst, van M.M.L. and Huijbregts, J.M., "Kwaliteit en internationale invloeden" (in Dutch), Informatie, April 1992


Garvin, David A., "What Does Product Quality Really Mean?", Sloan Management Review, Fall 1984


Hoondert, B.C.M., "Kwaliteit van informatiesystemen ingeschaald" [in Dutch], *Informatie*, December 1991


*Proceedings of the ACM*, May 1989

Janssens, Sander, "Het Probleem van de Onberekenbare Begrotingen" [in Dutch], *IT Forum*, July 1990


Kishida, Kouichi et al., "Quality-Assurance Technology in Japan", *IEEE Software*, September 1987


Schiffman, Stephen J. e.a., "An Examination of End User Types*, *Information & Management*, April 1992


Strijen, van B.M. and Broeren, R.J., "Kwaliteitszorg Binnen de Softwarebranche" [in Dutch], *Informatiemanagement*, June 1990


Twijnstra Gudde, *Annual reports (years 1980-1991)*

Twijnstra Gudde, *Profile 1992*


Vos, van der M.J., "Effect van Softwarekwaliteit op de Kostenbegroting van Systeemontwikkeling" [in Dutch], *Compact*, autumn 1990


Wijnen, Gert et al., *Projectmatig Werken* [in Dutch], Het Spectrum, Zeist, 1987
Willmer, Heidemarie, *Systematische Software-Qualitätssicherung anhand von Qualitäts- und Produktmodellen* (in German), Springer-Verlag, Heidelberg, 1985


Wood, Charles Cresson, "How many security staff should you have?", *Computer & Security*, August 1990

Appendix I: TG’s method for project management
What is project management

Project management is a complex set of management techniques, aimed at controlling project activities from the very start through to the very end. Project management distinguishes a number of phases in each project; these phases can eventually be divided into sub-phases. The life cycle of a project can be divided into six phases as follows:

- the initiation phase of a project is the only phase which does not have a clearly-defined beginning i.e. a beginning that usually only retrospectively can be identified. The central theme in this phase is determining whether or not something is a project and, if it is, deciding whether or not it can or may be approached as one. In other words, by the end of the initiation phase the project result must be known at least in general terms and the way in which this result is to be achieved must be determined (whether via a project-approach or in some other way)

- the specification phase, in which the project is specified in terms of external interfaces, design constraints, functional and performance requirements, and also with regard to lead times, cash flows, organisation structures and information procedures. In this phase the answer of the question "what"? is specified

- the design phase, in which the project (possibly split into sub-projects) takes concrete form in terms of choosen solutions which are acceptable in terms of the project specification: the question "how"? is answered
- the *preparation phase*, in which everything which will be required during the actual realisation of the project is made ready and available. Designs are organisationally as well as technically prepared by means of, for example, production planning, training programmes, task instructions and so on.

- the *realisation phase*, in which the project is finally implemented carried out or built in accordance with the requirements.

- the *post-delivery phase*, during which the project is used, maintained and, if necessary, modified.

Figure 1 shows that the life cycle of a project can be divided into three or six phases.

Figure 1: A project divided into three or six phases.
Phasing a project correctly makes it not only recognisable but also controllable.

Through projects certainly differ one from another, the control of all projects has many things in common.

This is particularly true with regard to the control of:

- **time/capacity**: ensuring that the project is realised on time, in well structured stages (phase) and with resources required (human and material)

- **finance**: ensuring that the project is completed within budget, and that it offers the expected economic returns

- **quality**: ensuring that the project conforms to the requirements started, by demanding controllable criteria measures, and conducting the planned for quality control tests

- **organisation**: ensuring that all parties involved know what each person's authority and responsibility is, how decisions are to be made, and how project organisation and formal organisation structure interrelate

- **information**: ensuring that all persons involved are informed in time with regard to all valid and relevant facts and figures and ensuring that these facts figures can be verified.
Both the phasing and the control techniques must be adequately integrated in each project (see Figure 2.).

Figure 2: Integrated controle techniques
Appendix II: Project result characteristics mentioned in the survey
Project result list

Functional:
1. the system will be used throughout the whole concern division BU local

2. place in the value chain of the automated function(s)

3. organization type according to Mintzberg

- simple structure
- divisionalized structure
- machine bureaucracy
- professional bureaucracy
- adhocracy

4. number of users

- 1 user
- 2-100 users
- 101-500 users
- > 500 users

5. stability of the environment of the organization

- stable
- unstabile

6. specification of the work processes

- high
- low

7. organization strategy according to Porter

- costleadership
- differentiation
- niche

8. system type

- horizontal
- vertical

9. emphasis on

- data collection
- data processing
- data distribution

10. contribution of the consultants

- organizing
- facilitating
**Technology:**

11. phase of use according to Nolan

12. system type

<table>
<thead>
<tr>
<th>TPS</th>
<th>SDS</th>
<th>DSS</th>
<th>ES</th>
</tr>
</thead>
</table>

13. system size in FPA

| small | large |

14. expected life time

| short | long |

15. use of the application

| central | decentral |

16. data collection

| central | decentral |

17. data processing

| batch | on-line |

18. importance of data processing

| not data intensive | data intensive |

19. importance of calculations

| many calculations | not many calculations |

20. importance of communication

| not communication intensive | communication intensive |

21. number of connections

| few | many |

22. familiarity with the technology

| proven technology | innovative |
Acceptance:

23. task content

24. competence of the users

low           high

25. vision about the personnel

theory X      theory Y

26. appraisal structure

group appraisal  individual appraisal

27. leadership style

28. level of "we" feeling

weak           strong
Appendix III: Survey results
On the next page the frequency of the scores per performance factor will be shown in a matrix. These are the scores for the complete sample. On the following two pages matrices will show under what circumstances certain performance factors become important (alpha=0.05). Only connections between single options and performance factors have been tested. Interactions between options have not been analyzed. In the matrices the options are given vertically. Only the options with a frequency of at least 5 could be tested. On the horizontal axis are the performance factors. In the matrices only numbers are given. They correspond with the numbers in the figure on the next page.
<table>
<thead>
<tr>
<th>Performance Factor</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Decision Support</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>2. User Support</td>
<td>-</td>
<td>3</td>
<td>1</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>3. Coverage Rate</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>4. Input Correctness</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>5. Processing Correctness</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>6. Input Completeness</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>7. Verifiability</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>8. Validity Of The Input</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>9. Security</td>
<td>-</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>10. Timeliness</td>
<td>-</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>11. Reliability</td>
<td>-</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>12. Robustness</td>
<td>-</td>
<td>1</td>
<td>6</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>13. Recovery Facilities</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>14. Graceful Degradation</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>15. Restartability</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>16. Availability</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>17. Flexibility</td>
<td>7</td>
<td>5</td>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>18. Maintainability</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>19. Testability</td>
<td>-</td>
<td>2</td>
<td>8</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>20. Portability</td>
<td>6</td>
<td>9</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>21. Connectivity</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>22. Reusability</td>
<td>3</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>23. Expandability</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>24. Response Time</td>
<td>-</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>25. Accuracy</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>26. Efficiency</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>27. User Friendliness</td>
<td>-</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>28. Trust In The System</td>
<td>-</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>29. Usability</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>30. Fit To Manual Processes</td>
<td>-</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>concern level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>2-100 users</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>101-500 users</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>stable environment</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>instable environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>high specification level</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>low specification level</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cost leadership</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>differentiation</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>horizontal system</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>mainly processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>organizing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>facilitating</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPS</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>large system</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>long life time</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>central use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>decentral use</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>batch system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>on-line system</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>non much calculus</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>much calculus</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>not much communication</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>much communication</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>many connections</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not many connections</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>proven technology</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>innovative</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

60
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>concern level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BU</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-100 users</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101-500 users</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stable environment</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>instable environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>high specification level</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>low specification level</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cost leadership</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>differentiation</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>horizontal system</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mainly processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>organizing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>facilitating</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPS</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>large system</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>long life time</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>central use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>decentral use</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>batch system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>on-line system</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>non much calculus</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>much calculus</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not much communication</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>much communication</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>many connections</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>not many connections</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>proven technology</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>innovative</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix IV: Discussion of the first three versions of the instrument
START

take first subsystem

user experience low?

Y

add factor set USEXlow to list

N

add factor set TVARhigh to list

N

add factor set TKNOlow to list

N

add factor set OCRhigh to list

Y

next subsystem?

N

END

Figure A3.1 Process diagram of the use of version 1
1st version

The first version incorporated four dimensions:
- task variety
- task knowledge
- output criticality
- user experience

These factors were related to performance factors in a rigid way. The analysis started with an empty list of important performance factors. The user of the instrument had to rate the level of every dimension. Only high/low ratings were possible. Based on the ratings additional factors were added to list of important performance factors. The analysis loop is repeated for every new subsystem, i.e. workplace. See figure A3.1 for a schematic illustration of this process.

2nd version

Two dimensions were added to the first version, namely expected life time and integration with other systems. As in the first version all dimensions were regarded as binary variables (only high or low was possible). This of course made it possible to use a very structured approach in the performance analysis as figure A4.1 illustrates. But at the same time it forced the user to choose between two extremes, while most probably his situation was not an extreme at all. This is illustrated by figure A4.2.

The General Mills manager stated he would be able to deal with the ambiguity of this approach by using more sophisticated tools to arrive at a choice for either low or high, e.g. via techniques inspired by Kepner & Tregoe's decision analysis. But of course, no matter how sophisticated the decision process the fact that instrument implied a huge difference in performance priorities between two situations which would score 'a little less than medium' and 'a little more than medium' respectively would still exist. Thus the first two version had the same problem as Evans' typology.

3rd version

In the third version the idea to create a typology in the strict sense, i.e. a structure with a limited number of classes was left.

The rigid connection between characteristics and performance factors was replaced by relationships based on more elaborate scales.

It also allowed other variables, not incorporated in the instrument to influence the importance of performance factors as long as was explained why deviations from the direction implied by the instrument were made.

In contrast with the first two versions the third one was above all a communication tool, an instrument to support analysts in making explicit the ideas and assumptions which had an effect on the discussion about performance issues.

The 'typology' in the third version was accompanied by an analysis instrument, incorporating scales to rate each of the six dimensions.

The complete third version is presented on the next pages.
In the first two versions the relationship between the task characteristics and the performance characteristics was a step function. Here this is illustrated for task significance and the importance of reliability. Especially for medium scores the conclusion "low" or "high" causes significant differences in the importance. This is not in accordance with reality. A linear relationship, as illustrated by the dotted line seems more realistic. In the next versions this last approach was operationalized.
Analysis Sheet

Identification:
(of application or major subsystem)

Name of Analyst:

Position of Analyst:

1. User experience:
Mark the one group which best describes the experience level of the user of the subsystem.

   group 1:
   non-programmers who access data through a limited, menu driven environment with a presented set of procedures
   group 2:
   command level users who are able to perform some of the specific functions of an application independently at an elementary level
   group 3:
   end-user programmers who are capable of developing their own applications and are able to use more advanced command languages
   group 4:
   functional support people who are reasonably sophisticated programmers; they support others in their functional areas
   group 5:
   MIS professionals

   The lower the number of the marked group, the lower the experience level.

   Score: group ....

2. Task variability:
Rate the level of agreement with the following statements on a 5-point scale. (1-do not agree at all; 5-agree very much)

   For this application people do the same job in the same way most of the time.
   1  2  3  4  5

   2.
   For this application unit members basically perform repetitive activities in doing their jobs.
   1  2  3  4  5

   Score [(score 1 + score 2)/2]: ....

64
3. Task analyzability

Rate the level of agreement with the following statements on a 5-point scale.
(1=do not agree at all;5=agree very much)

1. A clear known way to exist to do the major types of work normally encountered in this application.
   1 2 3 4 5

2. A clearly defined body of knowledge exists to guide in task execution for this application.
   1 2 3 4 5

3. An understandable sequence of steps exists that can be followed in executing the task for this application.
   1 2 3 4 5

4. To a large extent one can rely on established procedures and practices to execute the task for this application.
   1 2 3 4 5

The more the agreement with the statements tends to be high, the higher is the level of task analyzability

Score [(score 1+2+3+4)/4]:

4. Output criticality:

Mark the number of the most accurate description of the effect of a system failure for this subsystem.

a system failure
1. will only have internal effects
   (e.g. delay of monthly financial reports)
2. could effect the service level, without the customer being aware of it
3. could have an effect on the customer’s perception of the service level (e.g. late delivery of goods)
4. could lead to legal actions
   (e.g. securities trading system)
5. could endanger human lives
   (e.g. control system of a nuclear power plant)

The higher the marked number, the higher is the output criticality.

Score: ....
5. Integration with other systems:

1. List all the connected systems
2. Express the amount of data exchange per connected systems in FPA
3. Add up all the FPA's from 2
4. Calculate score as: total from 3/ total system FPA

The higher the score the more important the system interface.

6. Expected number of years to replacement:

State the expected life time in years.

Score: ... years

A life time of more than five years can be regarded as long.
Do you agree with the following assumptions:

1) The quality factors:
   - completeness of input
   - correctness of input
   - correctness of processing
   are always important
   agree/ do not agree

2) The quality factors:
   - efficiency
   - response time
   should only get extra attention when the requirements for all the other factors are fulfilled.
   agree/ do not agree

Please explain if you do not agree with these assumptions.

1) To prevent a software failure
2) To limit the effects of a software failure
3) Rate the importance you give to this quality factor as very high (vh), high (h), medium (m), low (l) or very low (vl)
4) Explain or comment on your rating
DECISION SHEET II

IDENTIFICATION
(of the application)

Name of Analyst:

Position of Analyst:

Date:

<table>
<thead>
<tr>
<th>The more the score for question nr. about tends to be</th>
<th>the more important is quality factor</th>
<th>Your rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 system integration high</td>
<td>connectivity provisions</td>
<td></td>
</tr>
</tbody>
</table>

rate vn, n, m, l, vi

Explanation of and comments on your rating:

<table>
<thead>
<tr>
<th>The more important the score for question nr. about tends to be</th>
<th>the more important is quality factor</th>
<th>Your rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. years to replacement high</td>
<td>portability</td>
<td></td>
</tr>
</tbody>
</table>

rate vn, n, m, l, vi

Explanation of and comments on your rating:
Definitions of quality factors

completeness of input
The extent to which input is only accepted when it provides all the data needed for a processing cycle.

connectivity
The ease with which the software product can exchange data with other software products.

correctness
of input
The extent to which input is only accepted by the system when it is in accordance with reality.

of processing
The extent to which input will be processed according to specifications.

efficiency
The amount of computing resources and code required by a program to perform a function.

flexibility
The extent to which the user or the application manager for the users can make adjustments or variances to the software products without affecting the code.

graceful degradation
The extent to which the system can continue with partial failure and be recovered in a modular fashion.

maintainability
Effort required to correct (corrective maintenance) or optimize the program (perfective maintenance) or adjust its functionality (adaptive maintenance).

portability
Effort required to transfer a program to another hardware and/or system software environment.

recovery provisions
Measures to limit the effort required by users or others to make the program run properly after a system failure.

reliability
The extent to which the program runs without failures.

response time
Speed of processing interactive transactions at a specified level of workload.

security
Extent to which access to software or data by unauthorized persons can be controlled.

testability
Effort required to test a program to ensure it performs its intended function.

user friendliness
Effort required to learn, operate and prepare input and interpret output of a program.
The need to solve the confusion of tongues in SQA
Roel van den Berg

The attention for quality has increased in many industries in the last decade. The software industry is no exception. To start an quality improvement program it is necessary the participants agree on a definition of software quality. A typology can contribute to clarification in the confusing field of metrics and measures.

Increased importance of quality

During World War II operation management was an important discipline. Military products require robustness of design and have to be manufactured in relatively controlled processes to guarantee interchangeability.

Although the defense industry stayed an initiator of quality improvement programs after the war, the relative importance of quality assurance decreased and disciplines like finance and marketing became better bases for careers in management.

At least in the West. Within the framework of the Marshall plan people like Juran and Deming implemented the techniques developed to control the production processes of the allied forces in Japanese industry. Because these Americans did not speak Japanese they needed easy access to the Japanese top management. This contributed to the acceptance of their ideas. The Japanese not only took over the concepts but also put effort in improving them and not without success. In the beginning of the 80s Western management theorists started to realize that the Japanese business success was highly related to their attention for quality.

In Europe these global effects were intensified by the creation of standards for quality management, normalisation, testing and certification in preparation of the realization of the internal European market.

Currently quality management is recognized as one of the
crucial pillars of a sound business policy in every industry. The information system industry is no exception to that (van Genuchten 1991). Specific reasons contribute to the concern about quality in this business.

1. The scope of the applications has expanded

Contrary to the initial years of information system development, when developer and user of an application were often the same person, developers are currently often confronted with applications they no longer automatically understand.

At the same time the current user is often someone with a low level of knowledge of IT. Consequently, more effort is required to prevent misunderstandings about the task of the user or technological limitations of the system.

2. The user has become more critical

Initially decisions about IT applications were left to administrators or heads of EDP departments. In many organizations no senior executive could be found with responsibility for the IT policy. But due to the expanded range of applications and the increased importance of IT for many organizations management's attention for it has increased. IT investments are more and more regarded as normal investments and suppliers are more and more confronted with critical questions about costs and benefits of their suggestions.

3. The software business is no longer a seller's market

The astronomical growth of the business is over. Suppliers experience increased competition.

Need for serious quality improvement is necessary in the software industry. A majority of the products is never accepted or never used after acceptance. Add to these failures the projects that eventually lead to a successful system but at the cost of significant budget problems and almost all software development projects are included.

Around the world initiatives to improve the quality of software were taken. Large ones like the establishing of the Software Engineering Institute at Carnegie Mellon University
and small ones (like the project "Quality of Information Systems" at Twijnstra Gudde that the author participated in).

The first question they all had to deal with was: "What is software quality?"

What is software quality?

Generally software quality is split up in three major segments:

1) quality of the product

   Most theorists have focused on product quality. Traditionally the major preoccupation of software quality assurance has been reliability ((Chow 1985), (Card 1990)).
   
   Boehm (1978) and McCall (1978) were the first to develop structures which were based on a broader definition of software quality. The elements of these structures were all related to system performance.
   
   The interpretation of software quality as software performance is still the most common. But at the end of the 80s broader conceptions of quality were promoted which also included functionality (Basili 1991).
   
   An even broader kind of definition, which is closely related to the intuitive notion of quality, takes the system’s success as a basis to judge its quality. The recent rise of this type of definition is related to the increased focus on the added value of the IT investments ((Strassman 1988), (Parker & Benson 1987), Sebus (1991)).

2) quality of the process

   Humphrey developed a five level software process maturity model (figure 1) and found that the majority of the software organizations is still in level 1. No company was at level 4 or 5.

3) quality of the resources (tools, developers etc.)

   Boehm found that with developers with a very low capability and experience level a project would require just over 10 times as much effort to completion than with developers with the highest ratings in experience and capability.
   
   These three segments are interconnected. For each pair an
example of interaction will be given.

**product/process**

Humphrey (1987) made clear active product control is only possible in the case of a certain level of maturity of the control of the software development process.

**product/resources**

Kemerer and Banker (1992) investigated the influence of experience of the developers on the maintainability of the product.

**process/resources**

Curtis (1992) investigated how CASE tools affected developer's productivity and found that the effect of CASE highly depends on the level of control over the development process. Once it is clear what is meant by software quality attention can be given to how it should be controlled.

**The necessity of control cycles in software development**

In many development fields it has been generally accepted that control cycles are needed to manage the development process.

Software development was an exception (DeMarco 1983). Practitioners claimed that because of the exceptional nature of both the software development process and the resulting software products the attempts to use traditional control techniques in this business would be doomed to fail.

Rockwell (1991) uses a cultural/historical analysis to explain this scepticism. But its persistence is probably most strongly related to the power distribution in the software market. Because of their strong position in a sellers market the developers were not forced to reconsider their attitude. Till the end of the '80s, when the market started to become less tolerant of the lack of discipline that was associated with it.

To survive software developers actually had to be in control, not just think they were. Basili (1991) is one of the people who claim that software quality control without using a traditional control cycle is impossible.

What are necessary steps before a control cycle can be
Steps towards improved quality control

1. It has to be clear what has to be controlled. Measurement takes time and money and should not be done without valid reason.
2. Once the characteristics are known it is necessary metrics are available for them.
3. To guarantee valid application of these metrics and make comparison of measurement results possible it is necessary measurement prescriptions are developed.
4. Once valid measurements are available it is possible to establish norms.
5. Control is possible by taking measures after reality is compared with the norms.

In this five step model, a next step is only possible when the previous steps have been completed satisfactorily.

Metrics development

With the last remark in mind it is surprising to see that much more has been done in the area of metrics development and measurement prescription than in the area of characteristics development.

product quality
Boehm and McCall both gave only a start to development of metrics for the quality factors mentioned in their structures but others made more elaborate sets of metrics (e.g. Gilb (1973) Willmer (1988), Rijsenbrij (1990), Wijers (1992)). Both Wijers and Rijsenbrij also deal with measures but they skip norms.

process quality
Humphrey's maturity model and the related instrument is an important breakthrough.
Hewlett Packard developed a set of metrics to control the maintenance process (Grady 1987). Kemerer (1987) developed
metrics for maintenance control too.\footnote{Roughly speaking, in the U.S. development of process metrics has dominated while most product metrics were developed in Europe.}

**resources quality**

Boehm (1981) developed ordinal scales to rate five personnel attributes and the use of software tools in his COCOMO model.

**Need for characteristics guidance**

In the area of characteristics development however not much has been done so far. Many quality theorists mention the necessity to focus the attention on a small number of performance factors. Humphrey (1988) mentions it is necessary to only put measurement effort in relevant characteristics.

This idea can be found in many project management methods as well (e.g. Wijnen 1985): before applying metrics it is necessary to define what should be measured and why. Measurement takes resources. It should only be done if it results in added value to the project. From the total domain of product, process and resources metrics a limited number has to be selected for the important factors only.

**Product quality differentiation**

Besides the efficient use of metrics two other reasons exist why it necessary to make choices regarding product quality, i.e. product performance.

First of all, costs are often exponentially related to the performance level of a factor. It would require a tremendous amount of resources to get the ratings for all factors at "very high".

A second reason is related to this problem. The techniques used to enhance the quality of a certain product in some respect are often conflicting. The response time is positively influenced by e.g. algorithm optimization but this has a
negative effect on the portability of the product.

Because of these reasons many people realized it is necessary to make decisions regarding quality in an early stage.

On the other hand, many others are still critical of prioritization. They are afraid of the "Pinto effect": the loss of trust from customers once those find out the developers have deliberately traded off quality.

It is important to understand that prioritization is not one of the ways to deal with performance assurance, but the only way. If one does not give thought to trade off problems at the beginning of the development process, the prioritization dilemmas will force themselves upon the developers in a later phase when there is no time left to really think through the consequences and the room for alternatives has already been limited significantly.

McCabe (1987) and Wijers (1992) state that it is necessary to have a discussion about what factors are important in a particular case but they do not provide any guidance about how this discussion should be organized.

Some have suggested a typology, which makes clear which factors should be selected as important in a specific situation, could be an helpful tool to deal with the trade off problems.

Many other theorists and practitioners in the MIS field (and not the least of them) are sceptical about the possibility to develop a typology. They think it has to be decided from case to case what the important factors are.

Those who do not agree with them have only been fairly successful so far. Evans and Marciniack (1987) discuss a typology which relates application and system characteristics to important performance factors. Bemelmans (1991) relates the importance of some performance factors to characteristics of the control situation the system is going to support.

What's new?

Here task technology is taken as a basis for the development of a new typology.
Four aspects are used as dimensions to define situation:
volatility of the environment
task analyzability
task significance
task interdependence

In figure 1 the typology is illustrated. The results were arrived at via literature research and interviews with both theorists and practitioners. The typology has not been systematically tested yet.

When taking the task as the central criterium in the process of establishing the importance of individual performance factors it has to be taken into account that often the task structure changes during the development process. Effective application of today's systems almost always requires major, sometimes radical changes in an organization's structure, personnel, roles and business processes. Thus it will often not be possible to use the typology as a tool in a simple, rigid analysis cycle.

The added value of the typology lies much more in its effect on the communication process.

Bemelmans (1991) states that during the requirements analysis only 10 percent of the time is spent on performance requirements.

Often many non technical stakeholders feel insecure in the area of performance. Designers' attempts to get feedback about the functionality of alternatives often leads to elaborate discussions but performance is easily left to the developers because it is assumed they know it best. When in a later phase users find out this is not true costly adjustments are needed.

The typology can facilitate discussions with the users about what aspects of system performance they really need and which are less relevant in their case.
Different stakeholders set different priorities regarding performance. To prevent that discussions about performance requirements are heavily politically biased and every party is only screaming to get his set of often unreasonable requests in the requirements document it is important the participants get a framework which makes the reasons for performance prioritization transparent. The typology can fulfill an important role in this process.