

QUALITY MEASUREMENT AND THE UTILISATION OF MEASUREMENT RESULTS IN A SOFTWARE DEVELOPMENT PROCESS

Jorma Hirvensalo

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Electrical and Communications Engineering, for public examination and debate in Auditorium S4 at Helsinki University of Technology (Espoo, Finland) on the 11 th of April, 2003, at 12 o'clock noon.

Helsinki University of Technology
Department of Electrical and Communications Engineering
Networking Laboratory

Teknillinen korkeakoulu
Sähkö- ja tietoliikennetekniikan osasto
Tietoverkkolaboratorio

Distributor:
Helsinki University of Technology
Networking Laboratory
P.O. Box 3000
FIN-02015 HUT
Tel. +358-9-451 2461
Fax +358-9-451 2474

ISBN 951-22-6423-4
ISSN 1458-0322

Otamedia Oy
Espoo 2003



HELSINKI UNIVERSITY OF TECHNOLOGY P.O. BOX 1000, FIN-02015 HUT http://www.hut.fi		ABSTRACT OF DOCTORAL DISSERTATION	
Author Jorma Hirvensalo			
Name of the dissertation Quality Measurement and the Utilisation of Measurement Results in a Software Development Process			
Date of manuscript March 7, 2003		Date of the dissertation April 11, 2003	
<input checked="" type="checkbox"/> Monograph		<input type="checkbox"/> Article dissertation (summary + original articles)	
Department	Department of Electrical and Communications Engineering		
Laboratory	Networking Laboratory		
Field of research	Telecommunication Software Engineering		
Opponent(s)	Professor Inkeri Verkamo (University of Helsinki)		
Supervisor (Instructor)	Professor Raimo Kantola (Helsinki University of Technology)		
Abstract The research topic of the thesis is quality measurement and the utilisation of measurement results in a software development process of a large switching system. The first problem studied, the utilisation, means that measurement results are poorly analysed; data is collected, but not enough conclusions are drawn about how that experience data could help in the planning and execution of software development projects. So, the problem is how to learn more about the collected material. The second research problem is further development of measurements from the point of view of the utilisation. We think this is because the existing metrics have too much focused on later phases instead of using purposeful driving metrics for controlling the process towards high final quality in early phases. The main objectives of this study are to examine the coupling of existing quality metrics to design practices, to generate new quality indicators, and to develop the utilisation of measurement results further. Basically, the study aims at finding out relationships between post-release measurement results and the driving attributes. It also aims at making proposals for methods to improve and introduce new measures; and to create a utilisation model for better use of measurement results. Research methods involve both empirical research and constructive design of models for measurement processes and utilisation. We apply statistical methods to sixteen data sets gathered from real software modules; and discuss correlation in the light of nineteen case studies. Eleven stated hypotheses are tested by their evidence based on the statistical analyses of the material collected during the 1990's from industrial projects. Generation of process and utilisation models is based on constructive modelling, on experience of good practices, and on adoption of elements from modern measurement theory as well as from goal- and attribute-oriented approaches. First, the results contribute to enhancement of empirical knowledge regarding implementation, problems, obstacles as well as lessons learnt on utilisation of measurements in an industrial environment. As a second result, the study provides analysed test results for each of the eleven hypotheses. A new result worth mentioning is a conclusion that 70-80 % of individual modules, which have been faultless in Function Test, are also faultless in System Test and during their first 6 months after delivery. Furthermore, the thesis presents an improved measurement framework including a renewed measurement process. The process is composed of four parts for planning, definition, performing - and especially - for utilising measurements. The thesis has also resulted in a novel utilisation model that can be used to improve the use of measurement results. Finally, we present a supplementary result that requires further research and validation. In particular, we have constructed a six-level measurement utilisation capability model.			
Keywords Telecommunication switching, software quality, measurements, metrics, capability, maturity			
UDC		Number of pages 154 p. + app. 29 p.	
ISBN (printed) 951-22-6423-4		ISBN (pdf) 951-22-6424-2	
ISBN (others)		ISSN 1458-0322	
Publisher Networking Laboratory/Helsinki University of Technology			
Print distribution			
<input checked="" type="checkbox"/> The dissertation can be read at http://lib.hut.fi/Diss/			

Preface

This study consists of work performed during the years 1995-2001. Reasons for and origin of this study were born in a Management Review concerning measurement results we had collected since 1987. A need for a book arose and I thought that a dissertation would serve this purpose. After my half-year stay at Ericsson Telecom in Sweden, in the mid of 1995, the management encouraged me to continue in research issues, so I put my energy to the topic entitled. In late 1980's I co-operated in the development of a fault content estimation model developed at Ellemtel (Lennselius, 1990). As this model was experimentally used at Ericsson in Finland, I provided Lennselius with project and product data from a few local projects. In co-operation with R&D management we paid attention on the creation of a method for analysis of Preconditions and Quality of software projects (Hirvensalo, 1990a).

I joined in Ericsson's Software Metrics project in early 1990's and continued active development of the measurement system as a member of Ericsson's PQT (Productivity-Quality-Time) group. I participated in the group where Ericsson has put a lot of effort in studying the introduction of metrics for new object-oriented software applications written e.g. in C++ and Erlang (Armstrong, 1996) languages. Participation in professor Fenton's seminar in Linköping (Fenton, 1996) and involvement in Ericsson's EPMG group (Nilsson, 1996) convinced me about the importance of improved and new measurement concepts. My role within EPMG group has been, considering the new terminology, to revise the instructions and templates for measurement definition purposes.

During 1996...1998 my work changed to focus more on research. As a member of the supervision group for several software research projects my role was to consider whether the results provide candidates for new metrics or not. In 1998, I joined as a partner manager in the second phase of PROFES (<http://www.elc.vtt.fi/profes/>) project sponsored by European Commission. In the PROFES project I had an opportunity to use and validate a "different" approach to measurements, e.g., based on Goal-Question-Metric. During this co-operation I had an opportunity to widen my perspective towards product focused process improvement methodology and goal-oriented measurements. In the Project Controllability (LUCOS) study (see Section 5.1.5) my participation involved definition of quality indicators that the new project can visualise and the investigation of the most important requirements for modernisation of the database as well as to consider links to Ericsson's PQT. Piloting of Utilisation Models presented was performed in close co-operation with quality people and management of two units.

This thesis concludes my work within measurement issues during ten years time span. Measurement activities have related to my daily work in industry and my company has permitted me to use a significantly large sample of material collected from software product development in an industrial environment. I have utilised my long practical experience and close co-operation with Ericsson's best measurement experts implementing real measurement systems. Many contacts with external experts in software research projects have supported my work in a valuable way.

Veikkola, March 5, 2003

Jorma Hirvensalo

Acknowledgements

I would like to thank Prof. Kauko Rahko for his very valuable support and guidance. During my study comments and suggestions. Prof. Reijo (Shosta) Sulonen kindly reviewed my material from the point of view of Software Engineering – many thanks to him. I thank the external pre-examiners of my thesis, Prof. Ilkka Haikala and Prof. Samuli Saukkonen, for their critique and suggestions for improvement. Especially I am very grateful to Division Managers of Telecom R&D at Ericsson in Finland, Lars Antman and Kristian Toivo, for their continuous encouragement and very positive support. I am also grateful to my Research manager Seppo Lindborg for the opportunity to use my working hours to a large degree during my stay at the research department. My new chief, Pekka Nikander, since August 1998, has greatly encouraged me with numerous valuable comments, in the later phase of the thesis. Later on, my current boss Ilmo Sirkka arranged me a possibility to concentrate on completion of my research. I wish to thank my company, that has thus economically supported this thesis.

I am also indebted to a number of fellow workers at Ericsson. Especially, Merja Elf-Mattila, Markku Jauhiainen, Johan Lahtivuori, Juhani Panula, Sakari Pulkkinen Harri Reiman and Henrik Sundell-Honkanummi have supported me with numerous comments. I want to thank Mika Ahola about a special examination of first 7-12 month's operational quality. Veijo Nygren and Esa Koskinen have supported me in generation of some graphs enclosed. HUT/CS students Markus Lindberg and Johan Sundström assisted in a lead time study and did a lot of collection and creative research within lead time oriented relationships. Co-operation with Tony Jokikyyry who implemented a web-based tool for inspection measurements has been very valuable. My fruitful contacts with other PQT team members, Anders Nilsson-Hallqvist, Lauritz Tuttunen and Kjell Wallin have made it possible to get my new ideas implemented in a practical measurement process in industry. Mia Meinander assisted me in testing of the created utilisation plan in a real project. My warm thanks to Risto Nevalainen, from Software Technology Transfer Finland, who has reviewed my capability maturity model for utilisation of measurements. Finally, I am very grateful for co-operation with all PROFES consortium members.

My daughter Pia Vuolanto and her husband Ville Vuolanto have kindly helped me in translating my "Finglish" into English. Special thanks are due to Christine Leidenius-Kourula for many linguistic corrections. I am very grateful for Björn Lundwall's scholarship for Study Trips within Ericsson, which made possible my visit to Ericsson in Holland. Financial support to this work from Emil Aaltonen Foundation is gratefully acknowledged. I also want to express my gratitude to my sons, Petri, Mikko and Markku, for their support. Finally, I thank my wife Raija, for her continuous support, endless trust and tireless encouragement during my work.

Contents

PREFACE.....	I
ACKNOWLEDGEMENTS	II
LIST OF APPENDICES	VI
LIST OF TERMS	VII
 1. INTRODUCTION	 1
1.1. BACKGROUND	1
1.2. THE MOTIVATION OF THE STUDY	3
1.3. THE RESEARCH PROBLEM AND APPROACH	4
1.4. STRUCTURE OF THE THESIS	5
1.5. CONTRIBUTION TO KNOWLEDGE	6
 2. MODELLING THE ENVIRONMENT OF MEASUREMENTS	 7
2.1. MODELLING SOFTWARE DEVELOPMENT	7
2.1.1. The project process model	7
2.1.2. The software product development model	9
2.1.3. The product model	10
2.2. MODELLING MEASUREMENTS IN THE SOFTWARE DEVELOPMENT	10
2.3. EXISTING QUALITY MEASUREMENT PROCESS	11
2.3.1. Presenting a practical process example	11
2.3.2. Evaluation of the existing process	13
 3. EXPERIENCE FROM PRESENT SOFTWARE QUALITY MEASUREMENTS	 15
3.1. MEASUREMENTS AT SOME INDUSTRIAL COMPANIES	15
3.1.1. Raytheon.....	15
3.1.2. Hewlett Packard (HP)	16
3.1.3. LORAL	17
3.2. EXAMPLES OF EXISTING QUALITY MEASUREMENTS AT ERICSSON.....	18
3.2.1. Examples of existing product quality indicators.....	18
3.2.2. Examples of process performance indicators.....	19
3.3. CASE EXAMPLES CONSIDERING THE PRESENTATION AND USAGE OF RESULTS	20
3.3.1. CASE 1. Fault density in Function Test.....	20
3.3.2. CASE 2. Classification of modules by the number of faults	21
3.3.3. CASE 3. Fault density for first 6 months after delivery.....	21
3.4. LESSONS LEARNED AT ERICSSON IN FINLAND	22
3.4.1. Positive experience from management and quality personnel.....	23
3.4.2. Difficulties experienced by management and quality personnel	23
3.4.3. Positive experience from senior designers and experts.....	24
3.4.4. Difficulties experienced by senior designers and experts	24
3.4.5. Obstacles of linking measurement results to the design process	25
3.5. NEEDS FOR FURTHER DEVELOPMENT OF MEASUREMENTS	26
3.5.1. Early needs to add company specific new measurements	26
3.5.2. Needs and expectations for utilisation of measurement results	29
3.6. RELATIONS TO CUSTOMER SATISFACTION	29
3.7. CONCLUSIONS ON PRESENT MEASUREMENTS.....	30
 4. GENERAL REQUIREMENTS FOR ANALYSIS AND USAGE OF MEASUREMENT RESULTS... 32	
4.1. CMM.....	32
4.2. EXTRACTS FROM ISO9000 QUALITY MANAGEMENT AND QUALITY ASSURANCE STANDARDS	34
4.2.1. General ISO 9001 standard.....	34
4.2.2. Guidelines for software work	34
4.3. QUALITY AWARD CRITERIA	35
4.4. TQM	36
4.5. THE ERICSSON QUALITY MANUAL	36
4.6. CONCLUSIONS ON REQUIREMENT CAPTURING	37

5. NEW OPPORTUNITIES TO USE THE KNOWLEDGE FROM THE EXISTING MEASUREMENT DATABASES 39

5.1. RELATED WORK.....	39
5.1.1. Studies at Ericsson in Finland	39
5.1.2. Studies in other Ericsson companies	40
5.1.3. Studies at University of Linköping (Liu)	41
5.1.4. Studies conducted by Ellementel.....	41
5.1.5. Other related studies	42
5.2. ABOUT MATERIAL USED AS RESEARCH OBJECT	43
5.2.1. Examples of project characteristics	43
5.2.2. Examples of module characteristics	44
5.2.3. Selected data sets	46
5.3. METHOD OF THE CASE STUDIES.....	47
5.3.1. Stating and handling of the hypotheses	47
5.3.2. About the research method	48
5.3.3. Views of presenting information in case studies.....	49
5.3.4. Selected relationships and correlation studies	49
5.4. CASE STUDIES - EXPLOITATION POSSIBILITIES OF RESULTS	51
5.4.1. Quality Productivity	51
5.4.2. Effort vs. volume	53
5.4.3. Lead time vs. quality	56
5.4.4. Post-release quality of modules faultless in Function Test	57
5.4.5. Post-release quality in modules having high fault content in test.....	59
5.4.6. Identifying occurrence of "Stinker" modules	61
5.4.7. Time based trend of cumulative fault density	62
5.4.8. Correlation between fault density in test and slippage	63
5.4.9. Quality by type of the module	64
5.4.10. Quality by technical group of modules	65
5.4.11. Module quality by modification grade.....	66
5.4.12. Module quality during 7-12 months after delivery	67
5.4.13. Proportion of faults causing fatal failures at the customer	68
5.5. CONSIDERING EARLY DESIGN ATTRIBUTES	69
5.5.1. Studying impacts of number of signals on faults and volume	70
5.5.2. Impact of inspection process on quality.....	71
5.5.3. About cost of quality assurance.....	75
5.6. OPPORTUNITIES TO USE MEASUREMENT DATA FOR PREDICTION OF QUALITY	77
5.6.1. Prediction opportunities on project level.....	77
5.6.2. Prediction opportunities on module level	78
5.7. CONCLUSIONS ON CASE STUDIES AND IMPROVEMENT OPPORTUNITIES.....	79
5.7.1. Lessons learned	79
5.7.2. New opportunities to improve measurements	80
5.7.3. Opportunities to improve use of measurement results	81

6. PLANNING AND DEFINING NEW MEASUREMENTS 83

6.1. DIFFERENT APPROACHES OF PLANNING NEW MEASUREMENTS	83
6.1.1. Defining a set of key metrics.....	83
6.1.2. GQM approach	84
6.1.3. MQG Approach.....	85
6.1.4. Goal attribute measure (GAM) method	85
6.1.5. The Balanced scorecard.....	85
6.1.6. Metric development process.....	86
6.2. QUALITY MODELLING AS A MEANS TO CLARIFY CUSTOMER SATISFACTION.....	86
6.2.1. Factor - Criteria - Metric (FCM).....	86
6.2.2. The ISO/IEC 9126 quality model	87
6.2.3. FURPS - an industry-oriented quality model	88
6.2.4. Summary of quality modelling.....	89
6.3. MEASUREMENTS SUPPORTING THE ATTAINMENT OF HIGH CMM LEVELS	89
6.4. MEASUREMENT FRAMEWORK	92
6.4.1. Useful new measurement concepts	92
6.4.2. A proposal for a Plan Measurements process.....	94
6.4.3. A proposal for a Define Measurements process	96
6.4.4. Measurement definition templates and instructions	97
6.4.5. Examples of well defined new measures	99

6.4.6.	General measurement implementation guidelines	99
6.5.	SYNTHESIS	100
6.5.1.	Inheriting issues from old measurements for future use	100
6.5.2.	Important new measurement subjects in the future.....	100
6.6.	SUGGESTIONS FOR IMPROVING MEASUREMENTS.....	102
6.6.1.	Implementing process-oriented measurements	102
6.6.2.	Inspection measurements – an example implementation	102
6.6.3.	Proposals for improved project measurements	104
6.6.4.	A proposal for improved quality cost measurement	105
6.6.5.	Proposals for ISP measurements	106
6.6.6.	Proposals for customer focused measurements	107
6.6.7.	Other suggestions to improve measurements.....	107
6.7.	CONCLUSIONS CONSIDERING PROCESSES AND MEASUREMENTS	108
6.7.1.	Conclusions on our approaches	108
6.7.2.	Evaluation of our process models.....	109
7.	PROPOSALS FOR MEASUREMENT PROCESSES AND UTILISATION MODELS	111
7.1.	A PROPOSAL FOR IMPROVED MEASUREMENT PROCESS	111
7.1.1.	Perform measurements	112
7.1.2.	Utilise measurements	113
7.2.	GUIDELINES FOR DEFINITION OF UTILISATION MODELS	114
7.2.1.	Background.....	114
7.2.2.	Purpose.....	115
7.2.3.	Capturing requirements for utilisation.....	115
7.2.4.	Requirements from CMM.....	115
7.2.5.	Essential issues to be considered	116
7.3.	CREATING A GENERIC UTILISATION MODEL	120
7.3.1.	Conceptual Framework for the Utilisation Model.....	120
7.3.2.	Measurement utilisation by phase and organisational level	121
7.3.3.	Mapping provided measurements to their intended use	123
7.3.4.	Use cases and practices.....	123
7.3.5.	Linking utilisation to work processes	126
7.3.6.	Marketing measurement utilisation opportunities	127
7.3.7.	Tailoring the Utilisation Model.....	127
7.4.	A MEASUREMENT UTILISATION CAPABILITY MODEL (MUCM)	127
7.4.1.	Background.....	127
7.4.2.	The Purpose of the MUCM model.....	128
7.4.3.	Capability levels.....	128
7.4.4.	How the MUCM compares to CMMI	128
7.5.	CONCLUDING AND EVALUATING UTILISATION MODELS.....	130
7.5.1.	Evaluation of Perform and Utilise measurements processes.....	130
7.5.2.	Evaluation of the Utilisation plan.....	131
7.5.3.	Evaluation of the MUCM model	131
8.	FINAL CONCLUSIONS	132
8.1.	CONCLUDING OUR STUDY	132
8.2.	OTHER CONCLUDING REMARKS REGARDING THE “MEASUREMENT JOURNEY”	132
8.3.	CONTRIBUTIONS OF THE THESIS	133
8.4.	PROPOSALS FOR IMPROVING UTILISATION	135
8.5.	PROPOSALS FOR FURTHER RESEARCH	135
REFERENCES	137

List of Appendices

APPENDIX 1	DEVELOPMENT HISTORY OF ERICSSON'S MEASUREMENTS
APPENDIX 2	THE QUALITY MACHINE
APPENDIX 3	DESCRIPTION OF ERICSSON'S PQT
APPENDIX 4	EXISTING QUALITY MEASUREMENT PROCESS AT ERICSSON IN FINLAND
APPENDIX 5	DEPARTMENT MANAGER REPORT
APPENDIX 6	EARLY QUALITY MEASUREMENT EXPERIENCE FROM ERICSSON IN FINLAND
APPENDIX 7	THE STATISTICAL METHODS USED
APPENDIX 8	RENEWED TEMPLATE FOR MDD
APPENDIX 9	RENEWED TEMPLATE FOR MRD
APPENDIX 10	DETECTED FAULT CONTENT OF PRODUCT - MDD
APPENDIX 11	FAULT DENSITY IN C/C++ SOFTWARE DESIGN
APPENDIX 12	A TEMPLATE FOR MEASUREMENT UTILISATION PLAN – GENERIC
APPENDIX 13	MEASUREMENT UTILISATION PLAN – EXAMPLE

List of terms

AC	Approved Correction
CAS	Channel Associated Signalling
CMM	Capability Maturity Model
ECAP	Ericsson CMM Assessment Process
EQM	Ericsson Quality Manual
ESI	European Software Institute
ESPRIT	European Strategic Programme for R&D in Information Technology
ESSI	Ericsson System Software Initiative
F/kPLEX	Fault per thousand PLEX statements
FAP	Fault Analysis and Prevention
FiSMA	Finnish Software Metrics Association
FT	Function Test
FURPS	Functionality, Usability, Reliability, Performance, Supportability
FP	Function Point
GQM	Goal-Question-Metric
HLL	High-Level Language
HLPLEX	High Level PLEX
HP	Hewlett-Packard
I/O	Input/Output
IS/IT	Information Systems/Information Technology
ISDN	Integrated Services Digital Network
ISO	International Organisation for Standardisation
ISP	In-Service Performance
ISUP	ISDN User Part
KPA	Key Process Area
KNCSS	kilo Non-commented Source Statements
KSLOC	Kilo source lines of code
LMF	Oy LM Ericsson Ab, Ericsson in Finland
MAP	Mobile Application Part
MDD	Measurement Data Definition
MHS	Modification Handling System
MRD	Measurement Result Definition
MS	Milestone
NUP	National User Part
PDCA	Plan-Do-Check-Act
PLEX	Programming Language for Exchanges
PP	Provisioning Process
PQT	Productivity-Quality-Time
PRA	Product Release A
QA	Quality Assurance
QFD	Quality Function Deployment
QMS	Quality Measurement System
RFA	Ready for Acceptance
SDL	Specification and Description Language
SEI	Software Engineering Institute
SPA	Software Process Assessment
SPICE	Software Process Improvement and Capability dEtermination
SQA	Software Quality Assurance
ST	System Test
STP	System Test Plant
SW	Software
SWAT	Software Action Team
TG	Tollgate
TQM	Total Quality Management
TR	Trouble Report
TSS	Trunk and Signalling Subsystem
TUP	Telephony User Part
UAB	Ericsson Utvecklings AG
VISCON	VISible CONsequences

Chapter 1.

Introduction

1.1. Background

About quality

Quality is a very wide and subjective concept. As the concept of quality depends on perspective and orientation, there is no generally accepted definition. The meaning of the word quality is usually refined within a company's Quality Policy¹, and may, for example, emphasise a customer-oriented view. A very essential issue in achieving customer satisfaction is faultless functioning of products. In the software industry, it has been common to focus on improvement efforts on defect avoidance. Therefore quality has typically meant "absence of faults".

About measurement

Already in ancient times people tried to make things measurable. For example, measurements of properties like time, weight and distance have been known in ancient times. Nowadays, measurements are an essential part of everyday life e.g. in electrical, medical and mechanical engineering. Measurements are common in car industry, in aviation; even in modern sport championships measurement data is actively collected and used.

At first, we need to define what is meant by measurement. According to a classical definition used in general measurement theory (Finkelstein, 1982), measurement is the assignment of numbers to properties of objects or events in the real world by means of an objective empirical operation, in such a way that the numbers represent or symbolise what they refer to. In this study we use a refined definition (Fenton, Pfleeger, 1996):

Measurement is the process by which numbers and symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules.

A measure is a mapping of the real empirical world to the mathematical world. By using mathematical models it is easier to understand the behaviour of entity attributes and their relationship to other entities. Here we want to emphasise the importance of the words, *attribute and entity*, which are not always clearly used in the context of software measurements.

Attribute is a feature or property of an entity.

Measurement is thus a way of describing real world entities by describing their characteristics. An *entity* in "software world" means either an *object* (e.g. a software product, a designer) or an *event* (e.g.

¹ Ericsson's policy is to stand for World-Class Quality and Business Excellency in everything and to commit not only to satisfy but even to exceed customer expectations.

testing phase of a software project). There are several ways of mapping the numbers to objects. We need, as a basic principle, some sort of a *measurement scale*. Scale types like nominal, ordinal, and ratio, known in statistical mathematics; are important because they also express which mathematical calculations are permitted for each scale. The term *metrics* (e.g. software metrics, software quality metrics) is used also to some extent in this study because it is widely used in the software community. In the context of software measurements it is usual to classify entities, attributes, measurements or metrics into three basic classes: *product*, *process* and *resource*. We make a clear distinction between *product metrics* and *process metrics*. Product metrics are tied to software work products (e.g. a requirement specification, a design document, a source code module, or a test data description) or end products delivered to the customer (e.g. a software package), whereas process metrics are tied to a certain process (or subprocess) like function specification, detailed design, and testing. Examples of *resource metrics* in software engineering are designer, tester, inspection team and software tool. Measurement theory holds several other useful fundamental concepts, which will be applied in this study. For instance, *validation* is a concept that helps to make sure that the metrics used are actually measuring what they claim to measure. Therefore, in Chapter 6, we pay special attention to concepts and discipline provided by the measurement theory. We realise that measurement objects in software are abstract and logical system elements and that it is difficult to apply measurement theory strictly. However, we will show how practical application of software measurements can benefit from scientific grounding and measurement theoretic-attributes and rules in order to turn measurements from “art” to engineering.

The role of measurements in software engineering

The importance of measurements in the area of Software Engineering has been acknowledged already early, even during 1968 in the ACM National Conference. It is interesting to note that one of the first measurement topics in this conference concerned software quality attributes (Rubey, Hartwick, 1968).

In practice, the quality of the results of work, for example, in the development of software products, has to be evaluated in some measurable way. In fact, measurement is becoming an increasingly accepted part of software engineering, but it is not yet a well-understood activity in that field. Measurement culture, metrics used, etc. varies in different companies. Most organisations have started to measure quality during their testing activities. The trends (Table 1) of industrial measurement within software engineering have been clearly explained by Shepperd (1995 pp. 4-9):

Table 1. Measurement trends

Era	Measurement trend
1970's and on into the early 1980s	Program code oriented measurements, e.g. Halstead, McCabe. Experiments on applying McCabe's measurements for predictive purposes.
In the 1980's	Exploration of other intermediate products to measure e.g. designs and specifications. Development of Function Points to make effort predictions feasible at an early phase of a project.
Early 1990's	The attention turns to the development process. Growing interest in the classical measurement theory. Measurements for object-oriented design.
Late 1990's	Use and interest in usage of Goal-oriented measurements is increasing. More attention is paid to automation of measurements.

One remarkable trend is that several enterprises have launched company-wide measurement programs, for example HP (Grady 1987, 1992), LORAL (Nusenoff, 1993), Motorola (Daskalantonakis, 1992) and Schlumberger² (Solingen, 1995), just to mention a few. The main benefit of such programs is that, by using measurable goals, these enterprises are able to indicate remarkable improvement of quality each year.

² Currently Tokheim, a company dedicated to fuel dispenser production.

Telecommunication software engineering is younger than general software engineering. Therefore, software measurement has fallen behind in the field of telecommunications. Anyway, several telecommunication companies have introduced standard quality measures. Inglis (1985, pp. 113-118) describes a set of metrics used at AT&T. Ericsson started a company-wide measurement program in 1992. Also other companies have instituted corporate-wide measurement programs, including for example, CONTEL, a US telecommunication firm (Pfleeger, 1993, pp. 504-505).

In spite of this positive evolution we can realise that software processes in a majority of companies still are quite immature. Therefore, it is difficult to “touch” the process and define measurement points. When software professionalism matures, then also the role of measurements becomes more important.

Measurements at Ericsson

Oy LM Ericsson Ab (Ericsson in Finland) has a tradition in the field of software quality measurement. The experience gained is based on data collected from hundreds of new and modified software units (products) developed for different applications in Ericsson's AXE10 telecommunication switching system during the past years. Furthermore, the Ericsson-wide PQT (Productivity-Quality-Time) measurement database contains material that is available for research purposes. The importance of this material³ is based on the fact that a comprehensive data set from real industrial projects and unique software products is provided. This material is further characterised in Section 5.2.

The objective of the Ericsson's PQT has originally been to implement a measurement system that serves as an integral part of the continuous improvement process by:

- Enabling visualisation of long-term trends and thereby finding areas for improvement,
- Supporting “management by facts” instead of “management by opinions”,
- Supporting the definition and follow-up of new quantitative objectives, and
- Increasing quantitative knowledge about the development process in a consistent way.

This study mainly uses Ericsson's measurements as a case, aiming at their further development and use. For further details, a survey on Ericsson's measurement history and trends is given in Appendix 1. That survey serves as an introductory chapter describing the historical background of a big company that had measurements at place before the beginning of the present study. It is recommended for readers interested in early stages of the process where the material was born.

1.2. The motivation of the study

In many other fields of engineering, for instance in electrical engineering, the measurements play a central role. Building the design artefacts is based on known theories, like Ohm's law, which clearly describes the relationships between resistance, current and voltage. In designing an electrical circuit it is possible to determine the voltage in different points of the circuit. It is easily measurable by using a specific instrument, the voltage meter. Measurement unit and scale type is also well defined – the unit “volt” and the scale type “ratio” are used. Measurement of physical objects is easier than measurement of such logical system elements like software. In the field of software we need to develop measurements to a more rigorous and well-defined engineering approach, such that measurements are not considered as an ad-hoc activity, but included into the software development processes.

Recently, several changes have taken place causing a growing interest in this type of research and motivating this study. These include the following:

- Embedded software applications are rapidly increasing and the organisations e.g. in the field of telecommunication are expanding. There is a need to develop the infrastructure in measurement handling and to clarify relations between different levels of the organisation. The interest in applying software quality measurement know-how – not only on upper but also on lower organisational levels – “everyone involved” in the spirit of TQM (Ericsson, 1995), is increasing.

³ Generally, the researchers have faced difficulties in getting material from industry or the material has been very limited.

- Software implementation techniques and tools are evolving. This also brings expectations for new metrics regarding totally new types of software written in new languages (C/C++, JAVA).
- Demands for further development of traditional measurements to correspond to the latest evolution in quality techniques, for example, due to renewed Quality System (e.g. based on ISO9001), Quality Award criteria, CMM etc.
- Trend towards “process thinking” in the software industry is moving measurement points towards earlier design phases. Thus, the things that earlier have not been measurable, should be made measurable.
- Management and organisation expect certain company specific improvement of measurements based on their experience. An example from our case company is discussed in Section 3.5.

In the academic world not many empirical results have been published relating to software measurements. The last but not the least valuable motivation for the present study is to increase the amount of such experience. We provide new knowledge and ways of utilisation examples derived from a large amount of measurement case material collected until now and made available for this study.

1.3. The research problem and approach

The research described in this study concentrates on two problem areas, the utilisation and further development of measurements.

The utilisation problem

In order to maximise the benefits of measurement, the results must be analysed carefully and used effectively. The emphasis should not be on the measurement itself. In many cases, the measurement data has been successfully collected for years, but not enough conclusions are drawn about how that experience data could help in the planning and execution of ongoing and new software development projects. Consequently, more has to be learned about the collected material. Could some rules and planning constants be derived? Could some new utilisation models be defined? What could be used for prediction, estimation and dimensioning purposes?

Need for new metrics

The focus up to now has been on measurements of later phases (i.e. post-release), not on the early phases of the process. The existing metrics used in the software industry express the software behaviours in the early phases of development only in a quite rough and limited way. It has not been easy to find driving metrics that can be used to control the process towards high final quality in early phases. Unfortunately, the couplings between early design parameters and final quality indicators are in most cases still unknown. Can *new* measures that increase the understanding of the complex software process be found? The present study tries also to find answers to this question.

Objectives of the study

The main objectives of this study are to examine the coupling of existing quality metrics to design practices, to generate new – in particular – process oriented quality indicators, and to develop further the utilisation of measurement results in Telecommunication Software Engineering. The practical objectives are defined as follows:

- To find out relations and correlation between post-release measurement results (used up to now) and driving attributes,
- To study the linking of existing quality measurements to software development processes and usual planning practices,
- To make proposals for methods to improve and introduce new measures, and
- To create a utilisation model especially for quality oriented measurements.

The approach of the research

The approach to solving the research problem is an empirical study by statistically analysing a number of measurement cases from the collected material; application of modelling and of the conceptual framework based on modern measurement theory. Measurement case material is used to test evidence of a number of generally interesting hypotheses. We aim at drawing generally applicable conclusions, whenever possible. Models are used to develop and apply measurements, to describe objects being measured and to build relationships between measures. Models are investigated to characterise how the data sets collected from previous projects can be used to predict or to determine the current status of process/products. Examples of models discussed and applied are Quality models, Goal Question Metric (GQM) and Product/Process Dependency Modelling (PPDM).

Our method of developing new process models, a utilisation model and a renewed measurement framework uses constructive design drawing on literature analysis and our long practical experience of implementing measurements in an industrial product development environment. In the conceptual analysis to renew the framework and to create the utilisation model we have been supported by active involvement in teamwork. The method of evaluating the models is mainly based on expert interviews, and to a limited extent, on experimental application in our case company.

1.4. Structure of the thesis

The rest of this thesis is organised as follows.

Chapter 2 - Modelling the environment of measurements – provides a survey of the industrial environment describing the product and process models in order to understand the measurement objects and the related development process.

Chapter 3 – Experience from present SW quality measurements – gives an overview of the existing software quality measurements, not only at Ericsson, but also in some other companies. Results from a problem inventory, needs for further development of measurements, and obstacles that prevent their effective usage, are handled based on interviews and local industrial experience.

Chapter 4 – General requirements for analysis and usage of measurement results – analyses the external pressures from Capability Maturity Model (CMM) into doing measurements. Furthermore, the measurement issues that are essential from the point of view of the Quality Award Criteria are reviewed at the end of the chapter.

In Chapter 5 – New opportunities to use the knowledge from the existing databases – relationships and correlation studies are created by a number of measurement case examples. Several hypotheses are presented and investigated whether they are proved to be true or false in the light of the collected statistical material. Suggestions are made for use of each particular measurement and opportunities are discussed for using the results to predict quality.

In the beginning of Chapter 6 – Planning and defining new measurements – we describe six different approaches for development of new measurements. The approaches, such as, defining a set of key metrics, Goal-Question-Metric (GQM), Metric-Question-Goal (MQG), Goal-Attribute-Metric (GAM), Balanced Scorecard (BSC) and Metric development process, are analysed. Measurements supporting the achievement of higher CMM levels are analysed in more detail. The usefulness of new measurement concepts is discussed in the context of a measurement framework. Processes for measurement planning and definition are modelled. Finally, a set of most useful indicators and improved ways of presenting results are proposed.

Chapter 7 - Proposals for measurement processes and utilisation models – presents an improved measurement process. The rest of Chapter 7 concentrates on the utilisation of this process. A generic utilisation model is presented. Its intention is to provide guidance on how to apply measurement results in an ideal way. The model states what the measurements are used for, who are the measurement customers and which are the analysis forums in each particular case. Guidelines are given for tailoring the utilisation model to consider different SW development applications and organisations. Furthermore, a few instructions for different uses of measurements like prediction, estimation and improvement analysis are discussed. Finally, a new measurement utilisation maturity model is created.

Chapter 8 - Conclusions – evaluates the results and concludes the essential findings and recommendations.

1.5. Contribution to knowledge

The present study makes use of a unique software development experience trying to generalise that experience as much as possible. We have used Ericsson's unique database and experience for trying to find answers to generally interesting questions such as early detection rates of faults and fraction of faults slipping to the customer. Numerical results and evidence for eleven hypotheses stated have been studied in the light of case examples based on material from industrial projects and software modules. New knowledge is provided about dependencies of attributes known during design and test phases to observed post-delivery attributes. The performed study provides some new research results from software product development projects regarding fault occurrence during development compared to behaviours in the field. For example, an interesting finding was that 70-80% of individual modules with zero faults prior to release from development (i.e. in Function Test) were also faultless up to end of their first 6 months in the field. The database also provided the opportunity to consider the impact of software modification on quality, an aspect that appears less in the literature.

The results also contribute to knowledge enhancement regarding implementation, problems, obstacles as well as lessons learnt on utilisation of measurements in a multi-national industrial environment.

Presenting measurement activities in a form of well-defined processes is not a frequently appearing topic in the literature. Definition of an international standard (ISO/IEC CD 15939)⁴ for software measurement process is going on in ISO/IEC JTC1/SC 7, but it takes time to get it completed. We provide proposals on well-structured measurement processes. In these proposals we clearly distinguish between planning and establishing of new measurements and measures, and actual running and use of measurements. We thus provide knowledge on how to systematise the development of measurement practices in a company.

We provide a novel *Utilisation model* that can be used to improve the use of measurement results in a company where software plays a central role. As the utilisation until now has been a less successful issue in the world of software measurement we come in with an improved framework. The framework takes into account users, phases, levels, practical usage applications and linking to other processes and systematises an area that has been quite incoherent.

We have also clarified how maturity in measurement utilisation is growing when an organisation is climbing up to higher maturity levels in software process capability. Finally, we contribute with a 6-level *Measurement Utilisation Capability Model* (MUCM). During the present study we have constructed a new model that has been tentatively evaluated. The result requires further research and more extensive validation. The MUCM provides guidelines for organisations that aim at achieving a higher maturity of their measurements and operations.

⁴ During 2001, the Committee Draft (CD) for this standard was under ballot. In October 2001, a Final Draft International Standard (FDIS) was prepared.

Chapter 2.

Modelling the environment of measurements

In this chapter, we present a measurement framework and the environment of measurements by describing the software development process model used in the context of this study. In order to understand what is measured we also present the product model.

2.1. Modelling software development

Measurement is reasonable only when it is associated with existing models. We need a model for the *software development process* with certain phases and milestones where we can establish the measurement points. This process is normally used in *projects*, from which the measurement data is usually collected. In order to understand phasing and work results, we come in with a number of concepts that define *project process*. We also need to explain which products we actually measure in the context of the processes involved. Therefore, a structural *product model* is discussed in Section 2.1.3 below. As a case example, we focus on the models, which are used in the AXE10 product development (Ericsson Telecom, 1996). In Figure 2.1, we distinguish between a *project process model* (upper part) and a *software product development process model* (lower part).

2.1.1. The project process model

Products are developed in projects, which have a fixed budget, time-limited duration and temporary organisation. Telecommunication suppliers usually run several projects in parallel; we have a multi-project environment. To run big projects we need a consistent project management method called *project process*. The project process is a common method to administer different types of development projects, either hardware or software. The *project process* at Ericsson is based on the company wide process model having transition points called *tollgates* (TG) between different phases of the development project (Ericsson Telecom, 1994). Between the tollgates there are points at which certain sub-events happen. These events are called *milestones* (MS). At first, it is necessary to explain these basic concepts relating to Figure 2.1.



A *tollgate* (TG) is a superordinate decision point at which formal decisions are made by the project sponsor concerning the aims and execution of the project.



A *milestone* (MS) is an intermediate objective that defines an important measurable event in the project and represents a result that must be achieved at a given point. *Milestone review* checks the completeness of intermediate results of the work at different stages of the development process. Milestones link, for example, the software *work models* to the project model.



An *internal* or *external release* denotes official publication and delivery of products. In our case example in Figure 2.1, e.g., the *internal release* of the products is marked with the status PRA (*Product Release A*) when their specification, design and test (Function Test) is completed and all corresponding documents are available. The *external release* is marked with RFA (*Ready for Acceptance*) when the products are delivered to the customer and are ready for acceptance.

The model is divided into four phases, which are prestudy, feasibility study, and execution and conclusion phases. In Figure 2.1 we can see the numbered TGs as important decision points at the end of each phase.

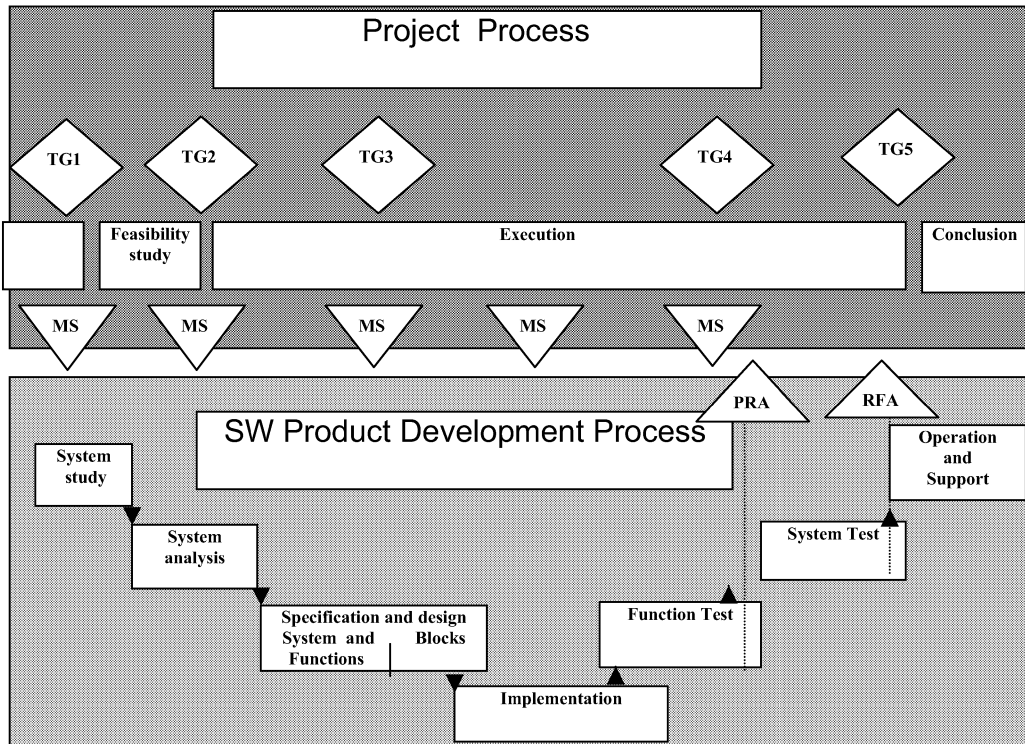


Figure 2.1 Project process and SW product development process models

The *Prestudy phase* assesses technical feasibility based on requirements and needs of customers. During this phase a set of alternative solutions is formulated and rough estimates of schedules, effort, etc., are made. Prestudy is finished at TG1 with a decision on start of the feasibility study.

The *Feasibility study* aims to form a good basis for the forthcoming execution project. During the feasibility study phase, the alternative solutions, their consequences and their ability to fulfil requirements are further analysed. As a result of this preparatory activity, a well-established *project plan* with goals, strategies and a defined organisation is available. At TG2 meeting, a decision is made whether the execution of the project is started or not.

The *Execution phase* aims to run the project as planned in order to meet the project goals and customer's requirements. There are several checkpoints to control the project during its execution. For example, in a software project, there is TG3 and the corresponding milestone between the design and implementation activities and also other TGs/milestones.

TG4 is a point in time to decide on use of the final results and hand-over to the customer. TG5 meeting makes a decision on project conclusion.

The *Conclusion phase* is a very important activity to record the experience gained and to suggest measures for improving the project model and the processes. A typical output from this phase is the *Final report*.

2.1.2. The software product development model

Software development needs a specific development model with carefully defined *phases*. Here the process used is an application of the traditional waterfall model (Royce, 1970) with specific phases and transition points. The software development model (lower part in Figure 2.1) defines a process that is divided into six phases which are the system study, system analysis, design, implementation, Function Test, System Test and operation. Between the phases, intermediate results are checked by using an *inspection* method. In order to provide as a good basis as possible for the next activity, any defects are corrected before transition to the next phase.

- ▼ *Inspection* is a well-organised and formal method, in which a group other than the author examines software requirements, design results or code in detail. The objective is early detection of faults and violations of development standards. The method is generally used as an integral part of modern software development process to find defects during design before release.

The *System study* analyses, clarifies and evaluates requirements. It determines if the existing software products are affected or if new system architecture is needed to fulfil the requirements. In general terms this phase is called Requirement Analysis or Requirement specification. Typical output documents are an implementation survey and a requirement specification. This phase also outlines a firm *design base*. The *System analysis* further analyses the requirements on new functions and describes the influence on the design base. This phase provides technical input for making a decision at TG2.

The *Specification and design* phase is needed in order to specify the functions, e.g. their operation and maintenance interfaces, in detail. Signal interfaces are designed and functions are described. The implementation of functions in the next level in our product hierarchy is designed in the block design phase including a detailed flowchart for the block. TG3 is the point in time that marks the beginning of the implementation phase. At TG3 all function specification and design, and block design documents are ready.

In the *Implementation* phase the designed *software units* are implemented as source code (i.e. coding). At Ericsson we call these units *implementation products*, and we really use these products as measurement objects in our measurement model in the following chapters. The implementation phase also covers *Basic Test* where the *software units* are tested individually by the designer in order to remove possible coding and design faults. After Basic Test, the *Joint Test*, the interaction test between the hardware and software controlling the hardware, is performed.

The *Function Test (FT)* phase includes a well-prepared activity during which the purpose is to verify the functionality of the developed new and modified software in a simulated or the target environment. Normally, it is planned in parallel with design, which means e.g. specification of test cases for each function. The Function Test box in our Figure 2.1 denotes the actual execution phase of the test cases. A *Trouble Report* is written, and registered in a database, concerning each fault found. At the end of this activity a result report, called *Test report*, is written including a careful analysis of all defects and faults detected. Function Test thus provides data for quality measurement purposes. After correction of errors in source code of the corresponding implementation products, the phase is completed and the products are set to PRA status and internally released for System Test.

System Test (ST) is a test of the entire telecommunication switching system. A selected application, in our case for example a local or international AXE10 exchange, is the test object. This test includes a wide test of functional requirements. Tests are designed to verify fulfilment of system requirements, such as capacity, maintainability, robustness and stability. Faults are corrected and reported correspondingly. After completion of the test, the developed products reach RFA status meaning that the products are good enough to be delivered and accepted by the customer.

During the *Operation and Support* period the customer receives support as needed and the discovered faults that were not detected in earlier phases are corrected.

2.1.3. The product model

Telecommunication switching systems are very large and complex, and therefore a well-structured product model is needed. We also need to explain which products we actually measure in this context. Figure 2.2 illustrates a hierarchical model for the product structure. A typical switching *system* consists of several *subsystems*. A *telecommunication exchange* tailored for a specific market is a combination of these subsystems. Similar *functions* (e.g. traffic control, charging functions) are grouped together in one subsystem. The functions are implemented in *function blocks*. Each function block is implemented as *function units* either in software (SW), in hardware or both. A software function unit is typically a computer program composed of *statements* (or lines) written in a high-level language or of instructions written in an assembly language.

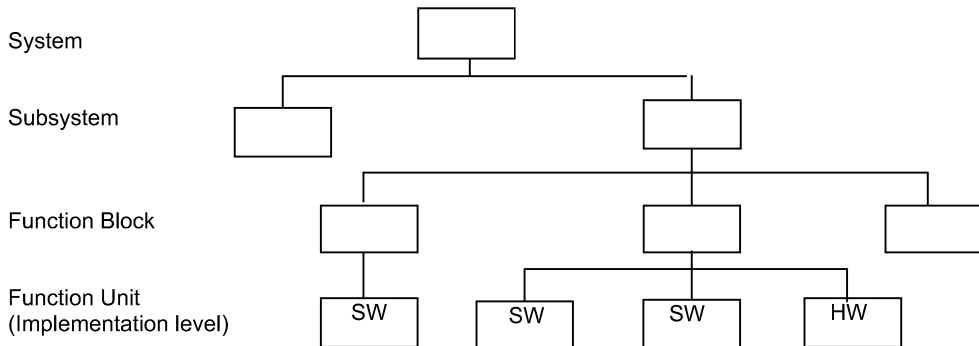


Figure 2.2 Product model

In this study, the word *product* denotes these implementation units on the lowest function unit level in the hierarchy. In order to follow general terminology used in other related studies, we simply call this product a *module*. The modules in the analyses and statistics under the succeeding headings are thus computer programs, which have a measurable *volume* (i.e. *size*). For example, in case of the present study, the modules are written in a high-level language called PLEX and non-commented source statements (NCSS) characterise the volume of each module. This is an issue that typically is defined in a measurement manual (Ericsson, 1995). In developing software for telecommunication switching systems, existing (old) modules are normally used as the base. Therefore, we will make a distinction between a *new module* and a *modified module*. A new module is a module that only contains new source code while a modified module contains new, changed and old (i.e. re-used) code. In this context we use either the term *total volume*, or *new and modified volume*. The total volume includes all statements of the module, both new, old (i.e. unchanged) and modified ones. The new and modified volume includes only new and modified statements and it is calculated by multiplying the total volume with *modification grade*. The modification grade expresses in per cent to what extent the new software module is new and modified, i.e., tells how much new and changed code the new software module contains.

2.2. Modelling measurements in the software development

How should the measurement activities and other quality-related issues work together? We can illustrate this by the “Quality machine” shown in Figure 2.3. It is presented here as a simple representation for a quality *measurement model*. There are “meter displays” which are needed all the time to keep control *during* the design and also to show the actual (final) quality *after* completion of the design. A more detailed introduction to different items of the Quality Machine is given in Appendix 2.

THE QUALITY MACHINE

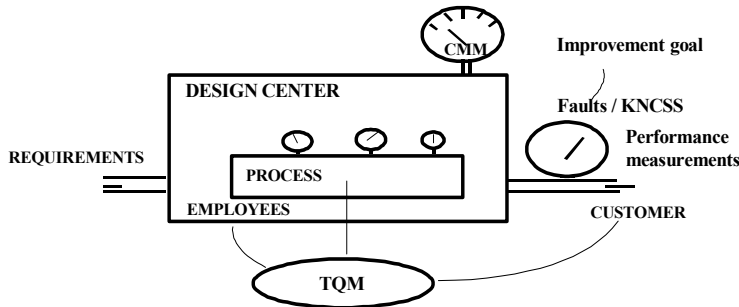


Figure 2.3 The quality machine for a design center

Modern thinking is demanding indicators and established measurements of each process to evaluate their adherence to process definition, performance and the effects of process improvement actions.

The *Capability Maturity Model (CMM)* "meter" is measuring the internal maturity of the entire software process. In order to get the CMM "meter" to display as high maturity as possible (out of five maturity levels), also some improvement in measurement practices is needed. Requirements for achieving levels 4 and 5, concerning measurements, are therefore analysed in Section 4.1. It is expected that high maturity that is indicated by the CMM "meter" should lead to lower fault density (F/KNCSS) and higher customer satisfaction.

TQM, Total Quality Management, provides tools for root-cause and statistical analyses and thus relates closely to the utilisation of measurements. TQM is described in more detail in Section 4.4.

However, in order to get the "machine" (like in Figure 2.3) running we need to characterise measurement activities in a more detailed level. The following framework (Figure 2.4) illustrates wider modelling perspective we take in this thesis.

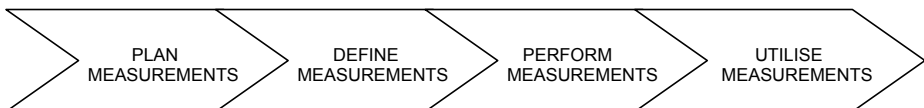


Figure 2.4 Measurement model framework

Models for planning and definition of measurements are discussed in Chapter 6. A case example of an existing model to define and perform measurements is found in Section 2.3. Renewed models how to perform and utilise measurements are presented in Chapter 7.

2.3. Existing quality measurement process

2.3.1. Presenting a practical process example

The simplified model in Figure 2.3 does not specify exactly when and what is measured. In order to keep the measurements running, some kind of measurement process, as shown in Figure 2.5, is a necessity. Here we present an example of an older process outline that has been in use for many years prior to the present study.



In Figure 2.5, Definition of Data and Results represents Measurement development activities for establishing measurement methods and tools. Development (either basic or further) starts from goals stated by the management, from new needs to see the influence of improvement actions or from other ideas to change the metric system. Development leads to accepted, well-defined and documented definitions for data and results.

Input and Data collection

COLLECTION OF MEASUREMENT DATA

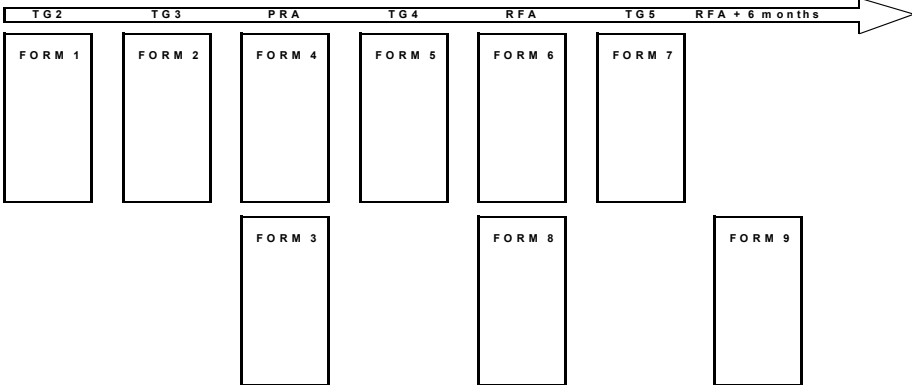


Figure 2.6 Data collection flow

Events in collection represent the review points, called *Tollgates (TG)* and *Milestones (MS)*, in the Ericsson's project process. Each rectangle visualises a specific Data Collection Form. Examples of *what* data, project-related or product-related, is collected on each form can be found in Appendix 4.

This process has been easily understood by design organisations and it has worked well to a great extent. However, people have sometimes ignored data events that should have been filled in forms. For this reason some "hunting" and control by the operator has been necessary.

Definition of Data and Results

The collected data is well defined by using a specific form in order to achieve consistency. *Measurement Data Definition (MDD)* is used for each collected type of data to explain briefly, for example, its purpose and method of implementation. *Measurement Result Definition (MRD)* provides a standardised way to describe all results to be presented to the user and their essential details, for example, their purpose, way of presentation, goal figures and hints for analysis. Similar types of definition templates are found elsewhere in the software industry, for example, in ESPRIT projects (Pulford, 1995).

Database administration and Result calculations

A dedicated person, the Measurement Operator, who has received training for using of the database system, performs administration. The measurement operator generates the graphs and *result calculations* needed by the management. Depending on feedback given by the organisation, some recalculations are sometimes necessary (e.g., due to faulty or lacking/delayed input data).

Management Review, commenting and Output

In the *Management Review* (in accordance with the ISO 9001 standard), the measurement results are studied and analysed. Observations regarding how the goals are met, findings like good news, any jumps in trends, possible reasons for deviations etc. are discussed. This meeting has a lot of manager's expertise to draw conclusions. In this forum, measurement results thus provide facts to make decisions on *improvement actions*.

Quality reports are prepared quarterly at each department and reported to upper organisational levels. Many kinds of listings for *organisation* and *projects* have been useful when analysing the actual quality product by product within line or project. A printout that has been very popular during past years has been the "*Department manager report*", where the key figures are presented on one page (see Appendix 5).

From the utilisation point of view the *output* part is very important. All instructions from the Definition of Data and Results phase are collected into the *Measurement Manual* that is distributed inside the organisation. The utilisation of this manual is sometimes neglected. If it is not followed, then a risk arises towards the quality of data and results.

2.3.2. Evaluation of the existing process

The model in Figure 2.5 is very simple and it does not describe activities for how to plan measurements. Neither does it tell anything about steps taken to define measurements. It is in a great deal collection and data administration oriented, an issue that is also well documented in the related PQT manual (Ericsson, 1995). However, the existing process is based on filling in a number of paper sheets and their manual distribution. The receiver of all these sheets has been the local "measurement operator", who should also validate and store the data. The process is quite centralised. Therefore, the system does not involve everyone in the organisation, a fact that in some cases might cause lower motivation.

When the existing measurement process was launched, the companies followed a standardised process model (see Section 2.1) in their software development, representing CMM level 2. Measurements were closely tied to the software development model. Therefore, measurement activities that are required on higher CMM levels are not represented in the model in Figures 2.5 and 2.6. Consequently, the activities for utilisation are not included, as well as a feedback loop to the

actual software development process is missing. The existing measurement process does not cover the use of results for quantitative analyses and project control. Neither does it include any visible feedback loop for evaluation of metrics. There is thus a good reason to look into CMM requirements considering measurements on higher CMM levels – an issue we discuss both in Sections 4.1 and 6.3. For this reason we aim at expanding this older process to cover our wider modelling perspective considering all four parts, plan, define, perform and utilise measurements (Chapters 6 and 7).

Chapter 3.

Experience from present software quality measurements

In this chapter, we present existing quality measurement activities. In the first section, we discuss aspects of present measurements elsewhere in industry. Next two sections are devoted to our case enterprise, Ericsson. Here we will have a look at the existing conceptual framework, and give three case examples of presentation and analysis of measurement results. Next, in Sections 3.4 and 3.5, we focus on experience concerning positive aspects, difficulties, and needs for company specific new measurements. Section 3.4.5 serves as an illustration of industrial experience of obstacles in linking measurement results to the design process, giving perspective to the Ericsson specific discussion. Finally, we discuss measurement relations to customer satisfaction.

3.1. Measurements at some industrial companies

In this section the main quality indicators used in three other companies, Raytheon, HP and LORAL, are surveyed in order to see what measurements exist and how are the results being analysed elsewhere.

3.1.1. Raytheon

Experience from Software Process Improvement at Raytheon Electronic Systems has been reported in a study report of Software Engineering Institute (SEI) of Carnegie Mellon University (Haley, 1995,1996). The main quality indicators used at Raytheon are *Cost of quality*, *Overall product quality*, and *Inspection metrics*.

Cost of quality is measured as conformance costs which consist of appraisal (e.g. reviews, inspection, testing) and prevention activities. A third component is the cost of non-conformance, which mainly include rework. Yearly trends of these costs, as percentage of project costs, have been followed up since 1988. Raytheon has succeeded in reducing rework costs to 20% of their original value. *Overall product quality* is measured as defect density in the final product. Defect density is expressed as number of Software Trouble Reports per Kilo Delivered lines of Source code (STRs/KDSI). Project defect densities are combined and monthly weighted averages are computed. Defect density versus time curve since 1988 shows significant improvement of averages. Data collected over 7 years period shows an improvement from an average of 17.2 STRs/KDSI to 4.0 STRs/KDSI (77%). *Inspection metrics* applying statistical process control were born in their stepping up improvements towards a CMM level 4 process. Data from detailed design and code inspection is collected. Defect detection rate, as action items per KDSI, is measured and analysed. Statistical analyses have shown that review speed (KDSI per hour), review package size (KDSI) and preparation time correlate with new code defect detection rate. By improving these three key factors it is then possible to improve the efficiency of the inspection process. Use of established control limits helps to influence on variability of the process. This statistical information has been used to derive some guide values for these driving

factors. For example, for an adequate preparation time Raytheon recommends a minimum of 5 total man-hours per KDSI.

3.1.2. Hewlett Packard (HP)

HP is a company that has been able to present experience from a company-wide measurement program (Grady, 1987) and from practical usage of metrics for project management and process improvement (Grady, 1992, 1993).

HP has used the GQM approach to develop the following nine *business management graphs*. The first two measurements relate to the goal: To improve software quality. Defects reported from any source during first 12 months after release are counted and divided by the number of non-comment source statements. Correspondingly, a graph for pre-release fault density as Defects/KNCSS in tests is used. The next four graphs are measure schedule accuracy, utilisation of best practices, productivity and defect repair efficiency. To maximise customer satisfaction, the last three graphs show open critical and serious trouble reports, mean time to fix defects (in days), and weekly hotsite status (urgent customer problems). Each graph includes a brief textual explanation with headings like, goal, question that this graph helps to answer to, and an explanation for interpretation of this graph. So, it helps the management to understand the way of the presentation all the time. Relating closely to the topic of the present study, a "Usage of software metrics" is presented in (Grady, 1992, Figure 15-10) showing the metrics, who is to use them and for what. For example, Defects/KNCSS is used for certification process by management to see fulfilment of company 10x-improvement goal⁵. Ericsson has selected similar approach in the ESSI improvement program (see Appendix 1).

Defect measurements have been expanded to include *categorisation of defects* (Grady, 1996), for example, by origin, type, mode, defect severity, phase of creation, and phase of detection. Defect occurrences in categories are measured and the most commonly occurred defect types will launch a root-cause analysis. Their method to analyse whether a defect was found in the same phase as created or in later phases (which means increasing cost to repair), is interesting. HP has used both post-project root-cause analyses and analyses as part of the continuous process improvement cycle approaches. These defect categories resemble the classifications used at Ericsson especially in Fault Analysis and Prevention (FAP) method. *Complexity metrics* are based on McCabe's cyclomatic complexity that is a count of the number of different logic paths through a module. Some correlation studies have recommended that the modules having complexity values greater than 10...14 are more defect-prone. However, Grady states that the acceptance of this metric has been slow. Anyway, HP has a tool that provides not only the values, but also a graphical representation of the code flow. HP engineers found this visualisation much better than pure values and the images are used to help in controlling the complexity. A similar tool at Ericsson is called CAPCO.

HP is well known of their extensive use of *Inspection metrics*. Typical attributes and metrics are shown in Table 2.

Table 2. Examples of inspection metrics at HP

Attribute	Metric	Used for
Effectiveness of inspection team	Total number of defects / total hours	Comparing different verification techniques and usage of resources
Defect density	Average number of defects found per page or line (defects per KNCS for code inspection)	Indication of the quality of the work products under inspection
Inspection rate	Number of pages or lines per inspection meeting hours	Assuring the optimal quality of inspection meetings

Based on measurement data collected from inspections, HP has also studied some typical guide values for attributes mentioned above. There are not many guiding values presented in the literature. However, in HP study (Grady, 1992), an optimum value at one HP division for code Inspection rate as 200-300 lines of code per hour, was recommended. Naturally, the guiding values are dependent on

⁵ For example, to improve HPs product post-release defect density by a factor of ten in five years

the type of the object to be inspected. Measurement results from inspections are also used at HP for determining ROI (Return on Investment) i.e. to calculate cost savings in engineering man-hours. It is worth to mention that HP has also measured adoption of inspection method as percentage of projects using inspections and as percentage of documents inspected (Grady, van Slack, 1994). Because the quality of inspections might also be different in different sites, the "extent of adoption" metric also includes inspection process maturity. Maturity definition is based on a CMM like five level models. HP's Software Engineering Systems Division has published some experience on implementing CMM level 2 (Lowe, 1996b). Some correlation between CMM efforts and attributes like quality, lead time, precision, cost have been obtained after passing level 2, as reported in the SEPG'96 conference (Lowe, 1996a).

3.1.3. LORAL

LORAL Space Information Systems is a company located in Houston, in the US developing critical SW systems for NASA space applications (e.g. Space Shuttle). This company has already achieved maturity level 5 in SEI/CMM assessments, so it is interesting to survey their measurement related issues here (Lee, 1994). LORAL has collected detailed statistics on error rate improvement concerning early detection (inspection) errors, independent verification (testing) errors and production errors since 1983. *Error rate* is defined as errors per KSLOC (kilo source lines of code) in new and changed code. As the focus is on early detection of errors, software quality measurements by using a few simple indicators are emphasised. These include the following.

$$\text{Early detection} = 100 \cdot \frac{\text{Major inspection errors}}{\text{Total errors}} \quad (\%)$$

$$\text{Process error rate} = \frac{\text{Valid errors}^6 \text{ Pre-Delivery}}{\text{Volume in KSLOC}}$$

$$\text{Product error rate} = \frac{\text{Valid errors Post-Delivery}}{\text{Volume in KSLOC}}$$

As an overall process measure, the following measure is used.

$$\text{Total inserted error rate} = \frac{\text{Major inspection errors} + \text{all valid errors}}{\text{Volume in KSLOC}}$$

More than 85% of errors are found in inspections. LORAL does not pay so much attention on testing but inspection. Especially requirements are carefully inspected by designers, testers and by a customer representant from NASA. During a ten years period, 1983-1993, LORAL has succeeded to decrease product error rate very close to zero. On the other hand, the intention is all the time to decrease total inserted error rate. Trend curves for both rates are presented as measurement results of evolution. In order to achieve low error rates LORAL uses error cause analysis and metrics. LORAL uses software error information systematically to correct and improve the process and thus prevent defects. Therefore a careful collection of data classified in detail is included in each Defect Report (DR) and analysed in causal analysis meetings. Defect elimination process does not only take steps to remove defects, but also to remove the root-cause of defect and to eliminate process escape deficiencies. As an interesting final step LORAL even searches and analyses other products where a similar escape might occur. The collected historical information is also used for quality forecasting in order to make error estimates.

⁶ Documented Discrepancy Reports

3.2. Examples of existing quality measurements at Ericsson

We start this section with a description of current measurement process and concepts. Second, we look into a few most important product quality metrics and process performance indicators.

3.2.1. Examples of existing product quality indicators

Faults per product

The number of faults per product is used as the most common collected quality indicator. With this primitive measure one is able to detect both low quality and high quality products. Data that until now is being collected relates to faults found, not to faults predicted. It does not express how many faults are remaining in the product. However, as it is used in present measurements, it is considered to be a *known fault* that is most concretely visible during testing.

Fault density per product

As products are of different size, then a normalised metric, *fault density*, is used for expressing observed quality (or, actually, lack of quality)

$$\text{Fault density} = \frac{\text{Number of faults}}{\text{Volume}}$$

where the *volume* is defined in KNCSS (kilo non-commented source statements) for HLL (high-level language) products or in KPSW (kilo Program Store Words) for assembly products.

Definition of KNCSS for volume at Ericsson differs from NCSLOC (non-commented source lines of code) that is commonly used in the software industry. The definition includes only statements (executable) which influence on functionality of the program. Variable definitions (declarations) are excluded. When a statement consists of several lines, it is counted only once. KNCSS resembles HP's way to define size of code as the number of non-commented source statements. Fenton (1996, p. 247) mentions that HP's definition is the most widely accepted. In fact, it is not as easy to compare sizes of two software products as lengths of people in centimetres. Implementation of the same function by two programmers may differ a lot in code length, because an efficient programmer is able to write a program with less statements. For further reading there are some good literature references (Levitin, 1986). Fault density is sometimes incorrectly called fault intensity. Philosophising about this type of problems may lead to an unfinished discussion about counting and fairness. For this reason, Ericsson has in 1992 established KNCSS to be used consistently and company-wide as volume measure for PLEX programs (Ericsson's PQT Manual). A tool for counting of KNCSS is also available. At Ericsson, fault density is measured for three specific phases:

- Function Test (FT),
- System Test (ST), and
- the first 6 months after external release to the customer (6MO).

Furthermore, the results for all the three phases are usually presented together. Using the existing reporting tools, it is possible to select the result figures for a particular collection of products (e.g. new or modified) or for a particular organisation or project.

Non-faulty products

A positive quality indicator is defined as follows:

$$\text{Occurrence of faultless products} = 100 \cdot \frac{\text{Number of faultless products in test or in operation}}{\text{All product that have passed test and 6 month operation}} (\%)$$

In order to learn about good quality, those products that have been faultless, especially during all the three phases (FT+ST+6MO), are identified and listed from the database. The reasons that lead to a faultless product are analysed. This is important in order to acknowledge the quality work of the designers involved.

3.2.2. Examples of process performance indicators

There are a lot of activities that are repeatable. When improving their quality we should focus on process quality issues. In the following, we describe a number of process-oriented measures at Ericsson world.

Quality seen after the end of the design process based on mean fault density

Let us assume that Function Test, the first extensive independent test phase, is in general thoroughly done. Then the *mean fault density* measured can be used to characterise the quality of the previous design process. Thus, the mean fault density for a given organisation can be defined as the total number of faults found in Function Tests divided by the sum of the total volume of products which the organisation has been responsible for (i.e. developed and released during a certain period). This is a good indicator of the overall process performance.

Precision

The quality of the entire development process is characterised, among other factors, by *delivery precision*, which can be defined as the percentage of delivery units ready on time. This measure can be applied to any type of released software units.

Quality in Trouble Report and Correction handling

Quality of Trouble Report and Correction handling process is measured with three distinct figures:

- Backlog of unanswered Trouble Reports,
- Percentage of answered Trouble Reports within time limit (e.g. 90% within 2 weeks), and
- Percentage of faultless corrections.

Results, which have, during past years, not been systematically stored into measurement database, but that are usually available in project and Test reports, are as follows:

Test detection rate

Useful quality metrics for the Test process is the *Fault removal rate* as Faults/man-hours or as Faults/Testcase. An essential matter in traditional testing of software is to find faults. Testing never guarantees that no faults are left. Final indication of *Test detection rate* can be seen afterwards by using a simple percentual capability metric:

$$\text{Test detection rate} = 100 \cdot \frac{\text{Faults detected in tests}}{\text{All faults detected in both tests and operation}} (\%)$$

In the equation, "all faults" denotes the sum of faults detected in the Function Tests, System Tests and during the first 6 months after external release.

Inspection detection rate

Some projects have reported inspection data in Quality reports. An example for a process measurement concerning inspections is *detection rate*, which is defined as follows:

$$\text{Inspection detection rate} = 100 \cdot \frac{\text{Faults found in inspections}}{\text{All faults}} (\%)$$

In the equation, "all faults" denotes the sum of faults detected in the inspections, Basic Tests, Function Tests, and System Tests and during the first 6 months after external release. In fact this metric represents a kind of early detection rate similar to LORAL's early detection rate (see Section 3.1.3).

Examples of other measurements regularly reported

Table 3 contains five other measurements characterising software design.

Table 3. Examples of measurements

Attribute	Measurement
Quality costs in SW design	Percentage of quality man-hours / all design man-hours
Occurrence of risky products	Ratio of "stinker" modules based on statistical control limits
Occurrence of top quality products	Percentage of zero fault modules
Duration of projects	Lead time in man-months
Productivity	Volume in KNCSS / man-hours spent

3.3. Case examples considering the presentation and usage of results

In the previous section we defined some indicators which have been used in practice. The intention in this section is to give a few examples of how to learn about results derived up to now. This is discussed below in the light of three case examples showing the way of graphical presentation and the usage of measurement results in Quality reports (described in Appendix 4).

3.3.1. CASE 1. Fault density in Function Test

In the Ericsson's quarterly Quality reports, created by the quality manager and distributed to the line management, the following trend curve (Figure 3.1) has been presented.

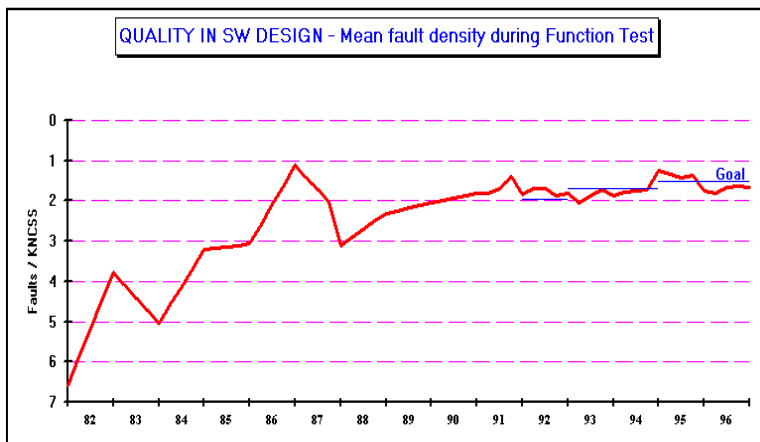


Figure 3.1 Mean fault density⁷

The curve represents a "rolling average" (or "moving average" as it is also known in statistical quality control (Montgomery, 1991)). After termination of each quarter, the mean fault density for the four preceding and finished quarters is calculated and the curve is updated. The method is suitable when only a small sample of statistic material is available during each particular quarter. Thus, the "expected" result for the on-going year is visible all the time. On the other hand, the curve "hides" the short-term statistical variations. The results are regularly analysed by the management.

⁷ Note the decreasing scale of Y-axis showing high quality to go upwards

Until 1987, there were many years of positive quality improvement, but e.g. rapid expansion and movement into new areas of software development caused quality to turn downwards. As a conclusion, actions were taken, for example by implementing:

- Yearly quality improvement programs,
- Extended fault analyses,
- Tool and method improvements,
- Training “just in time”, and
- Planning of quality assurance activities (e.g. inspections).

This resulted in a slowly improving quality. In early 1990's the trend has continued, still with some statistical variations. The quality during the past few years has proved to be quite constant. Thus it indicates that the process has become mature enough and the organisation has succeeded to use the process in a proper way.

3.3.2. CASE 2. Classification of modules by the number of faults

We can classify a sample of modules that passed their Function Test during a certain period (in this case during 1992-1994, a bigger sample of material included in Figure 3.1) to know how many faults are detected. The graph in Figure 3.2 represents the pareto of modules by number of faults found in Function Test.

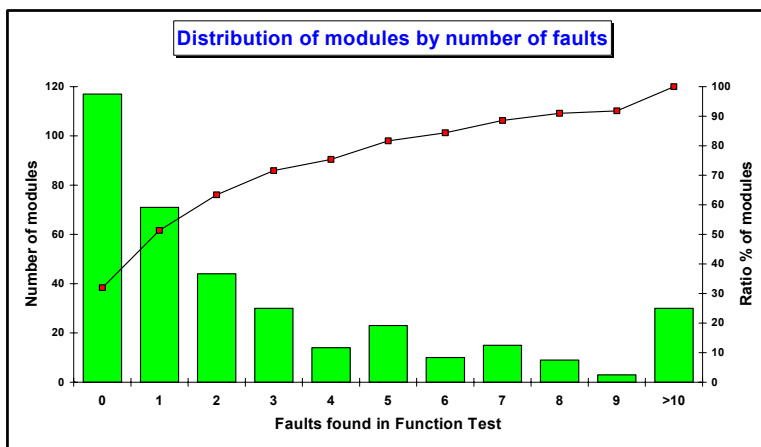


Figure 3.2 Pareto of modules

It clearly shows that 60 % of modules are of excellent quality (0 or 1 faults) and about one fourth has high number of faults (> 3). There are 30 modules (out of 326 modules, i.e. less than 10%) in which more than 10 faults were found.

As an example of actions taken by the management, an initiative was taken to produce a list of low quality modules from the database in order to analyse in more detail whether the modules were really bad or not. Correspondingly, a list of faultless modules was decided to be published because there is a good reason to use such information for rewarding purposes.

3.3.3. CASE 3. Fault density for first 6 months after delivery

Figure 3.3 shows one of the most frequently used graphs of those defined in the original version of the PQT Manual (Ericsson, 1992). Fault density for the first 6 months is a typical indicator of final quality.

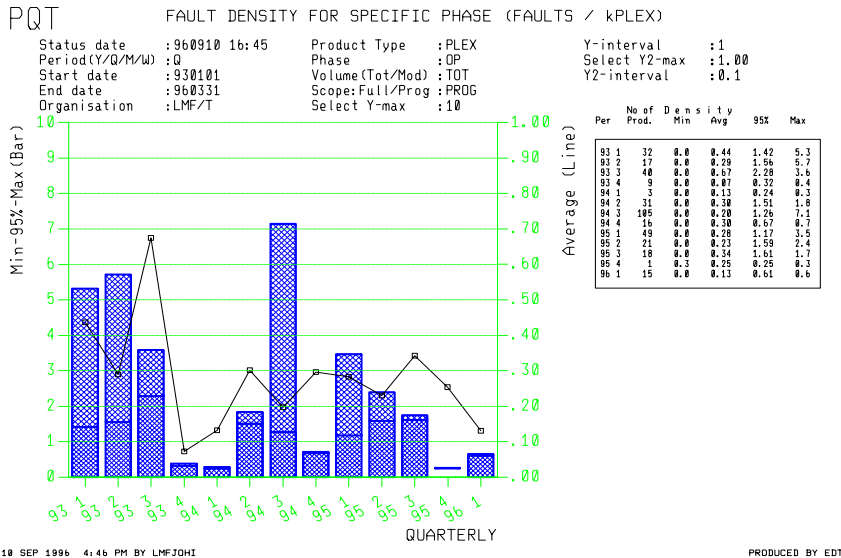


Figure 3.3 Example of a PQT graph - Mean fault density

The graph includes a collection of PLEX modules that passed their first 6 months after delivery to the customer since 1993. It is most useful when a significant sample of statistical material per period is available (e.g. several tens of modules per quarter).

The square-marked line indicates the average fault density as faults per kPLEX statements (see the scale on the right y-axis). This average, as presented here on quarterly basis, is also used to see the fulfilment of improvement goals (e.g. ESSI goal mentioned in Section 1.2). The bars referring to the scale on the left y-axis indicate maximum and minimum fault density. (Two y-axes are needed because the maximum values are usually greater than averages). The line between the two parts of each quarterly bar visualises the 95% fractile (0,61 for 96 quarter 1). The upper part of the bar thus represents the 5% of modules that have the highest fault densities.

It can be seen that the trend is decreasing even though there are quarterly variations. The modules that represent the highest fault densities are easy to list from the database in order to investigate and to take remedial actions.

3.4. Lessons learned at Ericsson in Finland

Since the measurement tradition has been long, some early experience and impressions are available on measurements (listed in Appendix 6). In connection with this study, some recent experience from management and quality people has been further surveyed. Positive experience from management (Section 3.4.1) includes aspects from measurements in early 1990's (Antman, 1993 and Hirvensalo, 1993). For improving the measurements, we performed a few expert interviews in order to know what has gone well, and what difficulties and problems have been experienced. The main argument in selecting interviewees was that they represented two major areas of software design, and that they had a close contact to measurements and quality. Two senior designers (Jauhiainen, 1996, Panula, 1996) have especially supported the experience collection.

We used a questionnaire consisting of seven specific questions. The questions concerned information distribution and awareness of existing measurements, strengths and weaknesses of measures, confidence in measurement results, their usage at departments, and obstacles to using. The rest of questions included suggestions for improving measurements and knowledge about driving factors and the dependencies influencing the results in a development organisation. Experience gained from management and quality personnel is discussed in Section 3.4.2 because it contains aspects based on numerous contacts with management and our own experience within quality management (e.g. as

a member of R&D Division's quality co-ordination network). The comments that were received from experts are discussed in Sections 3.4.3 and 3.4.4. Finally, we discuss the issue of obstacles separately in Section 3.4.5 in order to define improvement objectives.

3.4.1. Positive experience from management and quality personnel

Reviews of measurement results in management and expert groups raise new questions: "What, Why, How? What are the real causes behind?" etc. For example, the following concrete questions have arisen: "Are the most faulty modules also fault-prone in the field?" and "Are the non-faulty modules in tests still faultless in the field?". These types of questions can be considered as an indication of a desire to learn more. Sometimes the questions, as specified above, need a separate study. For this reason, the present study includes a deeper analysis of these two questions (see Sections 5.4.4 and 5.4.5).

Perseverance in systematic data collection has made it possible to use statistical methods to learn more from results. When the organisation has defined what data is a reasonable object for gathering then the collection activities are to be performed toughly and consistently in order to get good enough samples of statistical material. Without toughness, year by year, project after project, the material is imperfect and less reliable for statistical analyses.

Regular measurements have ensured that history is recorded in accordance with ISO9000. Quality records and maintained databases help to trace the quality and to see long-term trends. Measurement results have been considered as an important part of "Quality records" which are a repository required by the ISO9001 standard. Recorded history has helped to assure that the specified quality level has been achieved, the trends are known, and also the quality of those software modules that are re-used can be traced.

We get some support for the benefit of measurement as we mentioned in Section 1.1: "Decision-making and design is based on facts instead of opinions". Measurement results have provided facts and hard figures, for example, in baselining and goal setting (e.g. fault density objective values). The department manager report (see Appendix 5) shows a way of using yearly objective values in connection with measured figures. In addition to this "A4 pager", the measurement system generates more detailed calculations for projects and organisational units and thus provides knowledge for analysing what has happened during software design.

Management has been able to observe improved quality awareness due to measurements. When measurements are going on, or even when they are being planned or started, this has a positive effect on quality and result awareness. This has been experienced very early when the first measurements were started, but also later when new measurements have been launched.

Measurements provide a better possibility to give feedback about the work. For example, measurement material has made it easier to identify top quality software modules, to reward good success in projects, and to encourage people to become even better in their design skills.

3.4.2. Difficulties experienced by management and quality personnel

First, the management thinks that the existing measurements have provided too narrow focus regarding the whole business within software product development. Prior to this study measurements have focused too much on observed fault density and PQT, not enough on customer and people satisfaction and process. Based on this experience we analyse customer satisfaction issues in Section 3.7.

Second, the way of presenting the results to the management has become a problem. Measurement results have been presented with too many details on upper organisational level - this has caused some tiredness in Management Reviews. Ways of highlighting the results for upper managers should be improved. Management's decision making problems have also appeared when upper level managers and lower level managers get together to discuss all measurement results within the R&D division. It has been difficult for upper level management reviews to make decisions based on results because analysed results from lower level organisation have been missing. This has also been a

scheduling problem. Root-cause analyses and conclusions from the influencing lower level organisations should be available before the corresponding upper level management meeting is held.

Third, the lack of management understanding came up in interviews. The quality manager also experienced this problem. Misunderstanding of calculation rules for graphical representations has sometimes occurred. In principle, the rules have been available in measurement manual but not enough known or not read at all by the managers. This has taken place, for example, in management review meetings.

As a fourth difficulty, quality people have experienced that data collection and report preparation requires time in order to get reliable results. When data has to be collected from different sources (time reporting, trouble report databases, and source libraries), partly manually, it takes time. Producing graphs and inserting them into Quality reports is time consuming. Thus, some improvement in data retrieval and visualisation is needed.

Finally, quality people have often faced problems with incorrect or missing input data. Data is not correct in the database because it has been collected carelessly. In some cases it has become obvious that data collection instructions were not read and therefore the data on collection forms has been incorrect. Input data is sometimes missing or not delivered to “measurement operator” in time at the event when it becomes available.

3.4.3. Positive experience from senior designers and experts

Concerning information distribution and awareness of existing measurements the experts answered that project managers are quite well aware about how quality is measured today. Quality measurements have been specified in Quality Plans of projects. Project managers and designers have regularly filled in the collection sheets. Measurement results have been presented in Final reports and in Quality report of projects.

Measurements are documented well enough. As there has been a Measurement Manual with consistent definitions of measures it was concluded that the documentation is good enough. The measurement practices have been covered in the Operational Binders of the departments. Measurements after completion of design are quite well in order. Design results are observed during Function Test, and Faults/kPLEX as a measure in Function Test is well known, accepted and used. Instructions and support have been available for collection.

According to experience, Trouble Report/Correction handling works well, and makes it easier to collect measurement data. At Ericsson, all Trouble Reports containing detailed information of each accepted fault and their correction documents were registered by the designer directly into a company wide database during the past years. Trouble Report and Correction measurements by using a new automated collection tool on workstations work better than the tool on a mainframe. On a workstation, it is easier to fill-in a Trouble Report e.g. by using templates and copy and paste facilities.

3.4.4. Difficulties experienced by senior designers and experts

The first difficulty has been that measurements are not well understood on the line management level. Senior experts have experienced that the meaning of some metrics has proved to be unclear to their line managers. In spite of well-documented practices, the line managers do usually not have enough time to read measurement definitions.

Second, the experts pointed out that measurement related training and information distribution is not enough available. The collected experience has shown that it is necessary to know more clearly the responsibilities of data collection, when and why to do it, what are the results used for, etc. We can draw a conclusion that training and dissemination should be a continuous process considering both new and existing people.

Next aspect, that has been experienced to be a major issue, is that not enough data is collected and used during design. For example, knowledge from inspections is not utilised. Minutes of inspection meetings would provide a lot of information about the phase of detection, the types of defects, the

rework effort, etc. Lack of a tool for planning and handling the inspection data has been one reason for this difficulty. As a suggestion for improvement the development of such an inspection-planning tool has come up.

Lack of “driving metrics” during design is a reason for why the measurement results are not used that much on department levels. A good knowledge of which parameters (or factors) drive towards good quality, which ones lead to bad quality is missing. Such software metrics relating to driving factors are not systematically collected. This is an important task because by using such metrics it would be possible to influence on final quality better.

Certain measurement difficulties have often appeared in small fast projects. In smaller projects the standard design process is not always perfectly followed. For example, use of tollgate/milestones may differ from standard process, which is mainly intended for big projects. This is a typical issue in local design projects. Measurements (e.g. PQT) are based on the standard process and therefore the data provider should make some adaptations and interpretation in the data collection phase.

As a question it was stated “How good is designers’ confidence in measurement results?”. So, we collected answers regarding data quality. There are negative aspects that lessen confidence in reliability of measurement results in some cases. Sometimes it has been difficult to get correct data on planned and actual dates due to unexpected changes in project situation. Another difficulty has been to know exactly when the customer has put the product into operation.

3.4.5. Obstacles of linking measurement results to the design process

In order to develop utilisation it is reasonable to find out obstacles of utilising measurement results. When interviewing experts we therefore introduced one question in the questionnaire concerning utilisation: “What hinders of utilising do you see at your department”. We derived five essential groups of issues based on received answers and on our own experience. Our intention is to take these collected views into account in Chapter 7 where a new utilisation model is created and utilisation process is improved.

First, matters like low confidence in results and doubts about reliability of some collected data items have lead to a negative attitude in some cases. There is a doubt that the measurement does not really reflect our own performance or that it includes foreign drivers from other co-operating parties. There also appears to be a lack of trust whether the metrics used are expressing the behaviours of the processes well enough.

Second, we face another attitude issue. Measurements are considered to be a mandatory activity, only initiated by the top management. For this reason, there is a lack of thinking like “the measurements are for us”. It is not seen as a help in improving one’s own organisation or one’s own work.

Third obstacle relates to understanding and awareness. Metrics are not understood well enough which makes it difficult to interpret results. Line management has not always been good in improving awareness about the meaning of different indicators. People are unaware about the purpose and benefits of each measurement within the organisation, because enough information is not spread (e.g. how the goals were met).

Fourth, the experts felt that the lack of measurement data from early phases hinders the understanding of couplings between final quality and real driving factors. It is difficult to analyse correlation because the available data is incomplete. Some existing metrics are too coarse for project planning purposes and for derivation of planning constants. According to an earlier study (Rautakorpi, 1993), several existing measured attributes (e.g. modification grade, man-hours, type of product) are inadequate predictors (regressors) for quality. As an obvious cause, the lack of instructions and rules that are derived from measured facts complicates the transfer of knowledge from theory to practice.

Fifth, other obstacles regarding statistical significance, time allocation matters and access to the measurement system came up. Too small amount of quarterly statistical material on department level hinders from drawing any general conclusions. Due to time allocation problems, not enough time is allocated for analyses and to thinking about conclusions e.g. in teams. Finally, we draw a conclusion that one big obstacle is poor multi-user access to measurement system/database. If the users at

departments are not able to produce their own statistics for deeper analyses, it makes utilisation much more difficult.

3.5. Needs for further development of measurements

3.5.1. Early needs to add company specific new measurements

Expectations from departments were collected by making interviews in autumn 1995 and inquiries in electronic mail in spring 1996 (Hirvensalo, Nygren, 1996). This summary also includes the answers to a paper form inquiry collected by the quality group. Both department and quality managers participated in this investigation. Development suggestions were also received from some individual designers. People working with testing have also given very useful input. The needs identified on different organisational levels are presented in Tables 4, 5 and 6.

Table 4. Department level needs from PQT perspective

Improvement area	Development need
Software metrics	High Level PLEX (HLPLEX) volume and modification grade measurements PQT measurements for C and C++ software Faults after end of the first 6 months in operation Faults tied to subsystem and function block level Trouble Report severity and consequences in service Requirement stability Process measurements, measurements for "fast-track" activities Further development of department specific metrics
Inspections	Inspection man-hours and remarks per process stage
Search of data from measurement database	Subsystem name as a search parameter Measurements per subcontractor
Connections to other tools and databases	Delivery date retrieved from Product Information Management database
Engineering and Installation support	Delivery precision for correction packages Subproject specific measurements Ratio of released and inserted Approved Corrections (ACs) Quality of Approved Corrections inserted to customer packages "In-Service Performance" (e.g. number of restarts)

High Level PLEX (HLPLEX) is a high-level design language by which the programmer specifies module functions. As HLPLEX was published after the introduction of the first measurement system, a need to add volume and modification grade measurements arose. Besides the wide use of PLEX language, some departments started software development in C and C++ languages. Therefore, new measure types became necessary. *PQT measurements for C and C++ software* needed an expansion in the measurement system to allow volume and modification grade measurements, and the corresponding output graphs. *Faults after end of the first 6 months in operation* reflect the expectation to get a longer measurement period, e.g. 7-12 months in operation, and even from 13th month until the end of the product life cycle. The motivation was that the existing 6-month period was experienced to be too short for covering a broad operation. Thus, the number of faults detected at the customer during 7-12 months might express the final quality better. *Trouble Report severity and consequences in service* means that the faults should be classified by their effects on "In-Service" Performance (ISP). The existing measurement data model did not allow any separate storing of faults having an effect on ISP, but all operational faults were stored in the same data field. It is necessary to count the *Faults tied to subsystem and function block level* because certain faults can not be related directly to an implementation module. The existing measurement system handles faults detected only in implementation modules, on which the majority of trouble reports are written. *Requirement stability* is an essential driving factor in many projects, and therefore needed. It is important background information in analyses to understand why the measurement results look the way they do.

The need for *process measurements* reflects expectations to get more detailed data collected from software engineering processes. Further development of *department specific metrics* is necessary,

because the departments often use their own measures that might differ from standard measures depending on the application area and project specific goals. A need to improve the *inspection* process came up, e.g., in ESSI and therefore new measurements were a necessity. At least, inspection man-hours and the number of detected defects (remarks) were needed to be included in the measurement system.

Delivery date retrieved from Product Information Management database (PRIM) automatically, instead of asking it from the customer support organisation, has frequently been requested. Delivery date to the customer means the RFA date (see Section 2.1.1) when the modules are delivered to the first customer. Product Information Management database is a corporation wide repository of all official product, software implementation modules. We prepared a requirement specification in order to create a new mechanism between PRIM and the measurement system. *Subsystem name as a search parameter* is one development need for the database system. If the subsystem is marked in the fault information, then it is possible to use such a subsystem name as a search key in the generation of results from the measurement system. *Quality of corrections* can be measured for example by counting the fraction of rejected corrections. *Measurements per subcontractor* was needed because many projects in our case company are geographically and organisationally distributed. Often, one department has the main responsibility for a project; the other organisations act as subcontractors. Then it is useful to know the subcontractor's quality figures.

The last improvement area in Table 4 concerns measurements in the Engineering and Installation support organisation that works in close co-operation with the customer. As fault corrections are delivered to the customer, the *delivery precision of correction packages* is important to be measured. "In-Service" Performance (ISP) measurements need continuous improvement to cover new systems and applications (See also Section 3.6). For example, the numbers of large and small restarts are measures that can be used.

Table 5. Needs on upper organisational levels

Improvement area	Development need
Reporting	Results from each department presented in brief on the upper level report Analysed measurement results from departments prior to upper level review
Process measurements	Measurements per process and per design object
	Quality related man-hours retrievable from all departments Man-hours for quality activities identifiable in the new time reporting system Improved method of reporting and visualisation of results
Productivity	"Functional" volume based productivity measurement
Customer view	Customer satisfaction
Measurement database and reporting tools	Transfer of the measurement system from the mainframe to the PC and WS environment Information System and InformationTechnology support for measurements Connections to and automated data retrieval from other tools Deletion of such collected data, which no longer is necessary

The first improvement area in Table 5 is reporting. Two development needs are based on management's expectations (in our case company). *Results from each department* should be presented in brief on the upper organisational level report. Division manager also expressed a strong need to *get analysed measurement results from departments* prior to the division level (upper level) management review. The need for *process measurements* was kept active, for example, by the requirements coming e.g. from CMM and ISO9001. Therefore, we discuss these requirements further in Chapter 4. The need to measure *per process and per design object* includes at least the measurement of man-hours consumed, lead time and detected faults.

Quality cost measurement worked fairly well in the existing local measurement system. However, when development environments, time reporting systems and applications started changing, also the quality cost faced new challenges. The improvement of quality cost measurement was also motivated by their increasing emphasis on the latest ISO9000 standards. First of all, the development needs included retrieval of quality related man-hours from all departments and making the man-hours for quality activities easily identifiable in the new time reporting system (e.g. by using specific activity

codes). A need arose to get all departments involved, to ensure that everything works after introduction of the new time reporting system, and to improve the method of reporting and visualisation of results.

Productivity measurement is an issue that the software community has discussed a lot during the past years. A need to define “functional” volume based productivity instead of “implementation” volume based measurement came up also in our survey. *Customer satisfaction* here does not only mean the satisfaction of an external customer. So, the development organisation needs to measure, for example, the orderer’s satisfaction with projects and products. We refer to Section 3.6 for the discussion of the satisfaction issue. *Transfer of the measurement system* from the mainframe to the PC and Workstation environments was based on the evolution of computer technology. New systems provide better facilities for handling, fetching, and visualisation of data. Information System and Information Technology support for measurements was expected and connections to and automated data retrieval from other tools are needed to ease measurements. Deletion of such collected data, which no longer is necessary, for example, collection of faults for a period of the operation date +1 month was a need on upper organisational level.

Table 6. Corporate needs (for renewing PQT)

Improvement area	Development need
New collection events	Need to collect data from the prestudy phase (at Tollgate 1)
Deviation metrics	Collection of planned man-hours in addition to actual man-hours
Inspections, unit test	Defects detected during code inspections and unit test
Quality	Source of faults during the first 6 months after external release
Data security	Specific read and write privileges
Search of data	Project specific data retrieval Product groups (e.g. subsystem) with automated queries
Tool improvements	Need for porting the PQT measurement tool to a new operation system environment Development of measurement support tools for software written in new languages Improved connections to other tools (e.g. to software production tools)

The first PQT system (see appendices 1 and 3) was based on a mainframe implementation and a limited set of data. After many years of wide use, we collected corporate specific needs for renewing PQT shown in Table 6. Based on feedback from PQT users we discuss both some new measurements and a few issues concerning tools.

Collection of TG1 date, when the Feasibility phase is finished, enables the measurement of full Time-to-Market. *Collection of planned man-hours* in addition to actual man-hours (that were originally included in PQT) was based on a need to get measurement results on planning precision in effort dimension. *Defects detected during code inspections and unit test* were needed in order to create new measurement points earlier in the R&D process. On the other hand, we pointed out that it tells more about the process if the fault profile considering all phases is available for quality analyses in the measurement database. *Source of faults* during the first 6 months after external release means that e.g. faults found by Ericsson and by the customer are clearly distinguishable. *Data security* was needed to assign read and write privileges, and to create more strict access rules than in the old system.

The need for *porting the PQT measurement* tool to a new operation system environment included the idea to get the system ported to workstations. Further Development, e.g. the measurement of volume and modification grade for High Level PLEX, Erlang and C++ software were necessary to define also in the PQT system. The need for further development of support tools, e.g. the measurement of volume and modification grade for High Level PLEX, Erlang and C++ software appeared. In Finland, an Ericsson R&D organisation that is responsible world-wide for certain switching system development operations expressed their need to use product groups (e.g. subsystem) in the automated queries. We included the search by product group in the requirement capturing of the new PQT system (refer to Section 5.4.10 to see a case example of using it).

3.5.2. Needs and expectations for utilisation of measurement results

There are several reasons and motives for better utilisation of measurement results. The aspects presented are based on collected expectations from the PP (Provisioning Process) manager (Lappalainen, 1996) and the Research & Development manager (Toivo, 1996) in Finland as follows:

- To increase the use of measured facts in practical decision making.
- To change attitude towards measurements to more active and to create a positive measurement culture.
- To use results in order to compare different subsidiaries and the corporation.
- To increase the use of measurement results in different usage applications.
- To understand what the results tell about process and its stability.
- To make the message of graphical representations more understandable by using proper case examples.
- To contribute to analyses and to conclusions drawing in order to learn to avoid faults and to get better modules for the customer.
- To get utilisation ideas “marketed” better on all organisational levels.

Surely, the expectations listed above are very challenging. Three first mentioned issues require strong management commitment and an active measurement team. Fourth issue implies many examples of usage applications like, management and project planning (e.g. prediction, progress), performance assessments (e.g. goal vs. outcome), and the analysis of the real effect of improvement (e.g. is the trend positive). Fifth expectation, “understanding of what the results tell about the process”; can be approached by improving measurement processes, for example by enhancing pre-analyses and background information. More understandable graphical representations are partly dependent on tool improvements and models for use of results. The issues presented in this section will be used as one of the guiding inputs for measurement processes and models in Chapters 6 and 7.

3.6. Relations to customer satisfaction

Customer's SW quality expectations, which are to an increasing extent expressed in requests for tenders, in supply and support contracts, etc., state mainly requirements for software maintenance and availability performance. The external customer is not that much interested in design quality metrics like, faults/KNCSS, but the frequency and severity of failures. Thus, measurement feedback from times between failures, effects of faults and correction times of the most critical faults should be considered better by product development, not only by maintenance personnel.

A minor fault may make the customer very dissatisfied - there are these types of experience at departments working at customer interface e.g. the Field Support Office (Saarinen, 1996). On the other hand, there are software faults which, if removed, do not lead to improved reliability. Such a fault does not necessarily lessen the customer satisfaction even if Ericsson is not able to correct it quickly.

So, measurements should be further developed to identify occurrence of such faults that really make the customer dissatisfied. Proportion of the faults that the customer has set on the highest priority and that are most urgent to correct, compared to other faults, should also be indicated at product development departments.

In order to know more about customer dissatisfaction reactions, a distinction should be made between customer generated fault reports and faults discovered internally at Ericsson. This need already came up in interviews considered in previous sections.

The effects of faults should be classified. For example, the failures (e.g. restarts) caused by a fault should be classified by using the new Fault Analysis and Prevention (FAP) code VISCON (VISible CONsequences). The faults leading to severe failures, like hanging calls, major/minor restarts, or total stops of an exchange, should be analysed in order to get to know whether the fault was introduced in the design project and how could such faults have been avoided. The fraction of this type of faults,

measured separately, would express the relationship between quality and customer satisfaction better.

It is most important from the customer satisfaction point of view to be able to improve In-Service Performance (ISP) and to find its closer links to development projects. Such driving factors that affect on ISP should be measured already during the project. Both preventive measures to achieve higher ISP and metrics of actual ISP in the field should be included in the future metrics.

It is also reasonable to perform specific customer satisfaction measurements (CSM), not only on company level but also after each development project, and to compare the satisfaction results to quality delivered from product development units. This should be analysed by the organisations involved in the projects.

Several customer-oriented external quality factors (e.g. reliability, maintainability and usability) are defined in quality models (e.g. ISO/IEC 9126). Customer satisfaction can also be based on evaluation of these types of quality factors (Kitchenham, 1996). Therefore, some relevant models are discussed in Section 6.2 as means of clarifying software drivers behind customer satisfaction.

3.7. Conclusions on present measurements

The existing measurements surveyed are focusing on results seen after the completion of the design phase. It is still important to continue these measurements in order to see long-term trends and the effect of improvement efforts.

Collected experience during the present study indicates that measurements, like fault density in Function Test, have been well defined, documented and accepted. Also some data items collected are no longer useful (e.g. faults found during the first month in operation, because the observation period was too short).

Negative experience relates mainly to practical difficulties in getting data collected, in achieving correctness of the data, in understanding and presentation of measurement results, in motivation and ways of selling of the measurements in the organisation, as well as in confidence in measurement results.

Our interpretation of these problems is that the purpose of the existing measurements, like PQT, has not always been clear enough among people. That is why the existence and argumentation of the following two types of measurements have not been well realised. First of all, the measurements assessing the performance of design operations and final quality of results are necessary. Second, the predictive measures that help to achieve the demanded performance and high quality of results are needed. In the next chapters, as a recommendation due to these problems, we also will point out the measurement customer concept - it might be different for these two types of measurement. Handing over the right measurement results to the right customer may lessen the confusion and misunderstandings.

However, the lack of systematic data collection from earlier design phases was felt to be negative factor both by the interviewed experts and senior designers. This was also stated to be a very difficult obstacle, which prevents people from linking measurement results to the design process. This is a reason why the relationships between final quality and real driving factors remain unknown.

The designers also experienced, as one remarkable obstacle for utilisation, that there is not enough time allocated for analyses. A lack of instructions and rules derived from measured facts was also found. So, this issue is one of the elements included in the utilisation model in Chapter 7.

In order to improve confidence in the data, even to the data that is already widely used; it is all the time a question of training of attitudes and of refining measure definitions.

One problem in utilisation of the present assessment type measures is the long feedback loop. The measurement results are not available during the ongoing project, but only during the next project.

As a most important issue for development of measurements, there is a need to measure faults slipping-through, man-hours, requirement changes, and lead times per process/activity in more detail.

There is also a need to include major faults found in inspections, man-hours used for these quality assurance activities and using this knowledge for analyses.

In this chapter we have also presented how the quality as seen by the customer can be focused better by introducing quite simple changes in collection (e.g. classifications of faults). "Selling of metrics" is an important issue. Good hints are found e.g. in work performed at HP (Grady, 1987, 1992) (and this is a key element in developing a utilisation model and an organisational infrastructure of measurement activities).

Chapter 4.

General requirements for analysis and usage of measurement results

In this chapter, we give a survey of general requirements to facilitate better usage of measurement results. The purpose of this survey is to analyse measurement related issues in the Capability Maturity Model (CMM), Quality System standards, Quality Award Criteria, and TQM in order to be able to improve measurements also from the process maturity perspective. The last section is Ericsson specific, discussing measurement aspects in the Ericsson Quality Manual.

4.1. CMM

The Software Engineering Institute (SEI) has developed CMM (Capability Maturity Model for Software, also called SW-CMM) at the Carnegie Mellon University in USA (Paulk, Curtis, 1993). An initial version was released, reviewed and used during 1991-92. Since then, new CMM versions have been published like, P-CMM for People aspects, SA-CMM for software acquisition matters, SE-CMM for considering systems engineering and IPD-CMM for integrated product development. The latest trend is to integrate multiple models to a CMMI. At the end of writing, a public pilot version (CMMI, 2000) was available. The final version has been released in 2001. CMMI is also closer to international standard for software process assessment (ISO/IEC15504, 1998).

CMM provides a consistent method to assess and evaluate the maturity of industrial SW processes, and provides a framework with five maturity levels. The levels define an ordinal scale for measuring process maturity and evaluating process performance in an organisation.

The Ericsson CMM Assessment Process (ECAP) includes a complete set of guidelines and rules to perform CMM assessments within Ericsson. Yearly, trained assessors perform a total of 20-30 assessments. Each design centre is assessed every 1.5-2 years to determine evolutionary improvement in maturity, if any. Ericsson in Finland was assessed first time in spring 1995. The maturity level on different departments varied, some achieved higher CMM level, some did not fulfil all stated requirements. For this reason, research & development division as a whole achieved level 1 only. During late 90's some Ericsson subsidiaries have achieved level 3.

According to the basic CMM thinking, a mature software organisation shall measure modules and processes and use the results for continuous improvements. In this way, CMM issues closely relate to the theme of this study, and the demands on high maturity levels are discussed briefly below. We discuss what CMM actually requires from measurements, result analysis and utilisation. Improvement of process maturity and measurements are closely related as shown in Table 7 (Fenton, Pfleeger, 1996, p. 89).

Table 7. CMM maturity levels vs. typical measurements

Maturity level	Characteristics	Type of metrics to use
5. Optimising	Improvement fed back to the process	Process plus feedback for changing the process
4. Managed	Measured Process	Process plus feedback for control
3. Defined	Process defined and institutionalised	Product
2. Repeatable	Process dependent on individuals	Project Management
1. Initial	Ad hoc	Baseline

Table 7 presents an early classical overview of measurements suggested by each level. What can and should have been measured on each level depends on *visibility* into the software process. However, the table does not show the cumulative nature of measurements i.e. each higher level also includes the lower level metrics.

For Initial level 1, baseline metrics should be collected as a starting point for improvements of maturity. Level 2 measurements focus on project management because the focus in general is on projects. Regarding characteristics in Table 7, we can add that the process is not dependent on individual but also on projects. On this level, repeatable process activities are known and stabilised, especially for estimating, planning and monitoring progress of a software project (SPC, 1994). Typical measures are software size, staff effort and costs. It is also possible to collect statistics on trouble reports.

Level 3, defined process, has an extended visibility to intermediate activities, because those subactivities are defined and their inputs and outputs are known. The organisation's standard processes are defined and institutionalised. All projects use an approved and tailored version of this standard software process. Measurements are mainly product-related; early product measures can be useful indicators of later product measures (e.g. used to predict quality of code).

As CMM level 4 means "Managed" process it requires that detailed measures of the software process and product quality be collected. A key area for level 4 is Quantitative Process Management. Data analysis and utilisation requirement is expressed in the CMM document CMU/SEI-93-TR-25 (Paulk, Weber, 1993, p. L4-1) as follows: "Quantitative Process Management involves establishing goals for the performance of the project's defined software process... taking measurements of the process performance, *analysing* these measurements and *making adjustments* to maintain the process performance in acceptable limits. The organisation collects process performance data from the software projects and *uses* this data to characterise the process capability of the organisation's standard software process. Process capability describes the range of expected results from following a software process".

The same CMM document stresses also that the quantitative process management helps to achieve control over modules and processes by narrowing the variation of their process performance to fall within acceptable limits. Because the process is measured and operates within measurable limits, CMM authors suppose that this improves predictability in process and product quality. As a conclusion, measurements on level 4 are to be used to *control* the process.

Even more advanced utilisation is needed on level 5, "Optimising", where the focus is on a continuous process improvement. An important key process area is Defect Prevention: "Defect Prevention involves analysing defects that were encountered in the past and taking specific actions to prevent the occurrence of those types of defects in the future. The defects may have been identified on other projects as well as in earlier stages or tasks of the current project. Defect prevention activities are also one mechanism for spreading lessons learned between projects" (Paulk, Weber, 1993, p. L5-1). Furthermore, defect prevention on level 5 involves trend analysis to track the types of defects encountered and to identify defects that are likely to recur. It also demands specific actions to prevent recurrence of defects.

Thus, measurement results from activities at level 5 are used to improve the process, by removing something, adding process activities, and changing the process in response to measurement feedback. In other words, measurements are to be used to manage *process change*. It can be concluded here that a high maturity level of the process also involves a high maturity of measurements.

The existing measurements at Ericsson mainly relate to CMM levels 2 and 3. Some new fault analysis methods are in use that support achievement of level 4 and 5 needs. Concrete measurements by

level are discussed further in Chapter 6. The FAP (Fault Analysis and Prevention) process (Lahtivuori, 1994), which was introduced at Ericsson in Finland in 1994, has been one step towards higher maturity levels. According to this process each fault report received is analysed by phase of detection, fault introduction, technical category, reason and visible consequences.

4.2. Extracts from ISO9000 quality management and quality assurance standards

4.2.1. General ISO 9001 standard

ISO9001, a standard for quality systems (SFS-ISO9001, 1994), does not directly state any requirements for measurements. However, such requirements are given indirectly, for example, "Quality Policy" means that the organisation shall define quality goals. Naturally, these goals are to be followed up and this happens by going through measurement results. At Ericsson in Finland (see Section 2.3.1) such a forum has been the Management Review because ISO9001 standard clearly states that fulfilment of quality goals is to be reviewed (SFS-ISO9001, 1994, Sections 4.1.1 and 4.1.3, p. 10).

ISO9001 has also other relations to measurement and handling of results when it requires that the need to use *statistical methods* for establishing, controlling and verifying of *process capability* and *product characteristics* shall be identified (SFS-ISO9001, 1994, Section 4.20 p. 26). These methods must be documented. Measurements and use of statistical techniques are motivated in the guidelines part of the ISO9000 standards: "Guidelines for Quality management and quality system elements" (SFS-ISO9004-1, 1994 Sections 18.3.4 and 20). Objective and accurate means of measuring quality achievements are recommended. Focus is also on human aspects to encourage personnel to improve quality and to provide recognition of good performance.

Finally, this guide mentions that documented procedures should be established to select and apply statistical methods, for example, to product design, prediction, studies of process capability etc. Several examples for statistical methods are given like regression analysis and quality control charts. This guideline also refers to a separate ISO Handbook for further information about use of statistical methods (ISO, 1995).

The new ISO 9001: 2000, emphasises quality measurements, analysis and improvement even more than the previous 1994 version of the standard. It focuses on measurements that are also relevant on higher CMM levels (e.g., usage of measurements to maintain and improve processes).

4.2.2. Guidelines for software work

For applying ISO9001 requirements and methods on software, a standard "Guidelines for application of ISO9001 to the development, supply and maintenance of software" (SFS-ISO9000-3, 1993, Section 6.4), is available. Its Section 6.4 "Measurement" takes into account metrics for both software products and processes. The standard admits the lack of generally accepted measures for software quality. However, it recommends that, at a minimum, some metrics that represent reported field failures and/or defects from the customer's viewpoint should be used. For product measures it recommends that quantitative measures should be used to manage the development process and for the following purposes:

- To report metric values on a regular basis,
- To identify current level of performance,
- To take remedial actions if metric levels exceed the target levels, and
- To establish specific improvement goals.

By process measurements the standard means quantitative measures of the quality of the development and delivery process. First, these metrics should reflect how well the development process is carried out in terms of milestones and in-process quality objectives being met on schedule. Second, the metrics should express how effective is the development process at reducing the probability that faults are introduced or that any faults go undetected.

For both types of metrics the standard stresses that levels are known and used for process control and improvement, not what specific metrics are used for. Furthermore, the metrics chosen for the process should have a direct impact on the quality of the delivered products. In this respect, the standard encourages finding correlation between process measures and product metrics and to utilise them in design (as is the intention also in this study). ISO9000-3 standard is not used widely within Ericsson group because similar issues are included in CMM assessments.

After completion of the present research, ISO 9000-3 standard has been further developed. ISO 9000-3 was revised in 1997. A new updated committee draft (CD) version to comply with ISO 9001:2000 was made available in July, 2001.

4.3. Quality Award criteria

There are several Quality Awards in the world, including the following:

- Deming Price (Japan)
- Malcolm Baldrige National Quality Award (USA)
- European Quality Award
- National Quality Awards, for example, in Sweden and in Finland.

A very good survey about their main elements is found in (Ollila, 1995). In the following we limit to a few main aspects of the Finnish Quality Award concerning requirements for measurement results and their utilisation. Each company is evaluated in 8 areas (SLY, 1996). Here the main concern is in the key areas that give the highest number of points, i.e., the following four areas:

- Customer orientation and Customer Satisfaction (250 points)
- Operational results (195 points)
- People development (175 points)
- Process Management (125 points).

In the Finnish Quality Award, *Customer* means mainly the external customer on the market. Award criteria involve determination of methods for satisfaction by using measurements. Usually the measurements and investigations are performed by impartial consults. The company must be able to show customer satisfaction results as key figures to see development trends and improvements in comparison to competitors. The results are to be classified in a proper manner by product and service areas.

Operational results are focusing e.g. on products, operations and economy. Product results clearly relate to quality characteristics, which are important in customer's decision making. Examples of key figures to be presented are performance, number of faults, lead times and delivery precision. Development trends must be shown in relation to goals and benchmarking data. Both internal measurements and field measurements are needed.

People development involves resource-planning, development of competence and satisfaction as well as the related evaluations and measurements.

The *Process Management* section of the Award assesses at first the processes for development of products and services. The response should include how the design specifications are conducted from the customer needs and expectations, and how efficient processes are derived from design specifications. It is interesting that the award criteria also include measurement plans for processes. Furthermore, evaluation of process management activities involves that measurement results are utilised in maintaining process performance in order to make decisions on corrective actions.

On the other side, there are key areas that give fewer points but are very measurement oriented. Such a key area is for example "*Information and analysis*" (75 points) concerning the whole organisation. Several examples of useful analyses are given like:

- Impacts of improved product quality on customer oriented key figures (e.g. customer satisfaction, market-share)
- Influence of people satisfaction on personnel turnover

- Time-based trends of central quality related key figures.

It is important to take into account that a measurement system is at place for collecting relevant information and to support decision making. As a conclusion, it is common for all these areas within the Finnish Award that the evaluations not only expect to meet a number of measurements, but that the results are analysed and utilised as a basis for improvements.

Ericsson in Finland is well on the way towards this - in 1996, the company received the Finnish Quality Award. The score awarded was roughly 500 points out of 1000. This was one of the highest scores among those companies, which participated in the award competition during 1994-96. Below there are some strengths perceived by the jury:

- Many kinds of practices for measurement of customer satisfaction exist.
- People satisfaction has been measured since 1993 and the results are used to carry out improvements.
- CMM is used to assess and improve software design process.
- Management has committed "in black and white" to the ESSI program.

The jury also perceived a few weaknesses:

- No improvement process exists for the measurement method of people satisfaction.
- Processes are not measured in a full coverage (scope).
- No measurement results are available regarding effectiveness of processes.

Our conclusion regarding assessment at Ericsson in Finland is that two strengths and all the three weaknesses relate directly to measurements.

4.4. TQM

TQM is an old well-known management method expanding quality to cover all processes and operations in a company. During past years it has been common to apply TQM to improvements of software processes, as seen in the literature e.g. (Arthur, 1993). TQM has three basic principles: customer focus, continuous improvement, and everyone's involvement. Additional principles, for example at Ericsson, involve process orientation and decisions based on facts.

TQM provides tools and aids to affect processes and products in order to improve quality. These tools help in root-cause- and statistical analyses and are known as 7 Management (7MT) and 7 Quality Control tools (7QC). The 7QC are the most interesting tools in the perspective of the present study and include the following: Pareto diagrams, Histograms, Bar charts, Scatter diagrams, Run charts, Stratification, and Frequency matrix. According to Goodman (Fenton et al., 1996), an important dimension for successful TQM program is measurement and control, i.e.; it is vital to prove that improvement occurs. This needs baselining, goal setting and demonstration through indicators and measurements in order to ensure that the goals have been met. TQM's focus on causal analyses using statistical tools and decisions based on facts needs continuously running utilisation of measurement and learning from experience. This is basically demanded if TQM is applied in a company.

4.5. The Ericsson Quality Manual

Ericsson Quality Manual (EQM) (Ericsson, 1995) specifies a requirement: "We use measurement information to control and improve our products and processes and to ensure that we meet the specified requirements and expectations of our customers (external and internal)". One of Ericsson's basic strategies for quality is "Decisions based on facts" (instead of opinion based decisions). Measurements and analysing the results provide knowledge for this decision making. In connection with Process Management the Quality Manual also states a recommendation that performance indicators and established measurements for each process should be used.

Other important measurements, which this manual requires, relate to Customer satisfaction and costs of poor quality. Continuous improvement is one of the main quality policies - the work that is based

upon the PDCA (Deming's circle with Plan-Do-Check-Act segments). In this circle, the measurements act as an essential part of the "Check" segment to assess real effects of improvement actions and effectiveness of solutions. The manual also specifies why the measurements should be related to Planning and Control of design and development projects: "Data is collected for evaluation and improvement of *planning constants*, such as volumes, resources, and time schedules". This is illustrated in Figure 4.1.

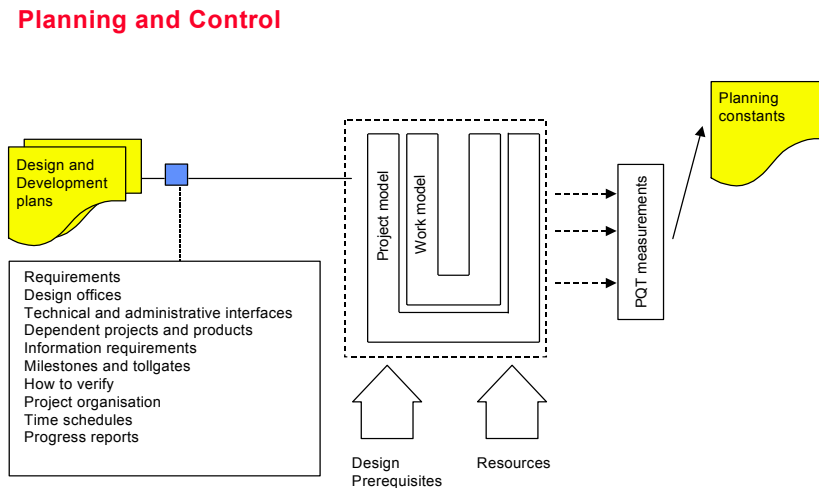


Figure 4.1 Use of measurements for improvement of planning constants

The latest version, called Ericsson Operational Quality Manual (EOQM) (Ericsson, 2001), requires the use of measurements as part of performance management and improvement. Performance measurements against business success factors imply the use of key indicators across a broad range of perspectives. The new manual fits to ISO 9001:2000 principles requiring e.g. process measurements. These measurements include measuring and analysing the results at key stages of processes as well as in-process measures helping to prevent problems.

4.6. Conclusions on requirement capturing

We have found obvious requirements and arguments for the analysis and utilisation of measurement results from CMM, ISO 9000, Quality Award Criteria, and Ericsson Quality Manual as well as from the customer. From the measurement point of view, a look into those sources has brought a few important issues to consider in development of new measures. Customer satisfaction and process measurements are important requirements captured from all methods discussed. All of them demand advanced measurements and their efficient utilisation.

For example, to fulfil needs for CMM level 3 we should introduce earlier measurement points for product data in order to be able to predict quality of code. Earlier product measurements (e.g. from similar projects) should be used for prediction of product quality produced in later projects. We should improve the measurement process to include a systematic way of studying and deriving useful planning constants. Inclusion of process measurements should be considered if an organisation aims at fulfilling level 4 requirements. It influences on the way of presentation of results because process variations should be visualised, and used for the control of the process to keep the quality in acceptable limits. Level 5 also requires that measurement results are used for thorough analyses, e.g., in order to find reasons of faults and to identify cause-effect relationships. In our development of measurements in later sections it is necessary to include analysis and modelling activities in the utilisation process, and to add a feedback loop to process management. As a conclusion we can state that a high CMM maturity level of the software process also involves a high maturity of measurements.

ISO9001 and its guideline for software work emphasises the importance of driving attributes that have a direct impact on the quality. Something important to be considered already in planning of the measurements is thus a survey of possible driving attributes (in Section 6.4.2 we therefore require some analysis to find the right driving attributes). It clearly stresses the use of preventive measurements in order to lessen the number of probable latent faults. ISO9000-3 includes a specific section for measurements and also gives hints for metrics regarding field failures. The standard guides to using the quantitative measures within the development and delivery process. As the standard encourages finding correlation, it gives us some support for our case studies in Section 5.4.

TQM elements like continuous improvement, everyone's involvement and a set of analysis tools encourage organisational learning. TQM provides useful tools to consider for analysing e.g. root-causes and presentation of results. According to TQM thinking, learning from experience and everyone's involvement can be taken into account by arranging feedback meetings on all organisational levels. TQM is included in Ericsson Quality Manual that also emphasises the use of measurement results for improving the planning constants and the use of measurements as an instrument for improvement (e.g. "Check" segment in the Deming's circle). Finally, we conclude that Quality Award Criteria specifies both customer-oriented measurements and measurement plans for processes as well as the analysis of measurement results as a basis for improvement.

Chapter 5.

New opportunities to use the knowledge from the existing measurement databases

In this chapter we survey a few past studies at first, mainly from Ericsson world, in order to see what correlation those studies have identified and what new chances they provide for further study in the present thesis. As a next step, three other related studies are introduced that have been going on during the present study. We have included a section presenting sixteen data sets that were extracted from databases. It has been necessary to limit our presentation to a selected collection of relationships and correlation studies. A number of case studies are done based on stated hypotheses that management has been interested in. Mostly these hypotheses are also interesting in general, like post delivery quality of those products that have been most faulty in design. Statistical methods are used to find out facts to prove whether each hypothesis is true or not. In case of positive correlation the utilisation topic of the work is covered by discussing under each case example about the possible use of the result. Finally, a special attention is paid on opportunities for prediction.

5.1. Related work

In Section 5.1.1 we first discuss about past related work covering studies at Ericsson in Finland. Next section, 5.1.2, concentrates on findings in a few studies performed in other Ericsson companies. Section 5.1.3 includes results from two closely related theses, which were completed during the present study at University of Linköping. Section 5.1.5 introduces a significantly related research that was run at Eltel in order to develop a fault content estimation model. The last section 5.1.5 includes two big research projects, PROFES and LUCOS, where also material from other than Ericsson companies was used.

5.1.1. Studies at Ericsson in Finland

An early attempt to use statistical analysis on the software products collected in the quality database is found in (Rautakorpi, 1993). The purpose was to see if any correlation between SW product parameters really exist, and to propose a statistical model in order to be able to predict the size of the product, the man-hour effort, and the number of faults. A sample of 177 software products developed in 8 middle-sized projects at Ericsson in Finland was studied. A few parameters related to the design process (e.g. number of signals, number of states, planned man-hours, volume, modification grade) were identified and a total of eight different correlation with quality related data (e.g. number of faults, fault density in Function Test) were measured. An example of an interesting finding in this study was a significant ($r = 0,70$) correlation between total number of signals and PLEX volume of modules as shown in Figure 5.1. A simple formula that could be used in prediction of the volume was derived. The more signals, the more the volume grows. Another result was that the correlation between man-hours consumed for design of the product and faults found in Function Test was observed to be quite high. Accuracy in prediction of man-hours was also studied. Accuracy between predicted and actual man-

hours was found to be very good. The accuracy is improving within the same project family as a function of time, so the study was able to show that the organisation has learned all the time.

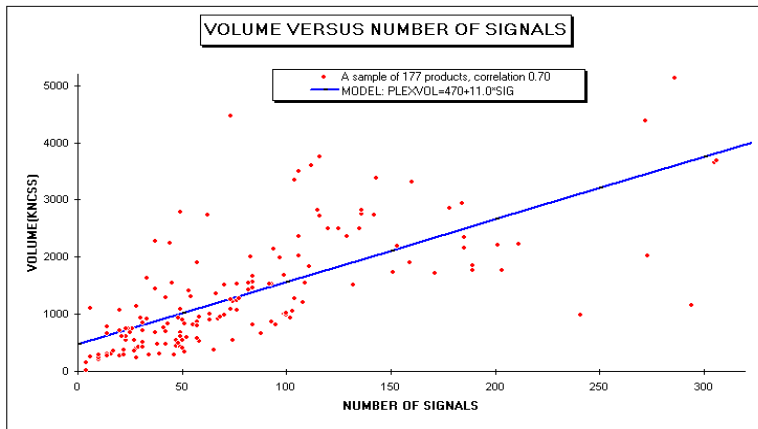


Figure 5.1 Relationship between volume and signals

According to this study several existing measurement data (e.g. modification grade⁸, man-hours used, type of product) are inadequate regressors for quality. For this reason these metrics are too coarse for project planning purposes and for derivation of planning constants. The limited scope of the study did not include other aspects (e.g. human factors, complexity, stability and effectiveness of inspections).

New chances of further usage of data from the databases

After this study a lot of new data has been collected from projects and there is a good reason to test whether the hypotheses are still valid when a larger sample of new material is used. The study was also limited to Function Test results collected at release. Now it is possible to study also post-release phases. In projects finished after this study, more accurate data is also available, for example, the number of new and modified signals, data from inspections and an improved product type.

5.1.2. Studies in other Ericsson companies

Ericsson Telecom (currently Ericsson Utvecklings AB) has run a few Master's Theses on quality issues. These studies concentrated on searching for design variables that would be good enough for prediction of error-prone software modules. One study (Johansson, 1993) of 4 geographically distributed projects within Ericsson collected the number of new and modified signals from 390 products. The product $S \cdot \text{SigFF}$ was chosen in order to study its linear regression with number of faults found in Function Test. The variable S is the size of the software unit in non-commented source statements (NCSS). SigFF , the number of new and modified signals, was collected from Function Framework (FF) documents that are produced in early phase of the software design process.

Another parallel study (Ohlsson, 1994) focused on predicting which software modules are likely to be error-prone. In this study, more than 130 modules were available. The researches found, among other things, a few new measures that could be used for prediction. For example, a complexity measure collected from SDL-objects is known before the coding phase. Modified McCabe Cyclomatic complexity was another example of many candidate predictors, which were included in the regression analysis and usefulness evaluation. An indicated fact from usual Ericsson projects was that approximately 80% of faults found in Function Test are focused on only 20-30% of software modules. Some further discussion based on these studies has been later going on, for example see (Johansson, Nord, 1995).

⁸ See Section 2.1.3 for definition of modification grade.

New chances of further usage of data from the databases

The number of new and modified signals, which is already available from several big projects at Ericsson in Finland, is an interesting data for further study. The correlation of this variable to faults found after test phases could also be investigated. If some correlation is found then this information could be used during design to concentrate corrective actions on these most error-prone products (see Section 5.4.6).

5.1.3. Studies at University of Linköping (Liu)

In his Licenciate thesis, Ohlsson (1996) has studied methods for early identification of fault-prone modules. The study was based on empirical data, i.e., design metrics and fault data from two successive releases of switching system software developed at Ericsson Telecom. A number of design metrics collected from design was analysed by applying a multivariate analysis. Ohlsson considered 21 design metrics from literature and from earlier studies. Identification of predictive design metrics that can help to make prediction as accurate as possible was done by studying correlation between two and three different variables in turn. The researcher succeeded in finding combinations that can be applied as prediction models at the completion of design. The best models identified were also validated by using a new approach, so called Alberg diagram (Ohlsson, 1994). The diagram shows the accuracy of the best prediction models in relation to the real fault data collected from software releases. As a conclusion, the best predictions can be based on number of signals and number of decisions. Both variables are known after completion of design (before coding). The analysis indicated that it should be possible to identify the 20% of the modules representing 50% of the faults. In his doctoral thesis, Ohlsson (1998) has further studied and refined his methods aiming at effective fault prediction. Empirical studies from real projects provide useful material to compare with results of the present study. An integrated fault analysis method is very similar to Ericsson's FAP analysis mentioned in Section 4.1. Based on this empirical data, some related quantitative analysis results are discussed in the light of 14 hypotheses (Fenton, Ohlsson, 2000).

New chances of further usage of data from the databases

Alberg diagram that is already implemented in the quality measurement system at Ericsson in Finland could be utilised to see how the diagram differs for different phases, FT, ST, and 6 months. When counted signal and decision data from the latest projects becomes available, it is possible to study dependencies between these variables and fault densities. The study result should be used to check whether these facts could be introduced in new/ongoing projects to improve quality. Studies of other driving factors, like project, process and resource could also be started.

5.1.4. Studies conducted by Ellementel

Since 1984 Ellementel (currently Ericsson Utvecklings AB) has run research projects concerning software reliability prediction models and experiments on software structure metrics (Rydström, Viktorsson, 1989). A research project carried out at Ellementel Telecommunication Systems Laboratories in co-operation with Telesoft Ab and Department of Communication Systems at Lund Institute of Technology (Lennselius, 1990) resulted in guidelines on how to develop a Software Fault Content Estimation Model. Data from Ericsson projects were collected to compare estimated fault densities with actual values. There were 7 industrial design projects involved in a total of 101 software modules. At first, these studies tried to find a correlation between ten software complexity and structure attributes and fault content. Complexity metrics derived from SDL description were included and correlation between each candidate metric and the number of errors were computed. The most promising variables, which positively correlated to the number of errors, were number of SDL symbols per module and number of unique signals sent to the module from other modules. The relations of this type of measures and the number of errors, and relations between coding time and each of the measures, was also studied. The complexity and structural analysis was proved to be a useful tool for identifying critical software products in order to be able to make process control actions in early phases of the development. However, the researchers concluded that differences in fault contents between projects could not be explained only by the software structure metrics. It is necessary to take into account both the development process and the software product to be developed. Studying also other factors affecting the number of software faults, like project ability and applied methods/tools

continued the research. Thus as a next step, these studies concentrated on such candidate fault drivers that likely have a major impact on fault content, for example, personnel capability and experience as well as project organisational factors. Guidelines on how to develop a software fault estimation model are given in (Lennselius, 1990). This model was also experimentally used at Ericsson in Finland. In the 1980's the systematic measurement methods did not yet exist, so the collection of data was troublesome and the samples of reliable material available were small. Therefore, the usage of these types of models at Ericsson was not broad.

New chances of further usage of data from the databases

Project distribution and size aspects can be utilised from the existing database. There are several data that unfortunately did not lead to a systematic collection. Project stability (e.g. stability of requirements), experience profile of project members and number of software signals would be useful and easy to collect from recently finished projects.

5.1.5. Other related studies

PROFES

PROFES (**PRO**duct **F**ocused improvement of **E**MBEDDED **S**oftware processes), a research project funded by EC, started in January 1997. The project that was divided into a baselining cycle and an improvement cycle was finished in September 1999. Ericsson in Finland is one of seven industrial and research partners co-operating with a few European University/research centres. The project's strategy was to link customer oriented quality factors to the characteristics of the software development process. PROFES also aimed to integrate goal oriented measurements (GQM), software process assessment (e.g. CMM, SPICE), Product/Process Dependency Modelling (PPDM) and the Experience Factory (EF) concept. PROFES addressed well exploitable results through the methodology, User's guide, guidelines to software process - product relationships, tool support and training material as well as application of the method in real projects. Project conclusion took place in October 1999. Results of industrial applications have already been presented in several conferences, see for example (Solingen, Derks, Hirvensalo, 1999). PROFES results are generally available (<http://www.ele.vtt.fi/profes/>).

Effectiveness measurement study

A research group at Ericsson in Finland has developed a model to measure the development effectiveness of AXE10 exchange's network signalling applications. The results are based on a finished research project "Effectiveness Measurement System" (EMS), which instead of traditional "implementation volume" (e.g. lines of code) approach, was looking into "functional volume" oriented effectiveness measurement. A study of several Function Point (FP) based metrics was done and a Functionality Increase Indicator was created. Calculation details for effectiveness were defined and Effectiveness baseline for software development within a subsystem was determined. The new attribute characterises the functional content of a project and the functionality increase indicator can be used for estimation of effort and improvement of effectiveness within this subsystem development. The project has improved readiness of using the new indicator in different design centres by deploying it in the form of training material, instructions, workshops and support.

Project controllability (LUCOS)

In order to find solutions to problems presented in Section 3.4 (e.g. lack of driving metrics during project) and needs for automated data retrieval, a new project was started in co-operation with Helsinki University of Technology (<http://mordor.cs.hut.fi/lucos/>) and, among other partners, with Ericsson's research group in Finland. The project was started during the present study and the duration was three years. The objective was to improve project controllability by defining a set of management indicators that could visualise the performance e.g. by using a web based control panel solution. Second, the focus was on automating data collection from existing data sources to a central database. The long-term objective was defined to support project prediction and planning, organisational learning and process improvement through an experience database (by storing more detailed information than during past years).

5.2. About material used as research object

The main intention of this section is to highlight characteristics, the amount, the type and the size of modules involved in our present study. Section 5.2.1 gives some facts and measured results from three real software projects at Ericsson. Section 5.2.2 visualises the time evolution of data collected into the database, the language and the size distribution as well as the degree of categorisation to new and modified modules. All selected sixteen data sets are presented in Section 5.2.3.

5.2.1. Examples of project characteristics

Characterising a new project based on past projects provides useful knowledge for planning. Understanding the profile of the current project to be launched helps to see how the project differs from other projects in key issues. A few typical sample constants collected from three projects included in the present study are given in Table 8.

Table 8. Characteristics of three projects

Characteristics	Project A	Project B	Project C
Modules			
Number of new modules	19	13	-
Number of modified modules	18	35	13
Total number of modules	37	48	13
Effort (total man-hours)			
Actual man-hours	75244	50711	46779
Planned man-hours	73175	51995	28800
Deviation from plan	2,8%	-2,5%	62,4%
Volume			
SW volume, Total (NCSS)	38008	50324	56275
Average size of module (NCSS)	1027	1056	4329
SW volume, Modified (NCSS) ⁹	21147	18638	5066
Average modification grade (%)	55,6	37,0	9,0
Number of detected faults			
- Function Test (FT)	105	120	102
- System Test (ST)	11	25	37
- RFA+6 months	14	22	11
Totally all phases	130	167	150
Fault densities (Faults/total KNCSS)			
- Function Test	2,76	2,38	1,81
- System Test	0,29	0,50	0,66
- RFA+6MO	0,37	0,44	0,20
Total fault density for all phases	3,42	3,32	2,67
Fault catching ratios			
Percentage of faultless modules in FT	45,9	35,4	15,4
Ratio % faults caught in FT	80,8	71,9	68,0
Slip-through RFA+6MO	10,8	13,2	7,3
Delivery Precision (%)			
RFA Planned at TG2 / Actual		93,7	96,7
Ahead / Behind		B	B
Lead Time (months)			
TG2 - PRA	13,5	9,3	8,0
TG2 - RFA	16,5	11,8	12,1
Productivity			
Total volume / man-hours	0,51	0,99	1,20
Modified volume / man-hours	0,28	0,37	0,11
Quality (Fault) Productivity			
- FT Faults / kilo man-hours	1,40	2,37	2,18

Numerical figures alone do not provide complete characterisation of projects. However, when discussed together with background material (like project specifications, Final reports, Test reports)

⁹ See Section 2.1.3 for definitions of modified volume and modification grade.

the measured numbers provide basic material for further analyses to understand whether the project went well or badly. As a conclusion, some comments on differences between projects A, B, C are given, for example:

Project A and B includes also new modules, project C only modified existing ones. However, we can observe that in project C the total software volume is large in spite of low degree of modification. Project C should modify big modules. Deviation from plan in effort is very small in project A and B. In general big software projects tend to exceed planned effort and delivery delays are usual risks (Jones, 1994, 1996). However, in project C, the effort was underestimated, deviation from plan is significantly higher. One reason is that requirements changed heavily during this project and caused extra hours. In all three projects, 70-80% of faults are detected in Function Test. Fault slip-through¹⁰ to first 6-month period is low, approximately 10%. Short lead time in project C seems not to have a negative impact on operational quality, the slip-through and fault density figures for project C are even smaller than in other projects.

5.2.2. Examples of module characteristics

In this Section, we discuss what kind of modules were available for the present study from the existing measurement databases. Figure 5.2 shows the material that is available for our study: more than 1000 modules on local company and more than 10000 ones on corporation level. Distribution of module data by programming language is presented. Size distribution and change of size by time is also discussed.

Evolution in data collection

Historical trend of stored data from PLEX modules is presented in Figure 5.2 to demonstrate that material has been collected regularly.

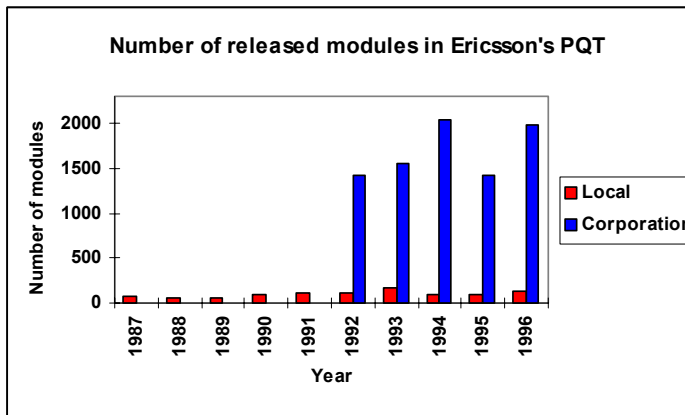


Figure 5.2 Flow of collected PLEX modules

Distribution by programming language

Programming language distribution of new and modified SW modules stored in Ericsson-wide PQT (global) database since 1992 is shown in Figure 5.3. In addition, Ericsson in Finland has maintained local database since 1987 containing data from 953 PLEX and 120 assembly (ASSEM) modules, which are shown in local column. Modules generated from High Level PLEX (HLPgen) represent an approach that has been used in several big projects. High Level PLEX (HLPLEX) is a high-level design language by which the programmer specifies module functions. There is a code generator

¹⁰ Slip-trough expresses in percent the number of faults discovered after a certain phase compared to the fault count for all phases.

available generating ordinary PLEX source code from HLPLEX specifications. In specific applications the corporation has also used a proprietary version of Pascal language (ERIPA).

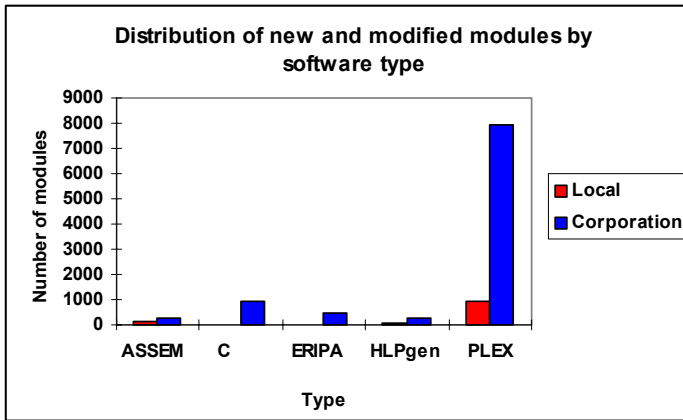


Figure 5.3 Module type distribution

Size distribution

Figure 5.4 shows the distribution by size class in columns for PLEX modules < 10 kilo statements representing whole of Ericsson. A curve shows the cumulative percentage of modules. From this percentage curve one can read that sizes less than 2000 statements represent approximately 50 % of modules.

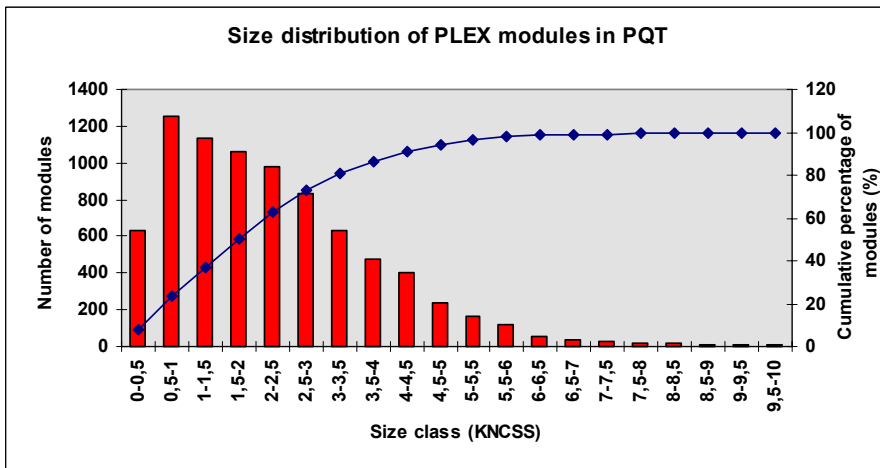


Figure 5.4 Module size distribution of PLEX modules

Fraction of new and modified modules

As the modules are either new or modified, this categorisation is presented in Table 9. This sample does not include modules without reported modification grade (HLPLEX generated modules are also excluded). It is usual to modify some existing base product; therefore only about 30 % of all modules are completely new. At local level the percentage is even smaller, 19%. However, one can observe that during late 1980's Ericsson in Finland has developed a higher proportion of new modules. This is probably due to the fact that new application areas were started in 1987-1991.

Table 9. Categorisation of modules

Module category	Number of Modules (Global PQT) 1992-1996	Percentage	Number of modules (Local) 1987-1991	Percentage	Number of modules (Local) 1992-1996	Percentage
New	2193	29 %	130	33 %	104	19 %
Modified	5442	71 %	267	67 %	439	81 %
Totally	7635	100 %	397	100 %	543	100 %

A study regarding size growth of modules

The local trend (Figure 5.5) shows that the average size of PLEX modules has grown from 1800 to 2500 statements during past years. The spread of size has been smaller in early 1990's. In 1996, 50 % of modules belong to size category between 900 and 3800 statements.

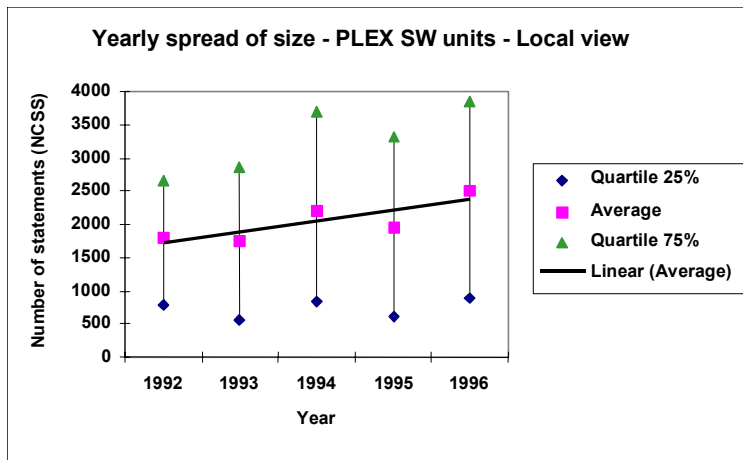


Figure 5.5 Yearly spread of module size - local trend

Summarising we can conclude that the modules to be studied are mainly written in PLEX language, the average size varies between 1800-2500 statements and modification of old modules is used to a great extent (only 20-30 % of modules are fully new).

5.2.3. Selected data sets

In this section we introduce data sets that are used for case studies in Section 5.4 including grounds for their selection. Samples of data fetched from Ericsson's databases are composed of sixteen project and product data sets explained in Table 10. Several different data sets were selected because certain data was not available in all projects or quality and consistency of data had to be ensured.

Table 10. Data set descriptions

Data set	Description	Section
1	Data from 16 finished local major projects having verified man-hour, milestone and fault information available for the whole life cycle of the project (see Table 8 for typical project characteristics). However, the sample is small composing of projects on different application areas.	5.4.1, 5.4.2, 5.4.8, 5.6
2	Data from 1405 new modules stored in global database. A random sample. Because of observed data inaccuracies, included are only new modules, having a design effort > 100 man-hours. Excluded are also HLPLEX generated modules because their code is not comparable to manually produced code.	5.4.2

Data set	Description	Section
3	Data from 127 new modules stored in local database to be used to study relationship between size and design man-hours of modules. Local modules were selected because quality of recorded man-hours in local database was proved to be particularly high.	5.4.2
4	Data from 25 finished local major projects. Selected were as many projects as possible having recorded dates available to determine planned lead time for module implementation and test.	5.4.3
5	Data from 242 modules having passed their first 6 months after external release and developed in all type of projects during 3 successive years in 1990's.	5.4.4
6	Data from 499 modules that passed their first 6 months after external release. A random sample from local database.	5.4.5, 5.4.11
7	Data from 7991 modules that passed their first 6 months after external release. A random sample from global database.	5.4.5
8	All modules developed locally during 5 successive years, covering also modules developed during the present study. The selection was based on programmed graphical facilities available in the PQT system. The selected time period was Function Test phase. This emphasises the importance of identifying low quality products as early as possible.	5.4.6
9	6055 global modules (of which 436 local ones). A large sample covering all delivered modules in global database to see the improvement trend during four years.	5.4.7
10	All 137 modules having an additional technical type stored in local database were selected because this data was only recorded and used in 3 mobile telephony projects on local level.	5.4.9
11	57 modules representing a certain product group recorded in the expanded global database. Product groups were collected during the present study within a specific application area. All available modules that passed 6 months their first 6 months after external release within the area were considered.	5.4.10
12	499 modules local products released during 4 successive years and passed their first 6 months in the field. A data set of 5618 other modules from global database is included in a study of modification grades.	5.4.11
13	Selected were 17 projects including 170 modules with their 12 months passed since external release. Only local projects of different size, in average 10 modules per project, were studied because their history was best known.	5.4.12
14	A separate sample of 1946 answered and finished Trouble Reports registered for modules with their first 6 months in the field passed. A two-year time 1997-1998 span was chosen.	5.4.13
15	A sample of 4 local subprojects representing 229 modules. The data set closely relates to the same collection of modules than in Data set 10. One additional project finished during the present study is included. Size of projects varies between 40-90 modules and 50-100 KNCSS in total.	5.4.4, 5.5.1, 5.5.2, 5.5.4
16	A subset of selected 8 new modules from the last finished project belonging to Data set 15.	5.5.2

Ten out of sixteen data sets relate to local projects because some data was collected only locally and information about background of these projects was best available. The rightmost column refers to the section(s) of the case study where the data set is used.

5.3. Method of the case studies

5.3.1. Stating and handling of the hypotheses

In our case studies in Section 5.4 and 5.5, a number of hypotheses are stated and handled systematically in the following way.

Background

Each heading includes motivation and argumentation how the hypothesis came up and why was it selected. It is discussed whether the hypothesis is fully new or does it appear in the literature. The most common literature references for equal or opposite claims (if any) are given and highlighted here.

Title of the hypothesis

The hypothesis is formulated in one sentence.

Facts

Measured facts based on provided data set are presented in a form of graphs and tables. This section includes statistical analysis and interpretation. Indications whether the material available provides evidence for or against the hypothesis is obtained.

Conclusion

This heading is used to conclude whether the hypothesis is true or not.

Use

Under this heading we provide hints for how this result and facts can be used by the organisation that is responsible for the production of software.

Discussion (optional)

Whenever applicable, possible error sources and general validity of results are discussed here. If appropriate, the result is compared to related research in the software engineering community. In case that the result significantly differs from other people's results, some possible reasons are discussed.

5.3.2. About the research method

In this study, methods and tools are used to analyse relationships that exist between different variables collected from the software. The purpose is also to prove the statistical significance of results. Common statistical basics are followed (Milton, 1995 and Oakland, 1990). Statistical analysis has been done based on a number of functions in Appendix 7. Examples of the most commonly used methods are the following:

Correlation analysis

Pearson correlation coefficient (r) is used to express linear relationship between two variables, x and y . It is based on covariance and standard variances. Correlation coefficient can vary between $(-1, 1)$. The closer the value of coefficient is to $r = 1$, the closer the two variables are positively correlated.

Regression analysis

The method is needed to indicate how variable Y depends on X , when a very good correlation between the variables has been obtained during the correlation analysis. Least squares method is used to calculate the coefficients for a straight line representing linear *regression*.

Trend analysis

In this study several time-based trends are investigated. There are series of results, which show upward, or downward tendency.

Rough Set Analysis

The technique is used to find the best possible combination of several independent variables, which have an impact on a certain dependent variable.

5.3.3. Views of presenting information in case studies

The intention in case studies is to present information from four different main points of view:

- Time based trend view where the results are presented to see the evolution in time e.g. as year by year trend curves.
- Organisation view that presents the results by using organisational unit (or a group of units) as a selection criteria. The view allows comparisons between organisations. The organisation view either involves material from many design centres, (e.g., *global view* concerning the whole of Ericsson) or *local view* that involves material concerning only one local design centre (e.g., Ericsson in Finland).
- Project view where the results are presented projectwise to compare results between projects on different types of software.
- Module view presents the results per module, per a certain *group* of modules, or the selection is based on the type of modules.

The view of presentation used is mentioned in Section 5.3.4 for each case study (Tables 11...14).

5.3.4. Selected relationships and correlation studies

There are many kinds of new issues that can be derived and learned from assessment type measurements. Having a look at measurement results usually raises new interesting questions. The case studies below are searching for new dependencies and correlation between attributes (e.g. quality, lead time, efficiency). The purpose/motives of each topic is argued. The view and the statistical tool/method used for each measurement case is given in four tables below. We have divided the case studies into four areas:

- General matters of interest,
- Case studies related to pre-release and post-release quality,
- Studies of quality by type of modules, and
- Other aspects of faults detected at the customer.

Three general issues that we study are summarised in Table 11 and discussed in Sections 5.4.1-5.4.3.

Table 11. General matters of interest

Attributes	Measurement	View	Tool/Method	Section
Quality vs. productivity	Number of faults/volume is compared with volume/man-hour	Project	Scatter diagram	5.4.1
Effort vs. volume	Development man-hours vs. software volume in statements	Project	Regression analysis	5.4.2
	Man-hours spent for unit design vs. volume in statements	Module	Regression analysis	5.4.2
Lead time vs. quality	Correlation between planned leadtimes and fault density	Project	Scatter diagram	5.4.3

Our first general study topic in Table 11 concerns *quality versus productivity*. One constant demand from managers is the improvement of quality and productivity. The argumentation for this study is to visualise organisation's capability to produce high quality with high productivity. As a second topic we study relationships between *effort and volume* both on project level and on software module level. We find out whether the collected data supports the fact that effort and volume should represent a nearly linear dependency. This topic was selected because similar results have in earlier studies been used to build effort estimation models (Putnam, 1978; Boehm,

1981). Selection of the third topic, *lead time versus quality*, was based on an intuitive management experience assuming that a tight schedule could even improve quality.

Four test related issues (Table 12): post-release quality vs. quality seen in Function Test, occurrence of "Stinker" modules, cumulative fault density, and fault density vs. slippage are discussed in Sections 5.4.4-5.4.8.

Table 12. Case studies related to pre-release and post-release quality

Attributes	Measurement	View	Tool/method	Section
Post-release quality vs. quality seen in Function Test	Post-release fault density in modules that were faultless in Function Test	Module, Project	Stratification, Rough set analysis	5.4.4
	Post-release fault density in modules that were proved to be low quality modules in Function Test	Module	Stratification, Rough set analysis	5.4.5
Occurrence of "Stinker" modules	Ratio % of stinker modules (both new and modified)	Module	P-chart	5.4.6
Cumulative fault density	Mean fault density in FT+ST+6 months	Organisation, Time	Trend diagram	5.4.7
Fault density vs. Slippage	Correlation between of fault density in Function Test and slippage	Project	Scatter diagram	5.4.8

Selection of the first two issues in Table 12 was based on questions raised in Management Review meetings. As there are a number of faultless modules in Function Test, it is interesting to know whether these non-faulty modules are still faultless in the field. On the other hand, it is reasonable to pay attention on the classical question: Are the most faulty modules in design error-prone also in the field? For identification of low quality ("Stinker") modules an application of Statistical Process Control (SPC) based on P-Chart is available. This method was used to study differences in the ratio of stinker modules between new and modified modules. Cumulative fault density expresses how well the faults have been prevented to flow from development to test and operation (e.g. caught in inspections). Finally, we try to find some indication of correlation between fault density in Function Test and slippage (slip-through). Slip-through attribute expresses a percentual amount of faults "delivered" to the customer and therefore a number of real slipping ratios collected from projects is studied.

Next, we selected three issues (Table 13) that we discuss in Sections 5.4.9-5.4.11 in order to see the impact of module type on quality.

Table 13. Quality by type of modules

Attributes	Measurement	View	Tool/method	Section
Quality by new module type	Fault densities by the new type introduced within new projects in FT+ST+ 6 months	Module type	Bar	5.4.9
Post-release quality by groups of modules	Post-release fault density by the technical product groups	Module group	Bar	5.4.10
Post-release quality by degree of modification	Post-release fault density by modification grade	Local, Global	Bar	5.4.11

The first topic in Table 13 is "Quality by new module type". A new technical type was introduced in the database system (in the local system only) and data from more than 200 modules was collected since 1993. Now there is enough statistical material available to utilise this data and therefore fault densities by type are discussed. The new type is explained in Section 5.4.9. The second topic concerning post-release fault density by module group enables a new cross-organisational view. As one of the further development needs was (see Section 3.5.1, Table 6) product group (e.g. subsystem), we initiated its implementation in the measurement system for the modules belonging to the Product Provisioning area TSS (Trunk and Signalling Subsystem). The first measurement results using this

grouping are presented in Section 5.4.10. As a third topic we selected a study regarding the impact of modification grade. There is not much such knowledge in the literature. The earlier studies (see e.g. Section 5.1.1) focused on fault density in Function Test, now we focus on 6-month post-delivery period (Section 5.4.11).

At last, our case studies include two customer oriented issues (Table 14), namely quality seen during 7-12 months after delivery and severity of customer detected faults, that are discussed in Sections 5.4.12-5.4.13.

Table 14. Other aspects of faults detected at the customer

Attributes	Measurement	View	Tool/method	Section
Quality at the second 6-month period	Fault density during 7-12 months after delivery	Project	Bar	5.4.12
Severity of faults	Proportion (%) of faults causing fatal failures at the customer	Organisation	Pie chart	5.4.13

As a first topic in Table 14 we focus on the second 6-month operational period at the customer. In practice it may happen that the modules are not yet in broad operation during the first 6 months and thus this period is relatively short and limited. Also a 12-month period (Grady, 1992) appears in the literature. The second period was thus needed (see Section 3.5.1) but no baseline data was available. Therefore, the purpose and method for a separate study about quality seen during 7-12 months was specified. The results of the study carried out (Ahola, 1996) are summarised and discussed in Section 5.4.12. The second study in Table 14 relates to on the severity of faults. It is essential to know how fatal the current faults are. Some results from earlier fault analyses are available. Further details are presented in Section 5.4.13.

5.4. Case studies - exploitation possibilities of results

In this section we concentrate on finding relationships between (external) attributes in the order specified in the previous Section 5.3.4. The most important relations and correlation studies are presented here case by case in a manner described in Sections 5.3.2 and 5.3.3, including graphics and findings. Some new hypotheses are stated and tested in the light of the collected material. We have formulated hypotheses for eleven out of eighteen case studies. The argumentation for stating a hypothesis has been a possibility to tie it to literature, or an aspect that it might be otherwise generally interesting. In cases with no formulated hypothesis we study correlation and discuss the meaning of results. Test results are presented by showing measured facts. Discussions include the usage (application) of each measurement case.

5.4.1. Quality Productivity

The graph in Figure 5.6 helps to visualise quality improvement compared to productivity improvement. Each dot represents a finished project. Quality is expressed here as faults detected in Function Test / new and modified kPLEX volume. Productivity is defined as volume (new and modified ¹¹) / man-hours spent to produce this volume.

¹¹ New and modified volume means the sum of volume in all new modules plus the sum of modified part (volume of the module x modification grade) in all modified modules developed within the project

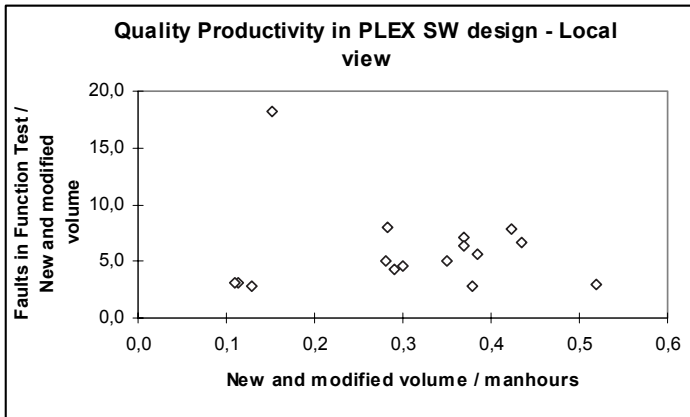


Figure 5.6 Quality Productivity

Facts (Data set 1):

A majority of projects are located in the lower right area that means high productivity with high quality. Correlation coefficient ($r = -0,11$) is showing a weak negative correlation. Trend curve is not meaningful here. One “exceptional” project is found in the upper left area - the reasons should be analysed in more detail. One observation regarding this project is a very low (11%) modification grade of modules.

Quality vs. project effort

Dependence between quality and total man-hours of projects was also studied. The following hypothesis was stated.

Hypothesis 1:

Amount of project faults correlates positively to the total effort of the project.

This simple evidence is also found in the literature (Putnam, Myers, 1997) showing that also in other databases an increase in the number of defects goes along with an increase of effort.

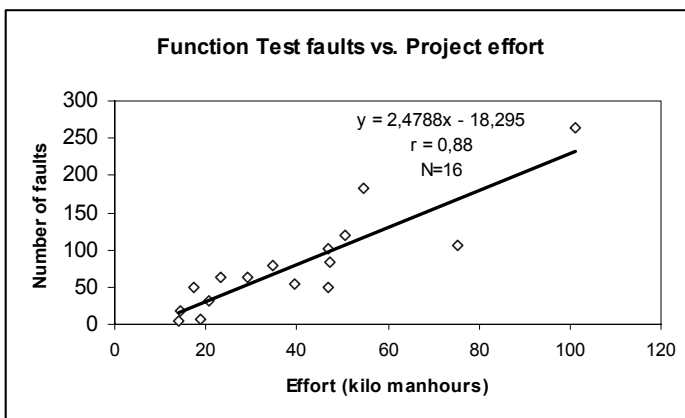


Figure 5.7 Relationship between Function Test faults and project effort

Facts (Dataset1):

Figure 5.7 depicts the effect of increasing project effort on the total number faults detected in Function

Test for the same projects as in Figure 5.6. Correlation between total Function Test faults and total project effort is significant because the value ($r = 0,88$) for the correlation coefficient is greater than the critical value (0,4973 in case of 16 items).

Conclusions:

The case study provides support for the hypothesis because significant correlation was found.

Use:

This fact can be used as a very rough fault prediction model. When you know the effort then you also know how many faults are expected.

5.4.2. Effort vs. volume

In this section, we are first searching how project effort depends on the source program volume developed in the project. Secondly, we discuss whether there is any correlation between coding effort and source program volume. As a third topic we try to find out a correlation between quality and coding effort.

Project effort vs. volume

The cost models, such as COCOMO (Boehm, 1981) and SLIM (Putnam, 1992) are based on the argument that effort has a positive correlation to volume estimated in SLOC or in delivered source instructions (KSDI). In order to see how the issues look as we use material from PQT database (modules written in PLEX) we formulated the following hypothesis.

Hypothesis 2:

Project effort correlates positively to the new and modified volume to be developed.

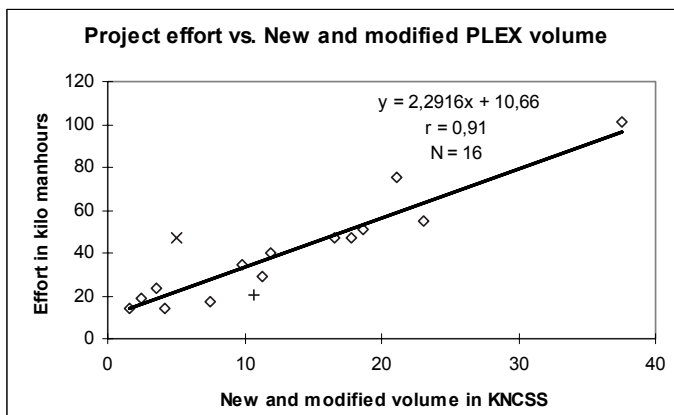


Figure 5.8 Relationship between man-hours and volume

Facts (Data set 1):

New and modified volume was selected because the project effort includes all man-hours spent to produce this volume. The study only includes the local subprojects, because validated data was available only from local database. As data from 16 past major projects (the same ones as in Figure 5.6, from 1990's) was studied, the following dependencies were found (Figure 5.8). Correlation is strong and significant because the value ($r = 0,91$) for the correlation coefficient is greater than the critical value (0,4973 in case of 16 items). In the graph there are smaller projects deviating remarkably from the regression line. For example, the project marked with "x" did modify the modules less than 10% in average, so the effort was mainly spent to produce other documents than the program code. All the modules in this project were modified products. Also the modification grade

has its inaccuracies. According to the Final report of the project (Mellberg, 1996) requirements were heavily changing during the project causing extra effort in design and testing.

Conclusions:

Project man-hours correlate well with the new and modified volume supporting our hypothesis.

Use:

These results can be used for rough estimation of project effort when the new and modified volume is predicted at the start of the project. It is possible to derive a simple equation for prediction.

Discussion:

In the light of literature the man-hours should not have a fully linear relationship with volume. Some estimation models, e.g. COCOMO in embedded mode, (Boehm, 1981, p. 76) state a slightly unlinear dependence. The reason for different result might be that our local subprojects do not involve all administrative man-hours, which are in some projects reported on the main project. An indicator set in a measurement guidebook (Jones, 1998) includes similar size–effort relationship. The intention is to use it to crosscheck effort estimates generated by other means to analyse reasons for the deviation if the estimates fall outside the 95% confidence limits. To get a useful method, enough historical data from finished projects for the same application domain is needed.

A study of Unit design (coding) man-hours and volume of modules

As man-hours for Unit Design & Basic Test and PLEX volumes per module were listed from the PQT database, we observed that a number of new big modules were found having only a small man-hour effort spent. A new module of 3210 PLEX statements can not be designed in 3 man-hours. Possible error sources are:

- A number of stored “new” modules are obviously not new, but considered being new when their module identity is new.
- Some man-hours are clearly estimates or rounded figures.
- A project may have several different modules with the same amount of man-hours for each of them.
- It is unclear how man-hours for beginners are handled in different companies - are their hours registered on job training or time-reported directly on software modules?

One can not rely so much on man-hour data - so we investigated new modules only (about 1400 modules, collected from all companies within PQT, Data set 2). The graph (Figure 5.9) shows that any reasonable correlation between man-hours and volume can not be obtained. However, Figure 5.9 indicates that there are two different behaviours in these modules, a set of big modules which required less man-hours and a set of middle-sized modules with a large effort consumed.

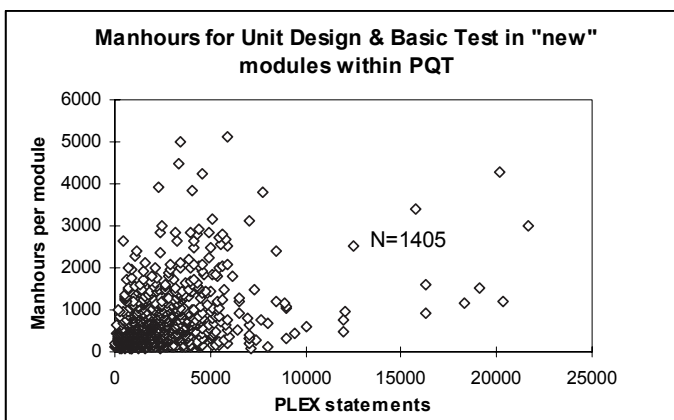


Figure 5.9 Coding man-hours vs. volume of modules

The reason would be the incorrectness of man-hour data mentioned above and, on the other hand, the differences in module difficulty and designer's skill levels. In connection with this thesis it was not possible to investigate in detail the data that was not available in the database. It would be reasonable to augment the PQT database by new data like module complexity, functional type of the module and design competence per module. The result also stresses the importance of *data validation* before inserting data items into the database. Ericsson in Finland has been more careful in classifying modules (new and modified) and registering man-hours. Thus, as shown in Figure 5.10, there is a significant linear dependency between man-hours and volume in new modules (Data set 3).

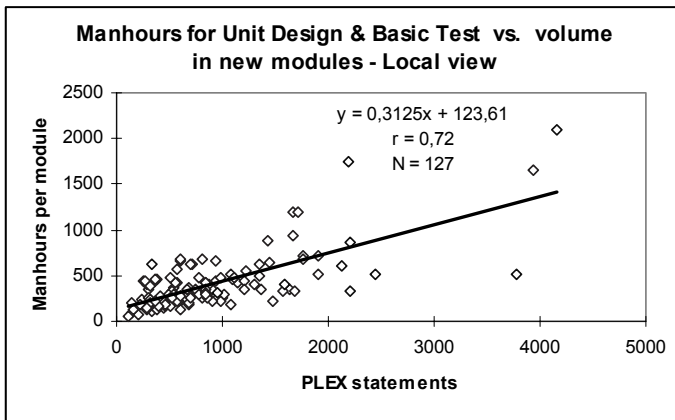


Figure 5.10 Coding man-hours vs. volume of modules

Correlation is significant because the value ($r = 0,72$) for the correlation coefficient is greater than the critical value (0,1966 if the number of items $N > 100$).

Quality vs. Unit design & Basic Test effort

Quality is expressed here as the number of faults detected during Function Test. Correlation between quality and design effort was studied for the same collection of modules as in Figure 5.10 (Data set 3). Quality depends on the man-hours as shown in Figure 5.11 indicating the fact: the more man-hours, the more faults.

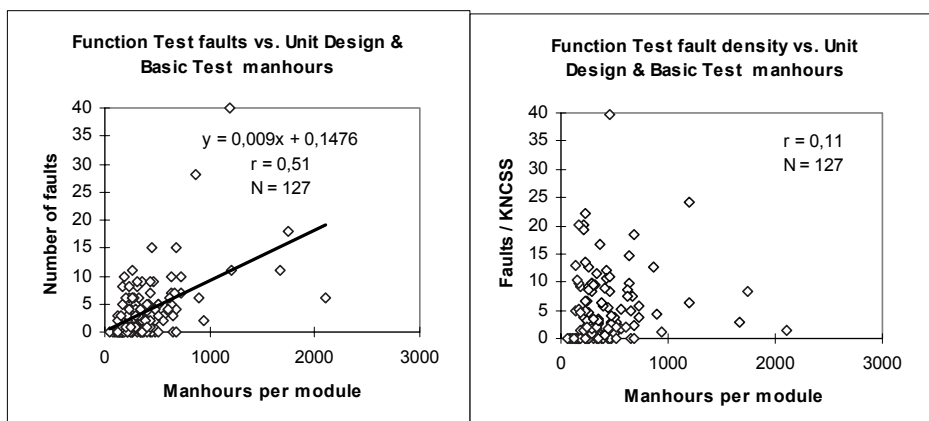


Figure 5.11 Quality vs. Unit Design & Basic Test hours in new modules - Local view

However, the correlation is not very good ($r = 0,51$). One reason might be booking inaccuracies in module man-hours. The rightmost picture shows that fault density does not correlate so well to man-hours ($r = 0,11$). Correlation is not significant because the value ($r = 0,91$) for the correlation coefficient is lower than the critical value (0,1966 if the number of items $N > 100$). Exceptionally high fault

densities appear in some smaller modules. The reason might be incomplete inspection or Basic Test. In the future it is important to register carefully the hours and faults for inspection and Basic Test activities to understand better the drivers to quality observed in Function Test. Competence and complexity are also important driving attributes for which collected data was not available per module.

5.4.3. Lead time vs. quality

Lead time means the calendar time from the start to the end of a defined phase or process according to the project model (see Section 2.1.1). In this study, lead times, either planned or actual, are based on milestone dates corresponding to tollgates or other important events (e.g. final release of modules) in a project. In order to study relationship between lead time and quality the following hypothesis was stated and tested.

Hypothesis 3:

Shorter planned lead times lead to better quality for the project.

The hypothesis is quite “revolutionary” because usually the exact opposite phenomenon is presented, i.e.; better quality can lead to shorter cycle times. Traditionally, e.g. the importance of early inspection has been argued by claiming that it reduces the lead time (Fagan, 1986 and Gilb, 1988). Poor quality was one of the most common reasons for schedule overruns in a survey of about 4000 projects (Jones, 1994). Abdel-Hamid (1991) presents some insight into effect of time pressure on error rate showing that error generation can increase by as much 50% under severe schedule pressures.

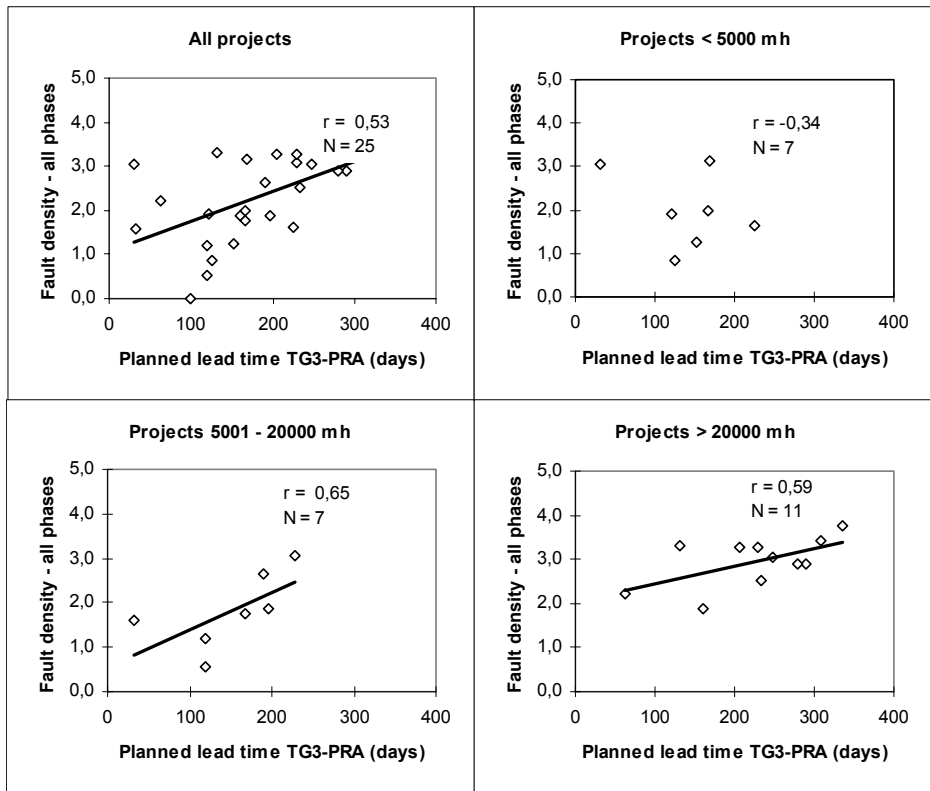


Figure 5.12 Fault densities to planned lead times

However, some software managers have presented indications (Toivo, 1996) that tight time schedules correlate with quality of the modules developed within a project. This is why the project that has a tight schedule should choose the resources very carefully. It is usually expected that a competent design team is able to produce good quality. Because any measure is not available expressing which project has a tight schedule it has been assumed that a short schedule is a tight schedule. A short schedule can be determined by comparing planned lead times of different projects. To get more facts from projects a separate study was specified and performed (Lindberg, Sundström, 1996).

Facts (Data set 4):

To test the hypothesis we use the planned lead times instead of actual ones. Quality for a project is measured as actual mean fault density for the project. It is also reasonable to classify projects with respect to effort in man-hours. We use three classes in order to distinguish between small and large projects: < 5000, 5001-20000 and > 20000 man-hours (mh). In each quadrant of Figure 5.12, x-axis shows planned lead time TG3-PRA¹² in days. Y- axis shows fault density for all phases FT, ST, 6 months. The leftmost upper quadrant includes all analysed 25 projects showing that correlation is not so strong. It is significant because the value ($r = 0,53$) for the correlation coefficient is greater than the critical value (0,3961 in case of 25 items).

In small projects we observe a weak negative correlation, -0,34 (not significant, critical value is 0,7545 in case of 7 items). Projects larger than 5000 man-hours have lower fault density when the planned lead time is shorter, same to eleven large projects (> 20000 man-hours) in the rightmost lower quadrant. Correlation ($r=0,59$) is not significant because the critical value for 11 items is 0,6021.

Conclusions:

Some support for the hypothesis but the correlation is not strong. The hypothesis was not true for small projects.

Use:

The results like this can be used to optimal break down of project schedule.

5.4.4. Post-release quality of modules faultless in Function Test

A significant number of modules appear to be faultless in Function Test (FT). A sample in Figure 3.2 (see page 21) shows that around 30% of modules are faultfree in Function Tests at Ericsson. Similar figures have been presented in other studies (Khoshgoftaar, 1992) discussing a random sample, where 30 out of 109 modules are faultless (28%). Other pareto analyses (Ohlsson, 1998) regarding Function Test show that 74% of modules are responsible for 100% of faults, so the rest of modules, i.e. 26% are faultfree. Software defect studies from different environments over many years in the area of software engineering conclude that about half the modules are defect free (Boehm, Basili, 2001). A proposal for a new study item popped up in the Management Review (Hirvensalo, 1994) to find out whether or not essentially more faults appears later in modules that have been faultless in Function Test. Because faultless modules have basically not been a concern in earlier studies (as they focus on fault prone modules) we state a question: "are the modules faultless in Function Test still faultless in later test phases and field operation". This implies the following hypothesis.

Hypothesis 4:

The modules that have been faultless in Function Test (FT) are also faultless after release.

Facts (Data set 5):

A number of modules have been studied below in order to know how the modules faultless in FT look after release. The study focused on 73 faultless modules out of 242 modules that have passed FT and also been in operation for 6 months (the uppermost row in Table 15 below).

First, the rightmost column shows that post-release fault density (i.e. in System Test (ST) and during 6 months (6MO) at the customer) has been the lowest in modules having zero faults in FT. Either the

¹² TG3 marks a point in time when coding of software modules starts; At PRA the coding and Function Tests are finished. For further details of definitions of TG3 and PRA, see Section 2.1.1.

FT did not succeed to find faults or the quality in design was really very high. Secondly, we observe that percentage of faultless modules also after release is the highest within these modules. Finally, Table 15 shows that the zero fault modules have the smallest size in average.

Table 15. Post-release quality - faultless modules in FT vs. faulty modules after release

Faults detected in Function Test per module	Number of modules	of which faultless modules after release in number and per cent (%)	Volume in total (NCSS)	Average size (NCSS)	Number of Post-release faults (ST / 6MO Sum)	Mean Post-release fault density (ST / 6MO)
0	73	59 (80,8 %)	94211	1291	15 / 11 26	0,16 / 0,12
1-2	73	51 (69,9 %)	106248	1455	20 / 17 37	0,19 / 0,16
3-5	46	21 (45,7 %)	93403	2031	38 / 26 64	0,41 / 0,28
> 5	50	12 (24,0 %)	128696	2574	129 / 86 215	1,00 / 0,67
Totally	242	143 (59,1 %)	422558	1746	202 / 140 342	0,48 / 0,33

Our further analysis showed also that 60% of modules in the first category have low modification grade (< 20%) while the last category mainly contains highly modified modules. From Table 15 one can easily calculate that also the number of post-release faults is the lowest in the first category.

On the other hand, it is also necessary to find out how often it is the same product that is faultless in FT and also after release. The third column in Table 15 shows that 59 out of 73 zero fault modules have been faultless also after release, i.e., in 81% of cases, the same single module has been faultless in all phases. This gives some indication that the reason for post-release faultlessness has not been insufficient execution of Function Test. However, further investigation of the data set showed that still the rest 14 modules were faulty either in ST or during 6 months. These modules contain those 26 post-release faults that were not detected in FT. Collected measurement figures, such as size and modification grade do not alone explain the reason for slippage, but each single module (e.g. difficulty, competence) and fault must be analysed individually. Differences of Inspection and Basic Test results between each group of modules should also be compared in further studies.

Facts (Data set 15):

In order to prove this hypothesis further, another collection of modules (Data set 15) was studied project by project (Figure 5.13). The graph in Figure 5.13 shows the trend per project that is similar to results presented in Table 15. We use lower granularity in classification for faults detected in FT because of smaller number of modules per project. The columns representing post-release fault density of modules that were faultless in FT (the leftmost columns marked in bold) have the lowest post-release fault density.

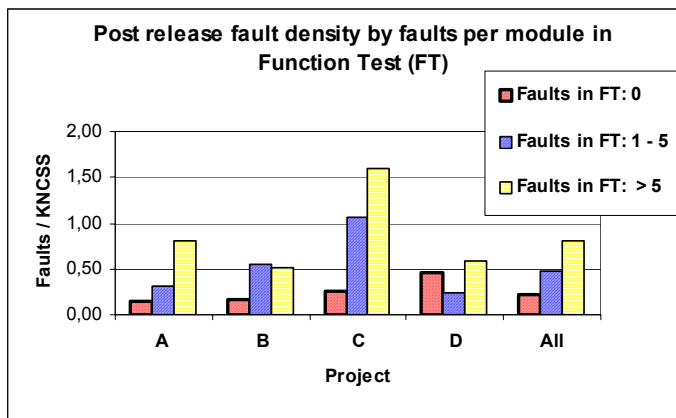


Figure 5.13 Post-release fault density in projects

Projects included in Figure 5.13 have succeeded to produce a number of modules that were faultless in all phases. The highest percentage of such modules appears in the category where faults in FT is zero. In these four projects around one third of all modules are faultfree in FT and more than 80% of them are also faultless in ST and during 6 month (Table 16). This percentage is significantly high with respect to occurrence of fully faultless modules within all modules (compare to the rightmost column).

Table 16. Faultlessness in four projects

Project	Volume in total (NCSS)	Number of modules	of which faultless modules in FT in number and per cent (%)	of which faultless modules in FT, ST and 6MO in number and per cent (%)	Occurrence of fully faultless modules out of all modules in per cent (%)
A	94999	89	27 (30,3)	24 (88,9 %)	27,0
B	56970	41	16 (39,0)	14 (87,5 %)	34,2
C	50324	48	17 (35,4)	14 (82,4 %)	29,2
D	82876	51	15 (29,4)	12 (80,0 %)	23,5
All	285169	229	75 (32,8)	64 (85,3 %)	28,0

Conclusions:

Studying two data sets provided evidence for hypothesis 4. In more than 80% of cases, where a module is faultless in Function Test, the same single module is faultless after release. Both the number of post-release faults and fault density is the lowest in this category of modules. For these reasons it can be concluded that these modules have not been problem modules in the field.

Use:

Trace the modules that have been faultless in FT, ST and 6 months, analyse the driving factors which led to good quality and learn about them. High percentage of such modules can be used as a goodness measure of the Function Test process. It is also useful to publish a list of these modules that were faultless in all phases to acknowledge the personnel about good quality.

Discussion:

The study neither considers complexity of faultless modules nor the severity of faults. In some modules there is a risk that the use after delivery to the customer has not yet been broad and therefore no faults occurs during 6 months.

5.4.5. Post-release quality in modules having high fault content in test

Testers traditionally believe that the probability of the existence of errors in a module is proportional to the number of errors already found in that module (Myers, 1979). This is especially true if a module is badly structured (e.g. spaghetti code) and thus error prone. Early during the present study a question arose: "Are the most faulty modules in design also error prone in System Test and in the field?". Quality management was interested in studying this question by using appropriate material available in the PQT database. We formulated the following four hypotheses.

Hypothesis 5:

- The higher number of faults detected per module in Function Test, the higher fault density in System Test.
- The higher number of faults detected per module in Function Test, the higher fault density after internal release (i.e. in System Test and during first 6 months).
- The higher fault density in Function Test, the higher fault density during first 6 months after external release (RFA).
- The higher number of detected faults per module in Function and System Tests, the higher fault density during first 6 months after external release (RFA).

Facts for hypothesis 5a (Data set 6):

The hypothesis 5a can be proved by investigating whether the modules that have been faulty in Function Test still remain faulty in System Test. The analysed small sample (Table 17) of Finnish modules shows that high amount of faults (e.g. > 5 faults) detected in Function Test per module means that a majority (> 57 %) of these modules are still faulty after internal release (i.e. in System Test). The higher is the number of faults detected in Function Test, the higher is the proportion of faulty modules in per cent. As a higher number of faults is detected in Function Test then also the mean fault density in System Test is getting higher. On the other hand, Table 17 shows that average size in NCSS is increasing, i.e., the faulty modules are bigger modules.

Table 17. Post-release fault density vs. faults detected in Function Test - local view

Faults detected in Function Test per module	Number of modules	of which faulty modules in System Test in number and per cent (%)	Volume in total (NCSS)	Average size (NCSS)	Mean fault density in System Test (ST)
0	154	14 (9,1 %)	201329	1307	0,12
1-2	146	42 (28,8 %)	259727	1779	0,22
3-5	96	33 (34,4 %)	214008	2229	0,32
> 5	103	59 (57,3 %)	287313	2789	0,76
Totally	499	148 (29,7 %)	962377	1929	0,38

Facts for hypothesis 5b (Data set 7):

We use a larger sample from global view to prove the hypothesis that is very close to the hypothesis 5a. Modules in Table 18 are divided into five categories by the number of faults detected in Function Test (FT). Two added categories are used for modules where more than five faults were introduced because absolute fault figures on global level appeared to be higher than on local level. Post-release fault density (see the rightmost column for ST+6MO) is constantly growing as function of faults detected in FT. On the other hand, it is also necessary to find out how often it is the same module that is faulty in FT and after release. This can be observed in column 3.

Table 18. Post-release fault density vs. faults detected in Function Test - global view

Faults detected in Function Test per module	Number of modules	of which faulty modules after release in number and per cent (%)	Volume in total (NCSS)	Average size (NCSS)	Mean Post-release fault density (ST+6MO)
0	2390	889 (37,2%)	4696539	1965	0,50
1-2	2043	1006 (49,2%)	4225358	2068	0,63
3-5	1404	881 (62,7 %)	3255502	2319	0,90
6-10	926	713 (76,9 %)	2497289	2697	1,27
>10	1228	1018 (82,9 %)	4219501	3436	2,06
Totally	7991	4507 (56,4 %)	18894189	2364	1,05

Facts for hypothesis 5c (Data set 6):

In our hypothesis we also expect to know the dependence between fault densities observed in FT and after delivery. We use Data set 6 again but we use normalised data (faults/KNCSS) instead of absolute faults per module. The following Figure 5.14 and Table 19 show the post-release fault density during RFA+6 months by fault density categorised in FT. The highest fault densities during 6 months are observed in groups of modules having a high fault density in FT.

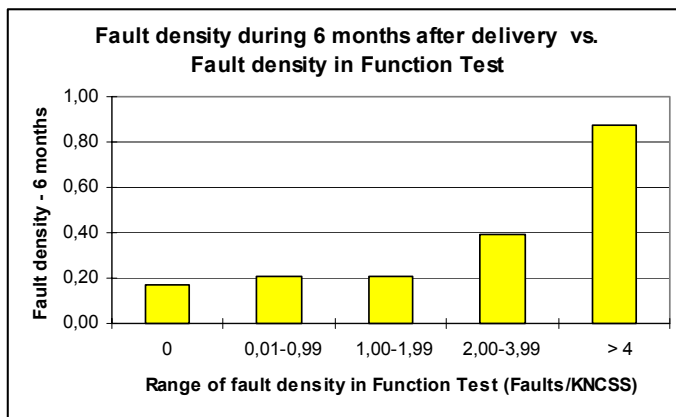


Figure 5.14 Comparison of fault densities in test vs. 6 months after delivery

Table 19. Detailed module data by categories included in Figure 5.14.

Fault density in Function Test per module (Faults/KNCSS)	Number of modules	Volume in total (NCSS)	Fault density during 6 months after delivery (Faults/KNCSS)
0	154	201329	0,17
0,01 - 0,99	84	263996	0,21
1,00 - 1,99	85	208951	0,21
2,00 - 3,99	83	162997	0,39
> 4	93	125104	0,87
Totally	499	962377	0,32

Facts for hypothesis 5d (Data set 6):

Finally, we categorise our material by summed faults detected in Function and System Tests (Table 20). Still the highest post delivery fault density appears in the lowermost category where also the highest percentage of faulty modules (48,4 %) occurs.

Table 20. Post delivery fault density vs. faults detected in pre-release testing

Faults detected in Function Test + System Test per module	Number of modules	of which faulty modules after delivery in number and per cent (%)	Volume in total (NCSS)	Average size (NCSS)	Fault density during 6 months after delivery (6MO)
0	140	21 (15,0%)	174701	1248	0,17
1-2	133	22 (16,5 %)	218111	1640	0,13
3-5	104	38 (36,5)	230099	2212	0,33
> 5	122	59 (48,4 %)	339466	2789	0,50
Totally	499	140 (28,1 %)	962377	1929	0,32

Now the next issue of analysing is how these faulty modules differ from other modules in the lowermost category. Regarding this category, our rough-set analysis showed that more than half of faulty modules represented highly modified (i.e. modification grade ranges 40-99%) modules. The majority of other modules (Fault 6MO =0) consisted of small new modules or modules with modification grade lower than 20%.

Conclusions:

In our study, we found evidence to support all the four stated hypotheses 5a ...d. Higher post-release or post delivery fault densities appear in those module categories where the higher number of faults are detected in Function Test (or in Function Test + System Test). Hypothesis 5c appeared to be true because we observed the highest fault densities during 6 month in groups of modules having a high fault density in Function Test.

Discussion:

Hypothesis 5a appeared recently also in other studies(Fenton, Neil, 1999; Fenton, Ohlsson, 2000) in a form "Higher incidence of faults in Function Testing (FT) implies higher incidence of faults in System Testing (ST)". Hypothesis 5d is closer to the statement presented in the related study (Fenton, Ohlsson, 2000) "Higher incidence of faults in all pre-release testing (FT and ST) implies higher incidence of faults in post-release operation (SI and OP)". In their investigation, the phase "SI" means first 26 weeks at a number of site tests that is exactly the same compared to the period RFA+6 months in the present study. Their phase "OP" means first year of operation after site tests. The data for the first year of operation was not stored into the measurement database and thus not available for the present study.

Use:

These facts can be used for example in planning and setting alarm limits during Function Test in order to catch and act on the modules that most likely will be faulty in the field.

5.4.6. Identifying occurrence of "Stinker" modules

The existing databases provide opportunities to identify modules having a very low quality. There are several possibilities to define a very low quality module (so called "stinkers"). It can be defined by using a high number of faults as criteria. However, the absolute number of faults is to a large extent dependent on module size. A better definition is found by using statistical process control method to

indicate whether the quality is deviating from "normal" level too much. According to Ericsson's PQT definition the "stinker" modules are modules having a fault density that significantly exceeds the capability level of the organisation and software process. It is supposed that the fault density of a mature process is below a certain limit. This control limit is set to 3σ ($3 \cdot \text{standard deviation}$) + Mean Fault Density. If a module is out of this limit this can NOT happen only because of random variation. In this way it is possible to count the sum of modules deviating from "normal" level and to calculate ratio of stinker modules as the number of assumed stinker modules divided by the total number of modules.

Facts (Data set 8):

Figure 5.15 illustrates the principle described above in the context of Function Test phase. Mean stinker ratio for the entire displayed graph is indicated as a straight horizontal (red) line. Upper control limit, UCL and lower control limit, LCL are indicated as the incremental (green) lines. The actual ratio of stinker modules is also given (black line).

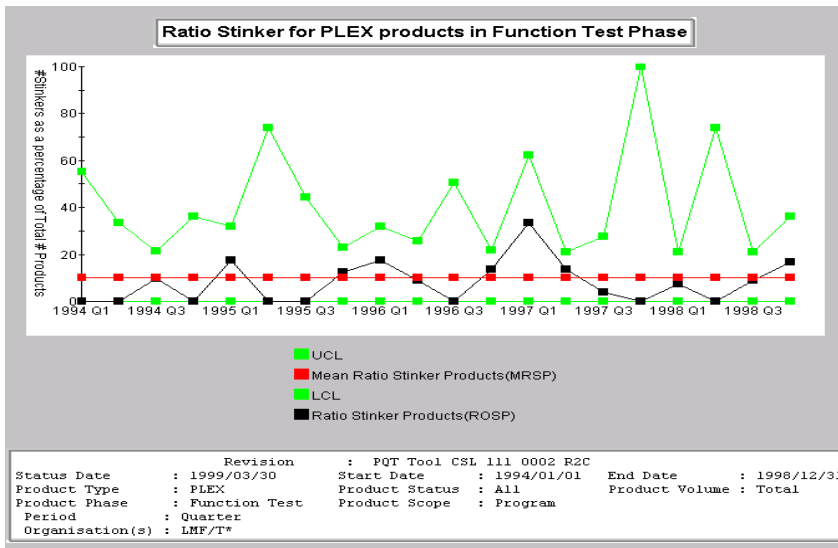


Figure 5.15 Ratio of low quality modules

As an additional information to the graph it is useful to list the modules (which are assumed to be stinkers) having "improbably" high fault density. This list contains information on module identity, name, and release date and project identity.

Use:

Ratio of Stinker modules can be used to analyse the stability of the design process, to identify potential problem areas and to obtain basic information for root-cause analyses.

5.4.7. Time based trend of cumulative fault density

In order to study the accumulated fault density for FT+ST+ 6 months, a time based trend presentation has been developed and introduced during this study (Figure 5.16).

Hypothesis 6:

Company-wide software process improvement actions lead to a real decreasing effect on total fault density.

Several companies have been able to reduce fault density in consequence of their process improvement activities. NASA/SEL has provided some error data taken from 60 projects over the full lifetime (McGarry et al., 1994). HP set a "10 x" improvement goal in 1986 to reduce their product post-

release defect density by a factor of ten in five years. The result (Grady, 1997) shows that the trend was year by year decreasing and HP achieved a 6 x company wide defect density improvement.

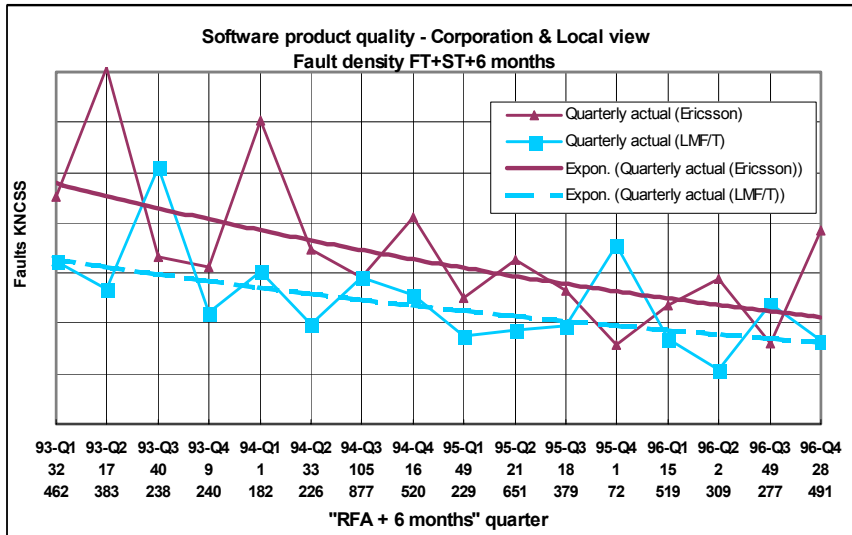


Figure 5.16 Trend of cumulative fault density (Note ¹³)

Facts (Data set 9):

The graph shows a decreasing trend of fault density. Quality shows significant improvement in comparison with baseline i.e. fault density during 1993. Improvement is 53 % on 1996 on corporation level. The graph also shows a higher quality (lower fault density) at local company (LMF/T) contra all design centres of the corporation.

Conclusions:

There is observable support for the hypothesis. During the period in question organisations have taken systematic improvement actions that have lead to a positive trend.

Use:

The presentation can be used by management to see that improvement actions performed during past year have affected significantly in reality. It is also easily possible to calculate savings or return on investment (ROI) when a large amount of trouble reports has been avoided. Management has also used first PQT results in the ESEPG97 conference to show the effect of corporate improvement program (Mobrin, Wåsterlid, 1997).

5.4.8. Correlation between fault density in test and slippage

A correlation should also exist between fault density in Function Test and slip-through to the period of the first 6 months after external delivery (RFA + 6 months). This attribute was used in Ericsson's ESSI program as a Key Performance Indicator for a vital action that aimed at improving the ability of testing activities to prevent faults slipping through to the customer. Now we have an opportunity to present some real figures.

Facts (Data set 1):

Slip-through was studied in the same projects as in Section 5.4.2. The following sample (Figure 5.17) shows only a weak negative correlation. The correlation coefficient is - 0,56 (significant) for fault density calculated with respect to total volume of modules, but a bit lower correlation, $r = - 0,46$ (not

¹³ Note. The figures under the x-axis represent the size of two statistical samples per each period, i.e. the number of modules that have passed 6 months.

significant), was obtained for new and modified volume of modules of same 16 projects. The critical value for the correlation coefficient is - 0,4973 in case of 16 items.

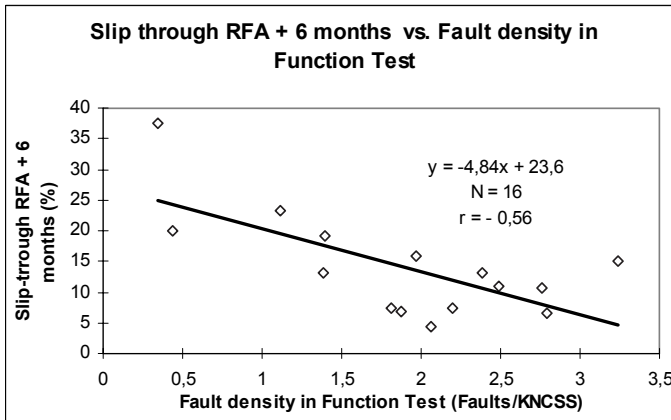


Figure 5.17 Influence of fault density on slip-through

Use:

When fault density in test is predicted or measured then it is possible to predict slip-through ratio.

Discussion:

Because the correlation is weak it is difficult to interpret the result. Fault density in Function Test may either tell about bad quality in design or goodness of testing. It is necessary to find other driving attributes for slip-through and perform multi-dimensional analyses. The graph only shows how the situation looks in circumstances where design and testing methods have varied.

5.4.9. Quality by type of the module

A study of fault densities by new module type was performed in order to see quality variations between different technical types of module. The new module type expresses the main task of a function block. The new module type, tailored especially for digital mobile telephony modules, is as follows:

T	Traffic	Traffic handling
C	Coordination	Coordinator tasks
I	I/O	Read from or write to an I/O buffer
M	MAP	Participate in Mobile Application Part signalling
D	Data SB	Store subscriber data
A	Analysis	Analyse numbers and series

Fault densities for each type in the three phases, FT, ST and first 6 months after delivery, look as shown in Figure 5.18.

Facts (Data set 10):

From the graph one can see that I/O modules have very high quality during 6 months after delivery. Only one fault is found (only in one module out of 45 modules). I/O modules are based on well-known macros and usually the same simple routines are repeated. Coordination and Analysis modules are fully faultless during 6 months. The average size of these modules (typically < 1000 statements) is smaller than the size of Traffic handling modules. Traffic handling and MAP modules are more complex and more difficult to test in Function Test. Faults in Traffic handling modules are found more frequently during operation.

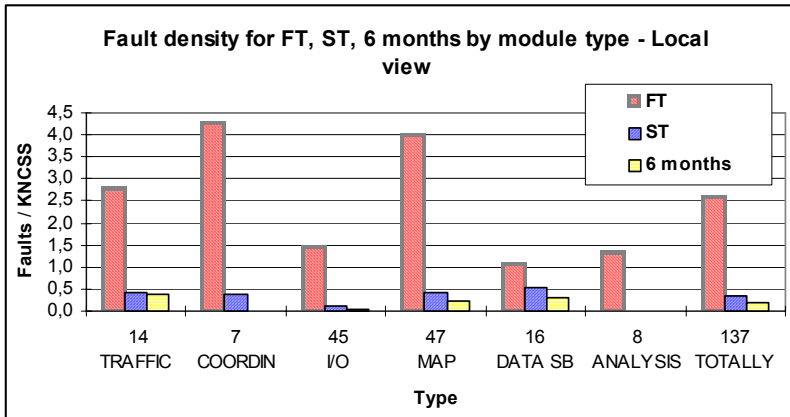


Figure 5.18 Quality by type of the module

Use:

This telecommunication switching oriented classification, set in an early phase of a project, provides information of the probable problem module that need more focus during design, inspection and test (for example, in design of preventive actions, in selection of test cases). The organisation can use the measured facts above as basis for a more detailed root-cause analysis within those groups of modules in which the faults typically slip (e.g. Traffic handling, MAP).

Discussion:

Because testability is different in different types of applications above it would be worthwhile to compare architecture of these applications and the impact of the architecture on testability of modules.

5.4.10. Quality by technical group of modules

The quality measurement database system (PQT) has been further developed during this study by introducing a product group. Within TSS (Trunk and Signalling Subsystem) the modules can be divided into four technical groups:

- TSS_ISUP (ISDN User Part),
- TSS_CAS (Channel Associated Signalling),
- TSS_TUP (Telephony User Part), and
- TSS_NUP (National User Part).

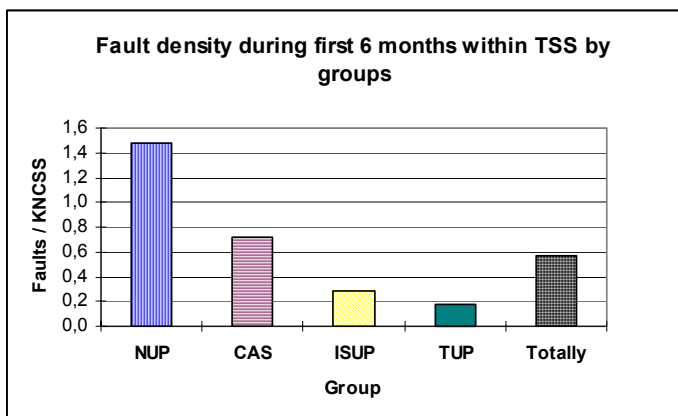


Figure 5.19 Quality by module group

The new concept is important for a geographically distributed company in which one local design centre has a prime responsibility for these modules developed in different countries. Figure 5.19 shows the 6-month fault density that has been defined as a key performance indicator concerning quality.

Facts (Data set 11):

The graph is able to show that the biggest quality problems have been in the NUP area.

Use:

The grouping like this or similar provides a cross-organisational view on technical area by category. It is the responsibility of the design organisation to analyse what are the root-causes for problems. For example, we suppose that root-causes maybe related to the clarity of NUP specifications etc, but those issues are not discussed here.

5.4.11. Module quality by modification grade

Already early experience (Myers, 1979) has shown that modifying an existing program is more error-prone than writing a new program. Existing (old) modules are normally used as base when new software units are developed for telecommunication switching systems (see e.g. Table 9 in Section 5.2.2). In our case, modification grade expresses in per cent to what extent the new software unit is new and modified, i.e., telling how much new and changed code the new software unit contains. This definition implies something opposite to degree of re-use that is widely used. Management of design organisations needed modification grade in fault density and productivity calculations because it was often more appropriate (i.e. rightful) to consider new and changed parts of the module. The modification grade used in this study is defined in more detail in the PQT manual (Ericsson, 1995). A tool for automating calculation of modification grade from source code was available thus providing an opportunity to study the effect of modification in the light of our data. Questions arose like "Is the quality higher in modules of minor modifications" or "How the quality of modified modules compares to quality in new modules". We stated the following hypothesis.

Hypothesis 7:

The lower modification grade, the lower number of post-delivery faults per module is expected.

Facts about faultlessness (Data set 6):

Table 21 contains data collected by modification category. Faultlessness in tests does not tell the whole truth about quality. In this section we identify amount of modules that have been faultless during their first 6 months. The results in rightmost column of Table 21 show that the number of faults per module is increasing in categories where modification is less than 40%. In the lowermost row we can also observe how the results compare to new¹⁴ modules.

Table 21. Faults and faultlessness by modification grade during first 6 months after delivery

Module modification category	Number of modules in the category	of which faultless during 6 months in number and per cent (%)	Faults per module during 6 months
1-10 %	161	123 (76 %)	0,40
11-20 %	87	56 (64 %)	0,68
21-40 %	76	44 (58 %)	1,07
41-99 %	77	54 (71 %)	0,81
100 % (New)	98	81 (83 %)	0,40

Facts about fault density by modification grade during first 6 months (Data set 12):

It is also interesting to see the influence of modification grade on fault density during the first 6 months. Here the fault density reflects the number of faults per total lines of modules, not per modified lines, because in the field operation the customer does not care whether a fault occurs in new parts or modified parts of software modules. In accordance with Figure 5.20, a significantly lower fault density

¹⁴ Modification grade for new modules is considered to be 100%.

is obtained in categories 1-10, 11-20 %. The trend in modules developed on local level is similar to the phenomenon at other Ericsson companies.

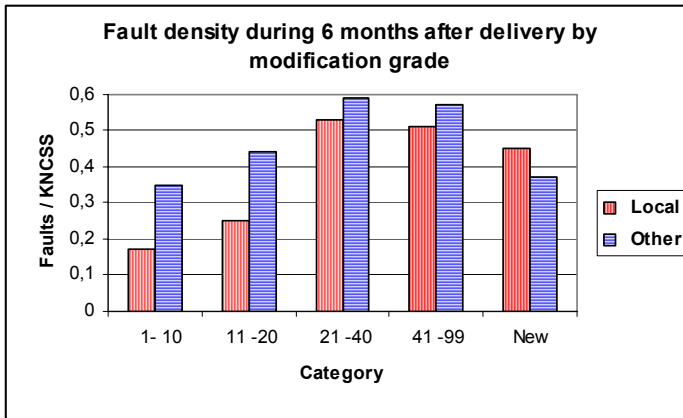


Figure 5.20 Fault density by modification grade category

Facts about occurrence of problem modules (Data set 12):

The material in modification grade categories was also studied in more detail in order to see whether there are less problem modules in categories with lower modification grades. Let us assume that a module having > 5 faults during 6 months after delivery can be considered as a problem module. The study of this aspect showed a low occurrence of such problem modules in the group of modules modified less than 20% (1 item out of 248 items, i.e. 0,4 %), but significantly higher occurrence (6 modules out of 153 items, i.e. 3,9 %) in highly modified modules (21-99 %).

Conclusions:

The results show that our hypothesis is true. The lower is the modification the less faults per module. New modules are less faulty than highly modified modules.

Use:

This knowledge can be used to achieve a lower fault density during 6 months after delivery by designing modules either with minor modification or by making a totally new module instead of modifying an old module to a large extent. Measurement results thus help to understand the effects of modifying software. The modification grade can also be used for comparing the design and the test effort between new and modified modules.

5.4.12. Module quality during 7-12 months after delivery

Hypothesis 8:

Fewer faults are detected during 7-12 months in comparison with first 6 months after external delivery.

Facts (Data set 13):

Quality seen during 7-12 months was examined in a separate study (Ahola, 1996) from 17 projects which, in addition to Function Test, System Test and 6 months, have passed their first 12 months after external delivery. Data from 170 modules was included. The graph in Figure 5.21 shows fault densities in all the 4 phases per project and the average over all projects. Average fault density for all projects is shown in the rightmost column of each phase.

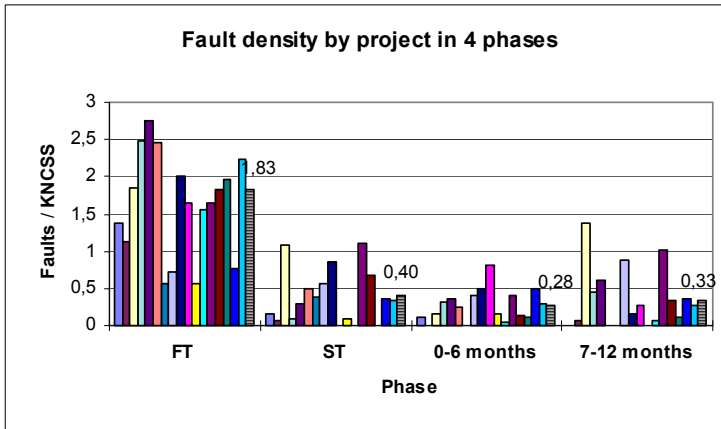


Figure 5.21 Fault densities measured up to the end of 12 months

Conclusions on results of the study:

- There is a weak linear correlation between 6 month's and 7-12 month's quality.
- Variations between projects and modules are high.
- 70% of modules are faultless during 7-12 months, but occurrence of modules having at a maximum 2 faults is more than 90%.
- The fraction faultless modules during 7-12 months nearly equals to 6-month figure, but faultlessness does not always occur in the same module.
- In nine projects, the fault density during 7-12 months is clearly equal or lower than during the first half of the year.
- On average, the 7-12 month's fault density over all projects is not lower than the 6-month fault density.
- In eight projects, the fault density for 7-12 months is even higher than during the first 6 months.

Discussion:

It is difficult to know whether the modules, during the first half year, have been in broad use or not – a fact that can explain the higher 7-12 month's fault content in several projects. Point in time when a single module has started in the real commercial traffic at the customer's site is also very difficult to trace. Thus, the variations between modules in duration of operational period have influenced on results.

Use:

The result can be used to predict roughly how many faults are left after the end of first 6 months. In new projects, it is thus reasonable to continue collection of faults after first 6 months and instead use statistics for the first year. This type of study also provides a basis for further analysis of type and severity of those faults that slipped through to the customer.

5.4.13. Proportion of faults causing fatal failures at the customer

In simple fault density measurements, no distinction between fatal and less serious faults is made, but all faults that cause correction of code are counted. However, information on severity of faults is usually available from trouble reporting systems/databases in software companies. Because this information was easy to search by using an existing function in the measurement system, an extended analysis is possible here.

In software of telecommunication switches, a fault is categorised to be *fatal*, for example, if it causes a complete system failure (total stoppage) or cyclic restarts. This means that the exchange has lost its ability to handle traffic and this situation can only be changed by manual intervention. In the second category, *severe*, a fault causes large or small restarts or traffic disturbance on a single route or for a few subscribers only. A fault is categorised as *minor* when single units are blocked for short periods of time without any traffic disturbance. Also failures in documentation that cause handling errors and opinion reports belong to this minor category.

Hypothesis 9:

The most severe faults detected in the field represent a minority of faults.

Facts (Data set 14):

When a big sample of faults collected during 2 years is classified in more detail we observe that less than 10 % of faults represented fatal failures (Figure 5.22). The sample includes a number of software faults tied to modules, which have passed their 6 months after external release, accepted by the designer to represent a real fault, and thus caused a software correction in the module.

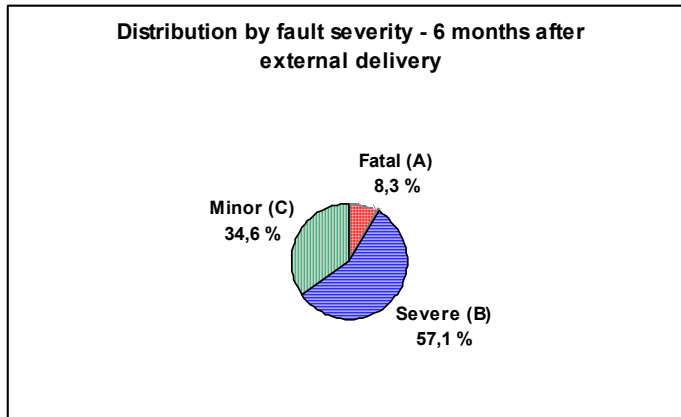


Figure 5.22 Fault classification by severity

Conclusions:

Our hypothesis is true.

Use:

This measurement result is useful in preventing and minimising the percentage of fatal faults on the condition that the reasons for and types of fatal faults are carefully analysed to learn how to prevent them. If the percentage of fatal faults is getting lower, the result can then indicate the positive effect of preventive actions. This indicator is more customer-focused than fault density measurement. The additional data related to modules having fatal faults can also be used in causal analyses of quarterly fault density figures.

5.5. Considering early design attributes

This part of the study mainly limits to a few *predictive* and *driving* measures based on data already collected until now. Four issues (Table 22) are discussed in Sections 5.5.1-5.5.3.

Table 22. Case studies considering early design attributes

Attributes	Measurement	View	Tool/method	Section
Fault content and SW volume vs. Complexity of software interface	Faults in Function Test vs. number of signals	Project, Module, Module type	Regression analysis	5.5.1
Percentage of faultless and faulty modules	Distribution of modules in Function Test fault categories	Module	Rough set analysis	5.5.1
Early detection rate	Inspection faults / faults in all phases	Project, Module	Bar	5.5.2
Effort and cost of detecting a fault	Preparation factor, Inspection hours, Man-hours / fault	Project	Bar	5.5.3

We move the measurement point to an earlier process phase and use collected signal and inspection data to study their relationships between results observed at later phases of the project. As the first driving attribute we selected the complexity of software interface to study its impact on the software volume and the number of faults detected in Function Test (Section 5.5.1).

The influence of module type (e.g. traffic handling etc.) is also studied. As the second issue we study the percentage of faultless and faulty modules by classifying signals and faults into four categories. Two last issues in Table 22 relate to the inspection process providing quantitative knowledge about the early detection rate (Section 5.5.2) and the cost of detecting a fault (Section 5.5.3). Because collected man-hour figures were available for the preparation of inspection, a study of preparation factor is included.

5.5.1. Studying impacts of number of signals on faults and volume

The complexity of a software interface in the AXE software is represented by the number of sent and received software signals. In the previous study (Section 5.1.1), the total volume of modules correlated significantly to the number of all signals. Instead of using all signals per module, as investigated in the related previous study, we focus on sums of new and changed signals collected from 6 projects finished after the previous study¹⁵. Normally the number of new and changed signals (SigFF) is known in an early phase and can easily be counted from Function Framework (FF) documents or from signal libraries.

Interdependence between volume and number of signals on project level

First we find out on project level how new and modified volume in statements correlates to total number of new and changed signals. The left side of Figure 5.23 shows a significant linear dependence because the correlation coefficient of 0,94 is greater than the critical value (0,8114 in case of 6 items). The correlation between Function Test faults and signals is also significant as shown in the right side of Figure 5.23.

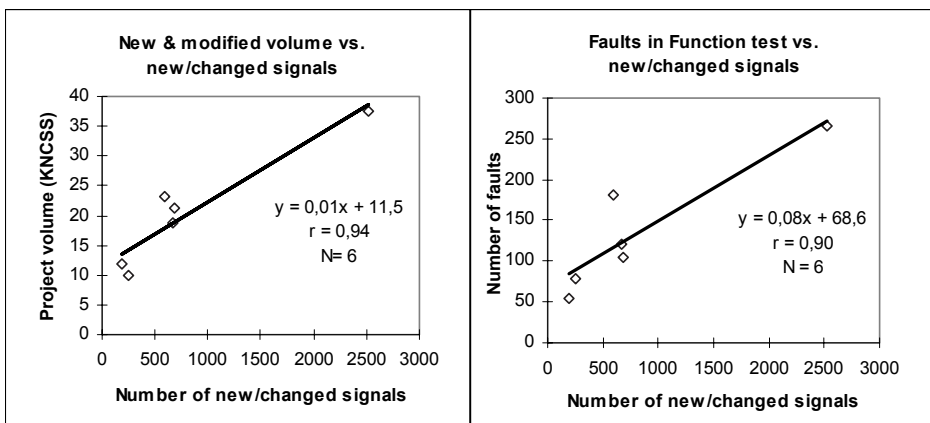


Figure 5.23 Relationship between volume, faults and signals

Signal studies on module level

Since another study (Ohlsson, 1996) has proved the number of new and changed signals (SigFF) to be a possible predictor for fault-prone software modules, we collected SigFF figures and compared them to faults detected from modules within a few Finnish projects (Data set 15). The type of module studied in Section 5.4.9 is expected to influence on results and the type is thus included.

¹⁵ Later projects only reported number of new and changed signals for each new and modified module

Facts (Data set 15):

Correlation coefficients of SigFF to Function Test (FT) faults, fault densities and volumes as dependent variable are presented in Table 23. Correlation was studied in four subsets: SigFF-T = SigFF figures in Traffic handling modules, SigFF-M in MAP, SigFF-I in I/O and SigFF-O in other types of modules, i.e. in the rest of modules (e.g. Coordination, Analysis). The SigFF appear to correlate best to FT faults and modified volume in traffic handling modules where signal communication plays a central role. The result is statistically significant because the values 0,65 and 0,63 for the correlation coefficient are greater than the critical value (0,4438 in case of 20 items). In Table 23 we also observe that correlation of SigFF to fault densities is very weak.

Table 23. The Pearson correlation between SigFF and variables

Variable	SigFF-T	SigFF-M	SigFF-I	SigFF-Other	SigFF-All
Number of modules	20	62	54	43	179
FT faults	0,65	0,30	0,18	0,12	0,26
Modified volume	0,63	0,57	0,43	0,49	0,47
FT faults/modified volume	-0,02				-0,09
FT faults/total volume					0,06

A study of signals and faults by category

As observed in this study, SigFF shows a clear dependence to Function Test faults. However, sizes and numbers of faults in modules were small, thus the prediction ability of SigFF on single module level is poor and variations are high. For this reason, a linear regression is not necessarily a best basis for prediction purposes. Instead, we classified SigFF's into "rough sets" as shown in Table 24.

Table 24. Distribution of modules in Function Test (FT) fault categories

		Distribution of modules in FT fault categories							
		0		1 - 2		3 - 4		> 5	
SigFF	N	n	%	n	%	n	%	n	%
1 - 5	49	25	51,0	16	32,7	5	10,2	3	6,1
6 - 15	46	15	32,6	15	32,6	9	19,6	7	15,2
16 - 30	42	11	26,2	17	40,5	4	9,5	10	23,8
> 30	42	5	11,9	14	33,3	8	19,0	15	35,7
Sum	179	56		62		26		35	

The number (N) of modules in each "rough set" is almost equal. In zero fault category the number (n) and the percentage of faultless modules is decreasing when SigFF is getting higher. In the rightmost column one can see an opposite phenomenon.

Use:

If the SigFF categories for modules to be developed within a project are known in early phase, then it is possible to identify modules, which need more attention on preventive quality issues.

Discussion:

In older projects, some SigFF figures may indicate a lot of new and changed signals but only a signal name has been changed. So, carefulness and tool support is needed in collection. All changes in a module are not affected by signals but other functional changes in code.

5.5.2. Impact of inspection process on quality**Improving early detection rate - project level**

In this section we first present some real measurement results on distribution of faults by phase including inspection. During past years many projects have stored inspection statistics on project level (Lahtivuori, 1997). Figure 5.24 shows some fractiles on percentual amount of major remarks. A risk relating to the statistics is that classification between major and minor remarks may vary in literature and especially in older projects. A traditional golden rule should be to find as many faults as possible in inspections. Early experience has shown software inspections to be a method finding 60-90 percent of all defects (Fagan, 1986). The following hypothesis can be stated based on this idea.

Hypothesis 10:

More than 75% of faults are found in inspections (regarding the whole project).

"Early detection rate" was one important quality indicator at LORAL (Section 3.1.3). The same expression was presented as "Inspection detection rate" at Ericsson (introduced in Section 3.2.2). Several studies report high rates, e.g. according to empirical results at Bull (Weller, 1993) inspections found 70 percent of all the defects detected after the code was completed. We based our hypothetical percentage of early detection rate on experience. However, we could not expect as high figure (85%) as LORAL because this company has achieved CMM level 5. Instead, we use 75%.

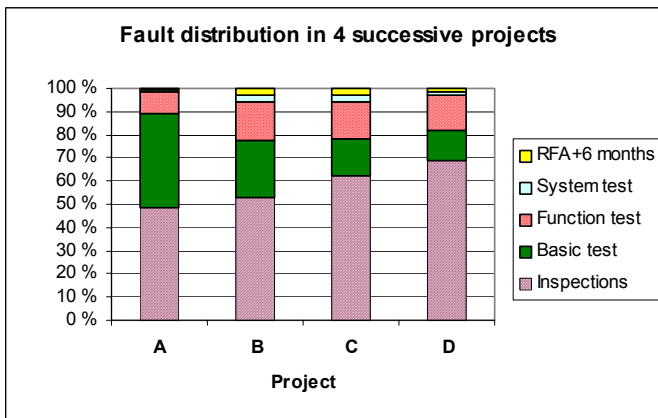
Facts (Data set 15):

Figure 5.24 Percentual distribution of detected faults

As data collected from practical projects (Figure 5.24) shows, the last project D has succeeded to find 69 % of faults in inspections whereas the other projects have found less. The early detection rate has all the time slightly improved in these successive projects (from 49% to 69%). However, in three last projects, the proportion of faults found in Function Test remains constant. In the light of this data set, also the percentage of summed faults found in inspection and Basic Test is also quite stable in projects B, C and D. Table 25 illustrates the relation of early detection rate to fault density in Function Test (FT) and during 6 months (6MO) after delivery.

Table 25. Fault detection in four successive projects

Fault detection rate	Project A	Project B	Project C	Project D
% detected in inspection	48,9	52,7	62,6	69,0
% detected in inspection and Basic Test	88,8	77,2	78,3	81,6
% detected in FT	9,6	16,8	15,6	15,6
% detected during 6 months	0,7	3,0	2,8	1,4
FT faults/modified volume	7,1	8,0	6,4	7,9
6MO faults/total volume	0,21	0,25	0,44	0,19

Some other empirical studies (Konradi, 1999) which also used Ericsson's material showed that 64% of total defects in the project were found in inspections. The percentage (64%) in this study neither includes defects found in code review nor minor defects. In another earlier case study (Ohlsson, 1998), 179 (57%) out of 316 major faults were detected during development. The percentage is lower despite of the fact that code review faults are included. Retrospective lessons learned from HP (Grady, 1997) reveal that inspections typically found 60-70 percent of defects.

Conclusions:

The hypothetical rate 75 % is still too ambitious to achieve, so our hypothesis is not true. Evidence is weak for any conclusion that early detection rate has a positive effect on post-release fault densities. In all projects we obtain low rates (< 3%) of detecting faults during 6 months.

Improving early detection rate – module level

It has been less usual to measure and study inspection data on module (source code) level. However, all modules should be inspected equally well in order to improve early detection of faults and stability of the inspection process. As the present measurement system only registered faults at post-release phases, we introduced also the measurement of inspection and unit test faults per module. It is then possible to compare faults found in inspections vs. later life-cycle phases, i.e. Insp, BT, FT, ST, 6MO phases for the same module. As some new data became available during the ongoing study, a question came up from an experienced section manager (Reiman, 2000) concerning the relation between faults found in code inspection and testing. The following hypothesis was stated.

Hypothesis 11:

A majority of implementation faults in a module are detected in inspection and unit testing; i.e. more faults are caught in code inspection and Basic Test than in Function and System Test.

Facts (Data set 16, Project D):

First, our results concern new modules developed in project D. We discuss new modules separately, because variations in modified modules appeared to be very high. Fault profile for new modules in project D (Figure 5.25) shows that overall the majority of faults is found in inspections and Basic Test. However, in 3 out of these 9 new modules the number of faults found in Function and System Test was higher than the sum of code inspection and BT faults.

For selected modified modules (Modification grade < 10%) the profile does not look that normal because of high column in FT. Many faults slipped to 6 months because System Test did not find all of them.

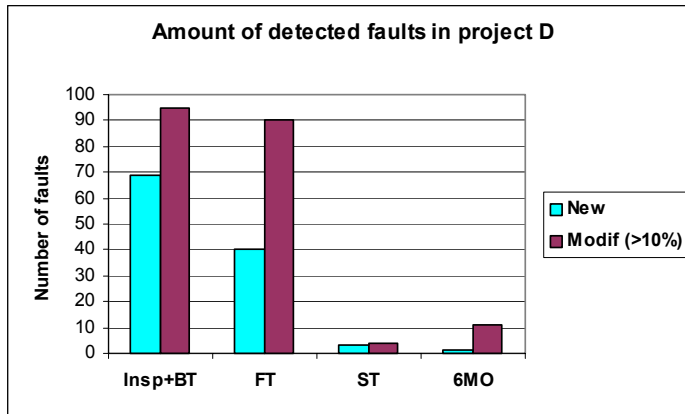


Figure 5.25 Total number of faults detected in different phases

A deeper insight into new modules within project D gives us results shown in Table 26. Let us denote:

Diff = Difference between majors/faults found in Inspection and Basic Test, and faults found per module in post development testing (FT+ST)

Cum FD = Cumulative (post inspection) fault density as faults found per module during post development testing and 6 months divided by module size (Faults / KNCSS)

Early detection rate for these modules representing negative Diff is naturally lower (32,4%) than in modules with positive Diff (73,4%). In three uppermost modules, where Diff is negative, we can observe a significantly higher post inspection fault density (Cum FD). Module D3 that contains the only fault slipped to 6 months period belongs to this category. Studying the modules modified more than 10% in project D we resulted in figures presented in Table 27.

Conclusions:

In new modules we get some support for hypothesis 11. In these modules we can also conclude that the negative Diff indicates higher post inspection fault density. Regarding modified modules, the results do not follow the stated hypothesis homogeneously.

Use:

The measurement can be used to evaluate capability of the inspection process to find faults early. Data to be collected from many modules (in the future) would provide more information about percentage of modules where the hypothesis is true. The profile shown in Figure 5.25 also helps to compare inspection results to amount of faults slipped through to the field. The profile provides a basis for root-cause analyses regarding modules where faults slipped from inspection to test phase. The result can also be used to identify possible risky modules (where the hypothesis is true) before delivering them to the customer. Aiming at positive Diff helps to decrease post inspection fault density and thus post development fault costs. A module is possibly a risky module when a majority of faults slip to test and operational phases.

Discussion:

In practice it is always difficult to make a distinction between major and minor defects. Some inspections tend to detect mostly minor defects. A high rate of detecting majors is an important goodness issue of inspection. Therefore, a possible new measurement might be the number of majors divided by the number of all defects. Experience from projects included in data set 15 shows that majors only represent 12-19% of defects (Lahtivuori, Karvonen, 1997).

5.5.3. About cost of quality assurance

Investing effort on inspections

Attention to inspection planning and execution can be improved in three ways by:

- Increasing planning and preparation effort prior to actual meetings,
- Ensuring that enough hours are planned and used for inspections in respect to overall project man-hours, and
- Supporting with proper tools.

One of the key issues is thus the invested effort. The four projects studied (Data set 15) also succeeded to collect inspection man-hours in order to calculate the following important factor:

$$\text{Preparation factor} = 100 \cdot \frac{\text{Preparation man-hours}}{\text{Preparation man-hours} + \text{Meeting man-hours}} (\%)$$

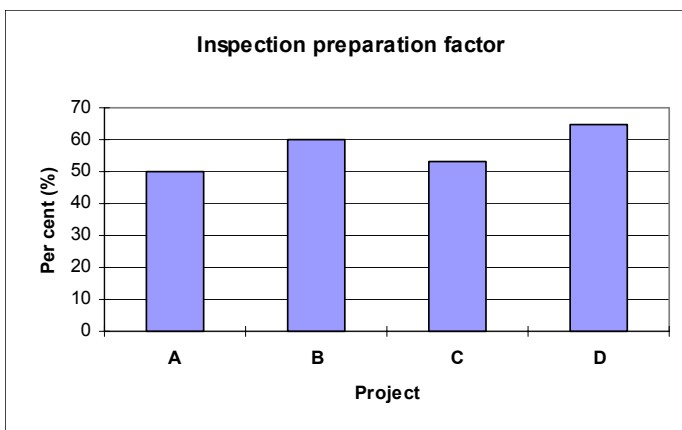


Figure 5.26 Improvement trend in preparation for inspections

Facts (Data set 15):

Results from our real projects show that preparation factor varies between 50 and 65 per cent (Figure 5.26). Results in other companies e.g. at IBM (Kan, 1994) provide some basis for comparison yielding 51 per cent inspection preparation factor for AS/400 computer system software. Variations of the preparation factor between individual inspections are high. Data collected from hundreds of inspections to computer based support system (WebLR) at Ericsson in Finland showed that preparation factor conformed to normal distribution the mean value being between 50 and 60 per cent (Jokikyyny, 1998).

Use:

The results can be used to show that improving the preparation factor helps to achieve higher early detection rate. This is a fact in case of project D as shown in Figures 5.24 and 5.26. The facts collected from real projects can be used for motivation e.g. in inspection training situations. Recently, the use of inspection preparation and meeting hour data was recommended for organisations beginning quantitative quality management (Paulk, 1999).

Inspection man-hours vs. project man-hours

Most projects usually register spent man-hours by phase and in total. As inspection man-hours are carefully collected it is possible to find out their relation to project man-hours per phase or to total man-hours spent. Facts from our 4 local projects in question are found in Table 28.

Table 28. Inspection effort in projects

Project	Project man-hours	Inspection hours	Inspection effort (%)
A	101266	5181	5,1
B	37059	2660	7,2
C	50711	2931	5,8
D	54528	3191	5,9

Project man-hour figures are taken from PQT and inspection hours from project Quality reports. The figures are fairly good, but some sources, like (Gilb, 1994), recommend even more: about 15% of project budget will probably be used when a company is mature. A classical article (Fagan 1986) mentions that typically, all design and code inspection cost amounts to 15 percent of project cost. Inspection hours are not completely accurate because, in project A, the man-hour data was not yet available in all inspection records. However, starting measurement of this kind of issues serve as basis for further planning and improvement of inspection activities. Inspection practices also change over time; for example project B introduced *weekly inspections*. In project D, a new inspection planning and recording tool was started for the first time.

Use:

The result can be used in project planning to ensure that enough hours are invested to inspections. A planning constant stating, that 5-7 % of project man-hours should be planned for inspections (see Section 5.5.2), can be used for similar projects.

Cost of detecting a fault

Even more interesting is to compare the cost of detecting a fault in inspection to costs in tests. Man-hours spent to detect a major fault for our four project ranges as shown in Table 29.

Table 29. Cost of finding software faults

Phase	Man-hours / major fault
Inspections	5...7
Basic Test	17...23
Function Test	83...111

The figures in Table 29 also show what inspections can pay-off compared to Function Test. It is much cheaper to find faults early. Many faults are introduced early, but detected late. The earlier fault analyses (Hirvensalo, 1990b) show that 40 per cent of faults detected in Function Test are introduced in function specification and detailed module design. Many examples of similar percentage (30-40%) can be found in Function Test reports in industry, e.g. at Ericsson. In another study (Ohlsson, 1996) a

conclusion was that 45 % of faults were introduced before coding phase. Ohlsson did not include the category "Where faults should have been found"¹⁶. For example in project D, 53% of analysed Function Test faults should (and could) have been found during inspections before coding. Because so many faults are present in earlier phases and they could have been detected earlier, it is reasonable to put more effort on inspection.

5.6. Opportunities to use measurement data for prediction of quality

Predictions are important for a project in order to monitor the project to meet the goals and to keep quality within accepted limits. In the previous section several opportunities to use collected data for prediction have come up. This section discusses and verifies a few possible practical approaches, techniques and models. The models provide prediction opportunities either on project level or facilitate prediction of fault content on module level. Some models are also able to identify fault-prone software modules.

5.6.1. Prediction opportunities on project level

Predictions based on fault distribution by phase within a project

Prediction of fault density for a project can be based on a known percentual distribution of faults between test and operational phases. Empirical distribution extracted from data existing in the quality measurement database (Data set 1 including the same 16 Finnish projects as in Section 5.4.2) looks as shown in Figure 5.27.

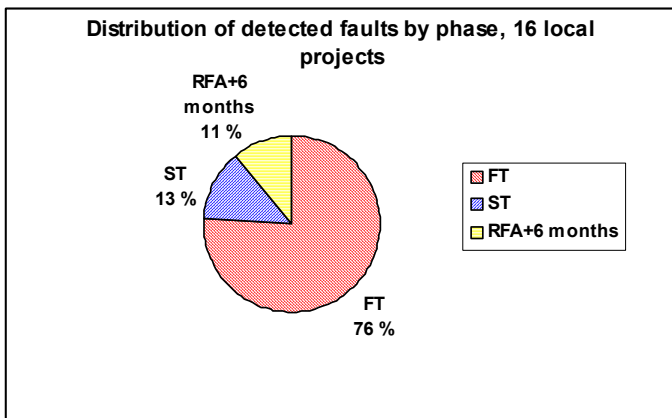


Figure 5.27 Fault distribution by phase

This knowledge can be used for prediction by estimating at first the total volume in KNCSS to be developed within a project. Fault density goal for RFA+6 months is used to derive the total amount of faults. From the fact that 75% of these faults should be found in Function Test (FT) it is possible to get an estimate for the number of faults to be detected during FT (in order to meet the goal for RFA+6 months) and use it for example as a test completion criteria. In order to clarify the method, the following example of steps is given:

1. Estimate the total software volume in KNCSS to be developed within project.
2. Identify the fault density objective in faults/KNCSS for RFA+6 months (as stated by project management).
3. Multiply the objective value by KNCSS to get the number of faults that are allowed to occur during 6 months.

¹⁶ This category is included in Ericsson's Function Test reports as a standard part of fault analysis

4. Calculate from historical data the distribution of faults by phase, e.g., 10% for 6 months, 75% for FT.
5. By using the ratio 10% and allowed number of faults for 6 months, determine the number of faults for all phases.
6. Use "known" distribution (step 4) of faults to determine an estimate for FT faults (e.g. 75% of all faults in FT) that should be caught in Function Test.

The above model can be called a *fault density prediction model*.

Predictions based on reliability growth models

It is possible to use the Reliability Growth Models (Conte, 1986) to forecast the latent errors left in software after testing. These models are mathematical expressions that describe a pattern of error discovery. Experience collected from software projects has shown that the number of errors as function of time fits more or less closely to a known statistical distribution, for example Exponential or Raleigh distribution (Kan, 1995). Choosing the right model needs measurement data from previous projects. This data is available on Test reports from start of testing up to internal release (PRA), but a new opportunity is to collect time based distribution even up to end of first 12 months after external release (RFA). Such distributions help to see per project the proportion of faults that are left in the field. This method is best suitable for big projects. The main assumption is that the defect rate observed during the development phase is positively correlated with the defect rate in the field. In software and in our case the defect rate is defined as defects per KNCSS in a given time unit (e.g., the first 12 months after delivery). In Ericsson case, for example, Figure 5.14 showed evidence for this assumption - the higher fault density in Function Test the higher is the fault density in the field.

According to Kan, the Raleigh model has been implemented in several software quality assessment tools, for example, in SAS and STEER, which are available in the industry. Software Productivity Consortium has also developed a PC based tool (Kan, 1995). An MS Excel based prediction tool supporting this Raleigh model based method is available at Ericsson in Canada (Miranda, 1998). This tool is a planning aid that is able to create a Raleigh distribution based on estimated fault content projecting it to desired intervals. The main advantage is that this type of tools can be used to monitor the project in order to keep the post-release fault content in desired limits (when the discovered number of faults can be compared to forecasted values). By using support tools for retrieving information from the Trouble Report database it is possible to automate collection of weekly actual figures.

This method is likely to be best applicable on the main project level because at the main level more statistical material than on subproject levels is available. The method seems to be most useful when earlier phases, i.e., defects detected in inspections, are also included.

5.6.2. Prediction opportunities on module level

In this section we discuss some prediction models which can be applied by using the measurement data already collected into existing databases. On the other hand, we analyse usefulness of a few methods that need new/additional data to be collected. The intention is to list possible predictors to be included in future measurements, not to develop any new prediction models. When feasible, some results from prediction accuracy (or comments on accuracy) are presented. We limit this discussion to models for prediction of low quality (stinkers) or risky modules. Models can be divided into two groups:

- "Check list " methods to predict and identify low quality modules
- Quantitative models based on one or more attributes, which are available before coding phase.

Check list methods

Check list methods are based on a questionnaire evaluating risks in different phases of development. We introduced a method where *attributes of the old base* module are first investigated. Second, the attributes of new design are audited based on a number of module specific questions. Analysis of attributes of the base module includes fault and correction history, readability of documents,

complexity (e.g. McCabe) and size of the module etc.¹⁷. In the questionnaire, many questions are stated concerning difficulty, goodness of structure etc. issues, which can be used to evaluate risks for quality. A good possibility to help prediction is to use results of statistical “stinker” analysis concerning old base products. Here the measurement data already existing can be used to identify occurrence of old stinker modules (see Section 5.4.6). The questionnaire includes numerous attributes concerning enhanced new design that is based on a certain old module. “Predictors” like the following are judged in the questionnaire:

- Experience of designers in function specification/design (who have designed input), and their participation and availability of their support in implementation of the module
- Optimisation in allocation of functionality and size
- Existence and complexity of alarm handling
- Amount of command parameters
- Difficulty/complexity of new design judged as a general “feeling”
- Number of new and modified signals.

Building in some scoring and weighting in questionnaires can make the checked things more measurable.

Quantitative models

Quantitative models enable predictions that are based on module attributes (driving attributes) available during early design phase. Driving attributes like size of software module, complexity of software interface, or number of conditions in Flow Charts, when measured, provide opportunities for prediction of fault content or other target attributes. Prediction can be based on simple mathematical formulas derived from regression models.

During the present study it was possible to collect data for a few driving attributes only, so opportunities to develop quantitative models were limited. Further analyses and studies based on measurement data to be collected in the future are thus needed to create complete mathematical models. For this purpose, we include an activity *Subsequent analyses and studies* in the utilisation process, which is defined in Section 7.1.2.

When the coding phase of the module is ready (but not yet function tested), modification grade and preliminary size can be used to calculate fault density for Unit Test. This early “forecast” of fault density can be used to identify possible bad software modules by running stinker analysis already during Unit Test phase (see Section 5.4.6). This would help to take some actions in order to prevent premature release of a risky module. During Function Test it is possible to make predictions repeatedly before the module is finished (i.e. released). This is done by collecting fault data accumulatively and by comparing actual numbers to “alarm” limits stated for fault count or fault density. Limits as such are very application dependent but determination of those limits can be based on analyses similar to those done in Sections 5.4.5 and 5.4.6.

Complexity of the software interface (e.g. in terms of new and modified software signals), categorised in accordance with Section 5.5.1, provides a possibility to predict the range where the number of faults in Function Test will fall. It is also possible to derive a regression model to predict the number of faults or project volume because correlation is significant (Figure 5.23). Prediction accuracy of this type of models, which are based on 1-2 variables, is rather low because so many other driving attributes (e.g. designer's competence, complexity of the problem) have an impact on the result.

5.7. Conclusions on case studies and improvement opportunities

5.7.1. Lessons learned

The first hypothesis stated that the amount of project faults correlates positively to the total effort. We

¹⁷ As a result of company-wide improvement project we made a suggestion to add this analysis to early System Analysis phase.

proved the statement that is also found in literature (Putnam, Myers, 1997), to be true with material of our case example.

Project effort in man-hours showed a good positive correlation to modified volume as our hypothesis 2 stated. When the modified volume is predicted then a simple formula is available for rough estimation of effort. We observed that this hypothesis was also true on module level in new modules.

Hypothesis 3, shorter planned lead times lead to better quality for the project, got some evidence from the performed study but correlation was not strong.

Hypothesis 4 stated: The modules that have been faultless in Function Test (FT) are also faultless in the field. The results clearly indicated higher post-release quality in modules that have been faultless in Function Test (FT). 70-80 % of individual modules that have been faultless in Function Test are also faultless in System Test and during their first 6 months after delivery.

A general truth in software community is that the higher numbers of faults in a module are found in Function Test, the more faults in the module also tend to appear in the field. The present empirical study of hypothesis 5 showed that it is true in practical projects. An interesting finding was that the modules, which have a fault density of more than 4 faults/KNCSS in Function Test have the highest fault densities during the first 6 months after delivery.

Already simple measurements provide possibilities to assess big issues like: have company-wide improvement actions affected positively on the total fault density (hypothesis 6 states yes). Three other companies Raytheon, HP and LORAL (Section 3.1) have presented similar lessons about decreasing trend of fault density.

According to the hypothesis 7 the lower modification grade, the lower number of faults per module is expected. Low modification grade (< 20%) of a module lead to higher 6-month post-delivery quality than in modules with high grade of modifications. In highly modified modules the quality is slightly lower than in new modules. A lower post-delivery fault density (during 6 months) can be achieved by designing modules either with minor modification or by making a totally new module instead of modifying an old module to a large extent.

Hypothesis 8 stated that fewer faults are detected during the second half-year period in comparison with first 6 months after external delivery. The performed study showed that in average, the 7-12 month's fault density is not lower than the 6 month's fault density.

Studying a sample of 2000 trouble reports received in the first 6-month field operation showed that less than 10% of faults represented fatal failures. So it is true that, the most severe faults detected in the field represent a minority of faults (hypothesis 9).

Hypothesis 10 proved not to be true for the investigated 4 projects. The early detection rate is lower than 70% while in organisations working on high maturity levels the rate is higher than 85% (Lee, 1994).

Hypothesis 11 stating that a majority of implementation faults in a module are detected in inspection and unit testing was studied separately for new modules and the modified ones in a project. We revealed that support for this hypothesis is best valid for new modules.

5.7.2. New opportunities to improve measurements

This section concludes some opportunities to improve measurements based on our findings in Chapter 5. Case studies provide a few issues that are useful to consider in improving measurements in Section 6.6. Some measurements presented in Sections 5.4 and 5.5 were taken into account in developing the new web-based version of the measurement system already during the present thesis. Some large projects made attempts to predict and identify occurrence of low quality modules based on theory presented in Section 5.4.6. Time based trend of cumulative fault density based on the case study in Section 5.4.7 was used in practice and included in quarterly Quality report. The second six-month period was also implemented in the measurement system. Slip-through presented in Section 5.4.8 was used as a performance indicator within the ESSI improvement project.

Occurrence of faults by registration date should be collected in order to see how well the data fits with Reliability models. This means a thorough collection of weekly actual fault figures in all phases of major projects including field operation. Using data from company databases to see how the methods fit with the reality can then test some models available. Installation and use of a proper tool helps both in projecting of estimates and handling of actual values.

Man-hours, both estimated and actual ones, should be measured in finer granularity than in existing measurements. For example, our case study in Section 5.4.2 argues for improvement in registering man-hours for unit design and Basic Test. Case studies concerning inspections in Section 5.5.2 have motivated the importance of careful registration of inspection man-hours. Therefore, we discuss an example implementation of improved inspection measurements (Section 6.6.2).

Influence of the old base quality on final quality of a module would be analysed easier if the faults detected in old base and the faults inserted in newly designed parts of the module are separated in the measurement database.

In the future it is useful to refine measurements expressing the complexity of software interfaces. Case studies in Section 5.5.1 and discussion in Section 5.6.2 gave some support for predicting faultlessness on a basis of new and modified software signals.

Some further background information in addition to hard figures would be extremely useful to store into measurement database. For example, information telling whether a module was developed as teamwork or as single programmer's work is important to register. Something more about test methods and test coverage would help to compare the projects in the right way. These are issues that have an effect on improvement of the measurement process (see Section 7.1.1).

There are also other aspects stressing our attention to improving the measurement process. Our experience has shown that quality, fault density and productivity need further discussion. It is not that simple to assign equal sign between quality and fault density as it happens often in practice. The meaning of quality arises questions like quality of what, quality of whose viewpoint, quality in what environment? If our focus is turning to the end customer (See also Sections 3.6 and 5.4.13), we realise that they do not care so much about fault density. They often care more about usability and ISP (i.e. availability and service performance). Quality of the entire system is their evaluation object. However, the fault content and fault density are useful internal measures for a software development organisation. These measures express fault introduction and removal capability impacting ISP, but basically they do not express product quality. Therefore, to avoid this type of problems, we pay attention to methods for planning and defining new measurements in Chapter 6. We are convinced that building up a systematic process for planning and defining meaningful measures helps best to improve measurements.

Fault density vs. ISP aspect is not the only issue that can be used as a basis in planning of new measurements. We can run the same reasoning for relevancy of productivity measurement that in the PQT system is based on lines of code / man-hour. Broader scope of productivity should be based on producing the desired result with the minimum expenditure of cost (effort). Therefore, the effectiveness measurement study presented in Section 5.1.5 turns our attention also to function points (Section 6.6.3).

5.7.3. Opportunities to improve use of measurement results

The data collected up to now into databases can be used for example in setting of alarm limits that are useful during software implementation and test phases.

Possibilities to use historical data for prediction are limited. However, we can point out that a simple method is better than nothing. Based on Function Test fault density it is possible to predict the 6-month post-release fault density, but only roughly because variations between projects are high.

Often, the measurement results indicate interesting deviations or facts, but reasons for them are difficult to trace. For example, the case studies done e.g. in Sections 5.4.6, 5.4.10 and 5.4.13 show that measurement results only provide a starting point for more detailed causal analyses. To improve utilisation we pay more attention on pre-analyses and availability of background information. This is to be taken into account in our improved Perform measurements process where a new activity is to be

added immediately after generation of measurement results, but prior to distribution of results. We refer to section 7.1.1.

Numerous other possibilities for utilisation of data collected similarly into databases than in our case examples are enabling further studies of effect-relationships like:

- Effect of planning precision on lead time
- Effect of planning precision on fault content after delivery
- Relationship between delivery precision and fault density
- Delivery delay relative to fault density during the first 6 months after delivery
- Lead time trends characterising e.g. "Time-to-Market"
- Quality in source code generated from High Level Design languages vs. ordinary "hand-made" modules written in high level language.

Based on collected simple data it is possible to see whether planning precision has any correlation to project lead time and fault content of modules developed in a project. Study of relationship between delivery delay and fault density is interesting because big delays often cause also quality problems. On-time delivery has traditionally been an important success factor in product development. It is interesting utilise data to study relationship between delivery precision and fault density. Lead time trends can be used for example to see whether the project lead times have been cut. Because of some inaccuracies of stored data information and of different scope of projects we did not include trend studies in our case studies. However, the lead time is a topic that is still interesting the management and improved measurement in this area should be used also in the future. Regarding the last item in the list above, we realised that it was not easy to compare quality of modules generated from High Level Design languages (e.g. HLPLEX that is mentioned in Section 5.2.2). Fault density is not a good measure for expressing quality, because generated source code is much larger in size than the corresponding hand-written code.

The present study supports the idea that the subsequent studies based on collected historical data provide the companies many kinds of "Experience models", as for example an *effort distribution model* that helps to allocate project man-hours for different *activities*. In Section 5.5 we have also shown that it is possible already from simple data to develop other models like Fault slippage models, Fault-effort models, Fault detection cost models, and Fault severity models. Those models can be used in project planning as well as in inspection and test planning. To create such models in practical life we need to include a visible activity for analyses and studies in the measurement process. This matter of generating experience models is to be considered in developing our Utilise measurements process in Section 7.1.2.

We need also better automation of the collection of data and the generation of results to improve the use of results. New technology provides good opportunities for automation of data retrieval and visualisation. We refer to discussion in Section 5.1.5 and Appendix 1 providing examples for tool activities. Our long experience has shown that development of measurement support tools and systems needs strong management commitment; otherwise the implementation of measurements may face problems. Therefore we saw it reasonable to outline general measurement implementation guidelines (Section 6.4.6).

Finally, we underline the importance of showing that the collected data is used and made visible inside and outside the company (i.e. disseminated). The researchers have also succeeded to use global database to prove some positive effects of CMM improvement efforts. An example comparison of fault densities and post-release faults (per module) between design centres by their CMM level was presented by Nilsson (1999).

Chapter 6.

Planning and defining new measurements

In this chapter we pay attention to two leftmost parts of our measurement model framework (Figure 2.4, page 11). As we aim at improving measurements we need a systematic approach how to plan and establish new measurements. First, we present the most common methods and discuss their utility value aspects. None of them are purely applied in this context, but the approaches provide elements that are useful in developing measurement practices in a company. Second, we discuss how quality models can help developers to achieve customer satisfaction. Third, we further look into measurements required when an organisation is climbing up to higher process maturity levels. Fourth, a measurement framework model is presented containing elements that are important both in the development of measurements and in the successful implementation of measurement activities. Based on knowledge discussed we suggest process models for well done planning and definition of new measurements. This chapter also includes a synthesis summarising the measurement subjects that are most useful in the future. In the rest of the chapter we finally make suggestions for improving measurements (e.g. indicators tied to processes). As some preliminary results have been available, for example Function Point based measures; these are included in suggested candidates.

6.1. Different approaches of planning new measurements

In this section we analyse methods and models that are useful in establishing new measurements.

6.1.1. Defining a set of key metrics

Many companies already follow a standard process model in their software development activities. As the management aims at establishing a company wide metrics program, an approach that we call here "Defining a set of key metrics", is available. In this approach, metrics are selected based on literature research and investigation into current metrics activities. Two examples from practical life are discussed below.

LORAL's way of selecting metrics

As a first example of this approach is LORAL's way of selection (Nusenoff, 1993). At LORAL a comparison to MITRE metrics (Schultz, 1988) was made and 10 key metrics were selected and introduced. The specification of metrics was a part of intensive implementation of the TQM program at LORAL. Mapping of SEI/CMM metrics collection requirements to the selected metrics was also made. The management metrics defined in this manner address to CMM levels 2 and 3. According to Nusenoff (1993), as a next step, the metrics required to advance CMM levels 4 and 5 were under further development at the time of writing the article. MITRE (Agresti, 1995) has presented a mapping of metrics to CMM levels giving a few examples of recommended indicators. We consider this knowledge in Section 6.3, where we will discuss typical measurements by CMM level.

A practical example of defining a set of key metrics

We present the definition of inspection metrics within Ericsson's System Software Initiative (ESSI). Using a similar approach, new metrics concerning measurements within the inspection process at Ericsson in Finland were also defined as a part of improvement program during 1996. Measurements were extracted from literature, adapted from other Ericsson design centres or developed basing on answers to a questionnaire made by the ESSI inspection team. The team resulted in an initial set of measurements expressing, for example, how many faults are found per hour, how many faults are found per inspected pages etc. In addition to definition of measurements the team made an initiative to develop a new tool for recording and analysing inspection data. In further detail, these results are discussed in Section 6.6.2.

6.1.2. GQM approach

Goal-Question-Metric (GQM) was originally developed at the University of Maryland (Basili, 1984, 1994). It is a method to address goals in a measurable way. Each goal involves aspects like Quality focus. The intention is to identify measurable goals, to set questions that provide information about the goal by clarifying which factors have a major impact on quality focus and how, and to specify metrics based on the questions.

GQM measurement tasks involve production of GQM models and Measurement Plans. Data collection and validation is performed with respect to the goals. GQM also aims at clarifying and communicating to the people involved, e.g., how to analyse and interpret data. Conclusions are drawn in Feedback Sessions. As a final step, results are packaged and process changes are proposed for future projects. GQM has obvious relations to utilisation of measurement results because the results are packaged and experience gained in form of models for future reuse. Ideas and models for how to implement improvements in future projects are stored in an "Experience Base".

In recent years, the use of GQM approach has increased in the software engineering community. Software Engineering Institute (SEI) has developed a guidebook (Park, 1996) that applies GQM. We can mention several examples where GQM approach has been used in the industry. HP has used GQM to define a set of maintenance metrics (Grady, 1987, 1992). The list of results consists of 5 goals, 31 questions and 35 metrics. GQM has also been used to some extent in electronic/telecommunication companies, for example in Motorola (Daskalantonakis, 1992), AT&T and in Nokia (Känsälä, 1996), to mention some companies. AT&T has used GQM for selecting inspection metrics (Barnard, 1994). Schlumberger Retail Petroleum System Division (currently Tokheim) has been active in adopting GQM measurements in industrial environment (van Latum, 1998). Dräger Medical Technology has also applied GQM in software development for new patient monitoring devices.

Ericsson has used it for several large and new projects. Several GQM Plans and measurement plans have been prepared and implemented in connection with major improvement programs (e.g. programs aiming to reduce the Time-to-Market (TTM) lead time. Some research projects (see Section 5.1.5, PROFES) have acted as a driving force for expanding the industrial application of GQM in Europe (Solingen, Derks, Hirvensalo, 1999).

Discussion

The method is probably most suitable when goals are set for new areas like process measurement. Quality management and project management can use GQM. It is intended to help to tie project measurements to both project and divisional goals - the task that traditionally has not been easy. GQM is a systematic method and requires that people involved are trained, motivated, well prepared, and have experience in using the method. It is important that one active and motivated person can act as a GQM champion during the program, supporting implementation of the plan, analysis and feedback sessions. A drawback worth mentioning is that GQM can produce plenty of questions and metrics. Setting questions and answers to the questions may also be subjective depending on the organisational tasks of the expert interviewed. Experience at Ericsson showed that GQM Feedback Sessions were valuable but even more time should be used for analysis of the results (Elf-Mattila, Hirvensalo, 1999).

6.1.3. MQG Approach

Bache and Neil (1995, pp. 59-68) have presented an alternative to GQM's top-down planning approach. MQG (Metric/Question/Goal) method is aimed at introducing metrics into an organisation for the first time. The authors suggest that managers should select a set of well-established metrics rather than resort to informally defining their own. The main idea is to collect metrics that lead to questions, which are then used to shape management goals. The three steps proceed as follows:

- **Metrics:** collect metrics and analyse the results.
- **Question:** ask why the results are like that, and if there is a need for improvement.
- **Goals:** managers are then aware of problems and deficiencies, and set goals accordingly.

Bache and Neil (1995) continue by characterising a well-established metric as having the following properties. For example, "It is well defined, objective and captures some well-understood attribute. It should be easy to collect, preferably it should be possible to automate the collection of data. It should have been well tested either in industry or at least in the research domain".

MQG focuses on investigating the current process in order to gain the understanding needed to set realistic improvement goals. The following benefits are found:

- Managers have easier choice over which metrics to use because the core set contains only well-established and defined metrics.
- By knowing the deficiencies managers have a practical starting point for setting goals and improving processes.
- Metrics from different sites can be compared because they will be defined in the same terms.

No references are available about the use of the MQG method. It does not appear much in the literature. To some extent Ericsson's PQT work resembles this approach.

6.1.4. Goal attribute measure (GAM) method

The GAM method resembles GQM and is based on Norman Fenton's (1991, 1995) ideas. It was developed at Ellemtel and it is at the moment owned by Ericsson Quality Institute (Nilsson, Rise, 1996). First, measurement customers and their goals are identified. Second, a set of target attributes, their driving attributes, and measurement objects are found out. These attributes are analysed in teams. Attributes are divided and decomposed into directly measurable sub-attributes and finally a collection of primitive and derived measures are defined by using specific templates (GAM Forms). Some parts of the GAM method, for example formalisation of measurement goals, are taken from GQM.

Rather than GQM, the GAM method addresses the evaluation objects. Examples of objects are products, process/subprocesses or resources. The GAM also focuses more on attribute structuring and definition than on questions. The method is in use e.g. at Ericsson Radio System AB and in Ericsson's SPI programs (Nilsson, 1999).

6.1.5. The Balanced scorecard

The model has its origin in strategic management (Kaplan, 1996). The Balanced Scorecard (BSC) provides a method to integrate company's strategies with measurements. The perspective is wide, considering financial, process development, innovation/learning, and customer etc. strategies. Strategic goals are decomposed into key success factors that would help to achieve the strategic goals. Key measures are set and a plan for action is created and followed up by using a coherent set of key performance measures. Management Dashboard is a way of presenting information to the management in connection with BSC.

Discussion

BSC is flexible and covers a wide range of measurements, including quality. Customer focus is strongly involved as one of the perspectives. Quality is usually one of the most important attributes in customer satisfaction. BSC can be applied also to drive software development processes. For example, NASA has applied Balanced Scorecard to the implementation of the strategic plan for a testing centre (Eickelmann, 1999). This strategic plan is very software-centred. The customer focus includes objectives to improve safety and reliability of the space shuttle mission program as well as to improve software quality. BSC in NASA includes both driver-oriented and outcome-based indicators (measures). Measures in the first category are called "Leading indicators" while the outcome-based ones are called "Lagging indicators".

BSC helps management to drive the company into desired direction and emphasises that it can and should be communicated on every level of organisation. Kaplan's model defines a basic structure for perspectives. The method, however, allows the user to invent their own perspectives, objectives and to plan new measures. BSC has already been used in R&D companies to monitor big strategic changes, but not widely used in companies producing software. European Software Institute (ESI) has build a Balanced IT Scorecard for Software intensive organisations (Buglione, 1999).

6.1.6. Metric development process

Metric development is based on a systematic standard process starting from customer needs, prototype development, analysis and feedback. This approach is used at Ericsson in the USA, where the process is called: "Software Metric Process Development Model". This process comes into play when the "measurement customers" have a metric that is needed. This metric can be requested by the line organisation or by the quality function at the management level. This request is reviewed for its usefulness to determine when the development of this metric will take place. The process for initiating metrics belongs to tasks of a specific metrics group that provides comprehensive metric services to Ericsson customers. These also include assistance in development and production of required metric services (e.g. for a certain project), and assistance in analysing the results.

6.2. Quality modelling as a means to clarify customer satisfaction

There are several quality models that are useful when attempting to break down product quality into customer oriented attributes and further into sub-attributes that can be measured directly from software. These models represent a wider quality view than just the code faults in software products. Typical models described in this section are summarised in Section 6.2.4.

6.2.1. Factor - Criteria - Metric (FCM)

This approach is based on tree-structured decomposition. Factors that normally represent external quality attributes (e.g. reliability, maintainability) are divided into software-oriented sub-attributes, called criteria (e.g. consistency, error tolerance, self-descriptiveness, modularity, and simplicity). A well-known early model of this type (Figure 6.1) was presented by (McCall et al, 1977).

The criteria are composed into metrics, which can be measured directly from software products or processes. The original model developed by McCall and Walters composed of 11 factors, 23 criteria and 175 metrics, so the full model is not shown in Figure 6.1.

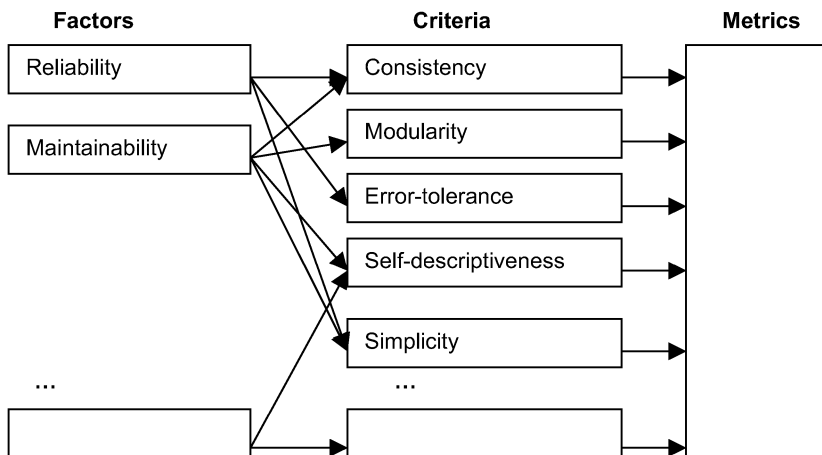


Figure 6.1 Principle of McCall's quality model

In studies at Technical Research Centre of Finland (Hirvensalo, Sepponen, 1983) we tried to adapt McCall's model for the determination and evaluation of quality of telecommunication switching system software. Lessons learnt showed that it was very hard to apply the model in full scope and for this reason the user should first make some selection of factors by importance in his product. The intention in collecting McCall's metrics is that values for certain metrics can even be collected from documents produced during specification and design documents, not only from program code. Even the developer can thus use certain metrics during the project for quality control. Metrics relating to program code like statement counts, looping, nesting, and Halstead metric used to measure conciseness depend on language and require implementation-dependent tools as well.

Numerous other similar models, like a model from TRW (Boehm, 1978), have been proposed, e.g., see a survey in Fenton (1995). These types of models are also known as SQM approach. The models break external quality attributes down into issues that help to understand which internal attributes have a driving effect on quality.

Models like these can provide valuable best practice metric information for example relating to module structure, effectiveness of commenting etc., but most metrics can not be standardised.

However, the quality models have been forerunners for standardisation efforts concerning factor and criteria levels. The FCM model has influenced international standardisation within IEEE and ISO. The standard "Software Quality Metrics methodology" (IEEE 1061) was published in 1992 and includes an appendix which describes the SQM approach. The approach is applicable in requirement specification phase to state detailed quality requirements. It was developed to allow the customer to evaluate the final product being delivered by a contractor. ISO9126 was developed for this purpose.

6.2.2. The ISO/IEC 9126 quality model

ISO and IEC have published an international standard (ISO/IEC 9126, 1991) called: "Software Product Evaluation - Quality characteristics and guidelines for their use". In this standard, the software quality is divided into the following six main factors: efficiency, functionality, maintainability, portability, reliability, and usability. The software quality model based on current ISO9126 is presented in Table 30.

The high-level quality characteristics provide a baseline for further refinement of software quality. This decomposition into sub-characteristics and metrics, the measurement method and rating is not standardised, but a proposal for definitions of quality sub-characteristics is given in Appendix A of the standard.

Table 30. The proposed sub-characteristics for each defined characteristic.

Characteristic	Sub-characteristics
Functionality A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that specify stated or implied needs.	Suitability Accuracy Interoperability Compliance Security
Reliability A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.	Fault tolerance Maturity Recoverability
Efficiency A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.	Resource behaviour Time behaviour
Maintainability A set of attributes that bear on the effort needed to make specified modifications.	Analysability Changeability Stability Testability
Portability A set of attributes that bear on the ability of software to be transferred from one environment to another.	Adaptability Conformance Installability Replaceability
Usability A set of attributes that bear on the effort needed for use and on the individual assessment of such use, by a stated or implied set of users.	Learnability Operability Understandability

The standard is intended for evaluation of quality of a software product from different points of view, not only the Users' view, but also the Developers' and the Managers' view. In spite of the fact that the six factors are clearly user-oriented, also the developer can utilise this standard. For example, the developer can evaluate the software specifications to see if they satisfy the customers' quality requirements during development and to evaluate the complete software package before its delivery. The six factors are to be considered as *external attributes* for software product quality. The current standard defines also a process model for evaluating software quality.

In 1994, revision of ISO9126 was felt to be necessary. The forthcoming version retains the same six characteristics but clarifies their relationships to internal and external metrics. According to the future plans, several supplements that define sub-characteristics, recommended metrics and their use in more detail, will be published. Preparing a new series of ISO/IEC 14598 standards is also going on in ISO/IEC JTC1/SC7, giving methods for measurement, assessment and evaluation of software product quality. A standard IS 14598-1 was published in 1999 and provides a general overview of quality characteristics and metrics. ISO9126 provides a software quality framework when used in conjunction with ISO 14598. In 2001, still a reorganisation project is ongoing to renew the structure of these standards to the form of a SQuaRE (Software Product Quality Requirement and Evaluation) architecture. We can conclude that ISO9126 is focusing on decomposing quality into measurable sub-attributes but the forthcoming ISO/IEC 14598 pays mainly attention on the process and the guidance needed in the application of metrics to the evaluation of software product quality. The relevance from the perspective of the present study is to consider the approaches of these standards in building measurements of customer-oriented quality attributes.

6.2.3. FURPS - an industry-oriented quality model

FURPS model (Grady, 1987) is used at HP. The model, that is later called FURPS+ (Grady, 1992), is based on a quality breakdown similar to the earlier models. Table 31 presents the five quality attributes and 27 sub-attributes of FURPS+. The five attributes on the left column of Table 31 are always used, but the selection of sub-attributes may vary depending on the project or organisation and are mainly used as a checklist. The model is used in two ways: to establish priorities and to define the priorities in measurable terms. FURPS+ aims at defining measurable product goals to achieve better customer satisfaction. Customer requirements are organised using the FURPS+ model.

Table 31. Structure of FURPS+

Characteristic	Sub-characteristics
Functionality	Feature set , Capabilities, Generality, Security
Usability	Human factors, Aesthetics, Consistency, Documentation
Reliability	Frequency and Severity of Failure, Recoverability, Predictability, Accuracy, Mean time to failure
Performance	Speed, Efficiency, Resource consumption, Thruput, Response time
Supportability	Testability, Extensibility, Adaptability, Compatibility, Configurability, Serviceability, Installability, Localizability

Quality Function Deployment¹⁸, (QFD) is applied to development of software products. QFD matrix diagram, that is used to map voice of the customer with current and new product features, helps the developer to focus on the primary customer desires. All customer requirements are weighted to represent very strong, strong, or weak relationship to product features. This relative weighting helps to quantify the level of quality desired and it is used in the requirement specification phase. Another typical use of QFD matrix is the setting of measurable objectives and metrics to be considered during each life-cycle phase. Thus, it relates to the topic of the present thesis and we use this method in Section 6.5.

6.2.4. Summary of quality modelling

Quality models provide a perspective that is useful in expanding our focus on most important quality attributes. Because the models include a wide range of attributes and sub-attributes it is not possible to implement all of them as a one-time effort within a product development project, but they must be considered by their importance case by case. These models also differ in definition and categorisation of attributes and sub-attributes. For example, in the FURPS model maintainability stands on sub-attribute level while in other models maintainability is a high level attribute. The FURPS+ model handles software and hardware together; for example mean time to failure is usually related to the entire delivered system. It can be concluded that the use of models like FCM might need a lot of tailoring and data collection effort to apply them in practice. In spite of the fact that these models have already been available for a long time, the approach has not been widely used in practice yet. However, a model like ISO9126, can be used already prior to a development project for identifying customer quality needs and especially for establishing the product quality profile. This helps to decide priorities of improvement goals and thus provides a good basis for planning of measurements. In the PROFES project (see Section 5.1.5), we piloted the use of the ISO 9126 framework in quality goal setting (Solingen, Derks, Hirvensalo, 1999). A created improvement methodology (PROFES, 2000) includes characterising and goal setting phases where ISO9126 model plays a central role supported by a goal-oriented measurement program. In order to get more attention to quality model approaches, the handling of quality attributes should preferably be integrated in the requirement specification phase of company's development processes and also communicated to customers.

6.3. Measurements supporting the attainment of high CMM levels

In this section the measurements that are necessary to exist on high CMM maturity levels are analysed in more detail. The discussion of the CMM requirements in Section 4.1 showed that it is important to consider CMM as a source of requirements for developing measurements. The analysis result in Table 32 is based on a careful review and deductions of the CMM technical report CMU/SEI-93-TR-25 (Paulk, Weber, 1993). CMM report only gives examples of suggested measurements because project environments in companies vary, and may lead to different measurement needs. The most thoroughly done mapping of measures to CMM levels is presented in CMU/SEI-92-TR-25 (Baumert, 1992). Looking over this document shows that a complete mapping is very extensive

¹⁸ QFD has its roots in Japan already during 1960s, used in requirements engineering also in software community. For details see e.g.(Sullivan, 1986).

covering thirteen categories of measures and tens of measurements by Key Process Area (KPA). The main difference between our analysis and Baumert's analysis is in use of references. Baumert's analysis is based on the earlier CMM version published in 1991, while the present study refers to later CMM version 1.1. (Paulk, Weber, 1993). Baumert's objective has been to provide software indicators that are consistent with the Key Practices of the CMM and that address the measurement related goals of the CMM key process areas. Baumert's document covers all KPA's in CMM and provides a comprehensive and cohesive set of indicators. Our objective is to find concrete example measures, not to define a complete set. So, Table 32 neglects for example the KPA "Subcontractor Management". We have added Function Points as an example of size measures, an item that does not appear in Baumert's mapping. Regarding levels 3, 4, 5; we limit ourselves in this context on quality measures including audits and reviews, trouble report handling and defect prevention.

Measurements required at CMM Level 2

Mapping CMM requirements to the measures to be selected is presented in Table 32 for a few level 2 KPA (Key Process Area). Measurements at level 2 are used for project management to estimate and track size, effort, cost, resources, schedules and risks. Data is collected and stored separately by each project. SQA (Software Quality Assurance) measurements are able to show that software quality assurance activities are well done. As required by CMM level 2, the SQA measurements mainly relate to schedule and cost status.

At Ericsson, these issues are typically reported in Project Progress, Test reports and Final reports. Some of the measures already belong to Ericsson's PQT, like lines of code, fault density and lead times.

Table 32. Typical measurements required at CMM Level 2 by KPA

KPA	Attribute	Example measures
Requirement management (RM)	Requirement stability	Number of requirement changes
	Requirement status	Status of requirement allocation in software
Project Planning (PP) Project Tracking and Oversight (PPT)	Size	Estimated vs. actual: Function Points Lines of code Number of requirements Number of pages
	Effort	Planned/actual man-hours
	Cost	Planned/actual costs Cost to date
	Quality	Number of Trouble reports open/closed Trouble reports vs. test cases
	Schedule	Lead time
	Status	Actual vs. planned milestones Percent tests passed
Software Quality assurance (SQA)	Status	Actual vs. planned number of audits Completions of milestones for SQA
Software Configuration Management (SCM)	Status	Number of change requests (CR) approved/ implemented/rejected Completions of milestones for SCM

Measurements required at CMM levels 3, 4 and 5

New issues at level 3, in addition to level 2 measurements are, for example, errors found and costs incurred by process activity, pareto analyses of defects and preliminary application of statistical control charts. Many kind of more detailed quality measurements come into play, like inspection and test measurements (Table 33). Faults found in inspections and tests are analysed by category and by severity. The analysis also covers identification of process activity where the fault was introduced. In addition to level 2 and 3 measurements, some new measurements on level 4 come in, for example, a deeper insight into process quality and quality costs (Table 33). Ability to perform quantitative process management activities on level 4 requires an organisation-wide measurement program including quantitative measurements goals, defined measurements, collection and analysis of organisation's

data and support tools. A standard set of measurements should exist and all projects should collect this standard set of measurement data. In quantitative software management product quality is measured and compared to the quality goals on an event-driven basis during projects. It is worth to mention that quantitative quality goal setting in CMU/SEI-93-TR-25 (see Paulk, Weber, p. L4-27) underlines the same quality characteristics than the quality models (see Section 6.2 of the present thesis). Organisation's process database also supports that quality goals can be defined for each life cycle-stage by making predictions based on process capability baseline and trends.

Table 33. Typical examples of quality measures required at CMM Level 3-5 by KPA

Level	KPA	Example measurements
3. Defined	Organisation Process Focus and Definition	<ul style="list-style-type: none"> -Organisation's SW process database is coordinated, established and maintained incl. definition of what data is to be collected and interpreted -Project's defined measurement plans
	Integrated SW management	<ul style="list-style-type: none"> -Project measurements are tailored from organisation's standard measurements -Projects record and share measurement data in the organisation's software process database
	SW product Engineering	<ul style="list-style-type: none"> -Number of Defects/Defect densities per product in inspections and testing -Number of TRs/faults compared with historical data -Number of modules containing/ affected by defect -Length of time TRs are unanswered -TR answer/fault correction times by severity -Defect type/category, activity where introduced, severity -Test coverage/readiness
	Peer reviews (SW Inspections)	<ul style="list-style-type: none"> -Number of inspections planned/held -Size and competence of team -Preparation times, Meeting times -Rework effort -Number of detected/ corrected defects in work products -Defects compared with/Fault densities per product -Pareto analyses of defects
	Training Program	<ul style="list-style-type: none"> -Quality of training program -Course quality profiles
	Intergroup coordination	<ul style="list-style-type: none"> -Cross organisational measurement team
4. Managed	Quantitative process management	<ul style="list-style-type: none"> -Process quality measurements including measured variation, control limits, expected mean -Process performance baselines -Causes of troubles -Test efficiency -Coverage and efficiency of inspections -Control charts on results of inspections
	Software quality management	<ul style="list-style-type: none"> -Quantitative Q-goals for SW products and measured actual progress towards achieving them in projects -Numeric values for product quality characteristics (reliability, maintainability etc.) -Cost of poor quality -Costs for achieving the quality goals
5. Optimising	Defect prevention	<ul style="list-style-type: none"> -Number of defects by type/category, life-cycle phase and activity introduced -Defect insertion rates -Defect category profiles over time -Time and cost of fault prevention activities -Time and cost to detect/correct a fault

On level 5 the organisation focuses on continuous improvement of the process. Measurement of process performance is made at key points in the process and used as input for changes of the process. Effect of changes made in the process is also measured. First of all, defect prevention needs predefined analyses to determine process-related causes of defects. Defect analysis is a routine activity with a purpose to eliminate defects caused by the process.

We can conclude that not so many new attributes and measures come up on levels 4 and 5 but the measured data is collected in deeper granularity and analysed statistically. Predicted data is also needed. Table 33 summarises the most common measures on upper CMM levels.

6.4. Measurement framework

Before going to measurement processes in detail we create a general framework model. The measurement framework helps to rationalise the handling of measurements in a company and provides guidelines for successfully launching, planning and establishing of measurement activities. It also helps to achieve comparability and avoid the misuse of the measurement results. In the methodological sense, the framework created is a new outline of our own that has been produced by constructing a collection of practical step-by-step instructions and processes. The framework is an adaptation of the model built by Ericsson's EPMG group that we participated in. The framework model shown in Figure 6.2 is used as a general umbrella for the issues to be covered in Sections 6.4.1...6.4.6, and also in Chapter 7.

General measurement implementation guidelines	Measurement concepts	
	Measurement processes	Defined measures
	Measurement definition templates and instructions	

Figure 6.2 Measurement Framework

In Section 6.4.1 we first refine the measurement concepts that are used in the rest of the thesis. Next sections 6.4.2 and 6.4.3 present essential activities for two created measurement subprocesses: Plan measurements and Define measurements. Sections 6.4.4 and 6.4.5 provide renewed templates of instructions and examples for defining new measures.

6.4.1. Useful new measurement concepts

In Section 6.1 we already referred to usefulness of new fundamental concepts that are necessary to introduce when improving a measurement system and developing new measurements. A few useful concepts have been presented during 1990's in connection with ESPRIT (European Strategic Programme for R&D in Information Technology)¹⁹ and in the latest books, like (Fenton, 1996) in order to understand better what each measurement characterises. These concepts, based on the latest applications of measurement theory to software measurement, are not yet widely used but their use is increasing in software engineering. In Section 1.1 we already defined the concepts like measurement, attribute and object and scale type. Below, the most important concepts used in the rest of this thesis are evaluated by their usefulness. First, we discuss about basic definitions like different types of measures, measurements, measurability and scales. Furthermore, we define a set of other advanced concepts grouped either into provisioning of measurements or into use of measurements.

Basic definitions

Measure (noun) is a concept that we distinguish from metrics. Metrics is widely used in software community. Metrics is a very general term that means the defined measurement method and the measurement scale (ISO/IEC 9126). Instead, we use the following definition: a *measure* is the number or symbol (or category) assigned to a certain object in order to characterise an attribute by making a measurement. See definition of measurement in Section 1.1. As an example of a measure, it can be

¹⁹ ESPRIT - the specific research and technological development programme in the field of Information Technology. ESPRIT was set up in 1984 as a cooperative research programme between European IT companies and academic institutions. It is managed by DGIII of the European Commission.

mentioned that months from start to finish is a measure for the lead time (attribute) of a completed project (object).

Measure type for a measure defined is either primitive or derived. If an attribute appears to be indirectly measurable, then it should be decomposed to one or several directly measurable (sub) attributes. A *primitive measure* is a measure that can be used for direct measurement of such (sub)attribute. A *derived measure* for an attribute requires that other (sub) attributes are measured and a value for the desired attribute is calculated (or inferred) from these. This classification is applied to our measurement definitions (see Section 6.4.4).

Measurement types can either be classified by measurability or by use. From the point of view of the measurability, their type most commonly classifies the measurements (or measures or attributes), either to *direct* or *indirect measurements*. Strictly speaking, only a direct measurement where a value can be given by a simple one-step measurement, is a real measurement. Classification to direct and indirect is important in order to distinguish measurement from calculation. A direct measurement of an attribute of an object does not involve any other attribute or object while an indirect measure of an attribute is derived (i.e. calculated) from measures of one or more other attributes. As the measurement types are classified by their use, two distinct types of using measurements exist (BOTH are needed!). First type is a measurement that can be used for the *assessment* of an existing entity by characterising one or more of its attributes. Second type of using a measurement is *prediction* of the values of some attribute of a future entity.

Scale types, including their definitions, exist widely in books handling statistical methods. Some early literature references in software engineering can also be found (Conte, 1986). However, these basic concepts have not been applied clearly in many existing measurement programs in industry. So was it also in our case examples in Chapter 3. We improve measure definitions by applying the known scale types. The list below includes a few examples of scale types:

- *Nominal* scale that means classification with no ordering (e.g. A, B, C) or labelling (e.g. faulty/faultless).
- *Ordinal* scale that involves ordered classification (e.g. criticality of faults). In the ordinal scale, arithmetic operations have no meaning.
- *Interval* scale that preserves order and differences but not ratios. It includes equivalent length and relative zero point (e.g. relative time). Only addition and subtraction are acceptable.
- *Ratio* scale where an absolute zero point exists, ratios between entities can be used, and arithmetic operations are meaningful.
- *Absolute* scale is based on pure counting (e.g. number of faults, number of statements) and all arithmetic is allowed.

For further details of scale types we refer to literature (Fenton, 1996).

Other advanced concepts

Below we define the concepts relating either to provisioning of measurements or to use of measurements (Table 34).

Table 34. Two groups of measurement concepts

Concepts for provisioning of measurements	Concepts for use of measurements
Measurement object	Evaluation object
Driving attribute	Target attribute
Measurement data	Measurement result
Validation of data and measures	Evaluation of measures
Measurement data provider	Measurement customer

Objects can typically be divided into two classes depending on whether they are objects of collection or evaluation. A *measurement object* is actually subjected to the measurement itself while an *evaluation object* addresses to an object in/of which the measurement customer needs to evaluate, control or improve a certain target attribute. Examples of measurement objects are a product, a process, and a resource. As examples of evaluation objects we can mention a set of products, a process executed by an organisation in the form of a project, and a resource (e.g. organisation, people).

Attribute, a concept that we defined in Section 1.1 is now refined considering its application to software artefacts. A *target attribute* is the attribute that the measurement customer needs to evaluate, control or improve. A *driving attribute* is the attribute that has a definite impact on the target attribute. Quantitative knowledge about influence of driving attributes is decisive when defining actions to improve or control a certain target attribute. Fault density, reliability, lead time, and delivery precision can be mentioned as examples of target attributes. Typical examples of driving attributes are complexity of a product, team competence, and effectiveness of a tool.

Measurement data provider is a person who provides measurement data and who has in his/her normal work a close connection to the measurement object. The designer, the tester, the project administrator, and the project quality coordinator are examples of measurement data providers.

Measurement customer is an individual or someone who is in need of facts about a certain object and uses the results of measurements. The measurement customer is also responsible for analysis of results. Different customers might be interested in evaluation of different target attributes. We introduce this concept because it helps to make sure that right measurements are handed over to right customers. Often, the persons like the line manager, the project manager, the process owner, the product manager, or the external customer may act as measurement customers.

Validation of data is of vital importance because the quality of collected data to be stored into databases must be good. Otherwise the user can not rely on measurement results. The way of validation is included in the definition of data and all collected data is to be validated against these definitions either visually or using build-in automatic checking. The person who makes sure that data is collected according to measurement data definitions (i.e. MDDs) should be independent from the measurement data provider.

Validation of measures is an important technique available that is not so often used in practice yet. It is useful when ensuring that the measure characterises the attribute in question. It can be applied as new measures are proposed or when the existing ones are to be evaluated. As there are two types of measuring, assessment type and prediction type ones, the validation activity is also useful for establishing the accuracy of the predictions. Fenton (1995, pp. 104-111) explains very clearly that "a measure is valid if it accurately characterises the attribute it claims to measure. On the other hand, a prediction system is valid if it makes accurate predictions".

Evaluation of measures is important, based on lessons learnt from the use of measures, to make sure that the measures are continuously useful, user friendly, well understood and not too hard to collect. We improve the measurement process to include evaluation of measures (see Section 6.4.3).

6.4.2. A proposal for a Plan Measurements process

As we evaluated the existing measurement process in Section 2.3.2 we concluded that activities for how to plan measurements were missing. Therefore, we started to outline such a systematic process that can be used for transforming management's goals and needs to measurable issues. The planning process shown in Figure 6.3 helps to link measurement planning to the aims that the management at each time needs to achieve. Our planning process takes the previously measured facts as one of its input in order to find right driving factors to achieve goals. In this sense we also thought it useful to add the cause-effect analyses as a feasible input.

Furthermore, we realised that the planning process should assist in widening the perspective whereas the previous PQT measurements were proved to be too narrow-focused (see Section 3.4.2). For this reason, we adopted elements from GQM (Goal-Question-Metric) and GAM (Goal-Attribute Measure) approaches, but use of these formal methods is not mandatory in full scope. The process is flexible, i.e. the first steps below can even be supported by using the BSC (Balanced Scorecard) approach. The main point is that planning proceeds in a goal-driven way. On the other hand, the approaches for planning new measurements that we discussed in Section 6.1, appeared to be quite separated what it comes to their application. Thus we thought it reasonable to make some integration and to result in a comprehensive planning process for measurements in the industry.

We also aimed at improving motivation and understanding that was felt to be a problem in our experience survey in Section 3.4.2. This aspect is implemented by involving the stakeholders already in the planning phase. The intention is that the planning process is run as teamwork. Finally, we aimed

at designing a process that applies modern measurement concepts provided by the framework discussed in Section 6.4.1. The most important items adopted from Table 34 are measurement object, driving attribute and the concepts for use of measurements.

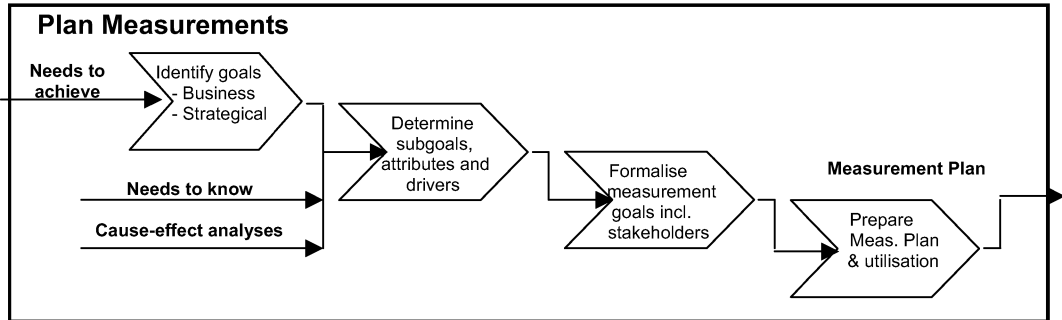


Figure 6.3 Measurement planning process

The Plan measurements process is composed of four activities: Identify goals, Determine subgoals, attributes and drivers, Formalise measurement goals including stakeholders; and Prepare measurement plan and utilisation.

Identify goals

Measurement planning starts with thinking what we really want to achieve, i.e. aiming at establishing a certain desired position. Identifying needs, established strategies and goals for business activities, projects, or products is a basis to break them down to measurable subgoals (or intermediate *objectives*). This activity is done e.g. in teams with upper-level managers participating.

Determine subgoals, objects, and attributes

The identified business goals may be expressed as very high level issues such as "Reduce Lead time" or "Improve customer satisfaction". For that reason, with goals identified, the next step is to determine what we really need to know, i.e., to achieve goals we need to know things (or properties) about objects of interest (also called entities). This activity aims at identifying subgoals that can be addressed by management or technical actions. Drivers, i.e., driving attributes, affecting fulfilment of each goal are also identified (if the subgoal is to improve, change or control something). Cause-effect analyses provide input for finding the right driving attributes.

Formalise measurement goals

We make a distinction between business goals and *measurement goals*. Formalising means that issues clarified above are translated into clearly stated measurement goals. Thus, this activity refines each goal (or subgoal) to cover the *purpose* why a certain *attribute* is measured from the *object* of interest, and who is interested in evaluating the results (stakeholder, i.e. the *measurement customer*) and in which *environment* the measurement takes place. Formalised measurement goals also help to improve utilisation of the measurements.

Prepare measurement plan

Further questions refining what one wants to know can be stated and listed regarding each measurement goal. One or several single measures (indicators) relative to measurement goals are identified in order to get answers to questions. The data elements (primitive data) needed to collect from the environment are listed tentatively. Finally, a *Measurement Plan* is prepared including collection procedures, sketches for graphical presentation of indicators, responsibilities, planned effort, schedule, intended use etc.

6.4.3. A proposal for a Define Measurements process

Already at the beginning of PQT, our case company had a formal way of defining measures. However, we did not have any systematic and described process to define measurements. Definition was in the hands of a dedicated corporate level measurement team and the commitment in subsidiaries was often poor. We also observed that the existing measure templates and instructions should be revised by exploiting the concepts provided by the measurement framework. One of the most important concepts introduced from Section 6.4.1 is the validation of measures.

To overcome the difficulty described in Section 3.4.4 stating that the line managers do usually not have enough time to read measurement definitions, our process involves managers (i.e. measurement customers) also in the definition phase. The define measurement process is intended for detailed definition of each of the measurement attributes (in accordance with the Measurement Plan), as well as for each data that is to be gathered. As a first step, the phase starts with gaining commitment and ensuring essential prerequisites to implement the plan. As a new issue, utilisation aspects are integrated to the last step of the measurement definition process (Figure 6.4).

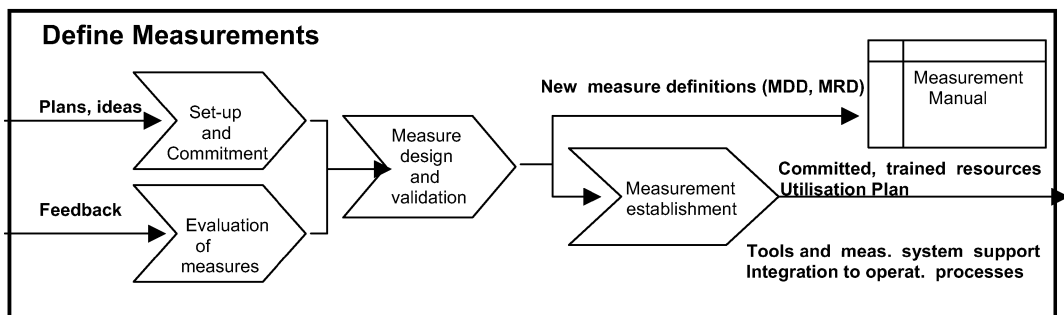


Figure 6.4 Measurement definition process

We also realised that an important activity is to evaluate the existing measures based on feedback from using them. Documentation aspects are covered by Intranet-based measurement manual in order to solve awareness and information distribution problems mentioned in Section 3.4.4. Correspondingly, the measurement establishment is attempting to ensure committed and trained resources for measurement activities, and integration to actual operative processes. Finally, for the sake of improving awareness for what and why are the measurements used, we saw it reasonable that the Utilisation plan (see Section 7.3.3) is to be completed in the final phase of the Define measurements process.

Set-up and commitment

A team including measurement customers (e.g. process owners) and quality or measurement experts is established. Measurement goals are validated with the measurement customer and the scope of the measurement plan is revised. Funding and resources are established (e.g., with an order letter) to define and introduce the planned measurements. Management's commitment is assured and the plan is approved.

Evaluation of measures

When a measure has been in use for a long time, it is possible to evaluate its usefulness based on feedback sent from *Utilise Measurement* process (see Section 7.1.2). A cost benefit analysis can be performed or the measurement can be prioritised to other measurements. A decision is taken or approved by the measurement customer.

Measure Design and Validation

Attributes should be decomposed and characterised in more detail. This step decomposes those attributes that are indirectly measurable to directly measurable attributes. Every primitive measure is described in detail by using *Measurement Data Definition* (MDD) that includes the method of

collection, triggering events, data provider, method of data validation, and tool support. Derived measures are described by using *Measurement Result Definition* (MRD) that includes the definition of result, measurement unit, calculation algorithm, presentation, measurement data used, and interpretation of result. Finally, MDDs, MRDs and way of presentation are validated with the measurement customer. Limitations in usage are explained.

Measurement establishment

In order to get data collected as documented in the MDD, establishment of data collection has to be done. This activity comprises appointing and training of persons involved, building tools and anchoring to other processes to make collection and validation activities as part of normal ways of working. Whenever MDD gives indication that something should be changed regarding processes, this is also implemented in practice.

As a second activity, establishment of result generation based on collected data is a necessity. Result calculations and graphs are to be "programmed" to make them available in databases. Other typical activities are selecting tools for data analysis (e.g. visualisation, statistical analysis), establishing threshold values and alarm limits, and guidelines for potential actions to take after analysis. Finally, a *Utilisation Plan* is prepared in the form of a detailed memo list (= whatfor, when, by whom and how are the results used - a template is available, see Section 7.3.3 and Appendix 12).

6.4.4. Measurement definition templates and instructions

In Section 3.2.1 and Appendix 4 we already discussed the concepts of MDD and MRD used as part of the existing measurement system. The templates for them need to be improved to get measure definitions to consider better the fundamental concepts like object, attribute and scale. A good practice is to support definition templates with filling-in instructions as follows.

Measurement Data Definition (MDD)

The purpose of the MDD is to provide a consistent way to describe the *primitive* measurement data used and to provide a brief standardised description of essential details of that data. Primitive measurement data is intended to show what data is to be collected from the objects to be measured. A primitive data is, for example, the number of non-commented statements, usually directly measurable by analysing the product (= object) to obtain the value. In the following we briefly argue the revised sections for MDD presented in Appendix 8.

The name of the MDD is preferably chosen to be reasonably obvious so that it is self-documenting. The name should contain the attribute of the measurement object to be measured.

Description of attribute gives an elaborate description of the attribute, providing enough information for those not familiar with the measurement. It is important to state what aspects and which object this attribute characterises.

Unit of measurement data is defined clearly. For example, the data might be Source Program Volume and the measurement unit could be defined as number of non-commented statements. This is similar to other measurements, such as distance, which might have a measurement unit of meters.

Purpose/Application of Measurement Data includes the purpose of the measurement data and the benefits of collecting and using this data. If it is not possible to provide a brief explanation of its benefits, then it is probably not a useful measurement data. No data is of any benefit unless it is known how to use it.

Scale type classifies measurement data to one of the following scale types: Nominal, Ordinal, Interval, Ratio or Absolute. See Section 6.4.1 "Measurement concepts".

Definition of Measure states the definition used to obtain the data, if any. This section is also used for describing how to support the different collection mechanisms proposed. Normally, this will result in a single description (or algorithm) which is used by all collection mechanisms. The algorithm for performing the calculation would be provided, or reference supplied (i.e. what to count what not to).

Method of Collection is expanded to cover *when* the data is collected, e.g., stating the event (name) that will trigger the collection and under what conditions a measurement data is collected. For example, data may be collected at release or at the end of each phase in the software design model. It is important to specify the person *who* is responsible for collecting the measurement data. This section also guides *how* the data is collected, i.e., what tools are used or steps taken to perform the measurement. Finally, *where/what* are the data sources (if they exist), that contain the information sought (e.g. name of database), is clearly stated.

Validation of measurement data section is added to describe *when* and *how* the quality of data (e.g. correctness of numerical values) is to be validated and who is performing the validation. This is important in order to ensure that data is consistent and reliable.

Measurement Result Definition (MRD)

The purpose of the MRD is to define an attribute of an evaluation object and describe how it is presented and to provide a brief standardised description of essential details, such as purpose of attribute, definition and interpretation of results, measurement customer, scope, and so on. The measure type is usually *derived*. A derived measure is one, which can be calculated from other primitive (or derived) measures, for example, number of faults per thousand non-commented statements. In the following we briefly argue the revised sections for MRD (refer to Appendix 9).

The name of the MRD is preferably chosen to be reasonably obvious so that it is self-documenting. The name should contain the *target attribute* of (or in) the *evaluation object*.

Description of attribute gives a more elaborate description of the target attribute, providing enough information for those not familiar with the measurement and stating what aspects and which object attribute characterises.

Unit of the measurement result is defined clearly. For example, the result might be Fault density in software design and the measurement unit could be defined as number of faults / kilo non-commented statements. This is similar to other measurements, such as measuring speed, which might have a measurement unit of kilometres/hour.

Scale type for the measure is mentioned if it is nominal, ordinal, interval or ratio. The scale types are selected from the same list as used for MDD. However, the scale type "absolute" is seldom used, because a measure defined on the MRD is usually of type "derived". See measurement concepts discussed in Section 6.4.1.

Purpose of Measurement on Attribute section is used to describe the main purpose *why* this measurement result is needed, and what the results should indicate.

Definition of Results includes the formula (mathematical operations) used to derive the result. The operations typically are averages, quartiles or percentages of some sort, but can also involve more complex algorithms like prediction.

Presentation of Measurement Result describes how the measurement result is presented. It mentions also when (events) the presentation will occur. When applicable, the result is presented in such a way that situations, which deviate from goals, are highlighted. Preferably, the section can include graphs that are used to illustrate the means of presenting the result.

Measurement Customer section is one of most important enhancements. It identifies the principal measurement customers *who* need to use the measurement result (who initiate result generation, perform analyses and define actions). Measurement customers' role is briefly described here.

Measurement Data Used section lists the measurement data and results used in calculating the measurement result. These primitive or derived data will consequently be described in further detail in the document Measurement Data Definition. The main argumentation is that the section lists the information to be captured when the data is collected and this additional information is used to put the measurement data in context.

Interpretation of Measurement Result gives hints and guidelines how the measurement result should be interpreted in order to understand reasons. It is important to list (in order) what the typical

actions/steps of the interpretation are and to tell how the measurement result is to be interpreted to see whether the process is under control or not (e.g. concerning the statistical techniques used). The section also refers to statistical techniques that may be applied and gives hints and guidelines on how to analyse the results and states who will participate in the interpretation activities (in addition to the measurement customer).

Scope of Application is intended to list the situations where the measurement result is valid and suitable for analysis/interpretation. We thus change the previously used MRD section heading "Application environment" to Scope of Application to emphasise also the cases (if any) where the measurement result is NOT valid and should not be used at all.

6.4.5. Examples of well defined new measures

In addition to instructions we also need model examples of well defined measures (i.e. filled-in templates). A central measure in quality control is fault content, but all trouble reports are not considered to constitute a fault. A fault content definition on MDD in Appendix 10 states what is counted and what is not counted. An example MRD concerns fault density in C/C++ software (Appendix 11).

6.4.6. General measurement implementation guidelines

This section provides some guidelines for important managerial and structural issues of measurements as well as for motivation aspects. In the following paragraphs we suggest the most important issues for successful launching, running and continuous improvement of measurements.

Management commitment and funding for measurement activities is a vital issue to get in order to avoid resource problems. A key success factor for setting up and running measurements in an organisation is continuous interest and support by the management. After all, the management should understand why to invest in a measurement program and what the value/benefit of the investment will be. This can be ascertained through a motivation presentation, briefing the basic measurement structure and plans, as well as reviewing advantages (e.g. how measurements provide visibility into improvement of business operations). Both top and middle management should be motivated. Finally, the main role of management is to provide resources and necessary funding for the effort.

Establishment of a measurement development team is a necessity. Development of measurements needs a daily driving force in the form of a team that should preferably be cross-organisational. Experienced persons, with well-defined tasks, who have energy to make measurements happen, are required. It should be ensured that there is some knowledge on statistical methods in the team. The team should also take care of the on-going measurements to see that they are run in a consistent way. The team should ensure that need for data quality is well understood, i.e., consistent over time as well as across the organisation.

Organising communication and training needs improvement as experts experienced it, for instance see Section 3.4.4. Regular training sessions should be organised to inform people about usage possibilities of measurements. Analysis results, improvement trends and success stories should be constantly communicated. It is also advisable that the new and existing software practitioners have knowledge of how measures are implemented (i.e. planned, defined, data collected, validated, processed and presented).

Architectural structure of measurements should be well planned. A modern measurement system should be build as an "open system" where practitioners and data analysts are able to take responsibility for their part. It should be ensured that the system provides real-time information, early warning signals and useful indicators to aid corrective actions. Information that is flexible and easily accessible is a necessity. Data collection principles include clearly defined points of collection, i.e., when and what is collected and who is responsible for providing the data. Data quality is a critical aspect of a measurement structure; thus, validation of data is of vital importance to include. The process for data collection should be event-controlled and also be automated as much as possible in order to ensure that collection is painless and accurate. Raw data which gets collected should be as

primitive as possible, for example dates instead of storing lead times, volume and faults instead of fault density etc.

Measurement database system/tool strategy implies important technological selections. Measurement tools play an important role in processing data collected from different sources. An organisation-wide database/warehouse supporting a well-defined *data model* and functionality for inputting data, storage and retrieval of results is of the utmost importance. The strategy might be to choose a system solution from a vendor or to apply own in-house Information systems/Information technology and competent resources. Whatever system is chosen, it should not be a too advanced system in the beginning. Other useful facilities are Internet/Intranet technology for collecting/sharing/publicising of data and statistical tools for analysing the results.

Other important general issues to ensure are integration *to organisation's work practices and usage aspects* (e.g. identification of main users, and realising the use of measurements in key decision making forums). These aspects are discussed in detail in Chapter 7.

6.5. Synthesis

Based on the requirements, lessons learnt, needs that have arisen and on the technological trends presented in previous chapters, it is useful to make a synthesis. Two main questions are discussed in this section. First, in Section 6.5.1, we try to answer to the question “what is good and transferable from “old” measurements for future use?” Second, in Section 6.5.2, we make a vision by keeping in mind the question “What new measurement subjects are important in the future?”

6.5.1. Inheriting issues from old measurements for future use

In this context we realise both a few useful attributes that could be inherited from old measurements for future. We also discuss legacy of routines because they as such are often transferable. Regarding useful attributes, we realise first that *fault content* is still a good application independent and simple quality measure in the future. Second we conclude that *Trouble Report answering performance* remains as a proper indicator of maintenance operations. We can list many routines, which are inheritable and useful to retain, such as:

- Systematic data collection and regular use of measurements in Management Reviews,
- Event-based collection tight to work product checkpoints in projects,
- A limited set of well-defined metrics documented in Measurement Manual (preferably web-based),
- The facilities enabling useful calculations per project but also allowing production of comparable results on organisational levels,
- Good support and close co-operation with those who are responsible for providing data, and
- A well-managed working group for requirement specification and development of the measurement system.

6.5.2. Important new measurement subjects in the future

We apply the QFD-based method here (see Section 6.2.3). Table 35 summarises *what* to consider in specifying and evaluating new measurements.

The vertical columns in Table 35 show *how* the different issues are implemented by introducing measurements. Implementation proposals for them are discussed in the remaining sections of this chapter.

Table 35. A synthesis of needs, desires, requirements and technological trends

WHAT to consider	HOW to implement	New measurements															
		Process change measures	Key Performance Indicators (KPI), and Dashboards	Measurements tight to new process models and languages	Defect prevention measurements	Inspection measurements	Time-based distribution of faults	ISP events, measures for attributes driving the ISP	Customer satisfaction profiles	Measurement of most frequently used functions	Cost of poor quality	Process oriented measurements	Proportion of faults detected by the customer	Customer-oriented product quality attributes	Function point based measures	Complexity measures	Competence profiles, personnel turnover etc.
Influencing In-Service Performance (ISP) in software design				x			x										
Application of Reliability Growth models						x	x										
Relations to customer satisfaction								x				x					
Requirements from CMM level 3					x												
Requirements from CMM level 4					x					x	x		x				x
Requirements from CMM level 5	x		x	x						x	x		x				x
Requirements from TQM		x						x									
Quality models, ISO 9126													x				
ISO 9001, ISO 9000-3										x	x						
Strategies to enable successful transfer to new technology areas (e.g. IP based mobile)		x	x														
New types of software development (e.g. OO, Incremental Design)			x														
New test methods (e.g. Statistical Usage Testing)						x			x								
Complexity of the problem / software																x	
Functional content of the problem / software														x			
Need to show measurement results in brief for management		x															
Quality Awards		x					x				x						
Measurements on-line during projects				x			x				x			x			

The proposals should be understood as typical examples because we keep in mind the golden principle included in our *Plan Measurements* process in Section 6.4.2:

**Actually, the primary question is not “what we should measure?” or
“What metrics do we want to use”
BUT
“What we really need to achieve?” and then consequently:
“What we really need to know and learn”**

A complete set of measures is thus not meaningful to give in this context because the measures are so much dependent on company specific goals and information needs. An organisation that has stated a goal to improve In-Service Performance (ISP) might choose measurements for attributes driving the ISP. As we observe quite many things are affecting on implementation of measurements. Our intention is thus not to go through how to implement all measurements that are mentioned in Table 35. We instead make a few general proposals, guidelines and examples for implementation of measurements.

Measurement attributes to be presented in succeeding sections are however tied to hypothetical goals, purpose and benefits in the form of an example of a Measurement Plan. Some measures are defined in a new formal way. We thus make principal use of the Plan Measurements and Define Measurements processes (see Sections 6.4.2 and 6.4.3).

6.6. Suggestions for improving measurements

This Section discusses several suggested improvements. First, in Section 6.6.1 we take a look at process measurements. Their importance came up already in Sections 3.4 and 3.5. Process measurements are essential in achieving high levels in accordance with CMM and also in ISO 9000 standards as discussed in Chapter 4. Our discussion is limited only to software engineering processes. Second, we take a closer look at inspection measurements suggesting a few practical measures that can be used as performance indicators for the inspection process (Section 6.6.2). Data was available from projects and we also used that data in case examples given in Section 5.5.2. Third, we make proposals for improving project measurements. We discuss introduction of Function Points, and measuring risks, and requirements. Section 6.6.3 also discusses issues to consider when the design process is changing to object-oriented design. The fourth topic is based on our experience on quality cost measurements in a software design process in the long run. Section 6.6.4 is basically a kind of contribution to refining quality cost components that are visible in a software process. Other suggested improvements, which we present in Sections 6.6.5...6.6.7 concern for example ISP measurements, customer view and human aspects. We go deeper than in Sections 3.6 and 5.7.2.

6.6.1. Implementing process-oriented measurements

Process-oriented measurements are measurements showing how well the organisation or user follows the process definition. Another motive is to indicate process efficiency. In order to be able to define measurement points and measures, the processes and subprocesses must be defined (i.e. as in accordance with CMM level 3). Based on our own experience and requirements analysed in previous chapters we propose a set of measures which are generally usable in all software engineering subprocesses:

- Detected fault content
- Slipped fault content ("slip-through" of faults to the next subprocesses)
- Effort spent in person hours
- Number of requirement changes
- Lead times
- Measures that indicate the efficiency of (sub)process
- Process adherence (how well the process definition is followed)
- Effectiveness of process specific tools.

Inspection measurements are typical examples of process-oriented measurements. We selected inspection measurements as examples for process-oriented measurements because we had an opportunity to participate in the ESSI improvement project, piloting of a new inspection tool was going on, and the case company put high attention on inspection process.

6.6.2. Inspection measurements – an example implementation

Inspection process is to be measured in order to assure that this important activity is well performed. The two key issues of performance are:

- Defect detection rate** - how well the time used for inspection is exploited
- Defect removal rate**- how well the inspected items have been checked

We focus on those two performance indicators that have been developed by the inspection team. *Defect Detection Rate* (DDR) is defined as:

$$DDR = \frac{F}{T}$$

where F means total number of major faults found during inspection,
T means time used for preparation + inspection by all participants in total.

Defect Removal Rate (DRR) is defined as:

$$DRR = \frac{F}{P}$$

where F means total number of major faults found during inspection,
P means number of inspected pages that were checked during inspection.

We propose the following principal way of presentation to visualise those inspections that have both high removal rate and high detection rate (Figure 6.5). The black “dots” in the upper rightmost corner of Figure 6.5 should represent the “high quality” inspections. The principle was implemented in the computer supported inspection process including a web-based tool called WebIR (Jokikyynty, 1999).



Figure 6.5 Defect removal vs. Defect detection rates in inspections

As we validate the usefulness of these measures above a strength is that the metric values can be collected immediately after termination of the inspection meeting. However, these measures pay no attention to how many faults were not caught (i.e. escaped). Therefore, a real *Early Detection Rate* (EDR) is defined as:

$$EDR = \frac{F}{F + FE}$$

where F means total number of major faults found during inspection,
FE means Faults “escaped” to Basic Test, Function Test, System Test and to the 6-month operation period after external delivery.

The main weakness is that EDR is possible to calculate only after termination of delivery + 6 months when all known faults, concerning the inspected product, are collected. Thus, the feedback loop is very long. However, EDR is a typical assessment type measure that is necessary. The measure definition equals to the existing inspection detection rate presented in Section 3.2.2.

Inspection performance measurements help to improve the **inspection process** so that as many faults as possible are detected and removed from the product (product improvement). It is also important to utilise the data collected from inspections to find defects in the **software design process**

to eliminate the cause (i.e., use for process improvement). Therefore some new fault codes and analyses are useful already in inspection phases.

If an organisation has a goal to improve the inspection process then also other measures in the area of inspections are to be implemented. The following additional measures are proposed to be systematically collected and analysed:

- *Preparation factor* that was defined and augmented in Section 5.5.3, shows the time used for preparation in relation to total inspection time (inspectors' preparation hours+meeting hours).
- *Defect severity* distribution as percentage of major/all inspection defects helps to control that as high proportion of major defects as possible are caught. Experience at Ericsson has shown that some defects classified as minors are in fact majors (Jokikyyny, 1998).
- *Size of inspected object* that is usually expressed in number of new or modified pages. In case of modified objects the designers should be very careful that they correctly report the amount of changed pages otherwise there is a risk of wrong results. This information can be used to calculate Defect Removal Rate (DRR) specified above.
- Other attributes that are interesting, for instance the number of inspectors, the experience of inspectors and, the duration of the inspection meeting.

6.6.3. Proposals for improved project measurements

Some measurement issues are strongly tied to a project, i.e., the measurement object is a certain project or subproject. Below, measures, which can be used during the ongoing project, are discussed and some ideas for new measures are given.

Introducing new Function Point based measurements

As generally known already long, software size measurement based on lines of code or number of statements have their weaknesses. If the goal is to improve productivity and estimation accuracy then the evolution that has taken place in the Function Point counting provides a new opportunity to move the approach in program sizing from "implementation" volume to volume of "functionality". Traditional Function Points, which were available late 80's, were hard to interpret, calculate and they were not well suitable for real-time applications. So, it appeared to be a too ambitious goal to get, for example, all subsidiaries in a multi-national telecommunication company to collect Function Points in a consistent way. However in late 90's, significant and promising solution, e.g. in the form of Full Function Points provided by COSMIC²⁰ consortium, and is available. COSMIC-FFPs, as they are named, enable collection already before coding phase in several checkpoints during a project. The first checkpoint is after completion of requirement specification. It is easy to automate counting e.g. from diagrams (models) written in UML (Unified Modelling Language), to mention one advantage. The Functionality Increase Indicator presented as a related study (see Section 5.1.5) instead required expert interpretation of calculation rules and needed manual collection from function specification documents. A company who aims at establishing COSMIC-FFP as a measurement method, should, however, do some piloting in practical projects to convince about their usefulness.

Measuring risk management

One of the success factors for a project is a well-managed risk handling. Measurements can serve this purpose. Risks *foreseen/overcome* can be measured during projects. After completion of different project milestones it is also important to recognise *actual risks* that became true. Risk measures were piloted during the present study and observed to be useful. A project maintained a risk register on INTRANET to follow risk status. Risk measures included in the GQM plan of a project related to a question "How well the risks have been managed?". Metrics are as follows:

- Degree of risk realisation (%) = $100 \times \text{risks that came true (fully, partially)} / \text{risks foreseen}$
- Risk severity and likelihood to occur (low, medium, high)
- Risk status control ratio = $\text{Preventive actions taken} / \text{actions defined to overcome a risk}$.

²⁰ Common Software Measurement International Consortium

Measuring requirements

Mapping to CMM models helps to identify some important measures. One of them is the number of changes due to requirement change requests (CR) during the project. *Number of CRs* was included in a GQM Plan (Hirvensalo, 1999), as well as the number of *estimated* and *actual hours* for raising from the CR. In addition to the “GQM” question “How stable are the requirements” is the question. “How good are the requirements”. A subjective measure for *clarity of requirements* is suggested. The project members can give their estimate for clarity in a nominal scale: Fully clear, clear or not clear.

Time-based distribution of faults

Distribution of faults by trouble registration date, for example weekly flow of incoming faults, is useful information for calibration of *Reliability models*. As shown in Section 5.6.1, this information could be used for prediction. The data should be made easily retrievable from trouble report handling systems to get weekly flow covering all phases of project life-cycle plus the operational phase.

Considering new process models and languages

During the present study we made some planning on measurement of Incremental Design. Data from module increments must be collected per increment as for example number of faults. On project release only the final size of the module is taken into account and fault over increments are summed up. In the future, incremental design will be applied also in the area of object-oriented (OO) design. Several literature references are available concerning OO metrics, see e.g. (Henderson-Sellers, 1996). A reference showing usage of OO code metrics is found at NASA/SATC²¹. Besides different way of design, the size (volume) measurement faces problems. We introduced Measurement Data Definitions for volume and Modification grade for C/C++ and JAVA programs. However, the simple line counting based volume appeared to be not feasible for C++. A better solution is that measurements should more consider C++ classes, methods, inheritance, include libraries, number of children, coupling etc. to get more information.

6.6.4. A proposal for improved quality cost measurement

Literature concerning quality, for example Juran (1980), Arthur (1993) and Ericsson's EQ (1989), accept widely that quality cost can be divided into the following three classes:

- Preventive “costs”,
- Quality control costs, and
- Fault costs.

Preventive “costs” represent rather *preventive quality investments* than costs. Quality control costs are often called *appraisal costs*. In connection with this study it has been refined what to include in each of the three classes. **Preventive quality** work includes quality planning and support, development of the operations system, improvement of methods, tools and processes, improvement projects or programs and tasks arisen in quality audits and assessments and analyses in order to prevent faults. Some sources include also training (Arthur, 1993), but in this model the training is excluded because it is difficult to make distinction between normal basic training and education having a particular impact on quality. **Appraisal costs** include design inspections and code check, reviews, audits and assessments, test work and its preparations, test configuration management (TCM) etc. **Fault costs** are divided into internal and external ones. Internal costs include the correction of faults discovered in tests and inspections by the developer and rework to be done due to changes. External costs cover all redesign, retest and correction due to faults/failures reported by the customer after delivery.

A proposal for a new way of visualisation of the Preventive quality investments in relation to Fault costs and Appraisal costs is modelled in Figure 6.6. From this type of illustration it is possible to see whether an increasing investment in preventive activities has a positive impact on reduction of total quality costs.

²¹ NASA Software Assurance Technology Center, see <http://satc.gsfc.nasa.gov/metrics/codemetrics/index.html>

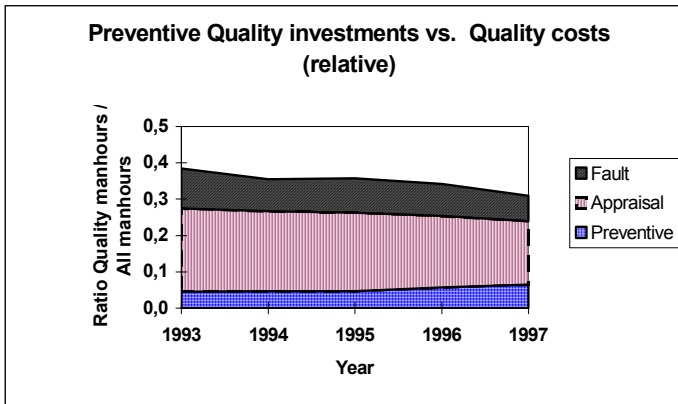


Figure 6.6 Visualisation of quality costs

Furthermore, the calculation routines needed to collect man-hours from the time reporting system are to be specified and implemented. Usually these systems are very company specific, but by using proper activity codes it is possible to separate man-hours into the three classes mentioned above. Man-hours in each three main classes can be summed up on upper organisational level from department reports and multiplied by average man-hour price.

6.6.5. Proposals for ISP measurements

As identified in previous chapters there is a need to get measured facts from influence of software design on ISP (In-Service Performance). Traditionally, telecommunication manufacturers and operators have been able to measure performance of the entire system by using software-related indicators for availability/reliability performance, for example (Ericsson Telecom, 1992)

- Number of Complete Exchange Failures (CEF)
- Number of major and minor restarts
- Number of major restarts with reload
- Subscriber line failure intensity.

ISP is continuously relevant also in new telecommunication equipment in future communication networks. Design is of vital importance to ISP. Some senior experts at Ericsson have presented that only design can reduce system down-time. However, it is not easy to find driving factors because there is a complex impact relation between ISP and design process/technology attributes. Thus, some sort of dependency modelling is necessary. During the present study we have identified a few useful modelling techniques to find design effect-ISP relationship, among others, the Product/Process Dependency (PPD) Modelling that was developed in PROFES project (Birk, 1998). PPDs concerning e.g. reliability issues involves a possibility for a company to build-up a repository containing experience of process/technology impacting drivers for ISP. The PROFES Repository of Product-Process Dependencies available from PROFES [www page \(http://www.ele.vtt.fi/profes/\)](http://www.ele.vtt.fi/profes/) is a good basis for an well-organised way to re-use and update knowledge on driving factors by subprocess. It is a responsibility of an organisation to prioritise the key drivers and define measures for them.

The models for a certain application can properly be created and updated in ISP improvement programs with experienced operational support and design people participating. We suggest a few measurements, which already during software development could focus on driving factors influencing the ISP, such as:

- Awareness and use of critical design rules is assessed by using simple measurements.
- Possible "stinker" modules are monitored by measured as in Section 5.4.6.
- Quality of old base modules is assured as discussed in Section 5.6.2.
- Effectiveness of inspection and test teams is regularly measured.

The reality should also be measured afterwards as feedback from field operation to development organisation. We recommend the following:

- Measure the frequency of disturbances on ISP per module (tools that are able to trace disturbance-module relationship are used).
- Count and analyse visible consequences (e.g. restarts) caused by each software module.
- Use new FAP Fault Analysis and Prevention process and fault codes.
- Make a distinction between a Fault and a Failure e.g. by calculating ratio of faults causing a real Failure in the field.

6.6.6. Proposals for customer focused measurements

Proportion of faults detected by the customer

When the software is running at the customer site, then the faults can come from different sources, either detected by the manufacturer's support organisation or by the customer. Proportion of faults detected by the customer expresses some type of disturbance or dissatisfaction reactions from the customer.

During the present study the measurement system was extended to cover both the number of faults detected by the manufacturer and the number faults found by the customer. The data about this external/internal classification has actually been available in the trouble report handling system but not used as a measure. High proportion of customer initiated fault reports is used to launch analyses concerning type of faults that the customer has categorised to be the most serious faults and most disturbing ones to act on. The measure helps to keep the customer-reported faults in a minority.

Customer's satisfaction with the development

As customer deliverables are most usually developed in projects, then an indicator based on simple ranking scores is proposed. The rating includes customer's general impression on quality of results based on a questionnaire and interview after a completed project. A compressed way of presenting for management, e.g., "Management star", which is also known as Kiviat diagram (Burr, Owen, 1996).

6.6.7. Other suggestions to improve measurements

Metrics for programs generated automatically from high-level design language are needed. During the present study the measurement system was expanded to include a separate measure type for PLEX programs generated from HLPLEX (a high-level design language). Our solution was to measure size of the program both on HLPLEX and PLEX levels. So, hand written code and generated code should not be mixed in measurement because the generated code appears to be much more "loose" and less optimised. Another example of this trend is generation source code from SDL specifications. It is the same trend in object-oriented world. Some tools are able to generate C++ code from UML models. Some code must however still be written manually and linked to the generated code. This causes problems for fault density calculations in combining the results. For this reason, in order to avoid mixing of apples and pears, the suggested solution is to measure both levels separately.

In the future, more and more new languages and tools will be used and measurement of software size on lines of code basis will face more problems. Therefore we suggest that the number of faults per design man-hours would be a proper measure to collect. This measure can express to what extent faults are generated with respect to the invested effort. It helps to compare quality of software written in different languages, methods and tools. Such measurement results can be used to choose cost effective tools.

Design metrics that can be used to identify error-prone modules during the design phase have been presented for example in (Zage, 1993). Looking into our related studies we also found very promising empirical studies toward effective fault prevention (Ohlsson, 1998). Those design metrics need, however, hard and careful collection of data about internal structure and interface of modules. In our investigations we found similar possibilities to predict faultlessness on a basis of software signals (see

Section 5.6.2). In future measurements any type of primitive data to be collected from specifications and interface would provide important information to derive regression models to fault content. It would be reasonable to augment the database by new data like module complexity, functional type of the module.

Very crucial for software quality are the human factors. Even more useful than to collect software structural issues is to consider designers or team competence per module. An attempt to link competence to modules was made already in late 80's (Hirvensalo, 1990a). Later on we suffered from many changes made in classification. Attitude towards collection was not always positive because the collection was felt to be quite sensitive. The studies made in SEI (Humphrey, 1999, 2000), however, show that human factors and teamwork have an essential impact on quality. So, the competence issues can not be neglected from future measurements.

An experienced quality manager (Elf-Mattila, 1998) has also pointed out the importance of getting some preliminary data stored into database. For example, if preliminary sizes of modules and number of faults are stored before coming of the final figures then the measurement system could better be used for prediction and project control.

In Section 5.7 we concluded that the faults inserted in newly designed parts of the module should be separated in the measurement database from faults detected in old base module. This would be more rightful to designers and helps to characterise degree of faults in the old design.

It should be "marked" in the database if a project/product has been developed by using new methods e.g. Fast-track design, Incremental design, new team work etc. Many kind of different test methods (automatic testing, statistical testing etc.) are applied in projects, so it should be marked, too. In this way it is easier to classify projects and trace process impacts to measurement results.

6.7. Conclusions considering processes and measurements

6.7.1. Conclusions on our approaches

Increasing use of goal-oriented measurements in industrial companies is a common trend. Measurement planning is thus more and more becoming an essential part of business and management processes. However, today's industrial projects are busy and do not have enough energy to run e.g. too heavy GQM programs. Instead, the projects need computer-supported possibilities to track their goals and drivers for them. We identified a problem in goal setting process on different levels of organisation. The upper level strategical goals should be broken down into lower level subgoals and project goals. The goal setting process needs more attention so that the lower level goals serve more the achievement of higher level goals. So, a kind of systematic analysis of driving attributes is often missing and therefore it was included in the Plan Measurements process.

The Plan measurements process presented before should therefore NOT be understood as a completely separate process but as a chain of activities to result in purposeful measures. Roles and responsibilities are discussed in text part of Sections 6.4.2-3 but linking to other (i.e. user) processes is analysed in Section 7.3.5. The process developed follows a goal-oriented approach and facilitates conversion to formalised measurement objectives in well-managed steps. As a final step a new Utilisation Plan (see Section 7.3) concept is included to ensure that the use of a measure is planned whenever a new measure is planned.

It was also found that the measures used are not always well defined and understood. As a result of investigation of measurement process issues we incorporated an improved *Define Measurements* process (Section 6.4.3). The process that is supported by measurement definition templates implicitly steers measurement customers to think about clearly specified measurement objects and attributes, validation of data etc. The Define measurements process also includes feedback to continuously improve measures and to validate them.

The combination of BSC and a quality model would be an excellent idea to get more attention to the customer-oriented quality factors of ISO 9126. Quality focus in current measurements has been quite narrow mainly concerning fault density. In lower levels of organisation or in certain process area (e.g.

requirements, testing) it would be feasible to visualise quality factors and their measurable driving attributes in a BSC Dashboard.

We expanded the focus to cover In-Service Performance and customer aspects. It was not easy to build a bridge from product development to the customer. As we looked into the problem we found PPD modelling to be a useful method to identify experience-based drivers. BSC is one solution to broaden the perspectives of goals and measurements and to improve management visibility into vital few indicators. However, it is important to remember that the granularity of information needed on different organisational levels is different. We can also conclude that even for a current switching system it was possible to introduce a tool that traces ISP failure events to individual software modules. Therefore, some refined measurements to consider this aspect in future measurements were proposed (Section 6.6.5).

As we strongly felt the goal-oriented approach to be useful, our approach in Chapter 6 was not to develop a complete set of new measurements. The input that drives development of new measures comes from many sources as summarised in Table 35. However some general goals in mind, we made suggestions on improvements of inspection, project, quality cost and ISP measurements which partly were also experimentally used during the present study.

We have analysed and evaluated which concrete measurements should be used in order to achieve high maturity of measurements in accordance with higher CMM levels, as for example, test coverage, efficiency of inspections and process variation. The improvements that are necessary relate to the use of control cards, quantitative goal setting and measurements towards achieving them in projects. An important CMM issue to consider is how to improve existing graphs in order to help to reduce statistical variations.

New methods in software engineering, e.g., Object-oriented design, component based development and re-use will change the nature of planning and using measurements. Measurement attributes should more and more be integrated into the used processes. Measures, like faults per man-hour tight to process activities and maintenance man-hours can be used instead of using only product-oriented measures.

6.7.2. Evaluation of our process models

The measurement processes presented in Section 6.4 were applied on R&D division level at Ericsson in Finland to derive a few Key Performance Indicators (KPI) corresponding to the performance objectives. The models proved to be suitable for use in connection with the BSC approach that we discussed in Section 6.1.5. The measurement process was thus closely connected to the management's strategic planning process where the perspectives, goals and indicators are first roughly specified by the management team. Practical examples of perspectives are Financial, Process, Customer, People and Innovation. All indicators were defined in detail using MRD and MDD (see appendices 8 and 9). The Measure design and validation activity (see Figure 6.4) was experienced to be useful. As we asked the primary measurement customer to validate the measure definition (MDD), they did it. KPI measurements were published and maintained on the Management Dashboard in Intranet and revised annually. Typical usage applications are performance assessment (goals vs. outcome), and the bonus program. For further details of KPI approach, see (Hirvensalo, 2001).

The process definitions described in Sections 6.4.2 and 6.4.3 were reviewed and evaluated by a few quality experts. Several iteration circles were performed and comments were discussed. Based on comments we made some improvements. For example, to mention a useful improvement (Hallqvist, 2001), we added cause-effect analyses to be considered as an input for the Plan measurements process. Cause-effect analyses might be an input from the strategic planning or other operational planning activities of the management. Our idea is that management teams try to find driving attributes for their goals and use e.g. TQM tools for doing cause-effect analyses. For this reason we added an arrow for cause-effect analyses as shown in Figure 6.3.

Another comment (Lahtivuori, 2000), concerned the feedback that should be handled by the Define measurements process. According to the received comment, the evaluation of an existing measure belongs to the Define measurements process if any feedback has been sent based on the usage of a certain measure. Therefore, we added a chevron for evaluation of measures as shown in Figure 6.4.

Prior to publishing, the models were approved by the quality manager and the descriptions of measurement processes were made a part of other process definitions of the R&D division (in accordance with ISO9001:2000).

The elements of the measurement framework, for instance the renewed definition templates and instructions for MDD and MRD have been reviewed and evaluated carefully by Ericsson's EPMG group composed of company's best measurement experts. MDD and MRD tools have been used in planning of KPI's in Finland as mentioned above. Ericsson has used them for example in renewing the PQT and in the Word Class Provisioning (WCP) program (see Appendix 1), to mention some examples.

We have analysed and evaluated which concrete measurements should be used in order to achieve high maturity of measurements in accordance with higher CMM levels, as for example, test coverage, efficiency of inspections and process variation. Analysis of new measurements was done as a literature review. However, we were able to evaluate a few of the measures by using them, for example, risk measures, measuring requirements and proportion of faults detected by the customer. The last mentioned measure was implemented company-wide in a new PQT tool in 1997. As risk measures and requirements measurements were included in GQM plans (Hirvensalo, 1999), their usefulness was evaluated by the project managers and quality experts. We gained also experience from measuring the inspection process. Inspection measures were validated by the ESSi inspection team and put into use in many projects. It was possible to collect a lot of data for inspection measures from real projects by using the new web-based tool (Jokikyyny, 1999). The measures as such were feasible but inspection measurements needed detailed data and it was not easy to get all designers to provide correct data.

Chapter 7.

Proposals for measurement processes and utilisation models

We presented in Chapter 2 the existing measurement process model (Figure 2.5, page 12). In this chapter we will concentrate on an improved version of this model, called *Perform Measurements*. The model provides a good practice for organising measurements in multi-project and multi-user environment on different organisational levels. The rest of Chapter 7 characterises the *utilisation* of measurements using this renewed Perform Measurements process model. The main focus in this chapter is thus on two rightmost parts of our measurement model framework (Figure 2.4, page 11). No accurate definition for measurement utilisation is found in literature. Therefore, we first define the term "*Utilisation*" more closely as follows:

All kind of analysis, use and application of measurement results to understand, evaluate, predict, control and improve software processes and products, as well as to manage software projects.

Utilisation is an issue that can be characterised by modelling. Utilisation activities are presented in Section 7.1.2 in the form of a process model. Use of measurements is discussed in Section 7.2 in a wider sense to introduce building elements of a *Utilisation model*. Before creating a model in Section 7.3 we define what we mean by the Measurement Utilisation Model. We adopt the following definition:

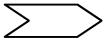
A utilisation model is a set of rules, practices and support tools for use of measurements in project management, process improvement and organisational learning.

In Sections 3.4.5 and 3.5 we presented expectations and a number of obstacles related to utilisation. A utilisation framework is analysed in Section 7.2.5 to consider the essential issues for overcoming obstacles. In order to promote a proper use of measurements it appeared to be necessary to consider differences in utilisation on different organisational levels and work phases (see Section 7.3.2).

In addition to a framework, the utilisation model presented covers a process representation for flow of analysis and information handling that aim at linking the utilisation to normal work processes. Finally, based partly on Section 6, we contribute with a few practical examples and models to different uses of measurements. The model created is in a great deal generic but we end the Chapter 7 by tailoring aspects.

7.1. A proposal for improved measurement process

The measurement process in Figure 2.5 is fully revised and renewed. The main improvements concern multi-level handling, data validation, and enhanced preparation for analysis, feedback and connections to action definition/evaluation. Immediate utilisation after performing measurements is vital, but in our pictorial presentation we divide the measurement process into two sequential

processes shown in Figures 7.1 and 7.2. In the following we make a survey on different activities (marked with chevrons ).

7.1.1. Perform measurements

This process (Figure 7.1) focuses on the actual "execution" of measurements. It resembles the existing model presented in Figure 2.5, but it is significantly expanded. Expansion is based on evaluation in Section 2.3.2, lessons learned (see Section 3.4) and on our long experience of executing measurements. The process is composed of four activities: Data collection from organisations' projects and processes, data validation and insertion, result generation including aggregation, and preparation for analysis and presentation.

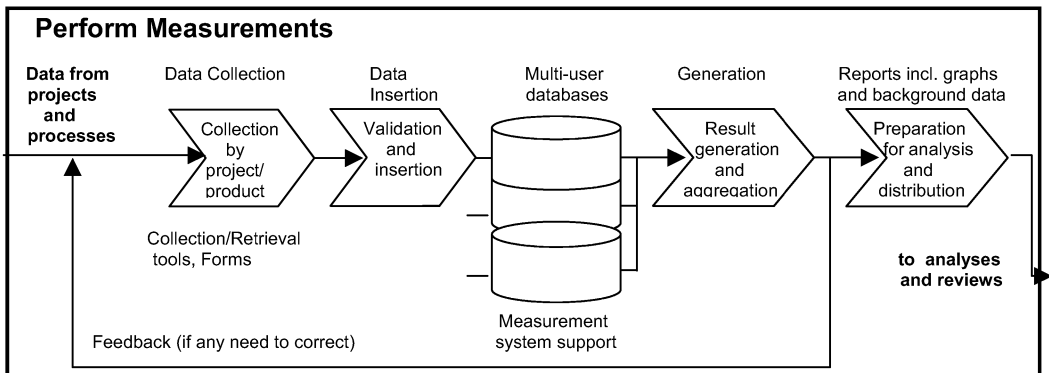


Figure 7.1 Perform Measurements process model

Data collection activities take place in parallel, concerning actual projects and products from different organisational units. Validation was included because data quality was proved to be an essential problem in our interviews (see Section 3.4.4). After the completion of generation phase we added a feedback loop that was missing from the old model. We also added a separate activity for preparation of analysis that includes the handling of background information. This was implemented because the background information helps to understand the results better e.g. reasons for possible deviations. Aggregation is illustrated showing that results from different organisational units can be merged. The Perform measurements process is composed of four activities: Data Collection, Data validation and insertion, Result generation and aggregation, and Preparation for analysis and distribution. In the following we describe the essential details of each activity.

Data collection by projects per phase

Data is gathered from organisations' projects and processes by using Collection Forms or automatic retrieval from data sources. Specific collection tools can be used (e.g. volume measurement tools). Data collection activity can be described as a separate subprocess similarly than in the existing process (see Section 2.3.1, Figure 2.6).

Data validation and insertion

Measurement data is validated to check that it is consistent and reliable as described in the MDD. Persons in role of "data administrators" at organisational units insert the data into multi-user databases. The data filled in collection forms is partly inserted into database manually or, as done in the most modern way, data is imported directly e.g. from spreadsheet files.

Result generation

Whenever enough data has been gathered (e.g. after end of a quarter, after a concluded project phase), graphs and a report based on that data are generated. Generation phase includes selection of output to be presented to measurement customers. Selected way of presentation is mainly based on

principles defined in MRD. Measurement experts, quality experts, or other proper analysts usually carry out the result generation. Depending on the level of organisation, the results from many organisations or sub-projects are aggregated on upper level. Whenever necessary, the input data might need to be further updated. Thus, feedback is sent to Data Administrators if any corrections or missing data appears.

Preparation for analysis and distribution

This activity includes preparation of material and some initial analysis of data. Presentation slides are prepared. Visualisation tools and Internet solutions are actively used. After that, the results are ready for input to the Utilise measurements process.

7.1.2. Utilise measurements

Because the utilisation was described insufficiently in the existing measurement process (Section 2.3.2) we decided to construct a more comprehensive model. Expectations from Sections 3.4.5 and 3.5 have been considered for example by enhancing pre-analyses. We have added a missing feedback loop to process owners. In Figure 7.2, a feedback link to Define Measurements process that was presented in Section 6.4.3 as a separate process is also shown (concerning evaluation of measures). Some activities are included due to requirements from higher CMM levels, for example an activity to compare with defined benchmarks. The model created also underlines the importance of Experience Base that contains derived knowledge for the future while the multi-user database in Figure 7.1 is only a storage for primitive data on low abstraction levels.

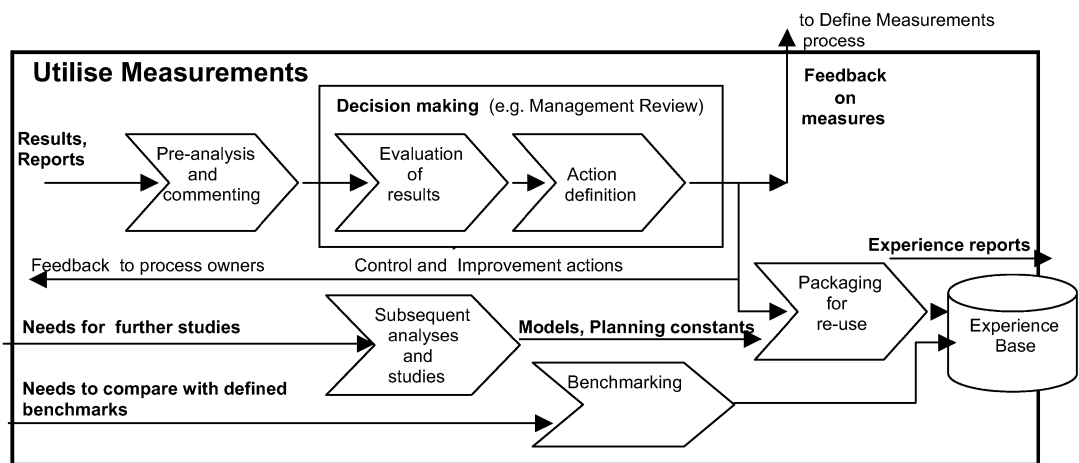


Figure 7.2 A process model for utilisation of measurement results

The usage process is composed of six activities: Pre-Analysis and commenting, Evaluation of results, Action definition, Subsequent analyses and studies, Benchmarking, and Packaging. In the following we describe the essential details of each activity.

Pre-Analysis and commenting

Relevant customers analyse and interpret the results, give their comments and findings in proper forums (e.g. in project Feedback Session). The report is updated including e.g. verbal text about root-causes. Some background information can also be added. Analysis and interpretation techniques that are suggested in the MRDs can be used.

Evaluation of results

Measurement results are evaluated (usually in Management Reviews) in order to know whether those results indicate any reason to suggest an improvement action of the processes and operations.

Evaluation may also concentrate on effects of already finished improvement actions based on measured facts. Results should always be evaluated jointly with project personnel or with persons who have provided data.

Action definition

Based on the analysis results it is decided whether further action is necessary or not. If actions should be taken, the suggested changes and modifications to process models are defined. The definition should also include who is responsible for modifications and when is each modification implemented.

Feedback on measures

Feedback is sent to *Define Measurements Process*, i.e., to design a new measure or to cancel an existing measure.

Subsequent analyses and studies

A need for "post-mortem" analyses may arise occasionally when data from many projects becomes available. Correlation and dependencies can be studied further whenever enough statistical material has been stored in databases. Outputs from this activity are, for example, *planning constants* or models derived. A simplified model of prediction the 6-month fault density based on known distribution of faults by phase presented in Section 5.6.1 is one example of such models. Subsequent studies based on collected historical data help companies to create many kinds of other "*Experience models*" (e.g. Fault slippage models, Fault detection cost models as discussed in Chapter 5). The analysis results are also used to revise *cause-effect models* for planning of future measurements (see Section 6.4.2).

Benchmarking

One essential use of measurements on higher maturity levels is benchmarking. Analyses are usually done separately by comparing company's performance with performance in other companies. Benchmarking covers much more than only comparing measured figures. It also includes causal analyses and active learning in order to become "Best in Class".

Packaging for re-use

Experience, findings, opportunities for improving and lessons learned are gathered. These are packaged for re-use, and stored into *Experience Base*. Headings for an Experience report are presented in Section 7.3.4.

7.2. Guidelines for definition of utilisation models

7.2.1. Background

Improvement of measurement utilisation means better **analyses**, **application** and **feedback**. In this context, an example from one big organisation, Goddard Flight Center, is given. In their Software Measurement Guidebook (NASA/GSCF, 1995), one can find the following golden rule:

Focus on applying results rather than collecting data.

Since 1976, the Flight center has succeeded in storing data from more than 100 projects, generating over 200 000 collection forms into their on-line database and has resulted in more than 200 reports. They have used this information to generate models and relationships that have been the basis for practices used in their software development. However, data collection has not been the dominant element, but the application of measurement results. Examples of use and application at NASA are planning and estimating based on measurements (e.g. derived models for estimating development

effort and annual maintenance cost) and measurements to increase understanding of software error characteristics.

7.2.2. Purpose

The main purpose for development of a utilisation model can be expressed as follows:

Can we learn lessons better?

This requires that measurement results **are used** and **seen to be used**. As soon as a utilisation model is available, the management's **commitment** to follow it is a key for success. Utilisation is a key element when an organisation aims to fulfil the requirements of higher CMM maturity levels as illustrated in Section 7.2.4. Because the utilisation practices have proved to be unstructured and unestablished, some kind of model is thus needed to improve utilisation.

7.2.3. Capturing requirements for utilisation

This section summarises the different sources causing requirements for improvement of utilisation. First, our main concern is to consider the requirements from CMM by gaining a deeper insight into maturity of measurement utilisation on higher levels (see Section 7.2.4). That is why the requirements arisen from ISO 9000, quality systems, Quality Awards and TQM are in a great deal similar to CMM, as we analysed in Chapter 4. Second, we aim at solving difficulties, problems, obstacles and expectations reviewed in Sections 3.4-3.7 by including those issues that are essential for utilisation (Section 7.2.5).

7.2.4. Requirements from CMM

Based on the survey in Section 4.1 and a deeper analysis of the CMM document SEI-93-TR-25 (Paulk, Weber 1993) by Key Process Area (KPA), the following utilisation requirements can be derived from CMM.

Table 36. Use of measurements at CMM levels

Level 2 Repeatable	Level 3 Defined	Level 4 Managed	Level 5 Optimising
			Defect prevention and use of feedback for Process change and improvement
		Quantitative goal setting, Defect analysis, Process analysis, Feedback for Process Control by using metrics, Statistical Process Control is applied.	
	Process tracking, Modelling Use of software process database for project planning and estimating Measurements are used to determine and predict the quality of products		
Project estimating and tracking of size, resources, effort, cost and schedule based on past experience. Estimation and tracking of critical computer resources			

Table 36 visualises how the measurement utilisation started at a lower level continues at upper levels.

Measurement utilisation required at CMM Level 2

Measurements at level 2 are to be used for project management to estimate and track (follow-up) size, effort, cost, resources, schedules and risks. Estimation and tracking should be performed according to a documented procedure. This means that estimations should be based on historical data from past projects. For example, estimation of critical computer resources from AXE software point of view can be interpreted as prediction of AXE processor's memory capacity in KPSW per each functional block²², checking of possible processor overload situations and availability of test channels on System Test Plant (STP). During and after the project the estimated issues should also be tracked by making measurements of actual values.

Measurement utilisation required at CMM Level 3

Measurement activities that help to reach CMM level 3 deal with the process and product in more detail. Emphasis of level 3 measurements shifts from projects to processes. From level 3 on, it is possible to make Process Tracking, which means documented procedures to measure, track and maintain profiles by key tasks of the software process. It is easier to utilise results because there should exist an experience database. As the processes are defined in more detail, it is possible to share experience-based metrics for estimating e.g. size, cost and schedules between projects. The KPA "Integrated Software Management" means that there is an independent group that reviews the estimating procedures and provides guidance in using historical data from the process database. Software Product Engineering involves quality measurements where the number of faults (found in inspections and tests); their types and severity are identified in products and tracked both cumulatively and per activity.

Measurement utilisation required at CMM Level 4

At level 4 it is essential to get a good touch to process factors driving towards high quality to be able to narrow the variation in the process. Therefore some predictive measures and use of measurement results for estimation are needed. Level 4 measurements typically specify that predefined analysis activities exist. Those who have been affected by the data review the results of analysis. A standard set of measurements should exist and all projects should collect this standard set of measurement data. Measurement results provide feedback for control of the process. CMM requirements on level 4 for a mature software process imply that statistical variations are to be reduced. Thus, measures that clearly show the variations in process and products and making statistical analyses happen are necessary. Therefore, Statistical Process Control (SPC)²³ is becoming increasingly important in the field of software development. This generates also the requirement to improve the ability of existing measurement methods to visualise SPC aspects better.

Measurement utilisation required at CMM Level 5

At level 5 there is a continued emphasis on process measurements and process methods for defect prevention. Results from measurements and careful process analyses are used to change the process. To consider level 5 needs in utilisation, it is essential to use measured facts in kick-off meetings of each project to prevent faults and in causal analysis meetings according to a documented procedure.

7.2.5. Essential issues to be considered

Several general key issues have an impact on improved utilisation and help us to outline the components to be included in the utilisation model. In this section we make an attempt to identify and characterise the most essential elements. Some reasons for importance of those issues are given. The issues are organised in six categories including:

- Provided measurements,

²² An activity that is in accordance to AXE software development model performed in Prestudy and System Analysis phases.

²³ SPC is a method for improving the capability and consistency of a product and a process by reducing process variability (Burr and Owen, 1996). The term "Statistical Quality Control (SQC)" is also used.

- Why are the measurements used,
- Levels and the viewpoint of using,
- When are the measurements used,
- Analysis methods, predefined actions, practices, mechanisms, and
- Meeting and reporting practices.

Provided measurements

At first, concerning either already existing or recently established measurements, it is necessary to identify *target attributes* (measured properties), *evaluation objects*, and *sources* of results available.

Target attribute expresses the property or feature (of an evaluation object) that the measurement user needs to evaluate, improve or control. A target attribute can further be composed of several sub-attributes. *Quality* is a good example of an attribute that is to be decomposed into several *sub-attributes* (e.g. fault density) in order to find the properties that are directly measurable from software. *Duration* or *Lead time* is a target attribute that is often tied to a completed project. *Effectiveness* and *Cost* are other target attributes that can be mentioned. The list of attributes is long; it is not reasonable to present any complete list here. The main point is that the measurement user can realise to which target attribute he/she is focusing with respect to goals.

Evaluation object is always important in order to know clearly what object each measurement user is actually evaluating and utilising. An object for developers might be an individual software module (either new or modified), a set of modules, a project, an organisation or a process/subprocess, while an end customer is interested in evaluating a complete system.

Source where the results are available. One difficult problem faced has been that measurement results are not always easily available. Improved availability and user-friendliness of information systems are keys for better usage. We need to define and communicate from what environment the measurements results are available, such as:

- Specific measurement systems,
- Specific retrieval and visualisation tools,
- Experience Base,
- Repositories of derived relationships,
- Storage of experience packages and measurement reports for future use, and
- Library for instructions and models for different uses of measurements.

So, the system environment providing measurement results should be well known, communicated and trained. Names of those systems, way of access (multi user/limited user group), and names of support persons should be well informed.

Why are the measurements used

This issue category includes definition of the *purpose*, intended *usage application* and *scope of use* for measurements.

Purpose is included because the lack of awareness of why different measurements are needed has been a problem for a long time. In Section 3.7 we concluded that even the purpose of PQT like measurements has not always been clear among people. In spite of the fact that each MRD contains a "Purpose" heading, the descriptions were not read or not communicated enough. In connection with utilisation model we will define clearly the purpose why the measurement results are analysed and applied. In brief the purpose can be expressed by one word, as for example: evaluate, improve, control, predict, characterise, monitor, baseline etc.

Usage application, when defined exactly in the utilisation model to address each vital measurement, helps to understand what the measurements are used for, e.g.:

- Project planning and control (Baselining, goal setting, status monitoring, pre-defined action)
- Operational planning and performance assessment (e.g. comparing organisation's goals to outcome)
- Quality assurance (e.g. inspection status, reviews held vs. planned etc.)

- Used to support process assessments (e.g. showing evidence of performed activities)
- Identification of improvement actions
- Checking the effects of improvement actions
- Use as criteria for awarding people
- Use of results and analyses in training
- Predictions and modelling (e.g. predicted fault content, effort/size models based on history)
- Internal/external Benchmarking.

The list can be continued and even more specific usage within the areas mentioned above can be defined when the utilisation model is applied. However, we conclude in this context by calling this entity "*Usage applications*".

Scope of use provides a possibility to express whether the measurement results are used internally or externally. Some results are to be reported externally for company or business unit management - which ones, should also be clearly stated. Especially, if certain measurement data is to be collected by lower level organisation, but not for their own use, i.e. for external use only, then it is extremely important to tell why that data is needed and how it is used. However, some results are suitable both for internal and for external use.

Levels and viewpoint of using - the measurement customer

In use of measurement results we distinguish between different *levels* of work. The "*measurement customer*" concept is introduced in connection with the utilisation model.

Levels, like organisational levels, should also be defined in the utilisation model because, for example, measurement utilisation on individual level totally differs from usage on management level. In our model we consider utilisation on many other levels, like on team, project or process level, on divisional level and on strategical levels. The level has an impact on content and scope of information what the measurement customer needs.

Measurement customer is a person to whom the results are handed over and who really is interested in evaluating the results. Viewpoint is thus important because there are different users (e.g. external customers, line managers, quality managers, process owners etc.) closely related to levels.

When are the measurements used

Phase of utilisation is needed because the use and application of measurements is also much dependent on work phase. Whatever work we aim to manage (getting completed), we can easily identify three usual phases: Planning, Execution and Control, and Conclusion (e.g. outcome perception after project termination). Measurements have different purposes in these different work phases. For example, in Conclusion phase we mainly need measurements assessing final results (also called "assessment type" measurements) while a "prediction type" measurement is performed to get a measure for software reliability.

Period of validity expresses the intended "life-cycle" of a certain measurement. Some measurements might only be valid within a project or during an improvement program while other measurements are valid continuously.

Frequency of analysis addresses the period telling how often the results are evaluated. Review of a measurement result may for example take place either at regular intervals (monthly, bi-monthly, quarterly, or yearly) or at certain milestones.

Analysis methods, predefined actions, practices, mechanisms

Analysis method includes definition of *statistical* methods and tools to support analyses and interpretation of results. Effective use of measurement results needs also established methods on how to analyse. Many useful methods are available, like trend analysis, root-cause analysis, regression analysis and 7QC tools mentioned in Section 4.4.

Predefined actions come into play for example on CMM level 4 as we have identified in Section 7.2.4. For this reason the model includes a possibility to specify certain predefined remedial actions or

further analysis activities to be taken if the measurement shows a critical situation. An example for a predefined action is an extra desk check that is triggered by a stated alarm limit.

Support and feedback mechanisms are necessary to consider in the utilisation model. Effective utilisation demands good support tools and techniques. Measurement material should be prepared and clearly presented in order to improve the usefulness of the material (e.g. careful visualisation of result data to make it understandable, readable and analysable). Tools/systems that allow on-line access to results are valuable (e.g. an easy on-line Internet access to measurement results). Utilisation of results can/should be divided into two methods depending on the type of feedback from measurements:

Usage application, when defined exactly in the utilisation model to address each vital measurement, helps to understand what the measurements are used for, e.g.:

- “On-line” method where the results are available immediately when measured, i.e. the results and graphs are available during the project to take immediate remedial actions.
- “Off-line” method where the results are available after end of the project and feedback is mainly used to suggest improvement areas or predictions for future projects.

Especially in the first method, it is possible to provide feedback to project personnel when feasible and analyse the result data timely in an ongoing project together with the project members. Thus, it is easier to identify root-causes when those are fresh in mind. After all, it is important to give feedback to individual developers.

Use cases²⁴ and **practices** as part of our utilisation model, means concrete examples, practical instructions and rules to be published in an organisation to help to use the measurements more effectively. The lack of instructions and rules was already identified as one obstacle for utilisation (see Section 3.4.5). For this reason we pay attention on “*Use cases and practices*” in our utilisation model. An example might include a prediction model for fault content to be used within an organisation. Any type of *Good Practices* in using measurements belongs to this category. Furthermore, in the model, the approach of using measurement results must be clearly defined in a form of instructions and rules:

- How much time should be reserved as a rule for analysis of results.
- What statistical analyses and SPC techniques are useful.
- How are the experience and the derived relationships stored for future use and made available (Experience packaging, Experience Base).

In spite of the fact that Use cases and practices are organisation specific we further treat them in Section 7.3.4 to give some generally applicable ideas.

Meeting and reporting practices

The measurement customer can either evaluate the results alone or several customers may analyse and take actions in meetings. The model guidelines the analysis methods and the meeting forums where the results are to be handled. Possible meeting forums are **Pre-analysis forums** and **Management's decision-making forums**.

Pre-analysis forums, like quality team meetings, expert meetings or project feedback sessions should be used actively for preliminary analyses. One good practice might be integration of feedback session to project progress-meeting routines. The intention of pre-analysis is preparation and commenting of the material before it is handed over to Management's decision-making forums, for example, Management Reviews, Technical councils, Project Tollgate and Conclusion Meetings etc.

What measurement results are attached to different reports is also a matter of clarification. Measurements should be integrated with normal reporting routines. **Reporting** routines include, for example Operational performance, Project progress and Final reporting. This is also applicable to “paperless” reporting on World Wide Web pages. Measurement / report relation to the most important selected measurements within an organisation should be “marked” to make utilisation more regular.

²⁴ Use case here has nothing to do with Jacobsons use case that is an essential concept in Object-oriented design.

7.3. Creating a generic utilisation model

In this section we will present a generic utilisation model of the measurements. The model gives rules, practices, tools and examples how software development activities can typically be supported by measurements. The model created includes six parts (Figure 7.3).

Marketing and Motivating	Conceptual framework
	A model example for the usage applications by phase and organisation level
	A template for mapping provided measurements to their intended use
	A model to link utilisation to work processes
	Use cases and practices including examples, instructions and rules

Figure 7.3 Structure of the utilisation model

The conceptual framework provides a summary of essential elements discussed in Section 7.2.5. The model considers how utilisation of measurements can support in different phases of actual work and on different organisational levels (like division, department, and individual). A template for a Utilisation Plan is created to ease the mapping of measurements to their intended use. A model to link utilisation to work processes guides how utilisation is integrated as a part of actual work processes like project process, software development process and management processes. Use cases and practices are presented as a separate part of the model. Lack of motivation might be in many cases a reason for unsuccessful use of measurements. Therefore, our model includes a few guidelines for marketing and motivating.

7.3.1. Conceptual Framework for the Utilisation Model

Based on building elements identified in the previous section, the model covers the elements shown in Figure 7.4. Each oval represents an issue category that is to be well organised in order to utilise measurements successfully. Bulleted lists inside each of the twelve ovals illustrate the meaning with two typical examples.

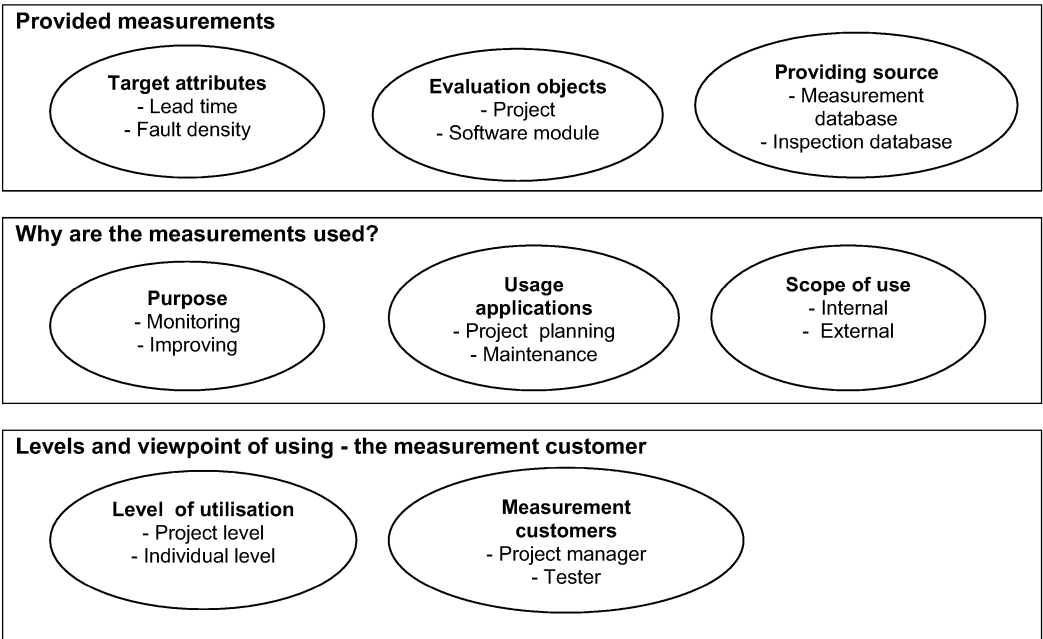


Figure 7.4 Conceptual framework for utilisation (each oval includes two typical examples)

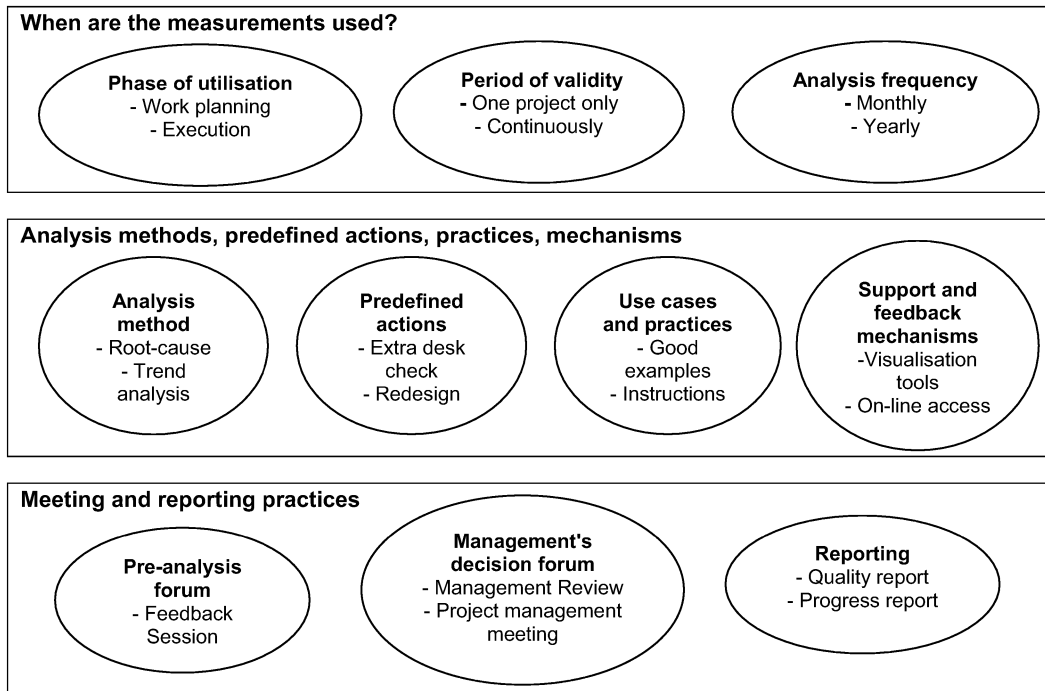


Figure 7.4 (cont.) Conceptual framework for utilisation (each oval includes two typical examples)

7.3.2. Measurement utilisation by phase and organisational level

Whatever task an organisation is performing the most common work phases are Planning prior to the work, Execution and Control during the work, and Conclusion (and onwards) to create outcome perception after completion of the work. Because different kinds of usage for measurements can be obtained on different levels we introduce a general model for recognising practical measurement applications. A software-producing unit can apply a template shown in Table 37 to its own environment to support and provide guidelines for the better use of measurements. On each level one should first think about the most important activities on that level, for example, a project usually runs its goal setting activity in planning phase. As a next step one can realise vital measurement usage applications related to this activity. All activities in the three phases are thus mapped to measurements by asking: "How can we use measurement results to support this particular activity?"

Table 37. A template for assigning utilisation topics by level and phase

Phase	Planning	Execution and Control	Conclusion and onwards
Level			
Business/Product Unit			
Company level			
Divisional level			
Department			
Process			
Project			
Team			
Individual			

The template in Table 37 provides a simplified method to encourage people to apply measurement results. It is recommended to use this template in workshops where everyone is involved. In this context we go through a few levels in more detail in the light of an example (Tables 38 and 39) where typical activities and usage applications on three levels are filled in.

Table 38. Usage of measurements on divisional level

Phase Level	Planning	Execution and Control	Conclusion and onwards
	Yearly operational goal setting <ul style="list-style-type: none"> - Results from previous years - Baseline/current level Strategic planning and long-term goal setting <ul style="list-style-type: none"> - Weightings based on market studies Education planning <ul style="list-style-type: none"> - Current competence levels - Quality statistics from past courses Establishment of awarding criteria <ul style="list-style-type: none"> - Validated results from earlier metrics Financial planning and forecasting <ul style="list-style-type: none"> - Traditional financial metrics used for forecasts Planning key indicators for improvement programs (e.g. contributing company-wide action programs) <ul style="list-style-type: none"> - Results from previous years - Baseline/current level for target 	Goal assessments <ul style="list-style-type: none"> - Gap vs. goal - Rolling averages vs. goal - Quarterly results Steering the implementation of strategies <ul style="list-style-type: none"> - Strategies acted upon vs. planned Execution of education programs <ul style="list-style-type: none"> - Courses to be held vs. planned - Course attendees vs. target - Measurements based on feedback from participants Follow-up of awarding <ul style="list-style-type: none"> - Checking if the criteria are being met Financial situation <ul style="list-style-type: none"> - Control of incomes/ expenses per period Use of prioritised indicators <ul style="list-style-type: none"> - Quarterly results - Gap vs. target 	Goal fulfilment <ul style="list-style-type: none"> - Outcome vs. goals Strategies <ul style="list-style-type: none"> - Strategies planned/come true Evaluation of education programs <ul style="list-style-type: none"> - Courses held vs. planned - Achieved competence growth Awarding people <ul style="list-style-type: none"> - Outcome vs. criteria Business results <ul style="list-style-type: none"> - Outcome vs. plan - Bookkeeping results Target assessment <ul style="list-style-type: none"> - Outcome vs. target

Table 39. Usage of measurements on project and individual levels

Phase Level	Planning	Execution and Control	Conclusion and onwards
Project level	Baseline determination <ul style="list-style-type: none"> - Current quality (similar projects) - Quality of base products - Current effectiveness Goal setting <ul style="list-style-type: none"> - Quality, time, precision, cost etc. Prediction and estimation <ul style="list-style-type: none"> - Size, effort, lead time, faults, risks Planning <ul style="list-style-type: none"> - Measurement-based planning constants (design, testing) - Quality planning (e.g. inspections) Project kick-off/training <ul style="list-style-type: none"> - Fault profiles 	Forecasting outcome <ul style="list-style-type: none"> - Identification of fault-prone modules Deviation handling <ul style="list-style-type: none"> - Measured/forecasted delays - Deviations from plans and - Deviations from entry/exit criteria Use for inspection status monitoring <ul style="list-style-type: none"> - Inspections held/planned - Inspections slipped from planned week Use in project tracking <ul style="list-style-type: none"> - Expected outcome vs. project goals Project progress meetings or Feedback Sessions <ul style="list-style-type: none"> - Use of accumulated measurement data for decisions on control actions 	Evaluation of results after delivery to the customer <ul style="list-style-type: none"> - Customer satisfaction - Product quality until customer's acceptance - Delivery precision - Time-to-Market Project conclusion <ul style="list-style-type: none"> - Measurement results generated for Final-, Test- and Quality reports - Goal vs. final outcome discussions Identification of Improv. opportunities <ul style="list-style-type: none"> - Measured facts used to support proposals for improvement Identification of Best Practices <ul style="list-style-type: none"> - Measured evidence for quality of a tool or method
Individual level	Designers personal estimates <ul style="list-style-type: none"> - Measured effort, schedule data Individual goal setting <ul style="list-style-type: none"> - Study of earlier references Quality of input documents and base products <ul style="list-style-type: none"> - Inspection/fault statistics - Complexity figures 	Monitoring the task <ul style="list-style-type: none"> - Inspection results - Detected faults in tests - Effort, deadline 	Evaluation of own job results <ul style="list-style-type: none"> - Goals vs. outcome - Accuracy of estimations Analysis of and learning from faults <ul style="list-style-type: none"> - Fault profiles - Occurrence of pitfalls

Table 39 shows that even an individual designer and tester can utilise measurements, if the results are properly available, e.g., to make his/her personal estimates. This helps management on project level to get as accurate estimates as possible. On project level we identify a number of measurement applications tied to different project activities in all those three phases we have modelled. For example, historical measurement data is very useful in project goal setting by providing baseline knowledge in order to set realistic goals.

7.3.3. Mapping provided measurements to their intended use

The basic tool provided for mapping is a Microsoft Word spreadsheet (Appendix 12) that is used to list the most important measurements and their utilisation aspects. The spreadsheet provides a template for a Utilisation Plan that can be used for summarising the essential issues discussed in Section 7.2.5 on a measurement basis. The structuring of issues follows the framework specified in Section 7.3.1. Here, the essential issues are "chained" by using understandable explanations (*in italics*) followed by an example (in square brackets):

Provided measurement results

Referring to the goal [goal number]
focusing on the [target attribute area]
with respect to [sub-attribute]
related to [other measure or driving attribute]
of/in the [evaluation object]
generated from [system]

Why are the measurements used?

for the purpose of [purpose]
for [usage application]
and scope [scope]
from the viewpoint of the [measurement customer]

When are the measurements used?

using in the [phase of utilisation]
valid during [period of validity]
with analysis frequency [period]

Meeting and reporting practices

by applying the method [analysis method]
Pre-analysed in/by the [meeting/expert]
to make decisions in the [management meeting]
attached to the [report]

In addition to the template in Appendix 12 we need also an example tied to practical life. In Appendix 13 we present a filled-in template including a few typical project measurements.

7.3.4. Use cases and practices

As one motivation of the present study was the expectation to increase the use of measurement results in management and project planning, performance assessments and improvement work, this section provides a few use case examples on possible instructions, models, practices, and hints for this purpose.

Using models and instructions for project planning

Measured knowledge provides opportunities to create planning constants, models for prediction and estimation. From Section 5.4-5.7 we can summarise some input regarding models for project planning. A simplified method for prediction of the 6-month fault density based on known distribution of faults (see Section 5.6.1) can be made available in a form of an instruction. A method can be a Reliability Growth Model supported with a particular tool.

Effort distribution models help to allocate project man-hours for different activities. A model can state for example that 5-7% of project man-hours should be planned for inspections (see Section 5.5.2). Using advanced inspection measurements helps to establish planning constants for preparation hours needed to inspect different types of documents, preparation factor (see Section 5.5.3) and create norms for defect detection/removal rates. An organisation can also make decision on effort estimation method referring to Function Point measurements.

An instruction could state that analysis of project Final reports, Test reports and Quality reports or any type of Experience reports is a mandatory practice in planning phase of a certain new project. In project planning and goal setting phase it is also important to analyse effect-relations between project/product goals and driving attributes. Possible methods are GAM, GQM and PPD modelling presented in previous sections.

Other instructions can relate to baseling, assuring quality of old base modules, use of alarm limits (e.g. when an extra desk check is to be performed), release criteria or to a method for identification of error-prone modules (see Section 5.6.2). For further ideas of use cases we refer to tables 38 and 39.

Allocating time for use of measurements

Our question "How much time should be reserved as a rule for analysis and use of results?" is relevant also elsewhere. Only a few large organisations have paid attention to this problem. NASA/GSFC recommends that at least three times as much effort on analyses as on data collection should be allocated (NASA/CGFC, 1995). It is necessary to start reporting the amount of time (in minutes, hours) used for analyses, feedback session meetings, Management Reviews, team meetings etc. forums where measurement results are used. We thus need a framework or *time reporting model* to classify different essential measurement activities. The activities are suggested to be classified as follows:

- Data collection and insertion
- Result generation, analysis, interpretation and evaluation
- Feedback meetings, management meetings
- Experience packaging
- Subsequent analyses and studies to derive relationships, instructions and models.

It is extremely difficult to give any general "rule of thumb" how the time should be divided between different activities, because the time spent for measurements and analyses might be hard to separate from the time used for other actual project activities. However, an organisation can start for example by making inquiries about person's time usage profile. This as such can help the organisation to pay more attention to utilisation and to change attitudes. As the next step, an organisation can start regular measurements and goal setting for the analysis effort by recording time reserved and spent for utilisation.

Measurement of utilisation

Because the measurements play a vital role as a part of management process it is also reasonable to measure use of results. Measurement of utilisation is a measurement, too. New measures that become into question can be based on time allocation measurements mentioned above or be simple counting of occurrences, such as:

- Percentage of time spent for utilisation in comparison with total development effort
- Number of managers using the measurement system in decision making / total number of managers
- Number of observable actions, number of inputs for preventive work, number of instructions and models generated or derived from measurement results
- Number of pre-analysis meetings and Feedback Sessions held
- Number of proposals (based on measured facts) for changing process and project models.

All the issues mentioned in the list are surely not easily and clearly observable but by trying to start measurement, one can see whether real things are happening or not.

Instructions on statistical methods and tools

An organisation should support the use of statistical methods and tools by instructions, software installations and references. Suggestions for useful statistical analyses/SPC techniques are given in references (Florac, 1999; Burr and Owen, 1996) considering the increasing use of SPC within software development. Some improved graphs are needed due to higher CMM level requirements. Showing a mean curve is not always enough but the graphs that are able to show reduction of statistical variations. An interesting method to apply when the variations in data are quite high, is the CUSUM (Cumulative sum charts) method (Oakland, 1990). The graphs showing only averages, maximum and minimum values should be changed to visualise individual observations and control limits to get graphs to show much more. Trend diagrams that are able to indicate (10 %, 90%, mean) interval of variations are also useful, for example, to analyse lead times and delivery delays.

As the variables to be studied (e.g. correlation coefficients, mean values etc.) are random variables, the “confidence interval” concept should be applied. For example, 95% interval means the range of values, which contain the true value of the estimated variable at the probability of 95%. Upper bound and lower bound for a confidence interval are calculated based on the number of data items and so called Fisher Transform (Milton, 1995). Bounds for 95% confidence limits can be presented in a graph meaning in practice, e.g., that 95 out of 100 projects are expected to fall within those bounds. Instructions concerning the way of presentation and the statistical techniques, that can be applied, are included in MRD (see Section 6.4.4). An example of organisation's statistical tool convention might be use of MINITAB²⁵.

Experience packaging aspects

What and how should be packaged can be based a real Experience report (Hirvensalo, 1999) prepared during the present study in connection with applying the PROFES method (see Section 5.1.5). A proposal for proper headings in an Experience report is as follows:

- Context and background description,
- Measurement programme results,
- Lessons learned,
- Re-use situations, and
- Recommendations for future projects.

Context and background information is intended to explain the reasons why the measurement programme was initiated and to introduce the project and products that were the measurement objects. The heading **Measurement programme results** is the most essential part of the experience package report. It is divided into six subheadings that are Goals and attainment, Process improvement, Validation of hypotheses, Answers to questions (in case of GQM), Data collection and storing and Feedback sessions. *Goals and attainment* section discusses measurement goals and explains their attainment or non-attainment. *Process improvement* section explains the changes taken in the process. *Validation of hypotheses* explains whether the hypotheses have been confirmed or not. Hypotheses may relate to a certain GQM question or a previously known experience model, for example we stated that inspection preparation factor (Preparation effort / total inspection effort) equals to 0,52. Validation based on measurements in a project showed that inspection preparation factor increased to 0,72. *Answers to questions* of the GQM plan for a project are discussed here. An example question, taken from our Experience report is as follows: “How well Function Test has been done”. The answer to this question that was formulated during the feedback session was based on the measured fault density in Function Test. Actual fault density (0,36 Faults/KNCSS) was significantly lower than in the previous project (0,62) and also significantly reduced in comparison with baseline and goal (1,0). *Data collection and storing* section is intended for explaining how the data collection was organised. The last section provides information on *Feedback sessions* held (incl. material discussed) and may refer to separate Feedback session reports.

Lessons learned section packages the experience that answer to questions “What went well” (e.g. most measures focused on right things in the project), or “what were the issues during the project” (e.g. it was not possible to collect all data as planned). Furthermore, all technical and managerial

²⁵ MINITAB that is available for Windows has a complete set of Statistical Process Control functions (e.g. Pareto analysis, many types of charts).

lessons learned are discussed. **Re-use situations** suggests opportunities for re-use of material, for instance can the measurement plan be re-used as base for future projects. **Recommendations for future projects** that come forward from the measurement programme might concern timing issues between the measurement program and a project, man-hour reporting profile, time allocation improvements for analyses etc.

7.3.5. Linking utilisation to work processes

As we have identified a number of issues belonging to the utilisation framework, we should also describe how to integrate them as a part of actual work processes like project process, software development process and management processes. It is not only important to realise the input from other processes to measurement processes but also to clarify how the other processes should use the output from measurement process (i.e. utilise measurement results). To do that, we list some typical processes and make suggestions on activities and roles concerning utilisation.

- Management process,
- Project management process,
- Software process management,
- Improvement work, and
- Quality management.

The **Management process** should consider measurements especially in strategic planning, in decision making, in resource handling and training. Measurement planning relates to *strategy management* where the success factors, priorities and goals have an essential impact on the selection of key performance indicators. Use of measurement results is clearly connected to key *management's decision-making* forums. Analysis and evaluation of results and action definition based on measured facts is done as part of the management process. Management uses performance measurement results as base for *bonus program*. Management evaluates the usefulness of measures and, if necessary, gives *feedback* on changing measures. Management shall ensure that Quality Coordinator's *job description* includes measurement responsibilities (pre-analysis etc.), and that other expert's time allocation for analyses is satisfactory. Measurement training is to be linked to actual *training programs* of the line organisation.

The **Project management process** (i.e. projects) needs measurement results in baselining, goal setting and forecasting, and tracking against stated goals. Project management utilises constants and packaged experience data for project planning. Often, in order to avoid mistakes, new projects hold their kick-off meeting at establishment phase and discuss the measurement results from previous projects. Feedback sessions are to be held during execution and at conclusion of projects. Project manager uses measurement results to make action decisions at checkpoints. Projects also provide new measurement-based experience data for future use.

Software process management should regularly use results from process-oriented measurements (see Section 6.6), e.g.; adherence to established process models is followed up by using measurements. Process owners and process model developers react to feedback from Utilise Measurements process. Changes of Software Engineering process models etc., are implemented.

Improvement work should utilise measurements both as input for new improvement programs and in evaluating results of the program. Analysed measurement results provide input for improvement work (e.g. for preventive work) showing where the improvement actions should primarily be directed. Improvement actions should be tied to measurable indicators, so that progress can be measured as well. In order to evaluate output, the measurements provide material for analysing effects of improvements.

Quality management plays a central role in development of new measurements and measurement plans. They also take care of maintaining the measurement system and updating the measure definitions. It is the responsibility of quality function to run improvement and measurement programs, and to communicate the results with software engineers. Besides this, the quality function uses measurement results in quality assurance (reviews, inspections, audits, and assessments). The Quality experts should supervise pre-analyses and subsequent analyses, and participate in the evaluation and updating of Cause-Effect models. Finally, we should not forget quality management's role in training. Measurement results, packaged experience, hypotheses, derived models provide

material for quality courses, seminars and other training events. This material should be utilised regularly rather than occasionally.

Examples of other processes, not discussed above, are **Customer support and maintenance** and **Product Management**, which have their links to utilisation of measurements. Process map is company specific. We suggest that the discussion conducted above is performed in more detail to link company's actual work processes to utilisation by using computerised process mapping aids.

7.3.6. Marketing measurement utilisation opportunities

In order to get utilisation accepted among people it is necessary to consider a way of selling useful measures. This means motivation by arguing measure usefulness. Especially, marketing means promoting good measurement practices, showing good examples and opportunities. Opportunities should be demonstrated in positive spirit to get people to use measurements as their own instrument. The Internet can be used to effective communication of the measurement results. One good way to disseminate results is also the use of regular wall posters in conference rooms. Other marketing situations are training events, project workshops, management meetings, quality seminars and road shows. Utilisation plan that covers all essential in-house measures in brief serves also as new marketing material. This material is to be used in different situations to increase people's awareness about measurements.

7.3.7. Tailoring the Utilisation Model

Organisational units within a company might be larger or smaller in size and operate in different business areas. Therefore it is reasonable to create a tailored model based on a generic model which the organisation units can adapt to their specific needs. In this section we discuss the most essential tailoring aspects. It is evident that different ways of using measurement results should be applied to different types of operations. So, the measurement areas corresponding to the organisation's operations should be classified, as for example:

- Traditional switching system software design
- Object-oriented design for Internet software applications
- Operations and projects on specific Business or Product Area (GSM, UMTS).

Some columns in the template created in Section 7.3.3 are optional, and in this way, the model is tailorable. The size of the company may influence on the amount of utilisation levels. Examples in lower part of the template (Appendix 12) might also need adaptation to organisation's processes and practices. Usage applications and examples are also influenced by the CMM maturity situation of the organisation. Thus a need of updating the template may arise when the maturity level grows. The model can also be tailored for a restricted area, like utilisation of inspection measurements.

7.4. A Measurement Utilisation Capability Model (MUCM)

7.4.1. Background

In a recently published book (Dumke, 1999, p. 109), Niessink and Vliet discuss measurement maturity and mention that well defined measurement guidelines and programs do not yet guarantee successful *usage* of measurement programs. Further discussion about maturity of usage is useful because existing attempts to define measurement maturity do not cover usage (utilisation) aspects to a full extent. In practical implementation of a measurement program the usage of measurement results is often the weakest link. As we have modelled measurement utilisation, we can present the issues also in the form of a utilisation capability model. In previous sections we have identified a number of success factors which can be used to characterise a mature utilisation of measurements. We use them to create a *Measurement Utilisation Capability Model* (MUCM). Measurement utilisation capability is defined as:

The extent to which an organisation is able to use and apply measurement results in evaluation, control and improvement of its products, processes, projects and resources in order to achieve business goals.

The MUCM model has been created by using the analogy method. With respect to utilisation, we made an attempt to find similarities to other models appearing in the software community. For example, in the literature one can find a maturity model for assessing inspections (Grady, 1997). The first version of the MUCM was based on ordinal scales similar to CMM. As CMMI was published during the present research we adapted our model to CMMI levels. Our discussions on measurement utilisation required on different CMM levels in Sections 6.3 and 7.2.4 have also provided input for the MUCM model. Furthermore, we have adopted issues from the Utilise measurement process (Section 7.1.2) and from the Utilisation model, trying to map the issues to proper capability levels. Finally, we identified a number of essential aspects that differ by capability level, e.g., primary measurement objects and the length of the feedback loop (See the more detailed list in the beginning of Section 7.4.3).

7.4.2. The Purpose of the MUCM model

Especially, the MUCM model aims at characterising measurement utilisation issues, not characterising organisation's software process maturity/capability aspects that are already represented by models like CMM, CMMI (CMMI, 2000) and SPICE (ISO/IEC15504, 1998). To some extent the model however translates CMM(I) concepts to utilisation process. The model focuses on all life-cycle phases of organisation's software product development process discussing measurement utilisation aspects by capability level (on ordinal scale). No new process areas are adopted. As an added value, the MUCM model helps to interpret what every capability level means for utilisation of measurements.

The MUCM model can serve as a basis for assessments concerning capability maturity in use of measurement results within an organisation. The model provides a supplement for maturity models like CMM. This supplement can be used as a guideline in assessment situations and as a means for organisations that aim at achieving a higher maturity of their measurements and operations.

7.4.3. Capability levels

The following essential aspects have been considered in characterising capability levels in Table 40:

- Primary measurement objects by capability level,
- Management and involvement by capability level,
- Extent to which the analyses of measurement data result in action items,
- Length (duration) of feedback loop by level,
- Awareness about how the measurement results are used as input for other processes,
- Growth of understanding about relationships between driving and target attributes,
- Goodness of analysis/visualisation/publication tools, and
- Examples of typical utilisation characteristics and usage applications by level.

7.4.4. How the MUCM compares to CMMI

The conceptual structure is similar to CMMI (the latest integrated version of CMM): Six levels for organisation's measurement utilisation capability are defined by identifying characteristics that provide an enhancement in the capability to perform utilisation practices (see Table 40).

A separate Process Area (PA) "Measurements and Analysis", that provides a basic support for all other Process Areas in measurement issues, has been adopted in CMMI. The presented MUCM model quite closely relates to Specific Practices of the CMMI, in the PA Measurements and Analysis, e.g., Analyse Measurement Data and Communicate Results.

Table 40. Measurement Utilisation Capability levels and their characteristics

Level	Characteristics
0. Incomplete	<p>Ad hoc, no defined utilisation of measurement results or only partially defined and performed practices exist. No goals for measurements. Only a few or no measurement results are communicated and analysed.</p> <p>Example characteristics: Occasional usage</p> <p>Examples of usage applications: -</p>
1. Performed	<p>Utilisation practices are largely being identified and performed but are not well planned and controlled. Utilisation dependent on individuals, only a few experienced persons use results. Utilisation is informal and does not follow documented procedures. Measurement goals are not formally defined.</p> <p>Example characteristics: Users of measurements unknown. "Measuring for measuring's sake" may destroy utilisation</p> <p>Examples of usage applications: Usage to increase common understanding only</p>
2. Managed	<p>Basic utilisation practices exist. Project and line Management Reviews measurement results and reacts to problems identified. Long feedback loop. Only a few final performance based measures used. Ambition level of utilisation dependent on projects, team or software producing unit (SPU). Use of measurements included in project/quality plans. Measurements loosely connected to track business and project goals.</p> <p>Example characteristics: Post-mortem evaluation type usage, rear mirror view, "off-line" feedback</p> <p>Examples of usage applications: Project planning and control (Baselining, goal setting, status monitoring)</p>
3. Defined	<p>Utilisation process defined and integrated to management and standard software engineering processes. All projects systematically use measurement results in accordance with documented utilisation procedures. Utilisation tied to goal setting, cause-effect modelling and action-impact measurement definition. The purpose, usage applications, measurement customers (stakeholders) and way of analyses defined for all measures. Responsibilities for usage defined on organisational level. A totality of well-defined meaningful metrics exists. Measurement data stored in software process database utilised for project planning and estimation. An easy access to measurement results by those who have provided with data.</p> <p>Example characteristics: GQM, Balanced Score Cards, Management Dashboards, Multi-user databases, "on-line" feedback. Systematic and regular usage both on organisation and project levels</p> <p>Examples of usage applications: Organisation's operational planning and follow-up, Quality assurance, Quality cost monitoring</p>
4. Quantitatively Managed	<p>Measures managed and updated consistently with respect to organisation's visions, strategies, business goals, quantitative product quality goals, measurement goals and driving attributes. Increasing use of early in-process, measures, i.e., measurements are used across the whole life-cycle. Measurement results are distributed and visualised on-line. Feedback sessions held regularly with short feedback loop. Enough time is allocated for analyses. Corrective actions to be taken based on analyses are defined. Analysed measurement results used to derive relationships between driving and target attributes and to introduce instructions and models. Experience and learning packaged for future use. Effectiveness of the utilisation process is measured. Usage of measurements is quantitatively controlled.</p> <p>Example characteristics: GAM method, Quantitative management towards achieving product quality goals, Control charts on inspections, Measurement data analysed both on organisation, project and process levels</p> <p>Examples of usage applications: Predictions and modelling (e.g. Product-Process Dependency Models), (Sub)process performance monitoring, Use of analysed results in quality training and organisational learning</p>
5. Optimising	<p>Improvement based on use of objective measures fed back to the measurement process. Measures evaluated and changed where necessary in accordance with established change management routines. Effectiveness of changes in utilisation process is validated. Predictive measurement results continuously compared to outcomes in order to improve models and estimation accuracy. Relationships between driving factors and final performance indicators are known. Measurements used in a proactive way to achieve desired technical and business results. Measurement results used as input for preventive actions concerning improvement of software processes. Measurement results actively compared to customer satisfaction and to searched benchmarks.</p> <p>Example characteristics: Build-in "improvement machine", Experience Base</p> <p>Examples of usage applications: Internal/external Benchmarking, Validation of process improvements</p>

The MUCM model elaborates the institutionalisation of measurement usage to cover aspects on higher process capability levels. Achieving goals for higher capability levels requires a many-sided use of measurements. Use of measurements relates to several other CMMI Process Areas than Measurement and Analysis. In this sense, the MUCM provides a cross-area view for utilisation of measurements by giving examples of good practices.

7.5. Concluding and evaluating utilisation models

We have created the Utilisation Model to correspond to needs, desires and requirements from different sources presented in Chapters 3-6. As one of the most important aspects the model facilitates clarifying the purpose and use of each measurement. Many usage applications and use case examples are given. The utilisation model includes a quite comprehensive framework clarifying both mandatory and optional building elements. To overcome obstacles discussed in Section 3.4.5 we have paid attention on the way of doing analyses like responsibilities for pre-analyses, frequency, analysis forum as well as allocating time for them. The model enables proper feedback giving and guidelines for taking actions. In the model we also discuss different levels of using because measurements are felt more useful if right measurement results are handed over to right customers. Capturing of CMM issues (in Section 4 and 7.2.4) has in a great deal provided elements for application and use case examples. A template (Appendix 12) was designed to help to map provided measurements to their intended use. Finally, we contributed with a Measurement Utilisation Capability Model (MUCM) based on CMMI levels. This model helps to characterise the “degree of capability” in using measurements. The MUCM in Table 40 is useful e.g. in (self-)assessing and improving utilisation of measurements in an organisation.

7.5.1. Evaluation of Perform and Utilise measurements processes

The Perform and Utilise measurements processes were evaluated by internal experts in the quality network of our case company, but also by an external expert. The Perform measurement process presented in Section 7.1.1 was also applied in our case company to measurements of Key Performance Indicators (KPI) relating to follow-up of the performance objectives. Correspondingly, the Utilise measurements process presented in Section 7.1.2 was applied to management's decision making on the basis of quarterly KPI results (Hirvensalo, 2001). A web-based solution made the access and result presentation much easier. In this web-implementation it was also easy to link to the process descriptions that specified the responsible person for each activity. However, during the present thesis it was not possible to use and evaluate our processes in full scope because the KPI process did not involve all activities (e.g. packaging for re-use). Anyway, we implemented the following changes based on experts' comments.

One of the most important comments (Lahtivuori, 2000) related to “Evaluation of measures” activity that belonged to the first version of our Utilise measurements process. In order to avoid redundancy, we moved the activity “Evaluation of Measures” to the Define measurements process. However, the feedback on measures still remains in the Utilisation process. Based on feedback, the measurement experts who are responsible for defining the measurements can evaluate the measure in further detail and, whenever necessary, change it.

As an external expert (Nevalainen, 2001) evaluated our process we received a suggestion to add Benchmarking. Because comparison with defined benchmarks is one of the most essential activities to achieve CMM level 5, we added a new chevron (see Figure 7.2). External expert also commented that the responsible person for each activity is missing. As we tailored the web-application for KPI measurements therefore we completed each subprocess description with particular company specific responsible persons. Furthermore, the graphical way of presentation used in the measurement process does not illustrate the interfaces to user processes. This aspect is discussed in Section 7.3.5. However, we think this is something that should be considered in possible further studies in more detail.

7.5.2. Evaluation of the Utilisation plan

The experts evaluated the Utilisation model and it was tentatively tested in a project. The model was presented for evaluation to the Ericsson's EPMG group. It was discussed in a quality seminar and evaluated by quality managers. EPMG group reviewed the early version of the model and proved that its purpose chapter (see Section 7.2.2) is valuable and agreed that it is important to present issues on 1-2 A4 pages as we do by using templates. Furthermore, one of the reviewers would like to see more concrete connection with addressing needs/problem areas. For this reason, we added a possibility to refer to certain goal and target attribute in the template (See Appendix 13). From quality managers we adopted several improvements e.g. concerning usage applications, measurement customers, and feedback mechanisms.

We considered comments from a project, for example by adding a new usage aspect regarding analysis frequency. Another very positive suggestion by a project manager (Meinander, 2000) was to expand the model with predefined actions i.e. to allow a possibility to specify certain predefined further analysis actions to be taken if the measurement shows a critical situation. Due to this suggestion we made improvements on the conceptual framework of the model (see Section 7.2.5 and Figure 7.4). According to one of the comments from quality experts the building elements of the model could be divided into mandatory and optional ones. This division is implemented in templates of the model by shading the optional columns (See appendix 12 and 13). Appendix 13 provides an example of how the model fits to real measurements used in a project.

7.5.3. Evaluation of the MUCM model

The MUCM model was evaluated by using expert reviews. Four experts reviewed the first versions of the MUCM. All reviewers were authorised assessors (e.g. CMM). The first comment (Nevalainen, 2001) raised regarding capability levels was that the sentence "Measurement objectives are not formally defined" was moved from level 0 to level 1. The sentence on level 0 was replaced by "No goals for measurements". Ambition level of utilisation is not only dependent on projects, but also on team and SPU unit. In accordance with comments, level 3 was expanded to include the existence of well-defined metrics. Furthermore, comparison to benchmarks and validation of effectiveness of changes in utilisation process on level 5 was added.

The second reviewer (Reiman, 2000) thought that the sentence "Corrective actions to be taken based on analyses are defined" is a typical issue on level 4, so we agreed. His evaluation also caused an improvement on level 5 stating that measurements are used in a proactive way. The third reviewer (Nilsson-Hallqvist, 2001) commented that the level characteristics are mixtures of abilities and practices. This is true, but we regarded this comment as an idea for further development of the model. According to another comment the connection between goal setting, action definition and action impact must be brought earlier in the model. To consider this comment we added the sentence "Utilisation tied to goal setting, cause-effect modelling and action-impact measurement definition". Finally, the fourth reviewer (Elf-Mattila, 2000) noted that some level sentences were in fact usage examples. For this reason, we made a distinction between example characteristics and examples of usage applications. The reviewer also considered that it was too hard to achieve level 3 if all projects should use measurements results to determine and predict quality. We agreed and therefore this requirement does not appear in the final version of the MUCM.

The MUCM model has been tentatively verified as the experts reviewed it. A contribution was also given in Fesma 2001 conference (Hirvensalo, 2001). Because the MUCM was developed as a final step in the present thesis it was not possible to test it e.g. in CMM assessment situations. Thus, the evaluation done in this thesis is incomplete, and the MUCM requires some further research and validation.

Chapter 8.

Final conclusions

8.1. Concluding our study

We fully achieved or even exceeded the objective to develop utilisation models but we only partly achieved the objective for correlation studies. We put more effort on modelling measurement processes including careful planning and definition of new measures. The modern approach for measurement planning should include analysis of driving factors together with target attributes to be evaluated before establishing measures. In this way it is easier to arrange data collection for a certain purpose. Trying to touch driving factors afterwards faced to data collection problems. So, in spite of many interesting findings in Chapter 5, our possibilities to make correlation studies were limited. We did not find as much correlation as expected.

It is hard to measure quality within software-oriented development. Technologies and strategies are changing faster and faster. Improving quality is a goal-directed activity. Based on this thesis we can conclude that well planned goal-oriented measurements are best purposeful in the future. We paid attention on methods how to develop new measures but did not aim at defining a complete set of new measures. However, some improved new measures proposed in connection of this research are already in use at Ericsson to transfer measurement point towards earlier phases in software development (e.g. inspection) and customer satisfaction measurements. Computer support was significantly improved as web-based tools were made available.

8.2. Other concluding remarks regarding the “measurement journey”

During the study it was possible to look into evolution of measurements in the long-term and influence on several eras of evolution in our case company. The evolution described in detail in Appendix 1 is highlighted in Figure 8.1.

In the first PQT era measurements were event-based and collection-oriented. This means that collection process was well defined and everybody could see that data was collected. Measurement results were used but everybody did not see how they were used. There were problems in the huge implementation of the system, fetching of data and generation of all graphics. The present measurements provided mainly a “rear-mirror” view for management. The focus concerning fault density and LOC-like measurements was quite narrow. The process was very simplified covering a centralised way of collecting, database handling and basic output. In spite of well-documented metrics we observed some data quality problems in the database. Mainly the department managers participated in analyses. Feedback loops to designers were not effective and regular. Analyses from lower organisational level prior to handing results over to upper level management were missing. In the beginning of the present research (in 1996) we started a new way of reporting so that lower level analyses get ready first. The measurement system was changed to include both multi-user input and multi-user output. This appeared however to be a quite long learning and deployment process.

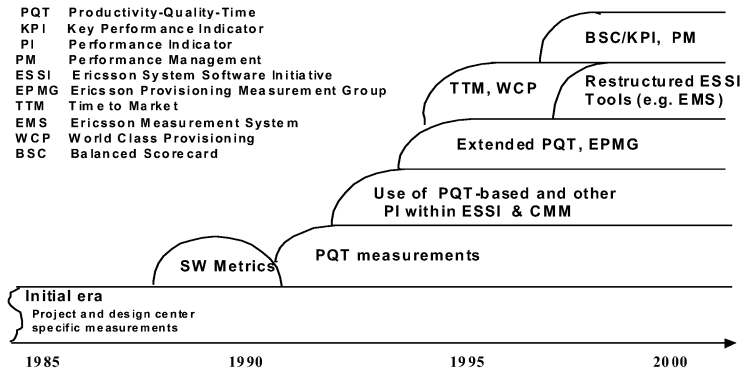


Figure 8.1 Evolution of measurements

What was good in past measurements – What experience can be transferred? We have made a briefing in Section 6.5 regarding our case company environment. We can summarise that use of measurement results in Management Review, a procedure that was started in early 90's, is worth continuing. The way of defining measures and their use in improved form has during the present study achieved the increasing acceptance in several measurement programs and organisation units. Later, the coming ISO/IEC 15939 standard that is currently going to be finished introduced similar measure definitions. Another good question, when adopting new systems and software technologies, is to think what can not be transferred from the past measurements, and if not, why.

Management Process should be updated to cover a clearly visible activity for analysis and other activities for utilisation of measurements. Furthermore, increased involvement of "measurement customer" thinking is recommended. Finally, the measurement customer should also take a more active role in planning, approving and using the measures. Handing over the right measurement results to the right customer may also lessen the confusion and misunderstandings among measurements.

8.3. Contributions of the thesis

The **main contributions** of this thesis include the following.

First, the results in Chapter 3 contribute to knowledge enhancement regarding implementation, problems, obstacles as well as lessons learnt on utilisation of measurements in a multi-national industrial environment. Both a few managers and individual designers were interviewed in the beginning of the study.

Second, a number of case studies using the data collected from real industrial projects were formulated in a form of eleven generally stated hypotheses. We revealed a few facts either supporting findings in earlier studies or providing new contribution to knowledge. As a new result we observed that 70-80 % of individual modules which have been faultless in Function Test are also faultless in System Test and during their first 6 months after delivery. In typical projects the fault slip-through to first 6 month is low, approximately 10%. However, faults causing a fatal failure at the customer represented less than 10% of faults in a sample of 2000 trouble reports detected during first 6 months after external delivery.

Third, we have renewed the measurement process to cover all essential phases and feedback. The result that consists of four main processes is summarised in brief in Figure 8.2. We provide proposals on well-structured processes for planning, defining, performing and utilising software measurements.

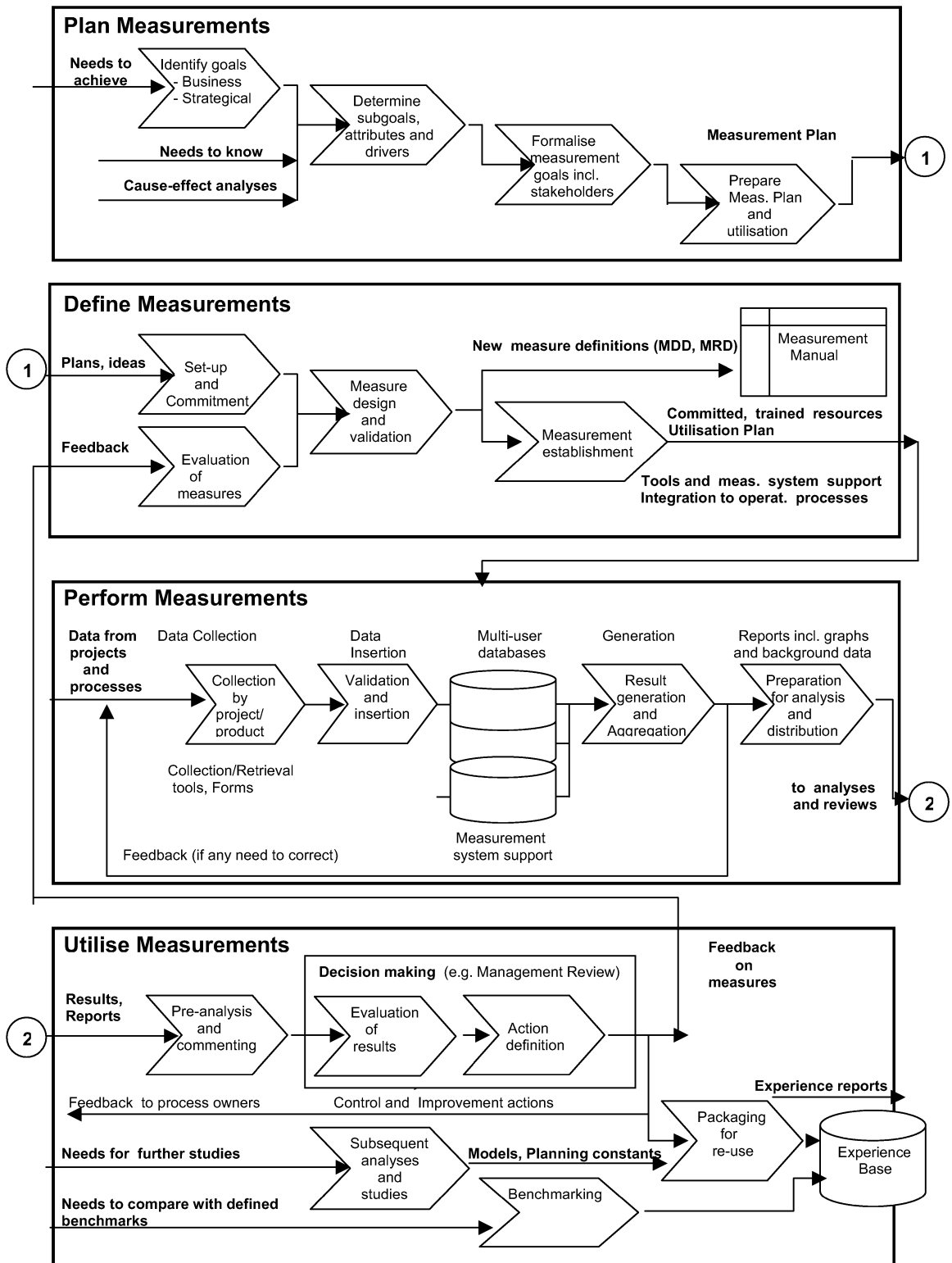


Figure 8.2 Measurement process overview

Here everything is presented on one page showing feedback loops between different subprocesses. The process overview created helps to put modern measurement theories and methods discussed in this study into practice. The process model involves a rigorous and well-defined approach that was one objective of this study.

Fourth, we allocated a significant effort to develop utilisation model that was a vital objective of the study. A novel *Utilisation model* is provided that can be used to improve the use of measurement results in a company where software plays a central role. As the utilisation until now has been a less successful issue in the world of software measurement we come in with an improved framework. The framework takes into account users, phases, levels, practical usage applications and linking to other processes and systematises the procedure that traditionally has been quite incoherent.

Fifth, we contribute with a six-level *Measurement Utilisation Capability Model* (MUCM). The MUCM provides guidelines for organisations that aim at achieving a higher maturity of their measurements and operations.

8.4. Proposals for improving utilisation

Based on the research and lessons learnt we make an attempt to list a few generally useful ideas to improve utilisation of measurements:

- The integration of data collection and reporting should be an integral part of project or software design processes. This is caused by including measurements into the development processes and by automating collection and presentation as much as possible.
- When new design methods and processes are created, corresponding measurements must be defined (not afterwards but as part of the method development).
- For large organisations establish a cross-organisational *measurement team*.
- Plan and define your measurements clearly by applying modern measurement concepts and the process presented, and by using use templates for measure definitions (e.g. MDD, MRD). Plan and do also utilisation of measurements.
- Management commitment, support and continuous interest are important for successful measurement activities. Do not present too many metrics on higher organisation levels; use Balanced Scorecard and Dashboards.
- Present well visualised measurements during project (it is easier for designers to feel them as their “own instrument”). E.g. improve retrieval of results and visualisation aids by using a web-based multi-user system.
- Put more focus on applying measurement results and feedback rather than collecting data. This is caused e.g. by allocating enough time for analyses of measurement results.
- Apply continuous measurement training, information distribution and showing results (including motivating).
- Take care of data quality: Ensure that collected data suits to its purpose, is stored in time, is consistent, complete and correct. As people have better confidence in reliability of data they are also more willing to use it.

8.5. Proposals for further research

Further studies of issues influencing on ISP (In-Service Performance)

Deeper analyses of design faults that have caused disturbances in the field are needed. In addition to root-cause analyses, a proactive work for *design to ISP* approach is a necessity. Studies of preventive measures having an impact on ISP should be continued by using Effect-Relationship analyses and Product-Process Dependency Modelling presented in Section 6.6.5. Also the methods presented for

planning and defining measures provide opportunities to pay more attention on driving attributes of ISP. It is possible to achieve better ISP figures by identifying and improving the right driving attributes and measuring them during design.

Cost of detecting a fault after termination of design project

In the present study (see Section 5.5.3) we succeeded to quantify fault detection cost for inspection, unit test and Function Test phases, but for the complete cost profile, only rough and old data based on general beliefs is available. More man-hour data is needed to determine costs of detecting a fault in System Test and operational phases. Based on current data it is difficult to trace fault cost to projects because maintenance man-hours are not collected per fault or per project after termination of design projects. Quantitative knowledge from latest cost profiles for all phases could be used in motivating people to reduce fault costs.

Piloting COSMIC Full Function Points

The further study should cover application and counting of COSMIC FFP e.g. with assistance of the Finnish Software Measurement Association (FiSMA) that is supporting the method. Tool support is one of major issues to be clarified. To do that, measurement analysts need some training in order to familiarise with COSMIC-FFP rules and conventions.

Proposals for hypotheses to be studied further

A majority of faults in code (module) should be detected in inspection, i.e.; more faults should be caught in inspection than in testing. A module, in which this is not true, should be considered as a potential problem module in tests. To prove hypothesis needs more statistical material to be collected from inspected modules and their quality in tests. Not enough statistical material was available for the present study because systematic storing of this data was started in the later phase of the study. The result can be used to create criteria for identifying and preventing potential low quality modules.

Further subsequent studies would relate to correlation between inspection attributes and post-release fault content. Application of fault prediction and fault history models is also worth of further studying.

References

Abdel-Hamid, T., Madnick, S., Software Project Dynamics, An Integrated Approach, Prentice Hall, Inc., ISBN 0-13-822040-9, 1991, 264 p.

Agresti, W., Software Metrics and Design Quality, NFK Seminar, Oslo, 1995-10-23.

Ahola, M., Selvitys ohjelmistotuotteiden 7-12 ensimmäisen käyttökuukauden laadusta, Helsinki University of Technology, Examination report, 1996, 25 p. (in Finnish only)

Antman, L., Mätning av utvecklingsarbete, Specialseminarium, chefer, Programvaruprocessen, Sveriges Verkstads Industrier, Stockholm, 1993-03-25, 19 p.

Armstrong, J. et al., Concurrent Programming in Erlang, Prentice Hall, ISBN-0-13-508301-X, 1996, 351 p.

Arthur, L., J., Improving Software Quality, An Insider's Guide to TQM, John Wiley & Sons, Inc. ISBN 0-471-57804-5, 1993, 287 p.

Bache, R., Neil M., Fenton, N., et.al., Introducing metrics into industry: A perspective on GQM (in Software Quality assurance and Metrics, A worldwide perspective), International Thomson Publishing Company, ISBN 1-85032-174-4, 1995, 320 p.

Barnard, J., Managing Code Inspection Information, IEEE Software, Vol. 11, 3(1994), pp. 59-69.

Basili, V., Caldiera H., Rombach H., Goal Question Metric Paradigm, Encyclopedia of Software Engineering, John Wiley & Sons, 1994, pp. 528-532.

Basili, V., Weiss, D., A methodology for collecting valid software engineering data, IEEE Transactions on Software Engineering, 6 (1984), Vol. SE-10, pp. 728-738.

Baumert, J.H., McWhinney, M.S., Software Measures and the Capability Maturity Model, CMU/SEI-92-TR-25, Technical Report, 1992, 304 p.

Birk, A., Derks, P., Hamann, D., Hirvensalo, J., Oivo, M., et al., Applications of measurement in Product-Focused Process Improvement: A comparative Study, Proc. 5th International Symposium on Software Metrics, 1998-11- 19...21, Maryland, USA, 12 p.

Boehm, B. W., Brown, J., et al., Characteristics of Software Quality, TRW Series of Software Technology, North Holland Publishing Company, ISBN 0-444-85105-4, 1978.

Boehm, B. W., Software Engineering Economics, Prentice Hall, ISBN 0-13-822122-7, 1981, 767 p.

Boehm, B. W., Software Defect Reduction Top 10 List, IEEE Computer, Vol. 34, 1(2001), pp. 135-137.

Buglione, L., Quintano N., Reo D., Balanced IT Scorecard and EFQM: A balanced approach to performance measurement for software intensive organisations, Proc. 2nd European Software Measurement Conference, FESMA'99, 1999-09-21...22.

Burr, A., Owen M., Statistical Methods for Software Quality, Using Metrics for Process Improvement, Intern. Thomson Publ. Company, ISBN 1-85032-171-X, 1996, 453 p.

CMMI Product Development Team, CMMI SM for Systems Engineering/Software Engineering, Version 1.02, Continuous, Representation, CMU/SEI-2000-TR-019. 2000, 606 p.

Conradi, R., Marjara, A., Skåtevik, B., Empirical Studies of Inspection and Test Data, Proc. International Conference on Product Focused Software Process Improvement, Oulu, 1999-06-22...24, pp. 263-284.

Conte, S.D., Dunsmore, H.E., Shen V.Y., Software Engineering Metrics and Models, Benjamin/Cummings Publ. Company, ISBN 0-8053-2162-4, 1986, 396 p.

Daskalantonakis, M., A Practical View of Software Measurement and Implementation Experiences within Motorola, IEEE Transactions on Software Engineering, Vol. 18, 11(1992) pp. 998-1010.

Dumke, R., Abran, A., (Eds.), Software Measurement, Current trends in Research and Practice, Deutscher Universitäts Verlag, ISBN 3-8244-6876-X, 1999, 269 p.

Elf-Mattila, M., Interviews, October, 1998 and December, 2000.

Elf-Mattila, M., Hirvensalo, J., Application of Product Focused SPI in Industry, Software Quality Management Congress, April 28-30, 1999, Cologne, 20 p.

Eickelmann, N., Software Measurement Frameworks to Assess the Value of Process Improvement, Proc. EuroSPI' 99 Conference, 1999-10-25...27, pp. 150-164.

Ericsson, The Ericsson Quality Manual, EN/LZT 101 839 R3, 1995, 58 p.

Ericsson, EQOM: Ericsson Operational Quality Manual, LZT 108 5252 R1, 2001, 25 p., (available on intranet only).

Ericsson, PQT Manual (Internal Binder), EN/LZB 101 2785 R2, 1995 (R1, 1992).

Ericsson in Finland, Telecom R&D, Annual Report, Booklet, 1995.

Ericsson Telecom AB, SW Metrics News 1(1991), (Internal Publication).

Ericsson Telecom AB, PROPS Project Manual, EN/LZT 101 1782, 1994, 88 p.

Ericsson Telecom AB, AXE Development Model, EN/LZT 101 1224, 1p.

Ericsson Telecom AB, AXE 10 Reliability-Maintainability-Quality of Service, EN/LZG 205 579/10 R1, Booklet, 1992, 58 p.

Eskelinen, J., Quality Machine & Tools within ESSI, Presented in ESSI seminar at Ericsson in Finland, 1995, 9 p.

ESPITI Seminar binder, European Software Process Improvement Training Initiative, Roma, 1995-10-23...26.

Fagan, M., E., Advances in Software Inspections, IEEE Transactions on Software Engineering, Vol. SE-12, 7(1986), pp. 744-751.

Fenton, N., Software Measurement Course, Linköping University, Sweden, 1996-10-16...18, 1996, 159 p.

Fenton, N., Pfleeger, S., Software Metrics - A Rigorous & Practical Approach, Intern. Thomson Publ. Company ISBN 1-85032-275-9, 1996, 638 p.

Fenton, N., Ohlsson, N., Quantitative Analysis of faults and Failures in a Complex Software System, IEEE Transactions on Software Engineering, Vol. 26, 8(2000), pp. 797-814.

Fenton, N., Neil, M., Software Metrics: Successes, Failures and New Directions, Journal Systems Software, Vol. 47, 2-3(1999), pp. 149-157.

Finkelstein, L., "What is not measurable, make measurable", *Measurement and Control*, Vol. 15, 1(1982), pp. 25-32.

Finnish Standards Association (SFS), SFS-EN ISO 9001, *Quality Systems. Model for Quality Assurance in Design, Development, Production, Installation and Servicing*, Helsinki, Finland, 1994, 31 p.

Finnish Standards Association, (SFS), SFS-EN ISO 9004-1. *Quality Management and Quality System Elements. Part 1. Guidelines*, Helsinki, Finland, 1994, 61 p.

Florac, W., Carleton, A., *Measuring the Software Process*, Addison-Wesley, ISBN 0-201-60444-2, 1999, 250 p.

Gilb, T., *Principles of Software Engineering Management*, Addison-Wesley, ISBN 0-201-19246-2, 1988, 442 p.

Gilb, T., Graham D., *Software Inspection*, Addison-Wesley, ISBN 0-201-63181-4, 1994, 471 p.

Grady, R., *Measuring and Managing Software Maintenance*, *IEEE Software*, Vol. 4, 9(1987), pp. 35-45.

Grady, R., Caswell, D., *Software Metrics: Establishing a Company-wide Program*, Prentice Hall Inc., 1987, 288 p.

Grady, R., *Practical Software Metrics for project management and process improvement*, HP professional Books, Prentice Hall Inc., 1992, 270 p.

Grady, R., *Practical results from measuring software quality*, *Communications of the ACM*, Vol. 36, 11(1993), pp. 62-68.

Grady, R., van Slack, *Key lessons in achieving widespread inspection use*, *IEEE Software*, Vol. 11, 7(1994), pp. 46-57.

Grady, R., *Software Failure Analysis for High Return Process Improvement*, *Hewlett-Packard Journal*, Vol. 47, 8(1996), 12 p.

Grady, R., *Successful Software Process Improvement*, HP professional Books, Prentice Hall Inc., 1997, 314 p.

Haley, T., Ireland B. et al., *Raytheon Electronic Systems Experience in Software Process Improvement*, CMU/SEI-95-TR-017, Technical Report, 1995, 70 p.

Haley, T., *Software Process Improvement at Raytheon*, *IEEE Software*, Vol. 13, 11(1996), pp. 33-41.

Hartkopf, S., Ruhe, G., Leippert, F., *How to make sense of empirical software engineering data – An integrated approach*, IESE Report no. 013.98/E, 1998, 33 p.

Henderson-Sellers, B., *Object-oriented Metrics: Measures of Complexity*, Prentice Hall PTR, ISBN 0-1323-9872-9, 1996, 234 p.

Hirvensalo, J., Sepponen, R., *Quality control and verification of software*, *Proc. EUROCON 83*, North Holland Publishing Company, 1983.

Hirvensalo, J., *A method for analysis of Preconditions and Quality in Telecommunication Switching System software projects*, *Proc. International Switching Symposium (ISS90)*, Stockholm, 4(1990a), pp. 105-108.

Hirvensalo, J., *An analysis of faults found during Function Test*, Internal Ericsson document, LMF/TK-90:005, 1990b, 11 p.

Hirvensalo, J., *Productivity and Quality measurements*, Ericsson Telecom AB, SW Metrics seminar, Stockholm, Nacka 1991-05-28...30, 1991, 6 p.

Hirvensalo, J., Experience of Quality Measurement, Temadag: Praktiska erfarenheter av kvalitetssatsningar inom programvaruindustrin, Svenska Förbundet för Kvalitet (SFK- DATA), Nov. 25, 1993, 20 p.

Hirvensalo, J., Management Review of LMF/T unit – 1994/Q1, Minutes of Meeting, 1994-05-11, 4 p.

Hirvensalo, J., Ohjelmistojen tilastollinen laadunhallinta, INSKO-AEL, Ohjelmistojen kehitys- ja toimitusprosessit, 1996a, 10 p.

Hirvensalo, J., Account of Jorma Hirvensalo's journey to Ericsson in Holland, LMF/T/R-96:075, (Internal document), October 1996b, 6 p.

Hirvensalo, J., Nygren, V., Needs for further development of Measurements - Results of Inquiries, (Internal document), 1996-09-14, 2+3 p.

Hirvensalo, J., GQM Plan and Experience Package (Ericsson), PROFES internal documents, 1999, 20+16 p.

Hirvensalo, J., Improving the Utilisation of measurements in a software development process, Proc. The 4th European Conference on Software Measurement and ICT Control, FESMA-DASMA 2001, Heidelberg, 2001-05-09...11.

Humphrey, W., Introduction to the Team Software Process, Addison Wesley Longman, Inc., ISBN 020147719X, 1999, 463 p.

Humphrey, W., What is your life dependent on Software?, Keynote in EuroSPI' 2000 Conference, 2000-11-07...09, Copenhagen, 38 p.

Inglis, J., Standard software quality metrics, AT&T Technical Journal, Vol. 65, (2) 1985, pp. 113-11.

ISO, Statistical Methods for Quality Control, Vol 1&2, ISO Standards Handbook, ISBN 92-67-10211-7, 1995, 492+384 p.

ISO/IEC 9126, Information Technology - Software product evaluation - Quality characteristics and guidelines for their use, ISO/IEC JTC 1, ISO/IEC 9126:1991 (E), Geneve, Switzerland, First Edition, 1991, 13 p.

ISO/IEC TR 15504, Information Technology – Software Process Assessment – Parts 1-9, Technical report, International Organisation for Standardisation, Geneve, Switzerland, 1998.

Jauhiainen, M., Interview, June, 1996.

Jokikyyny, T., Computer supported software inspection process, Graduate Thesis, University of Helsinki, Department of Computer Science, 1998, 96 p.

Jokikyyny, T., Lassenius, C., Using Internet to Communicate Software Metrics in a Large Organisation, Proc. of the International Globecom Conference, Rio de Janeiro, Brazil, December, 1999. 7p.

Johansson, Ö., Software Reliability, Royal Institute of Technology, Report no. TRI-TA/MAT-1993, 56 p.

Johansson, Ö., Nord, C., Using predictions to improve reliability, Ericsson Review, Vol. 72, 1(1995), pp. 30-35.

Jones, C., Assessment and Control of Software Risks, Prentice Hall Inc., ISBN 0-13-741406-4, 1994, 619 p.

Jones, C., Our Worst Current Development Practices, IEEE Software, Vol. 13, 3(1996), pp. 102-104.

Jones, C. (Cheryl) et al., Practical Software Measurement (PSM), A Guidebook developed by a Foundation for Objective Project Management, version 3.1a, 1998.

- Juran, J.M., Gryna, F.M., Quality planning and analysis, McGraw-Hill, 1980, (appr. 500 p.).
- Kan, S. H., Dull, S.,D., et al., AS/400 software quality management, IBM Systems Journal, Vol. 33, 1(1994), pp. 62-87.
- Kan, S. H., Metrics and Models in Software Quality Engineering, Addison Wesley, ISBN 0-201-63339-6, 1995, 344 p.
- Kaplan, R., Norton, D., The Balanced Scorecard, Harvard Business School Press, ISBN 0-87584-651-3, 1996, 322 p.
- Khoshgoftaar, M., et al., Early Quality Prediction: A case Study in Telecommunications, IEEE Software, Vol. 13, 1(1996), pp. 65-71.
- Kitchenham, B., Pfleeger, S., Software Quality: The Elusive Target, IEEE Software, Vol. 13, 1(1996), pp. 12-21.
- Känsälä, K., SW process improvement experiences within Nokia, SW Process Improvement Seminar, 1996-12-2...6, Brighton, UK, 1996, 6 p.
- Lahtivuori, J., Fault Analysis and Prevention method, Internal Ericsson Document, LMF/T/M-94:091, 1994, 24 p.
- Lahtivuori, J., Interview, February, 2000.
- Lahtivuori, J., Karvonen, S., Knuters, J., Quality reports for Switching System Software projects, 1994-1997, (Internal document).
- Lappalainen, P., Interview, April, 1996.
- Lee, E., SW Process Improvements and Maturity Growth at Loral Space Information Systems, Seminar arranged by Ericsson Quality Institute, May 1994, 135 p.
- Lennselius, B., Estimations of Software Fault Content for Telecommunication Systems, Report no. TR-104, Lund Institute of Technology, 1990, 20 p.
- Levitin, A., How to measure software size, and how not to, Proc. IEEE COMPSAC, 1986, pp. 314-318.
- Lindberg, M, Sundström, J., Study of project lead-times, A report submitted to Performance Measurement seminar, Tik 86.160, Helsinki University of Technology, Laboratory of Information Processing Science, 1996, 26 p.
- Lowe, D., A., Rapid design and Deployment Method for CMM Level 2, Proc. SEPG'96 Conference, Atlantic City, 1996a, 5 p.
- Lowe, D. E, Cox, G., The Fast Track to Implementing the Software Engineering Institute's Capability Maturity Model Level 2, Hewlett-Packard Journal, Vol. 47, 8(1996b), pp. 1-14.
- McCall, J., A., Richards, P., K., Walters, G., F., Factors in Software Quality, Vol. 1-3, Rome Air Development Center Reports, NTIS AD/A-049- 014, 015, 055, 1977.
- McGarry, F., Page, G., Basili, V. et al., Software Process Improvement in the NASA Software Engineering Laboratory, CMU/SEI-94-TR-22, Technical Report, 1992, 65 p.
- Meinander, M., Interview, January, 2000.
- Mellberg, R., Project Final report (Ericsson Internal Information), LMF/T-96:0333, 1996, 19 p.
- Mills, H.D., Cleanroom SW Engineering, IEEE, Vol. 4, 9(1987), pp. 19-24.
- Milton, J., Arnold, Jesse, C., Introduction to probability and Statistics, Mc Graw-Hill, 3rd edition, 1995, 811 p.
- Miranda, E., The use of reliability growth models in project management, Proc. IEEE Conference on

software reliability, Padderborn, German, 1998, 8 p.

Mobrin, J., Wåsterlid, A., The improvement engine of the Ericsson System Software Initiative, Proc. The European Software Engineering Process Group Conference, Amsterdam, 1997-06-16...19, C403, 15 p.

Montgomery, D., Introduction to Statistical Quality Control, John Wiley & Sons 2nd edition, ISBN 0-471-51988-X, 1991, 674 p.

Myers, G., The Art of Software Testing, John Wiley & Sons, ISBN 0 471 04328-1, 1979, 177 p.

NASA/GSFC, Software Measurement Guidebook, NASA-GB-001-94, 1995, 133 p.

Nevalainen, R., Interview, May, 2001.

Nilsson-Hallqvist, A., Interview, February, 2001.

Nilsson, A. T. Anselmsson, M., Johansson, E., Ohlsson, K., Impact of Measurements on an SPI-Program, Proc. The European conference on software process improvement, 1999-11-30...12-03, Barcelona, Spain, 1999, pp. 1-12.

Nilsson, A. T., Rise, J. L., Performance Measurements, Procedure to design measures – Goal Attribute measure (GAM), Ericsson Quality Institute, LME/Q-93:332, Rev. E, (internal publication), 1996, 27 p.

Nusenoff, R., Bunde, D., A Guidebook and Spreadsheet Tool for a Corporate Metrics Program, Journal Systems Software, Vol. 23, 3(1993), pp. 245-255.

Oakland, J., Followell, R., Statistical Process Control, A Practical Guide, John Wiley & Sons Inc., 2nd edition, 1990, 431 p.

Ohlsson, N., Predicting error prone Software modules in Telephone Switches, Industrierisiken, 1994, 87 p.

Ohlsson, N., Software Quality Engineering by Early Identification of Fault-Prone Modules, Dep. of Computer and Information Science, Linköping University, ISBN 91-7871-832-5, 1996, 151 p.

Ohlsson, N., Towards Effective Fault Prevention, An empirical study, Doctoral dissertation no. 522, Dep. of Computer and Information Science, Linköping University, ISBN 91-7219-176-7, 1998, 185 p.

Ollila, A., Quality Improvements through ISO9000 Standards, Doctoral dissertation, Helsinki University of Technology, ISBN 952-90-7057-8, 1995, 145 p.

Panula, J., Interview, May, 1996.

Park, R., Goethert, W., Florac, A. Goal-Driven Software measurement – A Guidebook, CMU/SEI-96-HB-002, 1996, 170 p.

Paulk, M., Curtis B., Capability Maturity Model, V 1.1, IEEE Software, Vol. 10, 7(1993), pp. 18-27.

Paulk, M., Weber, C., et al., Key Practices of the Capability Maturity Model for Software, V 1.1, CMU/SEI-93-TR-25, Technical Report, 1993, 444 p.

Paulk, M., Toward Quantitative Process Management With Exploratory Data Analysis, Proceedings of the Ninth International Conference on Software Quality, Cambridge, MA, 4-6 October 1999, pp. 35-42.

Pfleeger, S. L., Lessons learned in building a corporate metrics program, IEEE Software, Vol. 10, 5(1993), pp. 68-74.

PROFES Consortium, The PROFES User Manual, Fraunhofer IRB Verlag, ISBN 3-8167-5535-6, 1999, 400 p., (available on <http://www.ele.vtt.fi/profes/>).

Pulford, K. et al., A quantitative Approach to Software, Management - The AMI Handbook, Addison Wesley, ISBN 0-201-87746-5, 1995, 179 p.

Putnam, L. H., Myers, W., Measures for Excellence: Reliable Software on Time, Within Budget, Yourdon Press, Englewood Cliffs, NJ, ISBN 0-13-567 694-0, 1992. 378 p.

Putnam, L. H., Myers, W., Industrial Strength Software - Effective Management Using Measurements, IEEE Computer Society, ISBN 0-8186-7532-2, 1997, 309 p.

Rautakorpi, M., The prediction of the parameters of a SW product, Oy LM Ericsson AB (internal document), 1993, 41 p.

Reiman, H., Interview, October, 2000.

Royce, W.W., Managing the development of large software systems, Proceedings of IEEE WESCON, August 1970, pp. 1-9.

Rubey R.J., Hartwick, R.D., Quantitative measurement of program quality, Proc. of the ACM National Conference, 1968, pp. 671-677.

Rydström, L., Viktorsson, O., Software Reliability Prediction for Large and Complex Telecommunication Systems, Proc. of the 22th Annual Hawaii Int. Conference on System Science, January 3-6, 1989, pp. 312-319.

Saارينen, J., Interview, 1996.

Schultz, H.P., Software Management Metrics, MITRE Technical Report, ESD-TR-88-001, 1988, 52 p.

Shepperd M., Foundations on Software Measurement, Prentice Hall, ISBN 0-13-336199-3, 1995, 234 p.

SLY, Suomen laatupalkinnon arviointikriteerit, Booklet (in Finnish only), 1996, 39p.

SPC (Software Productivity Consortium), Software Measurement Guidebook, SPC-91060, 1994, 207 p.

Sullivan, L., Quality Function Deployment, Quality Progress, Vol. 19, 6(1986), pp. 39-50.

Taramaa, J., et al., Product-Based Software Process Improvement for Embedded Systems, Proc. of the EUROMICRO 98, Workshop on Software Process and Product Improvement, Västerås, Sweden, August 25-27, 1998.

Toivo, K., Interview, August, 1996.

Toivo, K., "Miten valitset ja kehität yksilö- ja tiimikohtaisia mittareita?", Performance Measurement & Cost Management 1996, Marina Congress Centre, Helsinki, 1996-06-11...12.

Trulsson, P., QMS Users Guide Ericsson Telecom, 1543-CSL 109 0003, (internal document), 1993, 83 p.

van Latum, F., van, Solingen, R., et al., Adopting GQM-based measurements in an industrial environment, IEEE Software, Vol. 15, 1(1998), pp. 78-86.

van Solingen, R., Derks, P., Hirvensalo, J., Product focused SPI in the Embedded Systems Industry, Proc. International Conference on Product Focused Software Process Improvement, Oulu, 1999-06-22...24, pp. 86-98.

Vassiliou, A., A Study of fault density during FT, ST and operation considering modification grade, Ericsson Intracom Report, ICM/DA-93:058 (internal document), 1993, 29 p.

Weller, E. F., Lessons from Three Years of Inspection Data, IEEE Software, Vol. 10, 9(1993), pp. 38-45.

Zage, W. M., et al., Evaluating Design Metrics on large-scale software, IEEE Software, Vol. 10, 7(1993), pp. 75-81.

Appendix 1

Development history of Ericsson's software quality measurements

Initial era

In the late 1970's and early 1980's Ericsson established a software development model. However, there were no accepted and consistent quality measurement methods for software development activities. Advanced measurements were performed in large projects and in different Ericsson design centers. The metrics were not defined accurately enough making it difficult to compare the results between companies. Some Ericsson companies were active in measurement matters by introducing their own measurement databases and quality reporting. As Ericsson in Finland has been a pioneer in quality measurements, it has been able to show results since 1981.

The first easily understood quality measure for design work as faults found per kilo Program Store Words (kPSW) after design (in Function test) was defined in 1979. Correspondingly, the period from release up to the end of the first month in operation was used to see the outcome from customer's point of view. In parallel, delivery accuracy measurements were started by using the concept of "Percentage of designs ready on time" as a measure.

A manual way of collection/result handling was sufficient to begin with when Ericsson in Finland was a small organisation and had a low number of delivered products. In 1981, a regular goal setting for quality and delivery precision was started. In the middle of 1980's measurements continued and followed up towards the goals, and a relational database system with flexible reporting facilities was created. Early experience showed e.g. a psychological effect on growth of quality (Antman, 1993). See also (Hirvensalo, 1993) and Appendix 6.

Forced support for measurements

At the end of 1980's a rapid expansion took place at Ericsson in Finland and new areas of software applications (e.g. NMT900) were launched. That is why more attention was also paid to quality issues. A dedicated person was appointed for quality coordination, measurement instructions and routines were improved and some new metrics, for example, to control trouble report answering performance were introduced.

A relational database system, FOCUS-QMS with flexible reporting facilities was created. A data collection form (so called Q-DATA) was introduced for reporting data of products within finished projects to the quality coordinator. The only measurement point was the release of the products. Inserting data into the database was centralised - so was the case concerning output reports. The main output generated quarterly for managers was the Department Quality Situation report (see Appendix 5). On the upper level of organisation, trend curves were presented quarterly (see an example in the Section 3.3.1).

In order to make deeper analyses possible, a number of calculations and reports were extracted from the collected data, such as:

- Quality calculations per project from Function test results (used e.g. for Project Final report, including also fault densities concerning each product,
- Mean fault density and delivery accuracy calculations per department to see actual outcome vs. organisational goals,
- Project Life cycle (FT, ST+ first month, 6 months in operation) listings for products that have passed their first 6 months in operation, and
- Other reports, for example lists of top and bad quality products, and pareto diagrams in order to know more about reasons for changes in trends and mean values.

In 1987-88, Ericsson Telecom ETX) in Sweden specified and implemented a new company-wide support (QMS, Quality Measurement System) for the production of reports and graphs concerning faults and corrections in faulty products. QMS is a multi-user system that has been widely used in Ericsson companies. The main functions were described in a User Guide (Trulsson, 1993).

More attention on preventive quality

In parallel with measurements going on, the technical management started "Quality meetings" in 1987. This forum consisted of the technical director, as well as some department managers and experts. The author was the main resource trying to find new metrics (e.g. complexity) and making some experimental correlation studies around Halstead and McCabe.

The Quality meeting was also a driving force launching development of a preventive method called "Precondition analysis" (1990a). The main purpose of this predictive method was to be able to influence on quality early enough. It was used in several projects to analyse risks and factors influencing quality. In this context the designer's competence was the key factor for design of a good product. In spite of the fact that competence and people metrics are very sensitive, Ericsson in Finland introduced and used them quite widely in competence planning and utilized a well defined profile for individual development. Every designer's competence was evaluated at least once a year in about 20 different areas that were directly related either to SW design model (e.g. function design) or to telecommunication knowledge (e.g. ISDN).

As a conclusion from this era until 1990, it can be mentioned that these measurements made it possible to follow up goals and to see evolution of quality project by project. As there were not many other companies within Ericsson that systematically collected measurement data the material from Finnish projects was also used for research purposes in cooperation with Ellementel and Qualisoft in attempts to develop estimation models for software fault content (Lennselius, 1990).

Not only the measurements based on the amount of faults, but also an extended fault analysis method was created. The author carried through several extensive analyses (Hirvensalo, 1990b) for finished projects to find out reasons and consequences of faults. These analyses covered the design faults detected in tests and also the faults found during operation.

Quite soon as the instructions, routines and measurement results became available, steps were taken to include them in quality training packages and usual information activities. Since 1990 measurements have been continued and further developed (e.g. new reporting functions).

SW Metrics - measurements from theory to practice company wide

The author participated in Ericsson's SW metrics activities that were started in 1989. A group of experts was established to drive the SW metrics work. In the early phase of Ericsson's metrics activities, a lot of knowledge and theories were collected, a network of internal and external contacts was built, and information was spread out by publishing "SW Metrics News" and arranging seminars (Ericsson Telecom, 1991). The work continued as a project with several subprojects in England, Irland, Denmark, Sweden and Finland. My assignment was to define metrics for productivity and quality (Hirvensalo, 1991). Many new ideas arose. As a "guiding star" in the later stage of this project, 3-4 standard metrics were defined for general use within the AXE design process.

SW metrics provided several useful methods and tools such as the SW metrics Process, Metrics Definition Forms and Leading indicator Forms, a Measurement Manual, and test results from several industrial methods (e.g. HP Software Process Assessment (SPA), IBM Function Points).

The metrics were based on state-of-the-art technology and also on experience from Ericsson subsidiaries, for example, some metrics were inherited from Ericsson in Finland. The value of the results has been obvious, because most of them were put into use after the end of the project. Furthermore, SW Metrics encouraged better focus on new methods, for example Cleanroom (Mills, 1987).

The arrival of PQT - standardized way of measurement

The following was concluded during the last SW Metrics Seminar (EricssonTelecom, 1991) concerning the future activities of the expert group:

"You are the best experts to define the Ericsson standard metrics in the area of Productivity, Quality and Time". So, this was a starting shot for a selected set of standard measurements within Ericsson - called PQT. A "Charter" was published by the top management for development of all metric definitions, forms, instructions and tools for aggregating the measurements. The objective of this project was to get the first complete PQT quarter report ready in April 1992 and to secure a continuous PQT reporting. Ericsson's approach here is a very good example of how the management says what they want: "all managers within Ericsson companies are responsible for implementing PQT into their respective operation". After getting the measurement system and collection started, a directive for application of PQT measurements was initiated by Corporate Executive Committee demanding performance measurements, as well as rational analysis and improvement work.

A basis for better understanding of public "consciousness" was the PQT Manual, which was distributed throughout Ericsson. A brief description of what PQT measures is given in Appendix 3. The intention with PQT was to introduce "good enough" measurements, i.e. not too complicated for general use. In spite of the fact that management support provides a good discipline, it is even more important to realize the principle of "Measurements for you". PQT reporting is not only needed to control achieved results within software development on the top level of the organisation, but it also serves as a useful tool for showing the actual situation and improvements within each particular organisation. Data collection that was expanded to cover several measurement points in the project process, supported data ownership and responsibility of the line organisation.

In the beginning, it was difficult to get each company involved. Therefore, a continuous PQT support to answer problems and further needs was introduced. Active data collection in tens of Ericsson companies has been continued since the first quarter of 1992. A multi-user database tool with a menu-system and pre-defined graphs was available on a mainframe. Input menus were strictly adapted to Data Collection Forms to ease the insertion of data. Up to now Ericsson has stored data from hundreds of projects and thousands of modules (e.g. SW modules written in High Level Languages) to be analysed. PQT data was at first mainly used on corporation level and in large projects at Ericsson Telecom (ETX). A study of fault densities considering modification grade was performed at Ericsson Intracom in Greece (Vassiliou, 1993).

ISO 9001 as a prerequisite

At the same time as SW metrics and PQT were introduced, Ericsson started a project to achieve ISO 9001 certification. As a result of this project, more attention was paid on use of measurement results due to the "Management Review" concept. Quarterly "Quality Meetings", held on upper and lower organisational levels adopted a mandatory practice to learn about results by going through well prepared Quality Reports and by making decisions on improvement actions.

ESSI as user of PQT

Ericsson System SW Initiative (ESSI) for AXE10 SW quality was launched by top management. It is an Ericsson-wide program to achieve break-through improvements in SW development activities. The creation of ESSI was based on ideas from Ericsson's Quality Management, SW Action Team (SWAT)

and Management's commitment at SW Seminar in June 1993. ESSI is a vehicle towards the goal of being the best telecommunication SW supplier in the world. This was to be achieved by having:

- World class quality software that is measured as Fault density during the first 6 months from external release to the customer, and
- World class SW development process, that is measured as Capability Maturity using the SEI CMM (Capability Maturity Model).

Corporate targets were broken down by using Policy Deployment including root-cause analysis. Vital actions expected to have an effect on Fault density during the first 6 months were taken. The actual effect was measured in Ericsson design centers by using PQT. Thus, ESSI is a very concrete example for utilisation of PQT measurement results because it was possible to see the present performance level in terms of faults/KNCSS (Kilo Non-comment Source Statements) as well as the actual rate of improvement (Mobrin, Wästerlid, 1997). Annually, a set of performance indicators was introduced.

PQT further development - effects in Finland

Because persons from Finland participated actively in the work within the PQT group, there were good possibilities to influence and to implement PQT in local practices at LMF. PQT brought more formalism and several points of collection. The use of the existing local database was continued and an automatic data transfer from LMF's FOCUS-QMS to central PQT database was created. Measurement definitions, instructions and processes were adapted to cover both corporate PQT and local practices. This integration led to the process that is described in Section 2.3. In Chapter 3 and in Appendix 4 this existing process and its utilisation is analysed in the light of a few case examples. See also (Hirvensalo, 1996a).

Metrics for object oriented programming

Around the same time another group at Ericsson put a lot of effort in studying the introduction of metrics for new object-oriented software applications written e.g. in C++ and Erlang (Armstrong, 1996) languages. Familiarization with new measurement theory and framework brought many new ideas concerning the definition and modeling measurements. Predictions for remaining faults, reliability modeling, failure detection and complexity metrics etc. were investigated. The ideas also influenced the further development of PQT.

Extended PQT, EPMG

At the end of 1996, a new project was established to develop a new WWW-based PQT system to improve its functionality, user friendliness and usefulness. All measurement definitions were refined based on the work done by **Ericsson Provisioning Measurement Group (EPMG)**. The Measurement Result Definition (MRD) for each measure was improved to introduce some new concepts, such as measurement customer, attribute and scale type. Tool support and training was significantly improved. The PQTweb tool was accessible on Unix and Windows environments and implemented in Netscape 4.x. In spite of many improvements, the tool as a first pilot WWW solution suffered from rather poor real-time performance. The new PQT system provided a remarkably improved PQT Manual and fully replaced paper distribution. Many new facilities for graphical representation of results were introduced (e.g. 25/75% percentile).

WCP Program

In early 1998, the top management of GSM systems started the **World Class Provisioning (WCP)** program. Ericsson has become a largest provider of infrastructure for mobile communications. GSM systems is one of the biggest Business Units (BU) within Ericsson and the measurement program was set up as part of the SPI program for WCP. The WCP program was supported by a high level goal regarding quality, productivity and lead times. As a result of initial workshops, 10 guiding principles were offered to the organisation as potential means to achieve improved performance. *Learning organisation* was one of the most interesting principles included, e.g. stimulating communication and feedback, root-cause analysis, and evaluating/packaging good practices. The WCP program applied a

very sophisticated goal setting process to interpret high level goals for setting of concrete measurable objectives as well as for tracking of the objectives. The method that the measurement team used was the Goal Attribute Measure (GAM) method similar to Goal/Question/Metric (GQM) model. Some metrics were PQT compatible, but many other metrics were included to address to specific objectives. For further details of experience, see (Nilsson, 1999).

Use of GQM at Ericsson

During the second half of 90's several GQM Plans have been prepared and implemented also in connection with major improvement programs (e.g. programs aiming to reduce the Time-to-Market (TTM) lead-time). An EC research project called PROFES has acted as a driving force for expanding the industrial application of GQM in Europe (Birk, 1998) and selected two application projects from Ericsson. Some examples are presented in references (Taramaa, 1998, Elf-Mattila, 1999 and van Solingen, 1999).

Restructured ESSI program

In 1998 Ericsson also realised that improvement programs like ESSI and incorporated measurements should become closer to business areas since targets are set by the business units (i.e. in BUs). One of the main areas of the restructured program is measurement. The program aims at utilising the methods inherited from previous activities, like PQT, EPMG and WCP as well as at supporting tool development. The measurement systems should also follow the new principle and so the three existing systems (including PQTweb), were replaced by a new WWW based EMS (Ericsson Measurement System).

EMS is an integrated, platform-independent measurement environment that consolidates project and organisational data and provides engineering management with factual decision-making information. It is intended for measuring performance both in product development and in maintenance activities. EMS focuses on automatic data collection, e.g. fault data is automatically retrieved from trouble report database, recorded to the phase of detection, and categorised to external/internal faults. The results are viewed from the perspective of projects, organisations and product lines. The new tool has significantly improved run-time performance but the development effort and updating of data are still issues.

Examples of local measurement tool activities

In a large multi-national company working on many and rapidly evolving business areas, the measurement tools supported by corporate programs are not enough. The organisational units working on lower levels often need support for the control of projects, and measurement and feedback during projects. Several measurement tools for specific local purposes have been developed at Ericsson subsidiaries. Some examples worth mentioning are local database systems at Ericsson in Holland (see Hirvensalo, 1996b), in US and in Canada. Our scope in this context is limited to tool activities at Ericsson's Finnish design center.

WebIR - a tool supporting inspections

As a local activity in connection with ESSI improvement program a team at Ericsson in Finland implemented computer support for inspections (Jokikyyny, 1998, 1999), (Elf-Mattila, 1999). The WebIR is a WWW-based tool that supports inspection planning, execution and feedback. It integrates measurement data collection and visualisation on-line during projects. It is accessible by everyone in order to motivate designers for higher inspection performance. The WebIR tool includes a graphical interface that gives users an ability to get information they want and how they want it. For example, defect detection rate (defects/page) can be visualised with respect to preparation rate (pages/man-hour). Many kind of other inspection measurements, like majors and minors detected per document type, or per process phase can be discussed.

LUCOS – support for project controllability

The goal of the LUCOS project (Jokikyyny, 1999), was to develop a complete framework for improving controllability of product development. The LUCOS project resulted in a method as well as WWW-based tools. The LUCOS visualisation tool presents real-time status of product development and the

achievement of goals from different perspectives. The JAVA based visualisation client (ViCA) provides a graphical interface to several existing data sources including WebIR. ViCA presents data in panels, which can contain one or more charts and pictures. Data shown in panels is always up-to-date as the metric servers execute the queries automatically.

LUCOS was a joint project between Helsinki University of Technology and industry. Its results were piloted in five industrial companies, one of which was Ericsson. The Mobile Switching Systems Unit at Ericsson in Finland piloted LUCOS in order to develop ways to break down project goals to driving attributes, project metrics, and visualisations. In total, 19 metrics were connected from visualisations to the project tracking plan.

Measurements as a part of Performance Management, BSC/KPI approach

Later on, the measurement program concentrated on business performance measurement, including performance analysis, further definition of measures, and actions for performance improvement. The Balanced Scorecard¹ method was used to define performance objectives across a broad range of perspectives. The measurement process presented in sections 6.4 and 7.1 was applied to derive a few Key Performance Indicators (KPI) corresponding to the performance objectives. All indicators were defined in detail by using MRD and MDD (see appendices 8 and 9). Practical examples of perspectives are Financial, Process, Customer, People and Innovation. KPI measurements were published and maintained in Management Dashboard on Intranet and revised annually. Typical usage applications are performance assessment (goals vs. outcome), and bonus program. For further details of this approach, see (Hirvensalo, 2001).

¹ Developed by Kaplan & Norton

Appendix 2

The Quality Machine

The following virtual machine is visualising how should the measurement activities and other quality related issues work together.

This can be illustrated by the “Quality machine” shown in Figure 2-1. The purpose of this appendix is to give an introduction to different items of the Quality Machine. The original picture presented by Eskelinen (1995) has been modified a bit and a brief interpretation of its “message” is given below. There are “meter displays” that are needed all the time to keep control **during** the design and also to show the actual (final) quality **after** completion of design.

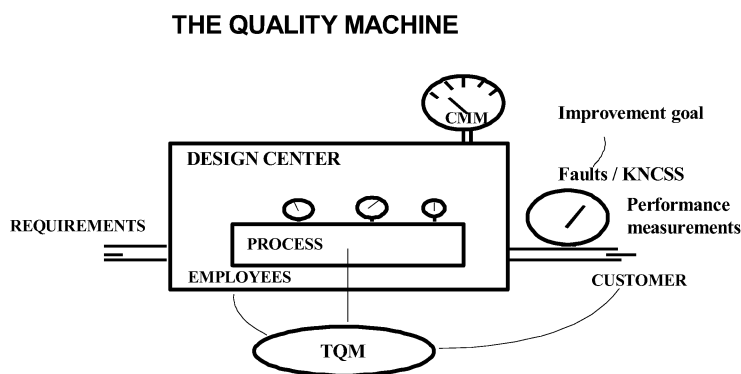


Figure 2-1 The quality machine for a design center

The Quality machine process “starts” from customer **requirements** and also ends at the **customer**. The final indication of quality is determined by the customer regarding the complete system of SW & HW (“In-Service performance” and customer satisfaction).

Faults/KNCSS¹ is an example of a quality indicator used during both test and operation phases to give feedback to the design organisation in relation to the volume of new and modified products developed. This study mainly concentrates on the quality of these new and modified parts, not on the quality of the complete AXE switching system.

Interdependencies between customer-oriented quality attributes, such as low system down-time, number of reloads/system restarts, and driving design factors are not well known. Relations to these factors from the design process and couplings to customer satisfaction are identified in this study. Some influencing factors are dealt with, but those factors are not the

¹ Expressed as Faults/kPLEX statements in ESSi improvement program

main subject in the study. Customer satisfaction measurement is included in TQM (Total Quality Management) methods.

TQM is a management philosophy that focuses on customer satisfaction, continuous improvement and everyone's involvement. As one important aim of TQM is customer satisfaction, TQM is an aid to meet the needs and expectations of customers (in accordance to Ericsson's Quality Policy even to exceed them). TQM focuses both on external and internal customers. Customer satisfaction is not only satisfaction with the product but it also includes technical support, processes and any kind of interaction with the customer.

TQM provides Management and Quality Control tools and aids to affect processes and products in order to improve quality. TQM also provides tools for root-cause- and statistical analyses.

Employees, their competence and satisfaction have a very essential impact on quality. Total Quality can be achieved only when everyone actively participates in tasks of improving quality. Employee attitudes and confidence on measurements (that just measure their work) are a basis for successful utilisation of measurement results. Therefore interviews and experience collection has been performed in connection with this study. Internal marketing and supporting activities of measurements towards employees are also needed in order to make measurements happen successfully.

The Process introduced has an essential impact on a company's performance. Well defined and consistent processes followed by every software *design center* are used both by the line organisation and projects. Process Management is a systematic approach to organising, managing and improving processes. Modern thinking involves indicators and established measurements of each process to evaluate their adherence, performance and effects of process improvement actions.

Capability Maturity Model (CMM) "meter" is measuring the internal maturity of the entire software process. CMM requires process-oriented measurements and analysis of results early enough in order to take preventive actions. In order to get the CMM "meter" to display as high maturity as possible (out of five maturity levels), also some improvement in measurement practices is needed. Requirements for achieving the levels 4 and 5 concerning measurements are therefore analysed in Section 4.1. It is expected that high maturity indicated by CMM "meter" should lead to lower fault density (F/KNCSS) and higher customer satisfaction.

Improvement goal (ESSI goal) relates to Ericsson-wide improvement program to halve the fault density of products seen during the first 6 months after delivery to the customer. It also involves an achievement of higher CMM level.

Fulfilment of ESSI objective is followed up by **Performance measurements** (PQT measurements) and CMM assessments.

Appendix 3

PQT description

PQT (Productivity, Quality, Time) is a collection of measurements indicating the performance of Ericsson's Research & Development operation. Ericsson's standard PQT was initiated in 1992 by Corporate Executive Committee demanding performance measurements, as well as rational analysis and improvement work. Ericsson in Finland (LMF/T) has been a pioneer, having active quality measurements already ten years before PQT was launched. LMF/T participates also in further work as a member of the PQT group.

The objective with Ericsson PQT has been to implement a measurement system that serves as an integrated part of the continuous improvement process by:

- ## Making it possible to visualize long-term trends and thereby identifying areas of improvements,
- ## Supporting "management by facts",
- ## Supporting the definition and follow-up of new quantitative objectives, and
- ## Increasing quantitative knowledge from the development process.

Productivity, Quality and Time were defined as the three entities that together provide the complete performance view.

Productivity

Volume/Manhour

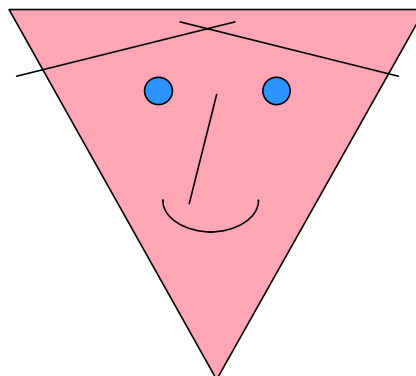
Quality

Fault density

Planning precision

Delivery precision

Quality in TR/AC handling



Time

Project **Lead Times**
between TGs/milestones

PQT metrics are 'assessment' type metrics that mainly "stare" afterwards at the final result of design, not "driving" metrics.

APPENDIX 3

PQT DESCRIPTION

Productivity is expressed as Volume / Manhours. As a first step Ericsson has used "implementation" volume (e.g. High level language statements) here. Volume of "functionality" is studied in connection with further development of PQT.

Quality is measured as Fault density (Faults / Volume) in three phases:

- ## during Function Test up to internal product release,
- ## between release and delivery to the customer, and
- ## 6 months accumulated from delivery to the customer.

Precision expresses process quality and is measured as the relation between Planned and Actual leadtimes. Both Delivery and Planning precisions are measured. Quality in TR (Trouble Report) and AC (Approved Correction) means high performance in answering TRs as well as in releasing corrections.

Time measurements - quite simply - mean measurement of calendar based Lead Time in projects. An active data collection in tens of Ericsson companies has been continued since the first quarter of 1992. Data collection supports data ownership and responsibility of the line organisation. Multi-user database tool with menu-system and pre-defined graphs are available.

Up to now Ericsson has results from hundreds of projects and thousands of products (e.g. SW products written in high level languages) to analyse. Results, for example, the fault density for delivery date + 6 months are widely used in ESSI (Ericsson System SW Initiative) improvement program to indicate whether the quality at the customer is getting better.

Appendix 4

Existing quality measurement process at Ericsson in Finland

The purpose of this annex is to present the measurement flow in Ericsson Finland in more detail. Figure 4-1 shows a simplified "process" view of Finnish application to software measurement procedure used since 1992 up to now.

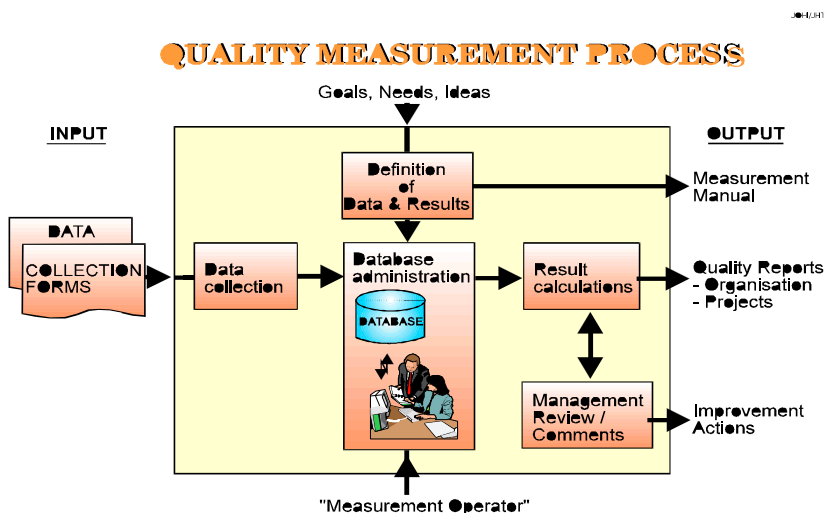


Figure 4-1 Measurement process

Input

Data is collected as primitive information from each ongoing projects. Main inputs are the *Data Collection Forms* that are sent from design (line) organisations to quality organisation. Input Information flows either on paper or as electronic mail.

Data Collection

Collection of data is *Event* controlled. This means that all data shall be reported as it becomes available at the time of the event. Flow of information - *when* and *what* data is collected from project

process - is shown in Figure 4-1. Points in collection represent the ordinary review points (events), called *Tollgates* and *Milestones*, in the Ericsson's project management process.

There is project-related data as well as product-related data. Data on project level consists mainly of date and manhour information whereas on product level of actual sizes and faults etc. Each rectangle in the picture (see Figure 4-2) visualises a specific Data Collection Form to be sent to quality organisation, which ensures that all the relevant data from projects the organisation is responsible for is received in time.

Thus, the data collection is made in a consistent and consequent way by reporting data when it is demanded.

Data collection is not fully automated at the moment. However, there are some tools available for supporting collection. Tools exist for counting of program size in statements and for collection of faults per product registered in Trouble report handling system etc.

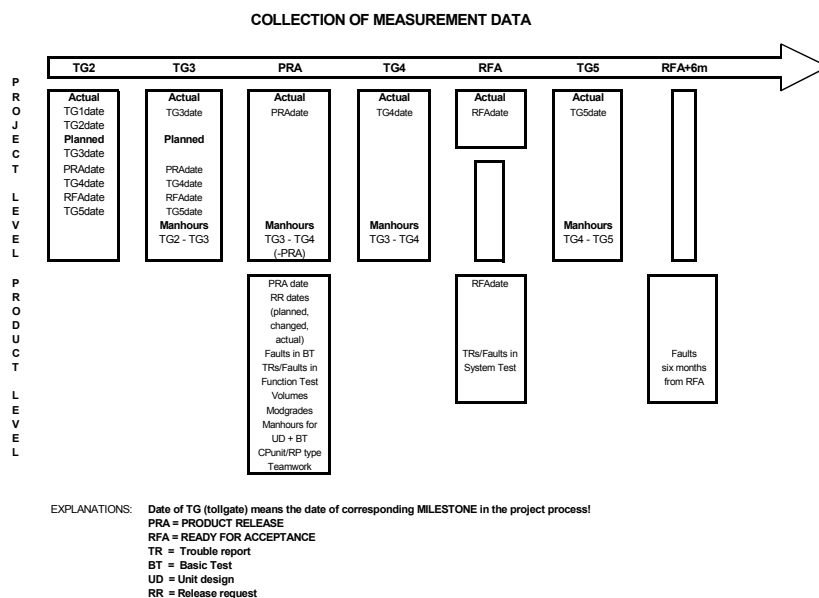


Figure 4-2 Data collection during the project process

Definition of Data & Results

The collected data is well defined by using a specific form in order to keep consistency. *Measurement Data Definition (MDD)* is used for each collected data to describe briefly:

- ≠ Name and Description
- ≠ Purpose/Benefit
- ≠ Measurement unit
- ≠ Method of collection
- ≠ Application
- ≠ Method of implementation
- ≠ Strengths
- ≠ Risks/weaknesses.

Examples of typical MDD :

PLEX Source Program volume, Fault per product

Measurement results are derived and presented by using another Form called *Measurement Result Definition (MRD)*, which contains:

- Name and Description
- Purpose
- Measurement unit
- Presentation
- Measurement data used (usually two or more data is used)
- Analysis of results
- Potential causes
- Remedial actions.

Example of a typical MRD :

Fault density in PLEX SW uses two defined Data: Faults per product divided by PLEX Source program volume.

Database administration

Administration is performed by a dedicated person, the *Measurement Operator* who has received training for using of the *Database system*. The Operator constitutes a link between data collection and reporting. He/she does validation of the measurement data and feeds the data into the database. When the system does not accept the data, the operator identifies the problem and contacts the person who has delivered an erroneous form.

Ericsson in Finland has a relational database with flexible reporting facilities. It contains both local data and standard items for Ericsson's PQT database. In this way each data is registered only once in Finland and then automatically transferred to the central database. For the sake of security, as a rule until now, only one operator in Finland has been authorized to insert data into the database.

Result calculations

The measurement operator generates the graphs and *Result calculation* listings needed by the management to be enclosed for example in Quality reports. Depending on feedback given by the organisation responsible for its own data, some recalculation is sometimes necessary e.g. due to faulty or lacking/delayed input data).

Management Review/Commenting

Management Review (in accordance to ISO 9001 standard) reflects managements' commitment to and conclusions from measurement results. Quality report is sent to this quarterly meeting, reviewed and analysed. Observations regarding how the goals are met, findings like good news, any jumps in trends, either positive or negative, possible reasons for deviations etc. are discussed and commented. This meeting brings a lot of expertise and knowledge at place, and measurement results provide facts to make decisions on *Improvement Actions*. As usual the actions to be taken are controlled by an action point list. To maximize the benefits of the management review it is used on all different organisational levels. According to the role of management review it should also make decisions on utilisation of some well proven facts, on introduction of new metrics or statistical methods to be applied in the future.

Output

All descriptions and instructions from Definition of Data and Results phase are collected into one binder: *Measurement Manual*. Its main content is listed below:

- Measurements in brief
- Data collection instructions and Measurement Data Definitions (MDDs)
- Measurement Result Definitions (MRDs)
- Quality during design and operation (e.g. fault density)
- Quality of planning and delivery (precision)
- Quality in Trouble report/Correction handling
- Lead time measurements
- Productivity, efficiency, cost measurements
- Other measurements (e.g. competence)
- Department and project specific measurements.

This manual and its changes are distributed to all department and quality managers. The modern way of distribution aims to computer stored manual.

Quality reports are prepared quarterly by each department and reported to upper organisational levels. The main structure of this report is as follows:

- Action point situation
- Quality goals vs. measurement results
- Product and service quality
- Delivery accuracy
- Customer complaints
- Audits and assessments
- Status of improvement program
- Cost effectiveness
- Satisfaction
- Analyses (e.g. Benchmarking).

A collection of trend curves and graphs extracted from measurement database is enclosed in Quality reports.

Many kinds of listings are produced per line *organisation* unit or per *projects* in order to analyse the actual quality within a project product by product.

Improvement actions decided by the management are handled in design organisations and included in improvement programs.

Appendix 5

Department manager report (example template)

Distribution
DEPARTMENT MANAGER

Objective values in parentheses
NOP = number of products

AXE SW-DESIGN - QUALITY SITUATION AT <DEPARTMENT>

Period: Q1/96

DESIGN QUALITY (... PRA)	GOAL	PLEX SW	NOP	ASS SW	NOP
Total program volume		xx stmts		yy PSW	
New & modified program volume		xx stmts		yy PSW	
Number of faults found in Function Test		xx		yy	
Number of faults / total volume	(x.xx)	x.xx	mm	y.yy	nn
Number of faults / new & modified volume	(x.xx)	x.xx	mm	y.yy	nn
Percentage of faultless products in Function Test:		xx.x %			

DELIVERY QUALITY (on internal release)

Number of delivered products:		zz
Percentage of products delivered on time	(xx.x)	xx.x %
Average delay in days from planned date:		xx.x

QUALITY SEEN IN SYSTEM TEST(PRA ... RFA)

Total program volume		xx stmts		yy PSW	
New & Modified program volume		xx stmts		yy PSW	
Number of faults found in System Test		xx		yy	
Number of faults / total volume	(x.xx)	x.xx	mm	y.yy	nn
Percentage of faultless products in System Test:		xx.x %			

SIX MONTHS OPERATION QUALITY (RFA ... 6 months)

Total program volume		xx stmts		yy PSW	
New & Modified program volume		xx stmts		yy PSW	
Number of faults found during first six months		xx		yy	
Number of faults / total volume	(x.xx)	x.xx	mm	y.yy	nn
Percentage of faultless products during in six months:		xx.x %			

Explanations of measurement units:

stmts = statements (written in PLEX)

PSW = Program store words (generated from assembly SW)

Appendix 6

Early quality measurement experience at Ericsson in Finland

Attitudes towards measurement

There are negative attitudes, usually when starting measurements:

- Our work can not be measured
- There do not exist any measurable parameters in SW
- You are not measuring right things
- It shows nothing
- The data is too sensitive
- How is the metric defined - it must be changed
- This is quality fuss
- Do not disturb our work with that foolery.

But, there are also many positive opinions such as:

If we run this and that operation, it is to be measured
If you measure, you know where you are.

Making measurements happen

First of all, strong management commitment is needed. A person who "sponsors" the measurement idea and encourages, is of great value.

Power is needed to keep the measurement process running, thus a "fighter" should be appointed. He/she takes the role of 'measurement operator', supervising the data collection, data base maintenance, validating, 'hunting' the lacking data, producing reports etc.

Patience, toughness, honesty are needed because of the repeatable nature of matters.

Data collection should be an essential part of daily work and processes, not an extra load. Points in time of collection should be linked to well known milestones within projects.

Measurements should not be considered as too "high science" - keep them simple, use easily understood metrics, just try to make them good enough.

It is also important to promote success stories.

Appendix 7

The statistical methods used

The main method is Statistical analysis. This method is commonly applied to Statistical Quality Management. In this study, methods and tools are used to analyse relationships that exist between different variables collected from the software.

The purpose is also to prove the statistical significance of results. Common statistical basics are followed. Below, the main concepts used in the study, are listed:

Correlation analysis

Pearson product-moment *Correlation coefficient* (r) is used to express linear relationship between two variables, x and y . It is based on covariance and standard variances. Correlation coefficient can vary between $(-1, 1)$. The closer the value of coefficient is to $r = 1$, the closer the two variables are positively correlated.

Regression analysis

The method is needed to indicate how variable Y depends on X , when a very good correlation between the variables has been obtained during the correlation analysis. Least squares method is used to calculate the coefficients for a straight line representing *linear regression*. Also *multiple linear regression* models, in which more than one distinct predictor variables can be used, will be applied in the study (when appropriate).

Significance

Because the regression and correlation coefficients are usually calculated from samples, their *statistical significance* is to be tested. Significance means the evidence that variable X affects variable Y at the probability of 95%. For example, Student-t test method can be used for significance test. Significance of correlation in our study is tested based on the table of critical values for product-moment correlation coefficient presented in literature (Conte, Dunsmore, 1986).

Trend analysis

In this study several time-based trends are investigated. There are series of results which show upward or downward tendency. Useful tools commonly used within Statistical Quality Control (Oakland, 1990; Montgomery, 1991) are:

SHEWHART Control Charts for measurements and attributes

- \bar{X} CHART (Mean) that monitors location in terms of means
- R CHART (Range) that monitors variability in terms of range
- P CHART (Proportion Defective)
- C CHART (Average number of Defects)

APPENDIX 7

THE STATISTICAL METHODS USED

In connection with the charts above so called 3-sigma limits are used to see whether a process is 'in control' or 'out of control'.

When a trend is presented, then a '*rolling average*' can be used to eliminate acute variations.

Sometimes, when a long term trend is analysed, then a smooth out curve is derived from data values of each period (e.g. quarter). This is best suitable for variations that are quite small.

Rough Set Analysis

In its wide sence, the Rough Set Analysis (Hartkopf, Ruhe, 1998) is a technique that is used to find the best possible combination of several *independent variables*, which have an impact on a certain *dependent variable*. We adopted this method in limited form.

Other concepts

Standard deviation, variance. For module sizes, also intervals of variations (e.g. 25%, 75% quartiles, mean) are used.

Statistical Tools

Statistical standard functions and graphical aids included in Microsoft EXCEL are used. It contains several functions that are based on those methods described above.

Furthermore, statistical facilities and build-in graphs implemented in Ericsson's PQT system (Ericsson, 1995) are used.

Appendix 8

Renewed template for Measurement Data Definition(MDD)

Title: <ATTRIBUTE> OF <MEASUREMENT OBJECT>

Description of Attribute:	Unit:
Purpose/Application of Measurement Data:	Scale Type:
Definition of Measure:	
Method of Collection: When: Who: How: Where:	
Validation of Measurement Data: When: Who: How:	
Strengths/Risks of Usage: Strengths: Risks:	
Measurement Data Relations:	
Other Comments:	References:

Appendix 9

Renewed template for Measurement Result Definition (MRD)

Title: <TARGET ATTRIBUTE> OF <EVALUATION OBJECT>

Description of Attribute:	Unit:
	Scale Type:
Purpose of Measurement on Attribute:	
Definition of Result:	
Presentation of Measurement Result:	
Measurement Customer:	
Measurement Data Used:	
Interpretation of Measurement Result:	
Scope of Application:	
Strengths/Risks of Usage:	
Strengths:	
Risks:	
Other Comments:	

Appendix 10

Example MDD

Detected Fault Content of Product

Description of Attribute: The number of faults detected during a specific activity (e.g. during a specific test activity, or a time period in operation) per individual of an implementation product, identified by product number and revision state.	Unit: Number of faults
Purpose/Application of Measurement Data: The number of known faults is an indication of functional quality of a product, which can be used for fault density calculations for a set of products and for identification of top quality (zero fault) products or low quality products. The number of faults is also used when doing root cause analysis where reasons and origin of faults are studied.	Scale Type: Absolute
Definition of Measure: By 'faults' here is meant problems that have been reported, analysed and got a decision for some corrective action. The number of faults are collected for a specific phase or activity in the life cycle of the product, regardless of parallel activities. In case of incremental design a specific activity might be split up over time, then the number of faults are summed. The phases / activities are: <ol style="list-style-type: none">1) Code inspection (e.g. Early code inspection, Final code inspection), where only MAJOR faults are counted. Definition of faults: see the document 'Glossary for Inspection Activities' in the process Review and Inspection Activities.2) Basic test, including all unit test activities like interpreter test, emulator test, module test (e.g. for AXE 10 up to MS7).3) Function test, starting after completed basic test (e.g. for AXE 10 at MS7, i.e. the first AD, when applicable) up to the release of the product (PRA).4) System test, normally starting at the release of the product (PRA) up to ready for acceptance date (RFA) for the product or equivalent point in time when it is handed over to the first customer.5,6) Operation periods 0-6 months and 7-12 months after RFA or equivalent point in time when the product is handed over to the first customer (faults found by customer or by Ericsson staff). When trouble reports are used as basis for counting the number of faults, then only primary analysed reports having answer code B are counted. The number of faults are collected per product identity. The collected faults might belong to the product itself (e.g program code), but also to documentation of the product (e.g. Data Change Information (DCI)). The TR REGISTRATION DATE, not the Answer date, is used to determine whether a fault is counted to belong to the period RFA+6 months. All priorities (e.g. A, B, C) of TRs are considered, but naturally matters of beatifulness can be excluded.	

Method of Collection: When: Data is collected for the product 6 months after RFA or equivalent point in time when the product is handed over to the first customer and reported into PQT when the majority of the trouble reports are analyzed (e.g. 4 weeks after passing RFA+6 months). Who: The organization responsible for the design of the product. How: E.g. QMS function: Number of TRs per product. Where: Original data source: e.g. MHS, TRtool, Clear DDTS	
Validation of Measurement Data: When: The quality of data is validated immediately after collection of faults. Who: Quality Manager/coordinator and/or PQT Operator. How: For example, the operator checks/controls that the specified collection method has been followed (by making samples etc.).	
Strengths/Risks of Usage: Strengths: The TR handling is well known and understood within Ericsson, which facilitates the collection of faults. Risks: Detected faults are not always registered in TR handling systems. Unanswered trouble reports may influence on the correctness of the measure. Designers might fear that fault density calculations are interpreted as an indication of their performance.	
Measurement Data Relations: The collected faults must be related to PRODUCT ID, ORGANIZATION, PROJECT, SUBPROJECT	
Other Comments: QMS is provided with functions specifically developed for the purpose.	References: Modification Handling Process User Guides of TR handling tools (e.g. Clear DDTS) PQT Manual, see 'Filling-in directives for PQT data

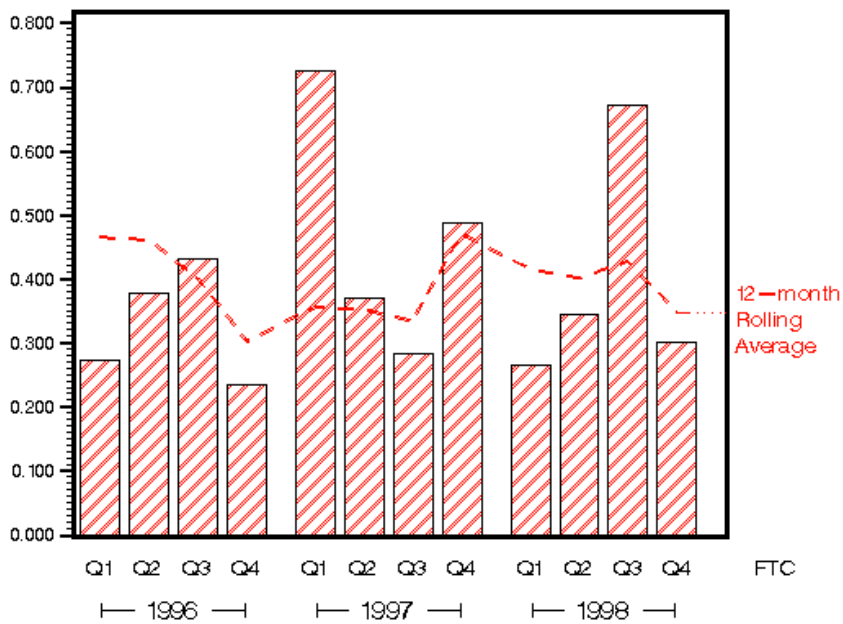
Appendix 11

Example MRD

Fault density in C/C++ software design

Description of Attribute: Fault density is a product attribute capturing the inherent quality of a set of implementation products tied to a development project and/or a development organisation. The attribute reflects the fault density detected in a certain project phase or during a certain product state.	Unit: Number of faults / kilo non-commented source lines C/C++
	Scale Type: Ratio
Purpose of Measurement on Attribute: The measurement is needed to monitor performance in the C/C++ SW design area and to visualise quality improvements stated by the management. The measurement is also used to calculate and analyse top and low quality products.	
Definition of Result: Mean fault density for a set of products is defined as the total number of faults detected and associated with the C/C++ products during the period in question (phases 1-6 below, see Presentation) divided by the sum of the volume of the product set. $\text{Fault density} = 1000 * (\text{Total number of faults}) / (\text{Sum of volumes in source lines})$	
Presentation of Measurement Result: The result is presented in the PQT report for the organisation and/or project in question: A) for each one of the following six phases as: 1) FAULT DENSITY in Code inspection 2) FAULT DENSITY during Basic test (including all unit test activities) 3) FAULT DENSITY during FUNCTION TEST (Integration Use Case Testing) 4) FAULT DENSITY during SYSTEM TEST, normally starting at internal release, up to external release or EQUIVALENT DATE when the product is handed over to the first customer. 5) FAULT DENSITY 6 MONTHS AFTER external release or EQUIVALENT DATE when the product is handed over to the first customer. 6) FAULT DENSITY 7-12 MONTHS AFTER external release or EQUIVALENT DATE when the product is handed over to the first customer. B) as FAULT DENSITY for a consecutive subset of THE SIX PHASES . How the graph in principle looks like is shown at the end of this MRD. X-axis shows quarter. Y-axis shows quarterly average fault density bars and a 12-month rolling average. The fault density figures are related to C/C++ SW design in the measurement report.	

<p>Measurement Customer:</p> <ul style="list-style-type: none"> - Line organisation managers who committed themselves in improvement goals, that is, management of the organisation/operation. - Project managers who committed themselves in specific quality goals for their SW design projects. - Quality managers
<p>Measurement Data Used:</p> <p>MDD: Detected fault content of product MDD: C/C++ Source Program Volume</p>
<p>Interpretation of Measurement Result:</p> <p>Check whether the goal has been reached or not. If not fulfilled, then analyse the measurement results in more detail to identify the causes. If the organisation has continuously met the goal for one year, then set a new goal. Pay attention to the time-based trend of the results.</p>
<p>Scope of Application:</p> <p>Fault density result is valid for the development organisation and/or project(s) under consideration. Measurement result is valid for projects that have concluded Function Test (at least).</p> <p>In all companies developing C/C++ software applications.</p>
<p>Strengths/Risks of Usage:</p> <p>Strengths:</p> <ul style="list-style-type: none"> - The defined measure is a part of Ericsson's standard measurements and provides a consistent way of presenting fault densities within C/C++ development. - Fault density is a key driver behind perceived product quality and development/maintenance costs. <p>Risks:</p> <ul style="list-style-type: none"> - Fault density is in fact not a measure of product quality, it reflects quality given in known and stable operating circumstances. If this distinction is not made clear, the results may be misused.
<p>Other Comments:</p> <p>The result does not indicate other quality factors (e.g. readability) or type and origin of faults. Neither is any regard taken to criticality of failures caused by the faults.</p>

Example Graph

Source:

The MRD is based on Ericsson's Measurement System (EMS) manual (formerly PQT Manual), 1999

A template for measurement Utilisation Plan

<p>APPENDIX 12</p> <p>A TEMPLATE FOR MEASUREMENT Utilisation Plan</p>

1(2)

APPENDIX 12

A TEMPLATE FOR MEASUREMENT Utilisation Plan

A template for measurement Utilisation Plan (cont.)

Users	When are the measurements used?				Analysis methods, Defined actions, Meeting and reporting practices			
	using in the	valid during	with analysis frequency	by applying the method	pre-analysed in/by the	Predefined actions	to make decisions in the	attached to the
from the view-point of the [measurement customer]	[phase of utilisation]	[period of validity]	[period]	[analysis method]	[meeting/expert]	[action]	[management meeting]	[report]
Example customers	Example phases	Example periods	Example frequency	Example methods	Example responsibilities	Example actions	Example Forums	Example Reports
Project leader	Planning	Project cycle	Monthly	Root-cause analysis	Quality team	Predefined activity e.g.: If the number of CRs exceeds alarm limit, contact to CR originator to check stability of requirements.	Management Review	Quality report
Test leader	Execution & Control	Improvement program	Bi-monthly	Trend analysis	Improvement team		Technical council	Intermediate report
Line manager	Conclusion & onwards	Fixed time span	Quarterly	Regression analysis	Project office		Project manag. meeting	Final report
External customers		Continuous use	Yearly	7 QC tools, etc. statistical methods	Feedback session			Test report
Development manager		Sofar		Identif. of low/ top quality products	Expert meeting			
Quality manager					Expert			
Process owner					Test team			
Sponsor/orderer								
Corporate manag.								
Product unit leader								

Appendix 13

A template for measurement Utilisation Plan

Measurement Utilisation for _____ at _____ level Example levels: Individual, team, project, process, organisation

Provided measurement results					Why are the measurements used?		
Referring to the goal	focusing on the	with respect to	related to	of/in the	Generated from source	for the purpose of	scope
[goal number]	[target attribute]	[sub-attribute]	[other measure or driving attribute]	[evaluation object]	[system]	[usage application]	[scope of use]
	Cost	Used effort		A1 & A2 subproject at T/M	TRAP	monitoring of the project budget	Both for subproject and main project use
	Planning Accuracy	Milestone planning precision		A1 & A2 subproject milestones	Delta / ECC and project milestone plans	Project tracking and planning constants	Both for subproject and main project use
	Quality	Detected fault content in Basic test		SW units participating in a WP of the subproject	AD Reports provided by designers	Quality assurance	Both for subproject and main project use
Example goals	Example attributes	Example sub-attributes	Example relations	Example objects	Example sources	Example purposes	Example scopes
Strategical goal	Quality	Fault density	Primitive measure	Project(s)	Meas. database	evaluating	Internal use (I)
Operational goal	In-Service Perfor.	System downtime	Driving attribute	Product (s)	Insp. database	improving	External use (E)
Improv. goal	Lead time	Lead time TG2-PRA		(Sub)process	Database xxx	controlling	Both (B)
Project goal	Cost	Planned/actual cost		Resource		predicting	
	Maintainability	MTTR		Organisation		modeling	
	Reliability	Fault tolerance				characterising	
	Precision	Delivery precision				monitoring	
	Planning accuracy	Deviation from plan				determining	
						Training	
						Planning constants	
						Awarding criteria	
						Operational follow-up	
						Project planning & control	

A template for measurement Utilisation Plan (cont.)

Users	When are the measurements used?				Analysis methods, Defined actions, Meeting and reporting practices			
	using in the	valid during	with analysis frequency	by applying the method	pre-analysed in/by the	Predefined actions	to make decisions in the	attached to the
from the view-point of the [measurement customer]	[phase of utilisation]	[period of validity]	[period]	[analysis method]	[meeting/expert]	[action]	[management meeting]	[report]
Project Manager	Feasibility and Execution		monthly	comparing against budget and forecast	project management team		Internal Steering Group	Project Survey in the Project Progress Report
Project Manager	Execution		monthly	comparing the status of different design documents against their planned status at a certain milestone	project management team		Internal Steering Group	Project Survey in the Project Progress Report
Quality Coordinator	Execution			Combining the information in the Basic Test reports	quality coordinator		project management team	WP delivery report
Example customers	Example phases	Example periods	Example frequency	Example methods	Example responsibilities	Example actions	Example Forums	Example Reports
Project leader	Planning	Project cycle	Monthly	Root-cause analysis	Quality team	Predefined activity e.g.: If the number of CRs exceeds alarm limit, contact to CR originator to check stability of requirements.	Management Review	Quality report
Test leader	Execution & Control	Improvement program	Bi-monthly	Trend analysis	Improvement team		Technical council	Intermediate report
Line manager	Conclusion & onwards	Fixed time span	Quarterly	Regression analysis	Project office		Project manag. meeting	Final report
External customers		Continuous use	Yearly	7 QC tools, etc. statistical methods	Feedback session			Test report
Development manager		Sofar		Identif. of low/ top quality products	Expert meeting			
Quality manager					Expert			
Process owner					Test team			
Sponsor/orderer								
Corporate manager								
Prod unit leader								