The Goal/Question/Metric Method:
a practical guide for quality improvement of software development

The Goal/Question/Metric Method:
a practical guide for quality improvement of software development

Rini van Solingen

and

Egon Berghout

THE McGRAW-HILL COMPANIES

London · Chicago · New York · St Louis · San Francisco · Auckland
Bogotá · Caracas · Lisbon · Madrid · Mexico · Milan
Montreal · New Delhi · Panama · Paris · San Juan · São Paulo
Singapore · Sydney · Tokyo · Toronto

Published by
McGraw-Hill Publishing Company
SHOPPENHANGERS ROAD, MAIDENHEAD, BERKSHIRE, SL6 2QL, ENGLAND
Telephone +44 (0) 1628 502500
Fax: +44 (0) 1628 770224 Web site: http://www.mcgraw-hill.co.uk

---

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 007 709553 7

Library of Congress Cataloguing-in-Publication Data

The LOC data for this book has been applied for and may be obtained from the Library of Congress, Washington, D.C.

Further information on this and other McGraw-Hill titles is to be found at
http://www.mcgraw-hill.co.uk
Authors Website address: http://www.mcgraw-hill.co.uk/vansolingen

Publishing Director: Alfred Waller
Publisher: David Hatter
Typeset by: The Authors
Produced by: Steven Gardiner Ltd
Cover by: Hybert Design

1 2 3 4 5 CUP 3 2 1 0 9

Printed in Great Britain at the University Press, Cambridge

Table of contents

Acknowledgements

This book is based on many experiences we had in GQM measurement programmes. These experiences were gained in cooperation with so many other people. We like to emphasise that this book could not have been written without the help of all these people.

First of all, we want to thank the Schlumberger/Tokheim project teams that participated in the GQM measurement programmes. Their positive experiences and enthusiasm motivated us to continue the enhancement of the GQM method. It is not possible to include everyone personally, however, special thanks to Wim van der Bijl, Henry van den Boogaert, Erik Rodenbach, Erich Sigrist, Frank Simons, and Anita Verwegen.

The (inter) national research projects ESSI/CEMP, PROFES and SPIRITS certainly also contributed to the development of the GQM method presented in this book. Through the cooperation with all the organisations involved in these research projects, practical application of GQM was realised, evaluated and improved. Again, for these projects it is also impossible to name everyone involved, however, special thanks to Andreas Birk, Janne Järvinen, Rob Kusters, Frank van Latum, Markku Oivo, Dieter Rombach, Günther Ruhe, Jos Trienekens, and Erik van Veenendaal.

Several Master of Science students contributed to the research through supporting the application in practice and through adding many suggestions for improvements of the GQM method. Many thanks to Erik Kooiman, Hans Leliveld, Shyam Soerjoesing, Paul Stalenhoef, Arnim van Uijtregt, and Cees de Zeeuw. A special thanks goes to Hans Leliveld for his contribution to an elementary version of this book. Marieke van Santen for her contributions to the web pages. Finally, we thank Rina Abbriata, Conny van Driel and Karin Nuijten for decoding our handwriting.

Rini van Solingen,
Eindhoven

Egon Berghout,
Rotterdam

The Netherlands
March, 1999

Foreword

The original ideas for the Goal Question Metric Paradigm came from the need to solve a practical problem back in the late 1970s. How do you decide what you need to measure in order to achieve your goals? We (Dr. David Weiss and I) faced the problem when trying to understand the types of changes (modifications and defects) being made to a set of flight dynamics projects at NASA Goddard Space Flight Center. Was there a pattern to the changes? If we understood them could we anticipate them and possibly improve the development processes to deal with them? At the same time, we were trying to use change data to evaluate the effects of applying the Software Cost Reduction methodology on the A-7 project requirements document at the Naval Research Laboratory.

Writing goals allowed us to focus on the important issues. Defining questions allowed us to make the goals more specific and suggested the metrics that were relevant to the goals. The resulting GQM lattice allowed us to see the full relationship between goals and metrics, determine what goals and metrics were missing or inconsistent, and provide a context for interpreting the data after it was collected. It permitted us to maximize the set of goals for a particular data set and minimize the data required by recognizing where one metric could be substituted for another.

The process established the way we did measurement in the Software Engineering Laboratory at Goddard Space Flight Center, and has evolved over time, based upon use. Expansion involved the application to other areas of measurement (such as effort, schedule, process conformance), the development of the goal templates, the development of support processes, the formalization of the questions into models, and the embedding of measurement in an evolutionary feedback loop, the Quality Improvement Process and the Experience Factory Organization. Professor Dieter Rombach was a major contributor to this expansion.

The GQM paradigm represents a practical approach for bounding the measurement problem. It provides an organization with a great deal of flexibility, allowing it to focus its measurement program on its own particular needs and culture. It is based upon two basic assumptions (1) that a measurement program should not be ‘metrics-based’ but ‘goal-based’ and (2) that the definition of goals and measures need to be tailored to the individual organization. However, these assumptions make the process more difficult than just offering people a “collection of metrics” or a standard predefined set of goals and metrics. It requires that the organization make explicit its own goals and processes.

In this book, Rini van Solingen and Egon Berghout provide the reader with an excellent and comprehensive synthesis of the GQM concepts, packaged with the support necessary for building an effective measurement program. It provides more than the GQM, but describes it in the philosophy of the Quality Improvement Paradigm and the Experience Factory Organization. Based upon experience, they have organized the approach in a step-by-step set of procedures, offering experience-based heuristics that I recognize as effective. They have captured the best ideas and offer them in a straightforward manner. In reading this

book, I found myself constantly nodding in agreement, finding many ideas I had not articulated as well. They offer several examples that can be used as templates for those who wish to have a standard set of goals and metrics as an initial iteration.

If you work on a measurement program, you should keep this book with you as the definitive reference for ideas and procedures.

Professor Victor R. Basili
University of Maryland
and
Fraunhofer Center for Experimental Software Engineering, Maryland

'Its not enough to do your best;
you must know what to do, and then do your best'
W. Edwards Demming

1 Introduction

1.1 Objectives

In the past few years we gained many experiences of organising software quality improvement programmes in industry. Although there are many textbooks available on quality improvement, we were surprised by the gap between theory and practice. Most literature on software quality is quite comprehensive, however, it often lacks the goal-driven nature of business. Business is not just looking for ultimate quality, but for the best quality to be given to other goals, such as timeliness, product features, complexity, or cost.

The Goal/Question/Metric method (GQM) supports such a business driven quality improvement approach very well, however, this method is merely published in scientific journals. This motivated us to write a practical GQM guide. We hope this book will inspire you during your quality improvement work in practice, and sincerely hope the practical material in this book prevents you making some of the mistakes we did.

This book has been written to support people that are working on quality improvement in the area of software development. It will focus on GQM and will provide:

- motives to start goal-oriented measurement;
- detailed steps to take for GQM application;
- examples of possible support for GQM;
- our lessons learned to prevent others making the same mistakes;
- templates of the necessary deliverables for GQM application;
- results from practice regarding goals we actually pursued and attained;
- suggestions for feedback material in which data is presented from projects we worked on.

This book is intended for:

- project managers setting up measurements towards a project or product goal;
- quality assurance personnel aligning the quality measures with the goals of their company and business;
- software engineers that want to structure their personal metrics;
- consultants that support companies in their process improvement and measurement programmes;
- teachers that want to explain to their students how to practically apply software measurement;
- last but not least, any other people interested in working actively towards a certain measurable objective.

1.2 Organisational setting

Our experience with GQM application has been developed in cooperation with many other companies and people mainly by participating in (inter)national projects. Three important projects are described below together with links to detailed information.

ESSI/CEMP: Customised establishment of measurement programmes.

The ESSI/CEMP project aimed at evaluation of the GQM-approach in the industry. The ESSI/CEMP project consisted of three practical case studies that investigated the introduction of GQM-based measurement in industry. The goals of the ESSI/CEMP project were:

- to provide cost/benefit data from three industrial case studies performed within ESSI/CEMP;
- to develop a set of guidelines and heuristics for the introduction of GQM-based measurement.

The ESSI/CEMP internet home-page contains free publications and deliverables of the ESSI/CEMP project: http:/www.iese.fhg.de/Services/Projects/Public-Projects/Cemp.html.

PROFES: Product focused improvement of embedded software processes.

The PROFES project particularly focuses at organisations developing embedded software systems, in sectors such as telecommunications, medical systems, retailing systems and avionics. The PROFES methodology provides support to an integrated use of process assessment, product and process modelling, GQM measurement and experience factory. More information can be found on: http://www.ele.vtt.fi/profes/.

SPIRITS: Software process improvement in embedded IT environments.

SPIRITS was a research project of Schlumberger Retail Petroleum Systems (RPS) which was executed in cooperation with the Eindhoven University of Technology, The Netherlands. SPIRITS developed concepts and methods for process improvement to accomplish high and quantifiable reliability of embedded products. The main objectives were the design of:

- methods for process improvement in embedded systems development;
- methods for measurement and evaluation of the effectiveness of process improvement activities on the reliability of embedded products.

The practical application of the concepts and methods was validated in several case studies within Schlumberger RPS. The set of instruments was based on practical experiences and validated by a number of pilot projects.

1.3 Schlumberger/Tokheim

Schlumberger Retail Petroleum Systems (RPS) has been working for many years on developing and producing high quality products for petrol stations. As with most electronic products, the amount and importance of the software included in products is getting more and more important. Because of this trend, Schlumberger RPS extensively worked with

software quality initiatives to manage the quality of their software. This book provides an overview of the experiences gained during these initiatives. It was written, because particularly in the area of GQM, there was little documentation available. This book is intended as a guide for GQM application in practice.

This book is based on many quality initiatives within Schlumberger RPS over the past few years, such as:

- ISO9001 certification;
- TickIT certification;
- CMM assessments;
- Goal-oriented measurement by the GQM paradigm.

Initially, Schlumberger RPS worked with software measurement based on ISO procedures and CMM. However, this way of working was considered unsatisfactory. Not all collected data were actually used and other data were missing. In other words, the measurement programme lacked a clear goal and was experienced as being inefficient itself (Latum et al, 1998).

This observation led to the conclusion that software metrics should be defined in such a way that the collected data is relevant, and that it helps in achieving goals of business quality, such as product reliability which is not an isolated business goal. Because of this observation, the Goal/Question/Metrics paradigm was adopted to structure the measurements. During the several years of GQM application it appeared that there was little or no documentation on 'how' these measurement programmes should actually be carried out. At many events we met people from several industries or academics who all supported the power of the GQM paradigm, but none of them could point us to practical procedures, templates, or detailed examples from practice. Based on this notion we decided to capture our expertise in GQM application and write it down in this book.

September 1998, the RPS division of Schlumberger was sold to Tokheim, a company dedicated to fuel dispenser production. This sale created the largest system and service supplier in the retail petroleum market. Currently, Tokheim employs 5,000 people world wide, and has a yearly revenue of almost 1 billion US dollars.

1.4 Origin of GQM

The GQM method was originally developed by V. Basili and D. Weiss, and expanded with many other concepts by D. Rombach. GQM is a result of many years of practical experience and academic research. With this book we aim at contributing to their original work. Our contribution is the detailed analysis of the method and the addition of several techniques. An example of such an addition is cost/benefit analysis. Also included are many practical examples and suggestions to realise successful measurement programmes.

1.5 Outline

This book intends to be a practical guide to the GQM method. It, therefore, includes many examples, checklists and document templates. The book is divided into three parts. The more knowledge you already possess of GQM, the later you are advised to start in the book.

- Part 1: General description of software quality improvement and GQM measurement.
  This part provides a background on the theory behind this book, contains motives to apply GQM measurement in practice, and reports cost/benefit information from GQM application in practice.
- Part 2: GQM method stepwise.
  This part describes the phases of the GQM method and guidance to apply GQM in practice. This includes procedures, documents and checklists.
- Part 3: Cases on application of GQM's four practical measurement goals.
  This part describes four measurement goals from Schlumberger/Tokheim programmes. Many details are described such as, the results of the GQM phases, lessons learned, and measurement results presented in graphs and tables. Each case also contains the GQM documentation that might serve as an example whenever you would like to measure a similar measurement goal. This documentation includes: GQM plan. measurement plan, data collection forms and feedback session reports.

We hope that this book will support you during measurement programmes in practice. Our aim was to capture our own experience of GQM application, and to present it in such a way that it will help you as much as possible and prevent you from making some of the mistakes we made. We wish you good luck in applying the GQM method and are looking forward to your results and experiences.

PART 1

Software quality improvement and goal-oriented measurement

'It wasn't raining when Noah built the ark'
Howard Ruff

2 Software process improvement

2.1 Introduction

Today's software development is still error prone. For instance, projects are completed too late, exceed their budgets, or require substantially more resources than planned. Often developers are working in an unstructured and stressful way. Resulting in a poor or unknown level of product quality. These problems are often addressed as the 'software crisis' (Gibbs, 1994), and by applying various software process improvement (SPI) approaches, organisations developing software try to resolve this crisis.

In this chapter, first the main improvement areas of software development are described, being, software products, software processes and software quality itself. Subsequently, the most common improvement goals encountered in practice are discussed: decrease of costs or risks; shortening of project cycle time; and improvement of product quality. Finally, the main approaches to improve software processes are introduced.

2.2 Improvement areas of software development

In this section a brief general introduction is given to the current status of software products, software development processes and software quality.

2.2.1 Software products

A software product is defined as the complete set of computer programmes, procedures, and associated documentation and data, designated for delivery to a user (IEEE, 1994). Software products often struggle with quality problems. Due to the enormous size and complexity of many software products, software developers are often not capable of providing reliable information on the quality of their products. As a result, many high-tech software projects eventually turn out to be disastrous.

Furthermore, today's software should comply with many implicit demands from a variety of users. Even if the software functions correctly, the functionality should also be logical for the users, clearly documented and supported by training of the users on how to use the software.

Another problem for the software community is the fact that software is not a static product, but a product that needs to be adapted all the time because of a changing environment. Flexibility is, therefore, considered to be one of the most important strengths of software. This flexibility is most apparent during the maintenance of software: as the requirements of a software product often change over time, the product should be easily maintainable. Flexibility is, however, also the largest weakness of software, as this flexibility makes it possible to change a product so completely that it's original architecture is put under a lot of pressure.

Finally, software products tend to become larger and more complex every day: the size of software in mobile phones, for example, increases 10 times every 1000 days (Karjalainen et al, 1996). Philips has noted that 70% of product development is spent nowadays on software development and has become the critical path in product development (Rooijmans et al, 1996). As a result of all these effects the quality of the product will often be at stake.

2.2.2 Software processes

Two major phases can be discerned in the software product life-cycle: the development phase; during which the product is initially created, and the exploitation phase; during which the product is used in practice and changes to the product are implemented in order to maintain the software. This book addresses the development of software only. This does not imply that the book is unusable for maintenance activities, however, it will not particularly address this phase. Software processes, therefore, refers to the development phase of a software product.

A software development process is defined as all activities necessary to translate user needs into a software product (based on ISO9000-3, 1991 and IEEE, 1994). Pfleeger states that such a process consists of a Requirements Analysis and Definition phase, a System

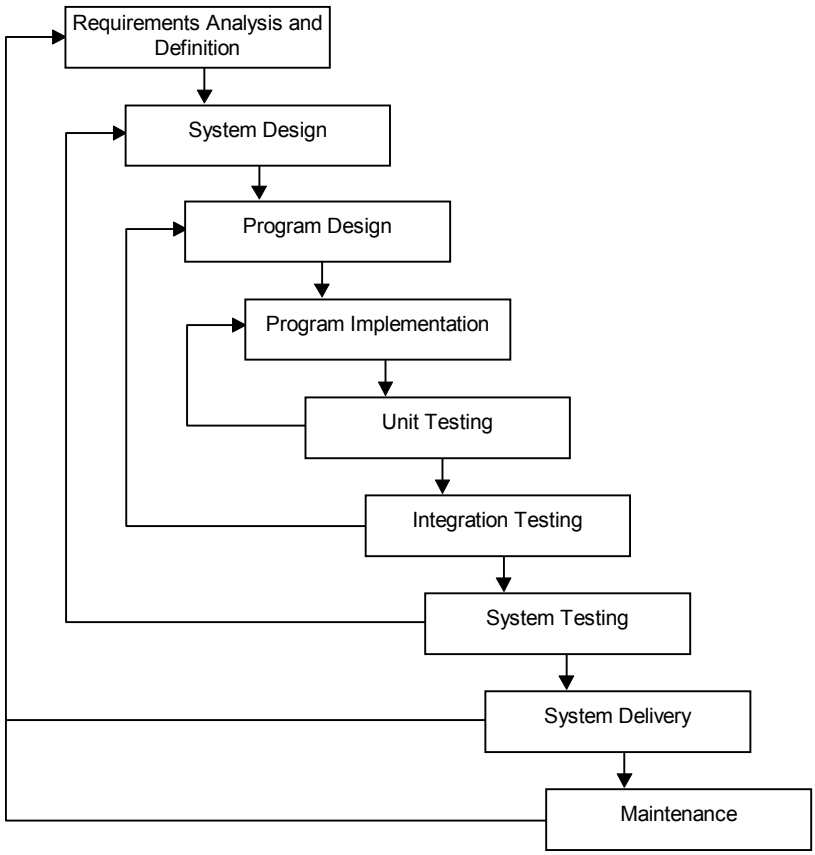


Figure 2-1: Phases within a software development process (Pfleeger, 1991).

Design phase, a Program Design phase, a Program Implementation phase, a Unit Testing phase, an Integration Testing phase, a System Testing phase, a System Delivery phase, and finally, a Maintenance phase (Pfleeger, 1991). This is illustrated in Figure 2-1.

Software processes are considered to be the main area for quality improvement, because they contain the activities during which the product is actually created. The earlier quality problems are detected, the easier and cheaper they can be resolved. Three typical problems exist regarding software processes:

- software processes are often not defined;
- software development is highly dependent on individual craftsmanship;
- Software development is difficult to manage.

Methods like the CMM focus on making software development manageable as a first step to continuous improvement (Paulk et al, 1993).

Software processes are often not well-defined. Although many methods are available, few are actually used in practice. Due to this lack of clearly defined processes, software development depends to a large extent on the individual skills and craftsmanship of the developers. Software development is still a very creative process (Glass, 1995), and individual developers maintain a significant influence on the end result (Bemelmans, 1991).

The Software Engineering Institute (SEI) has defined a five level model to describe the 'maturity' of the way in which the organisation addresses the importance of people: the 'People-CMM (P-CMM)' (Curtis, 1995). Humphrey, also the originator of CMM, has described an approach for the individual improvement of software engineers: the Personal Software Process (PSP) (Humphrey, 1995).

The dependence on individuals makes software development difficult to manage. It is often problematic to switch tasks between developers. Due to unclear or undocumented development processes it will be problematic to predict delivery times. A lot of research is still going on to enable organisations to improve their software development process, and, for instance, move to higher CMM levels (Humphrey, 1989).

2.2.3 Software quality

Software quality is defined as all characteristics of a product that bear on its ability to satisfy explicit and implicit needs of the user (ISO9126, 1991). An example of such characteristics is shown in Figure 2-2. Other subdivisions of quality in attributes have been described by, for example, Boehm, McCall and Cavano (Boehm, 1978; McCall et al, 1977; Cavano et al, 1978).

Quality is an important aspect of attracting customers. However, it is certainly not the only aspect. Other examples of important product characteristics would be price and delivery date. Also, the perception of quality will be different for many individuals. The software market itself is a typical example of a market where you may have a very successful product that is of poor quality, due to for example, missing functionality or software failures.

Several problems exist regarding software quality that are all related to two basic problems:

- It is difficult to specify software quality in measurable terms.
- It is difficult to select the most efficient and effective development process that produces a product with the specified level of quality.

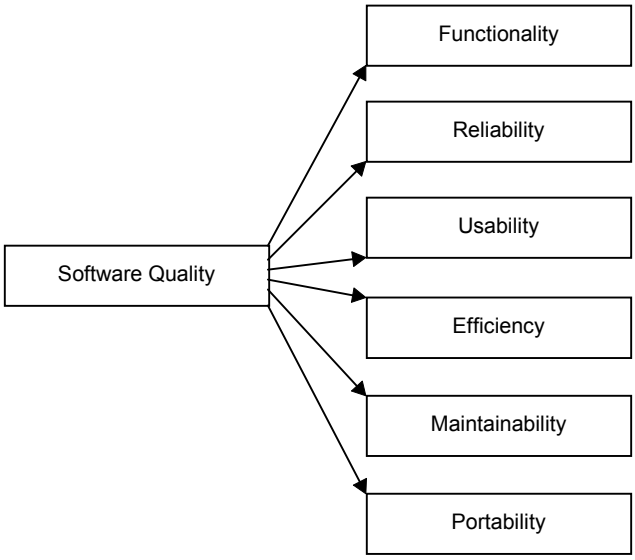


Figure 2-2: ISO 9126 software quality characteristics (ISO9126, 1992).

These two basic problems are also related to each other. Quality involves many aspects and these aspects often require subjective judgments. An example of an aspect that is often subjectively measured is: *usability*. An aspect such as *reliability* is probably more suitable for objective measurement. *Reliability* can, for example, be measured by means of particular failure rates. Even though a characteristic like reliability is also interpreted differently by different users.

In practice most quality requirements for software are not made explicit during the definition phase of software development. In such a case, it will be quite difficult for developers to create quality. Most development methods only specify functional requirements, leaving the fulfilling of other quality characteristics to the experience, craftsmanship and personal interest of the developers.

Even if quality is adequately specified, software quality is still difficult to build. Many software engineering methods, techniques, and tools, are used in the software industry, yet their effectiveness remains mostly unknown. As a result, it is very difficult to select the most appropriate development process. Furthermore, during the development process, insufficient measurement is often applied in order to create the possibility to take corrective action.

The quality of today's complex software systems should be achieved by combining experience and knowledge of many different disciplines. Integrating disciplines means cooperation of experts from particular disciplines, probably causing communication problems. Human failures then become more likely and can easily lead to lower product quality.

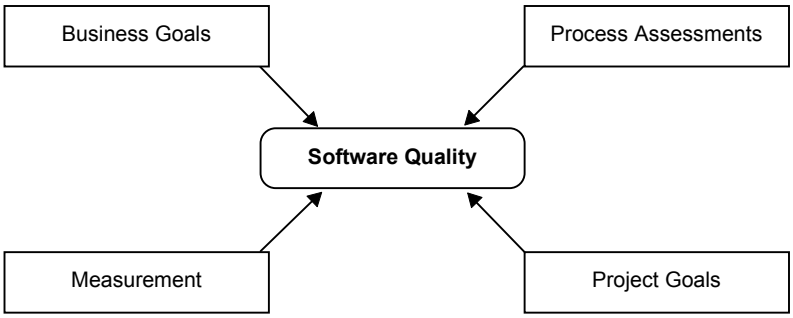


Figure 2-3: Identification of improvement goals.

2.3 Software process improvement goals

To improve their software development, organisations need a definition of clear improvement goals, otherwise the improvement activities will turn out to be as chaotic as the development process itself. These improvement goals should support business objectives in the best possible way. For example, it is not recommended to base improvements on a method that prescribes the installation of a software configuration management system, while most projects in the organisation fail because of bad requirements management.

Improvement goals can be defined in different ways. First of all, corporate management can prescribe quality objectives, from which improvement goals can be derived. Secondly, organisations can conduct software process assessments, in order to identify main areas suitable for improvement. Based on the identification of such areas, improvement goals can again be defined.

Refining high level business goals into specific goals for software development will always be difficult. It is recommended to use the Software Engineering Institutes (SEI) *Goal-Driven Software Measurement: A Guidebook*, for such a purpose. This report is publicly available and can be downloaded from the SEI home page on the internet.

Measurement also provides means to identify improvement goals. By applying measurement to a specific part of the process, problems within the process can be identified on which improvement goals can be defined. Finally, improvement goals may be defined with respect to particular project needs.

The next paragraphs describe four main areas to which software process improvement activities usually focus: the decrease of project costs, the decrease of project risk, the shortening of project cycle time, and the increase of product quality.

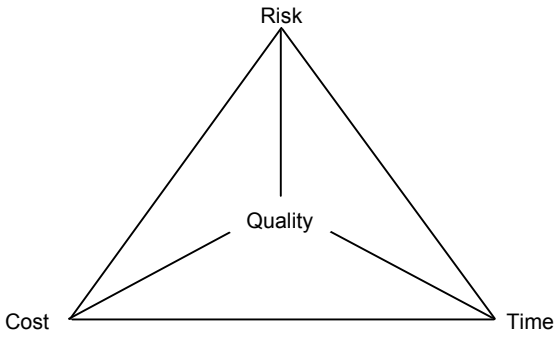


Figure 2-4: Process improvement focus.

2.3.1 Increase quality

If an organisation intends to improve the quality of its products, improvement goals in the area of product quality attributes should be defined, and thus the need for clearly specified requirements on functionality, reliability or other quality attributes arises.

By focusing on quality increase as an improvement goal, the software development process will eventually need to be fully defined, as having a quality improvement goal without an established development process seems unfeasible. The responsibility for the product quality lies with the people that create the product. By adopting quality improvement goals in an organisation, developers become aware of the need for high quality. Therefore, software process improvement supports the creation of a 'quality culture' within organisations very well.

When introducing quality increasing improvements in an organisation it is possible that project teams will ask whether that is really necessary. They might state that they already develop high quality products. Therefore, make sure you can show that a quality increase is indeed necessary, probably not because it is bad, but because business almost always looks for higher quality. And, it is always possible to improve.

Improvement goals in the area of quality often start with some kind of software defect measurement or detection topic. Both product and process failures will then be registered in order to identify the areas in the development process that have the highest need for improvement. Also initial defect detection improvements will be the implementation of Fagan inspection or other review techniques. However, more subjective approaches on quality improvement, such as focusing on customer satisfaction or keeping documentation up-to-date, can also be defined within a quality improvement goal.

2.3.2 Shorten project cycle time

Shortening time-to-market is an improvement goal which is frequently heard of in practice. Marketing and competition requirements make market introduction one of the most stringent requirements for product development. Especially in the embedded product area where cycle-time is currently most relevant. The project life-cycle or development time can be decreased by, for example, increasing productivity, parallel development, prototyping or reuse of already existing software. Productivity can be increased through, for example,

specialisation of software design. Parallel development will be encouraged through a modular design of the software.

Cycle-time reduction is often one of the reasons to define an improvement goal with respect to developer productivity. However, one should be careful choosing this approach. Measuring individual performance is difficult and if not fully supported by the software developers, it might very well destroy the improvement initiative.

Software process improvement is an excellent method to evaluate whether new tools or techniques that are pushed into the market are really decreasing the development cycle. Many 'silver bullet' stories are available on successful applications in organisations, though they never seem to fulfil all promises. We recommend to guide any method or tool introduction with measurement in order to evaluate whether such a change is actually an improvement.

2.3.3 Decrease costs

Decreasing the costs of software development is primarily realised by decreasing the software development labour, and therefore cost reduction will mostly aim at executing software development processes more efficiently. A first step to this is identifying current effort and expenditure. Examples from practice in which costs are expressed are costs per source line, costs per function point, costs per life-cycle phase, or costs per product (sub-system). Software measurement is an excellent tool to start improving on this type of data. The focus will not only be on 'doing things better' but also on 'doing the right-things', because eliminating unnecessary activities will be one of the most interesting cost cutting activities.

Embedded software products, however, should be examined more carefully with respect to costs: the cost of the product is not only related to the development cost of the software and to the development of the accompanying hardware. Often the product cost of an embedded system is largely determined by hardware, especially because eventually the production cost of hardware is higher than software, as production costs of software are negligible. Development of software is expensive and production is almost free (copy *.*).

Decrease of cost can also be established by reusing hardware designs, software components or documentation. Finally, an obvious reason to focus on cost reduction in an improvement programme is the fact that corporate management will often be interested in financial figures.

2.3.4 Decrease risks

In order to decrease risks that are involved in executing a project, project management has to be able to manage risk factors that are relevant for specific projects. This can be accomplished by identifying possible risk areas and applying measurements to those particular areas in order to track status and identify the need for corrective actions (Heemstra et al, 1998).

By increasing process maturity, the risks involved in executing the relevant process will decrease, because problem areas in the development process will be tackled by the software process improvement activities. Furthermore, explicit risk reduction makes projects more manageable, and is therefore a suitable and obvious way of improving software processes.

2.4 Approaches to software process improvement

Currently, many software development practitioners and researchers are involved in Software Process Improvement (SPI). Several improvement models, methods and techniques are available, divided over two major streams.

- Top-down approaches, such as CMM (Paulk et al, 1993), SPICE (SPICE, 1997) and BOOTSTRAP (Kuvaja, 1994). These approaches are mainly based on assessments and benchmarking.
- Bottom-up approaches, such as GQM (Basili et al, 1994a), QIP (Basili et al, 1994b) and AMI (Pulford et al, 1995), which mainly apply measurement as their basic guide for improvement.

The top-down improvement stream applies a normative model that is assumed to be the best way of developing software. By assessing an organisation, using this model, it becomes possible to identify the 'maturity' of that organisation, and propose relevant improvements (Humphrey, 1989). The bottom-up improvement stream measures software development to increase understanding within a specific context. Both streams are successfully applied in practice.

2.4.1 Assessment

A popular top-down approach to software process improvement that is based on assessments, is the Capability Maturity Model (CMM) of Figure 2-5 (Paulk et al, 1993). The CMM helps organisations improve the maturity of their software processes through an evolutionary path from ad hoc and chaotic to mature and disciplined. A low level of maturity incorporates a high level of risk in performing a process. As organisations become more capable, risks decrease and productivity and quality are expected to increase.

Each maturity level progressively adds further enhancements that software organisations typically master as they improve. Because of its progressive nature, the CMM can be used to determine the most important areas for immediate improvement. For this purpose, the

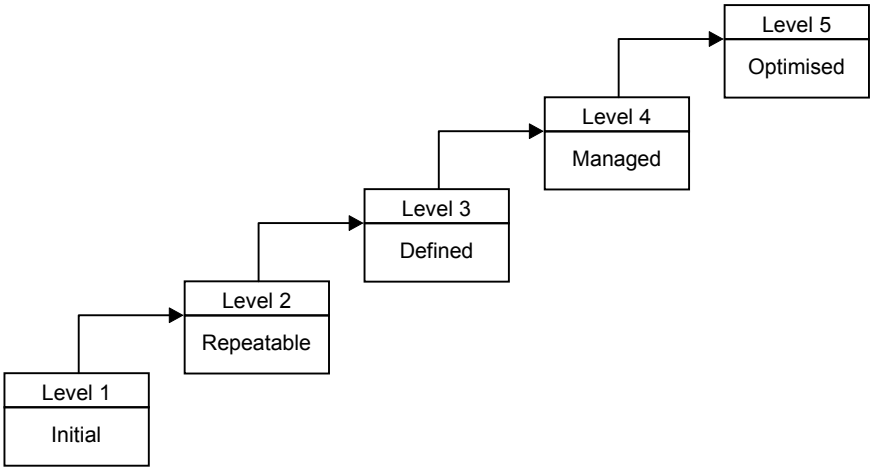


Figure 2-5: CMM maturity levels (Paulk et al, 1993).

CMM provides an assessment method to objectively and consistently assess the capability of software organisations and place them on one of CMM's five maturity levels. A software process assessment is a review of a software organisation to provide a clear and factual understanding of the organisation's state of software practice, to identify its major problems, and to initiate actions to make these improvements (based on Humphrey, 1989). After executing these actions, organisations can then return to re-assessing the established process, as process improvement is a continuous cycle.

Another well-known top-down approach to process quality is the ISO9000 approach. The theory behind the ISO9000 standards is that a well-managed organisation with a defined engineering process is more likely to produce products that consistently meet the purchaser's requirements, within schedule and budget, than a poorly managed organisation that lacks an engineering process. Within the ISO9000 family, ISO9001 is the most complete set of quality system requirements and consists of twenty clauses that represent requirements for quality assurance in design, development, production, installation and servicing, that define which aspects of a quality system have to be available within an organisation. ISO9001 does not, however, provide details on how those aspects should be implemented and institutionalised. ISO9000-3 provides guidelines for applying ISO9001 to the specification, development, supply and maintenance of software (ISO9000-3, 1991).

The ISO9000 family has the following structure (ISO9001, 1994; ISO9000-3, 1991, Looijen, 1998):

- ISO9000 Quality management and quality assurance standards, consisting of:
  - ISO9000-1 Guidelines for selection and use;
  - ISO9000-2 Generic guidelines for application of ISO9001, ISO9002 and ISO9003;
  - ISO9000-3 Guideline to apply ISO9001 to software development and maintenance.
- ISO9001 Quality systems: a model for quality assurance in development, production, installation and servicing.
- ISO9002 Quality systems: a model for quality assurance in production, installation and servicing.
- ISO9003 Quality systems: a model for quality assurance in final inspections and testing.
- ISO9004 Quality management and quality system elements guideline.
- ISO9004-2 Quality management and quality system elements part 2: guideline for services.

Receiving ISO9000 certification from an independent certification body enables purchasers of software products and services to demonstrate engineering and management capability. ISO9000 certification assures customers that an audited company has all its processes and work instructions documented to conform to the ISO9000 requirements, and that these processes and work instructions are being followed on a continuous basis. Frequent follow-up audits by the certification body will ensure that the certified company continues to comply with the prescribed quality requirements. However, ISO9000 certification does not give any guarantee for product quality, nor that the described process is actually executed. Even though some negative sounds are heard on ISO certified quality systems, many companies still claim success by ISO certification.

Many other top-down approaches have been developed during the past few years, most of which are based on the principles of CMM and ISO9000. Examples of such methods are

BOOTSTRAP, SPICE, and Trilium. The Bootstrap method (Kuvaja et al, 1994), which is the result of a European Esprit project, provides an alternative for organisations that are interested in improving their software development process and in attaining ISO9001 certification. It combines and enhances the methods provided by the CMM and the ISO9000 quality standards. ISO12207 describes a software product life cycle by defining a number of activities that should be performed in any organisation (ISO12207, 1995). ISO15504 adds metrics to identify how well practices are performed (Spice, 97).

The basis of the Bootstrap methodology is similar to CMM. Like the CMM, an assessment is based on five maturity levels, but the Bootstrap method uses another scale to measure an organisations' or projects' overall strengths and weaknesses. The ISO9000 quality standards (ISO9001 and ISO9000-3) are incorporated in the methodology because they provide guidelines for a company-wide quality system. The CMM does not include such guidelines. Furthermore, many European companies use ISO9000 as a primary quality standard. Bootstrap can be used by organisations to determine readiness for ISO9001 certification.

2.4.2 Measurement

The other stream contains the bottom-up approaches, which are based on applying measurement on the current software practices within an organisation. Bottom-up approaches are based on careful analysis of applied software practices, on selection of improvement goals derived from these analyses, and on management of improvement activities supported by measurement. Examples of such approaches are the quality improvement paradigm (QIP) as a framework for process improvement, and the Goal/Question/Metric paradigm to support measurement on the goals that were defined in QIP. These two paradigms will be discussed in detail in the next chapter that deals with goal-oriented measurement.

2.5 Conclusions

Many approaches are available to improve software processes, in order to solve problems that occur during software development. The two main streams are *assessment based* and *measurement based*. In practice, both streams are rarely applied together, although they complement each other very well. It is recommended that assessment approaches are used to create a first overview on the status of a software organisation.

Based on the results of this assessment, improvement goals should be identified that suit the organisation in the best possible way. Guidance of the improvement activities toward these goals should be done by applying some form of measurement to these activities, since measurement provides an overview and the opportunity to evaluate whether implemented changes are actual 'improvements'. Because measurement supports the execution of improvement activities towards clearly defined goals, a so-called 'goal-oriented measurement' method should be selected to implement measurement. The subject of goal-oriented measurement is described in the next chapter.

For embedded software development an improvement methodology is developed that combines both assessments and goal-oriented measurement. This method is made in the PROFES project. More information can be obtained from the internet address: http://www.ele.vtt.fi/profes/.

2.6 Questions and assignments

2.6.1 Questions

1. Which are the six ISO 9126 characteristics of software product quality?
2. Which improvement goals typically exist for software development?
3. From which sources can software process improvement (SPI) goals be derived?
4. Which are the five levels of the Capability Maturity Model (CMM)?
5. What is the difference between *development cost* and *production cost* of software?
6. Which two particular approaches are discerned for SPI approaches? What are their main differences?
7. Consider the statement that software quality improvement should always be based on a particular business goal, such as market share, or increasing profit margins. Do you agree with this statement and if so, do you find adequate support in the particular approaches to operationalise this linkage?

2.6.2 Assignments

A small company develops a very innovative mobile phone. It is expected to outclass all existing phones by providing revolutionary special features. The competition is working on such a product as well. However, the company has a lead of at least one calendar year. The product is developed within a team of 20 highly creative people, of which 15 have less than 2 years experience in software development. You are assigned as SPI co-ordinator.

1. Prioritise the four improvement focuses in Figure 2-4.
2. Describe four possible SPI goals for the two most important improvement focuses.

'Projects without clear goals will not achieve their goal clearly'
Tom Gilb

3 Goal-oriented software measurement

3.1 Introduction

In this chapter goal-oriented measurement is described. Its relation with general quality improvement initiatives is explained and the GQM-method is positioned. The GQM-method itself will be elaborated upon in Chapter 5 and beyond.

3.2 What is software measurement?

In general, 'measurement' is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules (Fenton and Pfleeger, 1996). The numerical outcome is called a 'measurement'. This can be applied to both a software development process and a software product.

Software measurement is the continuous process of defining, collecting, and analysing data on the software development process and its products in order to understand and control the process and its products, and to supply meaningful information to improve that process and its products.

3.3 Software metrics

Measurement on a software development process and its products is performed by applying particular software 'metrics'. Measuring will normally comprise several metrics, again resulting in several measurements per metric.

A lot of categorisations and examples of metrics can be found in literature, some examples are (Pfleeger, 1991; Fenton and Pfleeger, 1996; Grady, 1992):

- product and process metrics;
- objective and subjective metrics;
- direct and indirect metrics;
- explicit and derived metrics;
- absolute and relative metrics;
- dynamic and static metrics;
- predictive and explanatory metrics.

The most common types of metrics are described below.

Product and process metrics

First of all, measurement is applied to a process and/or a product, which results in process and/or product metrics.

A *product metric* is a measurement of an intermediate or final product of software development, and therefore addresses the output of a software development activity. Examples of such metrics are a size metric for the number of requirements, a complexity metric for the software code, etc.

Process metrics measure the characteristics of the overall development process, such as the number of defects found throughout the process during different kinds of reviews, etc.

Objective and subjective metrics

Objective metrics are absolute measures taken of the process or product, and count attributes or characteristics in an objective way (Humphrey, 1989), such as number of lines of code, number of faults discovered. These metrics have a fundamental starting point, a natural zero.

Subjective metrics are measurements of a process or product that involve human, subjective judgement. Examples of subjective metrics are expected complexity and degree of conformance to coding standards. These measurements are classifications of observations.

Direct and indirect metrics

A *direct metric* is a measurement of a process or product characteristic that does not depend on the measurement of any other characteristic. Examples are the number of faults in a product, number of hours spent during certain process, etc. An *indirect metric*, on the other hand, is a measurement of a process or product characteristic that involves the measurement of one or more other characteristics, such as productivity, fault density, etc. An indirect metric always contains a calculation of at least two other metrics.

3.4 Software measurement goals

Measurement provides a valuable tool for understanding the effects of actions that are implemented to improve a software development process. Examples of results are (Möller and Paulisch, 1993; Pfleeger, 1991):

- increased understanding of the software development process;
- increased control of the software development process;
- increased capacity to improve the software development process;
- more accurate estimates of software project costs and schedule;
- more objective evaluations of changes in technique, tool, or methods;
- more accurate estimates of the effects of changes on project cost and schedule;
- decreased development costs due to increased productivity and efficiency;
- decrease of project cycle time due to increased productivity and efficiency;
- improved customer satisfaction and confidence due to higher product quality.

Software measurement data is interpreted by people to provide information that can be applied for three different purposes (Figure 3-1). In the first place, the data provide visibility of the current development process and the characteristics of the software products. This visibility is required to reduce complexity and increase *understanding* of the process and products. Understanding means determining the different variables that exist during execution of a process. Once basic understanding has been established (the variables

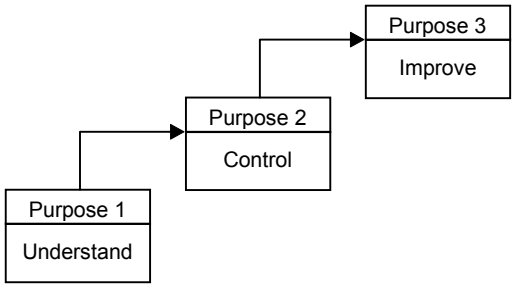


Figure 3-1: Applying measurement for three purposes.

are known), the collected and analysed data can be used to *control* the process and the products, by defining corrective and preventive actions. This means that the relationships between the process variables have to be determined. Once these relationships are known, they can be used to control the process. Furthermore, based on analysis, the collected measurement data can be used to assess the process, and therefore, act as an indicator of development process problem areas, from which *improvement* actions can be identified. Improvements can be made by influencing or changing process variables and their relationships. This hierarchy of measurement is illustrated in figure 3-1.

3.5 Management commitment

Attainment of business goals will be a result of several activities. Measurement by itself will of course not result in attainment of business goals. Specific actions are needed in an organisation to give such results. However, successful implementation of a measurement programme will facilitate continuous improvement and makes benefits more visible. One important prerequisite for successful implementation of a measurement programme is the level of support provided by the management of the company. Many of the positive effects are based on a positive change of attitudes in a project team towards quality improvement. Management should, therefore, actively support a measurement programme to influence these corporate culture changes. Business goals should be clearly defined and the measurement programme goals should reflect and support those business goals.

In order to facilitate a successful measurement programme, management needs to be sufficiently involved to support the entire initiative and operationalise business goals. However, at the same time they should also keep a certain distance to leave some privacy and encourage the personal responsibility of the software developers. Talking about quality, is also talking about mistakes and problems.

Implemented successfully, the measurement programme and the related software development process improvement activities will, in time, become an integral part of the software project management culture, and metrics will become an ongoing practice adopted and used by all project personnel (Möller and Paulisch, 1993).

3.6 Concepts of goal-oriented software measurement

Software measurement should only be performed towards an explicitly stated purpose. This so-called ‘goal-oriented measurement’ is especially used for improvement programmes as described in this book. The method applied in this book, is the GQM method (Basili and

Weiss, 1984). The following sub-sections introduce the basic phases and concepts of the GQM method. In Part 2 of this book, the GQM method will be described in detail.

The GQM method contains four phases:

1. The Planning phase, during which a project for measurement application is selected, defined, characterised, and planned, resulting in a project plan.
2. The Definition phase, during which the measurement programme is defined (goal, questions, metrics, and hypotheses are defined) and documented.
3. The Data Collection phase, during which actual data collection takes place, resulting in collected data.
4. The Interpretation phase, during which collected data is processed with respect to the defined metrics into *measurement* results, that provide *answers* to the defined questions, after which *goal attainment* can be evaluated.

The four phases of the GQM method are illustrated in Figure 3-2. The planning phase is performed to fulfil all basic requirements to make a GQM measurement programme a success, including training, management involvement and project planning. During the definition phase all GQM deliverables are developed, mainly based on structured interviews or other knowledge acquisition techniques. The definition phase identifies a goal, all questions, related metrics and expectations (hypotheses) of the measurements. When all definition activities are completed, actual measurement can start. During this data collection phase the data collection forms are defined, filled-in and stored in a measurement database. Then the 'real work' can start: using the measurement data. During the interpretation phase, the measurements are used to answer the stated questions, and these answers are again used to see whether the stated goals have been attained.

Activities such as packaging measurement results to be used in other parts of the organisation are not considered to be part of the GQM method, but of a wider perspective, such as a company wide improvement programme.

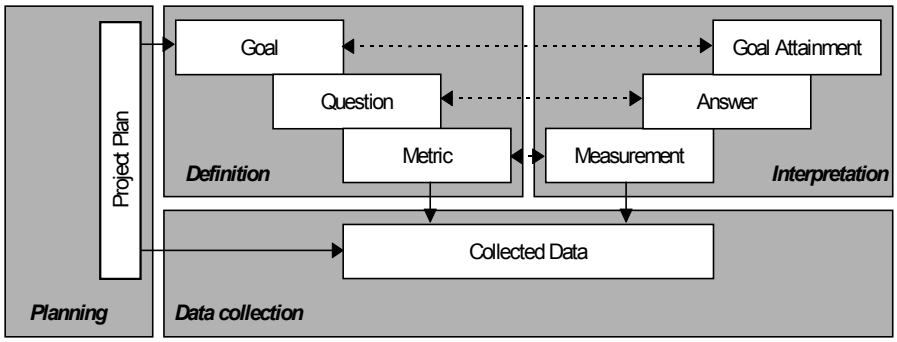


Figure 3-2: The four phases of the Goal/Question/Metric method.

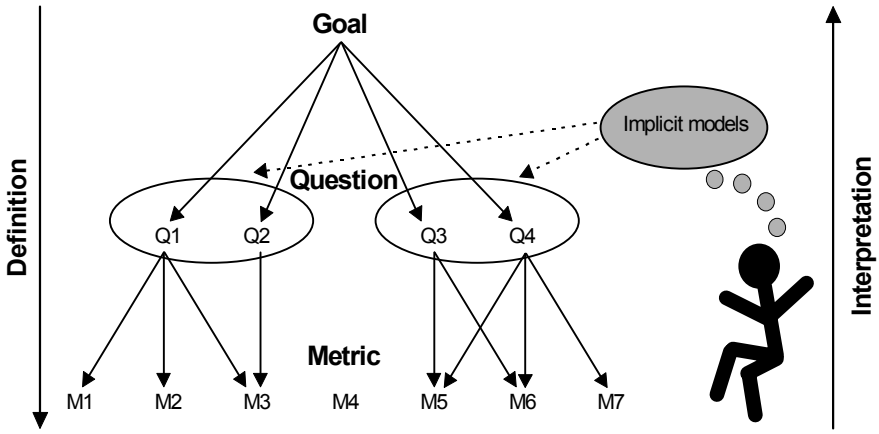


Figure 3-3: The GQM Paradigm (Basili and Weiss, 1984).

3.6.1 The measurement concept: the GQM paradigm

GQM represents a systematic approach for tailoring and integrating goals to models of the software processes, products and quality perspectives of interest, based upon the specific needs of the project and the organisation (Basili et al, 1994a). The result of the application of the GQM method is the specification of a measurement programme targeting a particular set of issues and a set of rules for the interpretation of the measurement data.

The principle behind the GQM method is that measurement should be goal-oriented. Therefore, in order to improve a process, organisations have to define their measurement goals based upon corporate goals and transform these goals into activities that can be measured during the execution of the project.

GQM defines a certain goal, refines this goal into questions, and defines metrics that should provide the information to answer these questions. By answering the questions, the measured data defines the goals operationally, and can be analysed to identify whether or not the goals are attained. Thus, GQM defines metrics from a top-down perspective and analyses and interprets the measurement data bottom-up, as shown in Figure 3-3.

The GQM model starts top-down with the definition of an explicit measurement goal. This goal is refined into several questions that break down the issue into its major components. Each question is then refined into metrics that should provide information to answer those questions. Measurement data is interpreted bottom-up. As the metrics were defined with an explicit goal in mind, the information provided by the metrics should be interpreted and analysed with respect to this goal, to conclude whether or not it is attained.

GQM trees of goals, questions and metrics should be built on knowledge of the experts in the organisation: the developers. Therefore, knowledge acquisition techniques are also applied to capture the implicit models of the developers built during years of experience. Those implicit models give valuable input into the measurement programme and will often be more important than the available explicit process models.

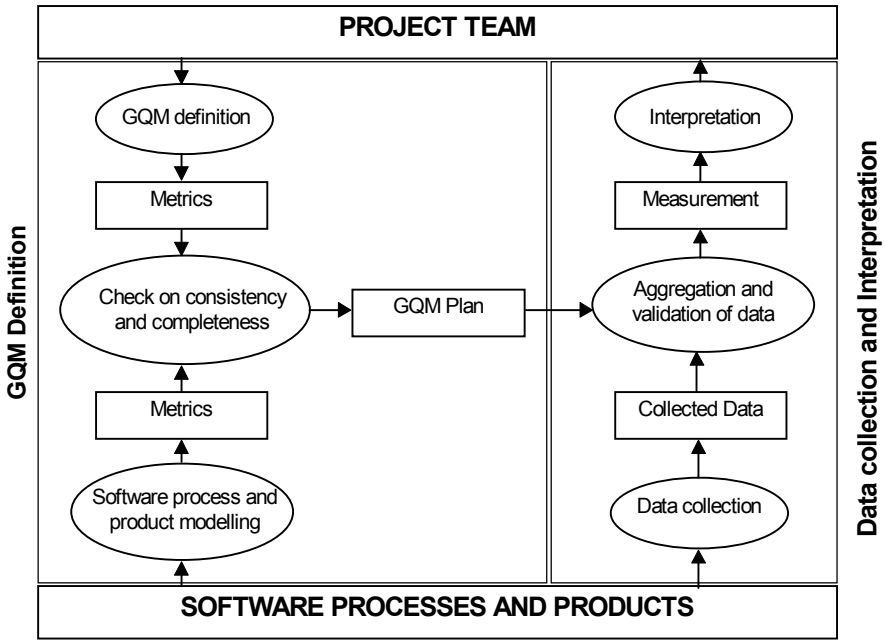


Figure 3-4: Metrics modelling from two perspectives (Solingen et al, 1995).

Over the years, GQM has evolved to include models of software processes and products, resulting in a model-based GQM approach (Solingen et al, 1995), that defines metrics from two different perspectives, as shown in Figure 3-4:

- Metrics definition by members of the project team, using GQM techniques.
- Metrics definition based on models of software processes and products.

By modelling both perspectives, two sets of metrics are identified that can be mutually checked for consistency and completeness. This will identify subjects that are missing, or badly defined. After the models of these two perspectives have been checked and enhanced, a GQM plan is developed. The GQM plan is the documented description of all the information that the measurement programme is based on. This document represents the measurement goals, related questions and identified metrics. Based on the GQM plan, a measurement plan is developed that defines procedures for collecting data.

When the plans are approved, measurement can start. Data are collected on the software development process and products, aggregated and validated. Finally, the measurement results are returned to the project members for analysis, interpretation and evaluation on the basis of the GQM plan (Solingen, 1995).

Figure 3-4 illustrates that GQM modelling can be checked for consistency and completeness on the basis of software process and product models. The following activities are required:

1. Check on the presence of all GQM based direct metrics in the software development process model.
2. Adjust the software development process model, adding the missing direct metrics.

3. Check the GQM definition on missing metrics that are defined in the software development process model and identify their relevance.
4. Adjust GQM definition, adding the missing direct (or indirect) metrics.
5. Decide on accepting the set of direct metrics.

The consistency and completeness check between the software development process models and GQM means that all metrics defined in a measurement programme, also need to be defined in the software development process model. If a certain direct metric is not defined in the software development process model, however, as required in your GQM model, the software development process model should be enhanced, adding the specific metric. In this way GQM also supports improving your software development process models.

3.6.2 The improvement concept: QIP

The quality improvement paradigm (QIP) is a quality approach that emphasises continuous improvement by learning from experience both in a project and in an organisation (Basili et al, 1994a). This learning from experience is built on experimentation and application of measurement. Each new development project is regarded as an experiment and available results of every foregoing and ongoing experiments should be packaged and reused, too. This reuse is applicable to experience on, for example, processes, products, problems, methods and resources. All the available information should therefore be packaged and reused on both project and corporate level to improve current and future performance.

QIP identifies a six step cyclic approach, which is based on the plan-do-check-act approach of Shewart/Demming (Basili et al, 1994). The QIP cycle consists of the following six steps (Figure 3-5):

1. *Characterise*. Understand the environment based upon available models, data, intuition, etc. Establish baselines with the existing business processes in the organisation and characterise their criticality.
2. *Set Goals*. On the basis of the initial characterisation and the capabilities that have a

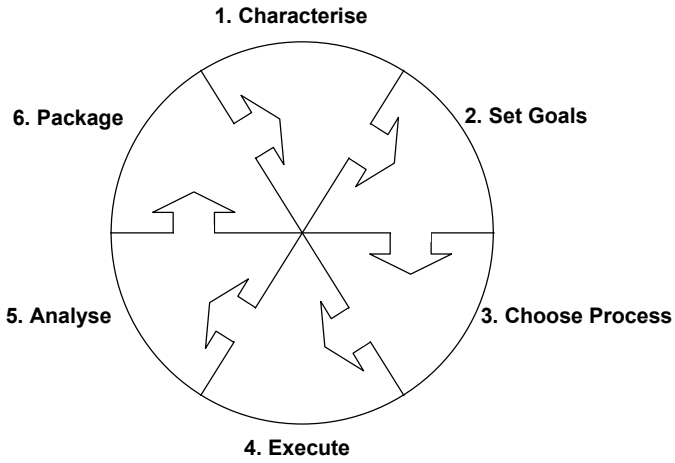


Figure 3-5: QIP cyclic approach (Basili et al, 1994a).

strategic relevance to the organisation, set quantifiable goals for successful project and organisation performance and improvement. The reasonable expectations are defined based upon the baseline provided by the characterisation step.

3. *Choose Process*. On the basis of the characterisation of the environment and of the goals that have been set, choose the appropriate processes for improvement and supporting methods and tools, making sure that they are consistent with the goals that have been set.
4. *Execute*. Perform the processes constructing the products and providing project feedback upon the data on goal achievement that are being collected.
5. *Analyse*. At the end of each specific project, analyse the data and the information gathered to evaluate the current practices, determine problems, record findings, and make recommendations for future project improvements.
6. *Package*. Consolidate the experience gained in the form of new, or updated and refined models and other forms of structured knowledge gained from this and prior projects and save it in an experience base to be reused on future projects.

Once the experience has been packaged during the last step, a new cycle is initiated toward further improvements and additional experience and knowledge. This emphasises the continuous character of QIP. The quality improvement paradigm implements two feedback cycles:

- *Project feedback cycle*. This control cycle provides the project with feedback during project execution. Resources should be used as effectively and efficiently as possible by the project and quantitative information is used to prevent and correct problems.
- *Corporate feedback cycle.* The corporate feedback cycle provides knowledge to the complete organisation, by comparing the project data with the nominal ranges in the organisation and analysing concordance and discrepancy. This experience is accumulated and, based on analysis, can be reused to improve performance of future projects.

Within QIP, software measurement is considered an indispensable component to capture experiences and retrieve knowledge on development activities. To collect the information on software development in an organisation, software measurement should be integrated in the software development process. A feedback process should exist in order to establish a continuous learning process in the organisation.

QIP suggests the GQM method as the measurement mechanism to define and evaluate the improvement goals of QIP. The organisational concept for such improvement programmes is described in the next section.

3.6.3 The organisational concept: experience factory

An experience factory consists of the *GQM definition* and *GQM analysis team* and is defined as: 'The experience factory is a logical and/or physical organisation that supports project developments by analysing and synthesising all kinds of experience, acting as a repository for such experience, and supplying that experience to various projects on demand' (Basili et al, 1994b). It packages experience by building informal, formal or schematised and productised models of measures of various software processes, products, and other forms of knowledge via people, documents, and automated support'.

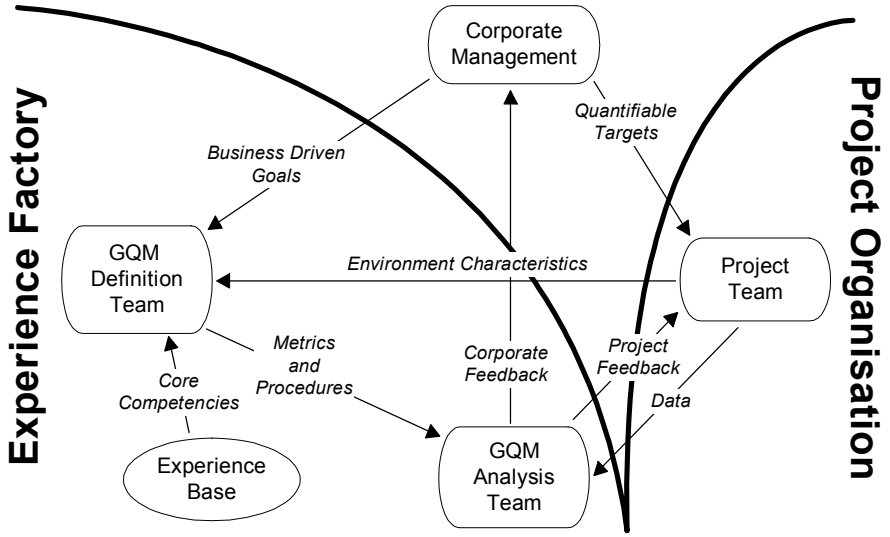


Figure 3-6: The Experience Factory (Basili et al, 1994b).

The stakeholders that are distinguished in a measurement programme are corporate management, GQM definition team, GQM analysis team and project team (Basili et al, 1994b). The experience factory is illustrated in Figure 3-6.

Corporate management provides a project team with project objectives for software development, and a GQM definition team with business-driven goals. The *GQM definition team* translates these goals into measurement goals and refines these measurement goals into related metrics and procedures, based on environmental characteristics provided by the project team and previous experience that is packaged in the *experience base*. The *project team* provides the *GQM analysis team* with measurement data. The GQM Analysis Team processes these data and presents results to corporate management and the project team.

In the experience factory an GQM analysis team processes gathered data and provides feedback to projects. A GQM definition team maintains the information, makes it efficiently retrievable and controls and monitors access to it.

In this book we do not make such a distinction between two GQM teams. We will use the term 'GQM team' for all those activities that should not necessarily be carried out by the software development team (project team). We think this is a more practical approach which encourages learning by the GQM team and minimises overhead.

A GQM team and project team principally have overlapping tasks. Two extremes are conceivable. First, all support is provided by a GQM team and merely the results are presented to a project team. In this case a feedback process is limited to the presentation of conclusions. The GQM team interprets these data and decides upon improvements, resulting in a new model of the development process. In this case the role of the project team is limited to the execution of improvements. On the other hand, the entire improvement process could be carried out by the project team itself, without support from a GQM team. This will result in substantially more commitment towards the results. However, other priorities may lead to a lack of effort spent on the measurement programme: when deadlines are approaching, the project team will focus on finishing their core activities and forget about the measurement programme.

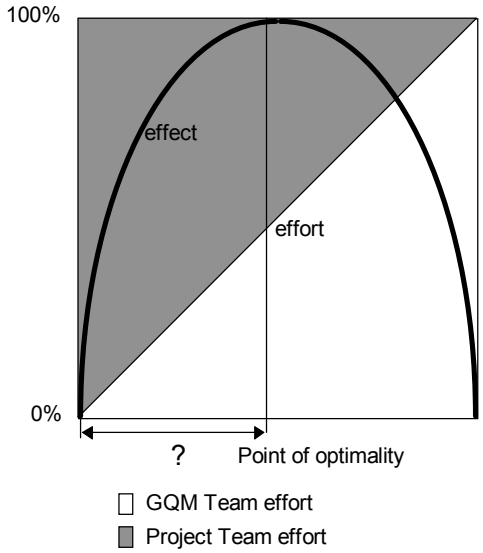


Figure 3-7: Effect depending on distribution of effort.

In our opinion, ideally, the entire quality effort should be executed by the software developers themselves, being most knowledgeable in their field. However, practice shows that in these cases most quality programmes stop when project deadlines emerge. A separate GQM team that safeguards the quality effort is therefore indispensable. However, if such a GQM team also interprets the measurement data and establishes the conclusions (the suggested improvement actions), the quality effort may well loose the commitment of a project team. A kind of balancing of effort is therefore required. This balancing of effort of a GQM team and project team is illustrated in Figure 3-7.

Our experience is that a good balance is often found when a project team carries out 30% of the overall effort of a GQM measurement programme and 70% of all work is done by a GQM team (see also Birk et al, 1998).

3.7 Questions and assignments

3.7.1 Questions

1. Which are the four phases of the GQM method?
2. What is the difference between:
   a. A product and a process metric?
   b. An absolute and a relative metric?
   c. A direct and an indirect metric?
   d. An objective and a subjective metric?
3. Which three purposes of measurement exist? What are the differences between them?
4. Explain the paradox in management commitment and involvement of management.

5. Which phase of the four phases of the GQM method is considered to be the most important one?
6. The GQM paradigm has a top-down and a bottom-up process. Where do these two processes interconnect?
7. What is the most important benefit of linking GQM metrics to the software development process?
8. Which are the six steps of the quality improvement paradigm (QIP)?
9. What is the difference between a project team and a GQM team?
10. Explain the advantages and disadvantages of having most of the effort of the measurement programme put in by either members of the project team or GQM team.

3.7.2 Assignments

1. Write down two typical measurement goals that have an improvement purpose (purpose 3 in Figure 3-1).
2. Derive from each of these goals, pre-conditions that should be under control (purpose 2 in Figure 3-1). Rewrite these pre-conditions to measurement goals with a control purpose.
3. Derive from the control type measurement goals, the underlying topics that must be understood (purpose 1 in Figure 3-1) before control can be striven for. Formulate these understanding pre-conditions as measurement goals too.

'A cynic is a man who knows the price of everything, and the value of nothing'
Oscar Wilde

4 Costs and benefits of GQM measurement programmes

4.1 Introduction

The cost/benefit analysis of this chapter will illustrate that setting up a measurement programme where benefits exceed costs is not as straightforward as may be expected. Our hypothesis is that in practice many measurement programmes are abandoned, because management perceives them as unbeneficial. When a cost/benefit analysis is included in a measurement programme such perceptions are made more explicit and open to discussion. Including this type of analysis is, therefore, regarded as an essential element of a successful measurement programme. First, costs of measurement programmes will be discussed and secondly benefits.

4.2 Cost of GQM measurement

The operational cost model of GQM measurement programmes which is presented in this section has been derived from six GQM measurement programmes (Birk et al, 1998). Almost all costs are costs of labour eg effort of personnel for the particular activities of a measurement programme. The operational cost model will be described for two particular situations, being:

- initial application of GQM measurement in an organisation (first time usage); and
- routine application of GQM measurement (any other usage than first time).

This distinction is made because a first measurement programme requires substantially more effort than any subsequent one. The two variants of the cost model are described in the two following sections. For each variant first, the underlying assumptions are stated. Second, the effort figures are provided for each process step and role involved in the measurement process. Third, the cost structure is explained, and fourth, typical variations of the scenario are discussed. In both costs models the following typical activities are discerned:

1. *GQM programme planning.* This includes the identification of available input, preconditions and constraints, the set up of an infrastructure, the selection of an improvement area, the selection of a project, initial planning of the measurement programme, and the preparation and training of the Project team.
2. *Identify and define GQM goals*. This includes the characterisation of the project and organisation, identification and selection of improvement goals, definition of the measurement and GQM goals, modelling of the relevant software processes involved in the measurement programme, and the identification of artefacts to be reused.

3. *Conduct GQM interviews.* This includes studying documentation, defining, scheduling and inviting interviewees, briefing of a project team, conducting GQM interviews and reporting them.
4. *Develop GQM deliverables.* This includes definition, reviewing, and refining of a GQM plan, the definition of a measurement plan, identification and definition of data collection procedures, reviewing and refinement of a measurement plan, and development of an analysis plan.
5. *Data collection.* This includes a data collection trial to test data collection procedures and forms, briefing the project team and kick-off of the measurement programme, collection, validation, coding and storage of measurement data.
6. *Data analysis and interpretation.* This includes analyses of the measurement data, preparation of the presentation material, planning, conducting and reporting of the feedback sessions.

The operational cost model for routine application of GQM measurement programmes is addressed first, because it refers to the standard situation once GQM is introduced. This model is also less complex. In section 4.2.2 the cost model for first application will be described.

4.2.1 Effort model for routine application of GQM

In this section a cost model for routine application of GQM is described. First, a typical project for which the total effort will be calculated is introduced. Second, the model itself will be stated. Third, the model will be explained.

The cost model is applied on a typical GQM measurement programme containing the following characteristics:

- *Participants.* The cost model is based on a typical software project where four engineers and one project manager need to be interviewed. The GQM team consists of one person.
- *GQM goals*. The measurement programme contains one (major) GQM goal, however, three different viewpoints (i.e., management, quality assurance, and software engineer) are possible.
- *Size of project team*. The project team consists of ten software engineers.
- *Support infrastructure*. There is an existing tool infrastructure for supporting a GQM measurement programme (including computer tools and paper forms).
- *Training and briefings*. There is no special need for training and briefing, because the participants of the measurement programme are familiar with GQM measurement.
- *Feedback session.* As feedback sessions are both the most valuable and costly part of a measurement programme, it is assumed that five feedback sessions are sufficient to attain the specified goal.

Task	GQM team	Manager	*Single engineer*	For 10 engineers	Total
GQM programme planning	4	2	-	-	6
Identify and define GQM goals	8	1	*1*	10	19
Conduct GQM interviews	40	2	*2*	8	50
Develop GQM plan	40	1	*1*	6	47
Data collection	24	1	*1.5*	15	40
Data analysis and interpretation per feedback session[1]	*48*[1]	*2*[1]	*2*[1]	*20*[1]	*70*[1]
Data analysis and interpretation (5 feedback sessions)	240	10	*10*	100	350
Total	356	17	*15.5*	139	512

Figure 4-1: Effort model of routine application of GQM (effort in person hours).

Figure 4-1 shows the operational cost model for routine application of GQM. Required effort is given in hours. Please note that the total effort is, of course, based on the project description stated before. The cost model, however, contains sufficient information to calculate other project settings. The cost model is referred to as an 'effort model' here, because we will focus on required effort only. In order to calculate costs, the required effort needs to be multiplied by the appropriate labour costs of an organisation.

The following cost structure is typical for routine application of GQM:

- Roughly 30% of the effort is spent on defining the measurement programme, while 70% is spent on continuation of the measurement programme, which is almost completely spent on the feedback sessions.
- 70% of the effort on the measurement programme is spent by the GQM team, and only 30% of the total effort is spent by the project team.
- The effort spent by the project team is less than 1% of their total working time.
- A total three person-months of effort is required for a typical GQM measurement programme, distributed over one calendar year

Planning and definition normally requires a period of 4 to 6 weeks. Availability of the interviewees is critical for the duration of the definition. Duration of data collection and feedback depends on the frequency of feedback sessions and on quickness of goal attainment. Experience shows that an elapse time of a year is normal to attain a measurement goal.

[1] The cost data for data analysis and interpretation apply for one data collection cycle lasting 2 months and being concluded by one single feedback-session. However, experiences showed that for goal attainment 3-7 feedback sessions are normally needed. So, the table contains data under the assumption that 5 feedback sessions are held within the measurement programme.

GQM definition phase

All activities for setting up the measurement programme from preparation to start of data collection, take on average 11½ working days (92 hours) for the GQM team. The effort needed for the project manager and the engineers is relatively little for all measurement definition tasks, respectively 6 hours, and 4 hours. Clearly, in this stage most effort is performed by the GQM.

Data collection phase

An opening briefing is held with an effort of half an hour per participant. The effort of the GQM team is spent on a full day for the briefing and another two days for data validation.

GQM interpretation phase

The GQM team typically needs five days to prepare a feedback session. Feedback sessions take approximately two hours per participant, and another six hours for a GQM team member are needed to report the feedback session. After goal attainment, the experiences of the measurement programme should be documented to capture gained expertise.

There are many factors that influence the required effort of a GQM measurement programme. Three important ones are:

- *Number of GQM goals*. An increasing number of GQM goals also increases the required effort of all activities. Furthermore, the complexity of a measurement programme increases rapidly as more goals are included. Practice shows that three goals are really a maximum and even that is already difficult.
- *Size of a project team*. The increasing size of a project team also increases the required effort, both absolutely and relatively. More project team members need to be interviewed, and feedback sessions will require more time.
- *Number of feedback sessions*. As feedback sessions are the most important and also the most time consuming part of a measurement programme, the number of feedback sessions is an important factor. It is, however, advised not to save on feedback sessions, because a measurement programme without appropriate analyses will loose all its benefits. Benefits are derived only from feedback sessions.

4.2.2 Effort model for initial introduction of GQM

In this section the cost of a first GQM measurement programme is described. The cost categories are identical to the previous model of routine application, however, required effort per category is significantly higher. A typical first GQM measurement programme has the following characteristics:

- An *external GQM expert* is a member of the GQM team and acts as a coach to the project team and the other members of the GQM team.
- The project team contains *less engineers*. The subsequent example mentions eight engineers and this is even quite large for initial GQM introduction. A first project with five engineers is more than sufficient.
- There is *no measurement infrastructure*, such as tool support for data collection or measurement forms. Usually data collection support is developed gradually, starting with paper-based data collection forms and is later moving towards on-line data collection.

Activity	GQM expert	GQM member	Manager	*For single engineer*	For 10 engineers	Total
GQM programme planning	24	32	4	*4*	40	**100**
Identify and define GQM goals	18	8	2	*1*	10	**38**
Conduct GQM Interviews	35	35	2	*1*	4	**76**
Develop GQM deliverables	196	138	2	*1*	10	**346**
Data collection	-	16	-	*3*	30	**46**
Data analysis & interpretation per feedback session	*12*	*48*	*8*	*4*	*40*	*108*
Data analysis and interpretation (5 feedback sessions)	60	240	40	*20*	200	**540**
Total	**333**	**469**	**50**	*30*	**294**	**1146**

Figure 4-2: Effort model for initial introduction of GQM (effort in person hours).

Figure 4-2 shows the cost model for initial introduction of GQM. The effort is again stated in person hours. In order to calculate costs, these hours again need to be multiplied with the appropriate labour costs per hour.

The following cost structure is typical for initial introduction of GQM measurement:

- An initial GQM measurement programme needs approximately eight person-months of effort. This is significantly more than the three months mentioned for routine application. A first project requires a lot of initial work. An example is the initial training of the GQM team.
- Approximately 50% of the total effort is spent on defining the measurement programme. This differs significantly from the 30% in routine application. Especially the definition of the GQM and measurement plan requires more effort.
- 70% of the total effort on the measurement programme is spent by the GQM team (including expert), and 30% by the project team, which is identical to routine application.
- The effort spent by the project team is less than 2% of their total working time, which still is double the amount compared to routine application of GQM measurement.

Planning and definition requires approximately three months (compared to four to six weeks in a routine application). The additional time is primarily spent on training and learning. For data collection and feedback the same rules of thumb apply as for routine application. Effort depends on the frequency of feedback sessions and on speed of goal attainment. Experience shows that approximately one year should be sufficient to attain a measurement goal. However, first feedback sessions proved to be difficult for some project teams. The interactive, open and learning character of feedback sessions might be quite new for engineers. Therefore, it is not unlikely that a first measurement programme takes even longer.

Other differences compared to routine application of GQM measurement are:

- *GQM programme planning*. Contains significantly more effort for the GQM team. The effort is three days for the external GQM expert and four days for the other members of the GQM team.
- *Identify and define GQM goals*. The GQM team again needs additional to allow training and clarification by the GQM expert.
- *Develop GQM deliverables*. Again significantly more effort is needed, because the GQM team needs to be instructed how to set up such a plan. Furthermore, a measurement infrastructure needs to be developed for the first time. This type of effort, containing for example the set-up of data collection procedures, documents, tools and forms, requires a lot of effort.
- *Data collection*. The additional effort is one day for coaching, validation and "trouble shooting" of the measurement database and data collection forms. Some additional effort is also required to motivate the project team during the start of data collection.
- *Data analysis and interpretation*. Interpretation is the most important part of a measurement programme. The project team will have to learn how to analyse measurement data. It will be especially difficult to learn that the performance of individuals should not be evaluated, but that a development process needs to be improved. Interpretations should focus on improvement actions and answering of GQM questions.

Again there are many factors that influence the required effort of a GQM measurement programme. The most important ones are:

- *Resistance to change*. There will always be resistance to change. This can be overcome by focusing on process aspects, showing the need for the change and make sure that the project team receives sufficient training in order to be prepared. Including the goals and questions of a project team is also an important element in overcoming resistance.
- *Size of project team*. If one is less experienced the size of a project team becomes more important. All phases will require more time and overall it will take considerably more time until the measurement programme will produce any results. This increases the chance that people loose their interest. It is, therefore, recommended to start with a project team of approximately five engineers.

4.3 Benefits of GQM measurement programmes

Measurement programmes may have various kinds of benefits. The most desirable benefit is achievement of the explicitly stated improvement goals. Such improvement goals can be product- or process-related (e.g., product reliability or process effectiveness). In addition, measurement can have numerous other effects such as improving communication within a project team, attitude of personnel, process definition, and process execution.

Eventually, benefits should result in financial gains. However, it will often be difficult to make a financial appraisal of quality improvement, because many indirect links are involved. For example, the improved software development process, will lead to better embedded software, increasing product characteristics, increasing sales and ultimately profits. However, many other elements will also change in the mean time and it will therefore be quite complicated to link the additional profits to the process improvements. In this type of situation it is advised not to calculate the exact financial benefits and to remain

with a quantitative description. Management should then evaluate whether the costs can be expected to be worthwhile. Knowledge of how customers will value a better quality is essential here.

However, it may also be relatively easy to quantify financial benefits. For instance, in one of the measurement programmes (Chapter 11), how developers were interrupted during their work was investigated. Based on the findings, work and communication practices in the team were improved, and the number of interrupts was decreased significantly. The improvements had a measurable impact on productivity and the associated cost reduction could be calculated.

In any case every software measurement or improvement programme has a set of goals to be achieved. These goals are made explicit and measurement will play an effective role in achieving them. These goals can address product and process aspects. Based on the measurement results, the developers can (re-)focus their development activities towards achieving these goals. Team members will identify process changes themselves. This ensures that improvement actions are very well supported and therefore implemented successfully. This is experienced as a significant benefit from measurement and GQM feedback sessions in particular.

As stated before, many other benefits will result from measurement that are not directly related to the achievement of explicit goals. Typically, they are related to communication and interaction in a project team, attitude and knowledge of the team members, and the degree to which quality assurance and improvement actions are integrated with software development.

A general impression from the measurement and improvement programmes is that GQM measurement makes improvement initiatives solid and successful. The project teams use GQM as a change agent for improving their work practices and the products they produce. Furthermore, the goal-orientation of GQM helps to keep improvement programmes focused: it helps limiting the scope of an improvement programme to what is feasible in a given situation, and it helps to identify the necessary improvement actions in a well-informed and appropriate manner.

4.4 Questions and assignments

4.4.1 Questions

1. Which two typical applications are there for the GQM method in an organisation?
2. What is the typical effort distribution in percentages, for GQM measurement programmes, looking at:
   - 2.1. Distribution over project team and GQM team, in a routine situation?
   - 2.2. Distribution over defining and continuation of the measurement programme, in a routine situation?
   - 2.3. Distribution over project team and GQM team, in an initial situation?
   - 2.4. Distribution over defining and continuation of the measurement programme, in an initial situation?
3. How long (in calendar time) does the typical set-up of a measurement programme take in a routine situation? And in an initial situation?
4. Why should benefits of GQM measurement programmes not only be evaluated financially?

4.4.2 Assignments

A project team consisting of ten engineers and a manager, that have already been involved in two measurement programmes, are working on the definition of a new measurement programme. The objective is to identify product features that need most customer support during the first six months after release. Based on these findings improvements will be made regarding, for example, documentation, help-files, and customer support desk manuals.

Develop a time plan in which the participation of the project team members is listed. Use the effort table in Figure 4-1, to estimate the required effort. Please, do not forget to include work that has to be done by GQM team members. Also note that during the period in which this measurement programme is defined, the project team is preparing a release of new software. They therefore work under pressure and appointments with the GQM team have the chance of being changed at the last moment.

PART 2

The GQM method stepwise

'If a had eight hours to chop down a tree, I would spend six sharpening my axe'
Abraham Lincoln

5 GQM planning phase

Checklist GQM planning phase		
No.	**Checklist item**	✓
5.1	Separate GQM-team is assigned and installed	
5.2	GQM-team has sufficient resources available	
5.3	Corporate improvement objectives are formulated and approved	
5.4	Project team is established and supports improvement objectives	
5.5	Project team has at least 2% resources reserved for GQM programme	
5.6	Project plan is available	
5.7	Communication and reporting procedures are defined	
5.8	Training and promotion is planned and described	
5.9	Management is committed and regularly informed	
5.10	Management has approved the project plan	

5.1 Introduction

In Chapter 2 the four main phases of the GQM method were introduced, being *planning*, *definition*, *data collection* and *interpretation*. In this chapter the first phase is described. In this phase a framework is given to introduce a measurement programme. The position of a planning phase in the GQM method is illustrated in Figure 5-1.

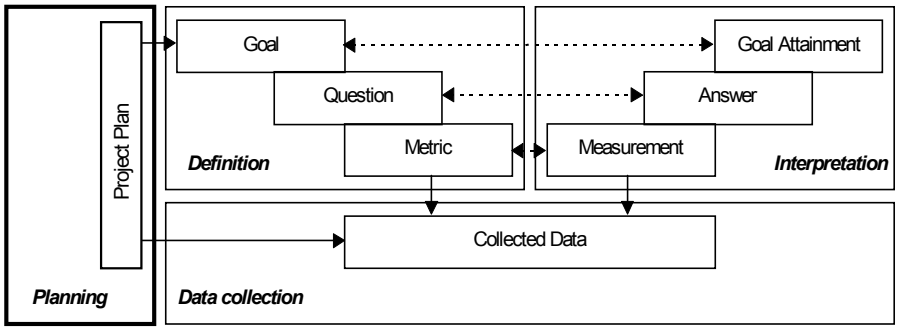


Figure 5-1: The planning phase of the GQM method.

5.2 Planning procedures

The primary objectives of the planning phase are to collect all required information for a successful introduction, and to prepare and motivate members of an organisation for a measurement programme. A *project plan* is an important deliverable of a planning phase. Such a project plan documents procedures, schedules and objectives of a measurement programme and provides a basis for promotion to and acceptance by management. A project plan should also contain a planning for training of the developers involved. The planning phase consists of five steps which are illustrated in Figure 5-2. In the following sections these particular steps will subsequently be described.

5.2.1 Step 1: Establish GQM team

Practice shows that without a separate independent team responsible for the continuity of a measurement programme, measurement activities tend to fail. When deadlines emerge, measurement activities often receive less attention, unless an independent GQM team is established (Basili et al, 1994b). Although this team should be independent, it is of course

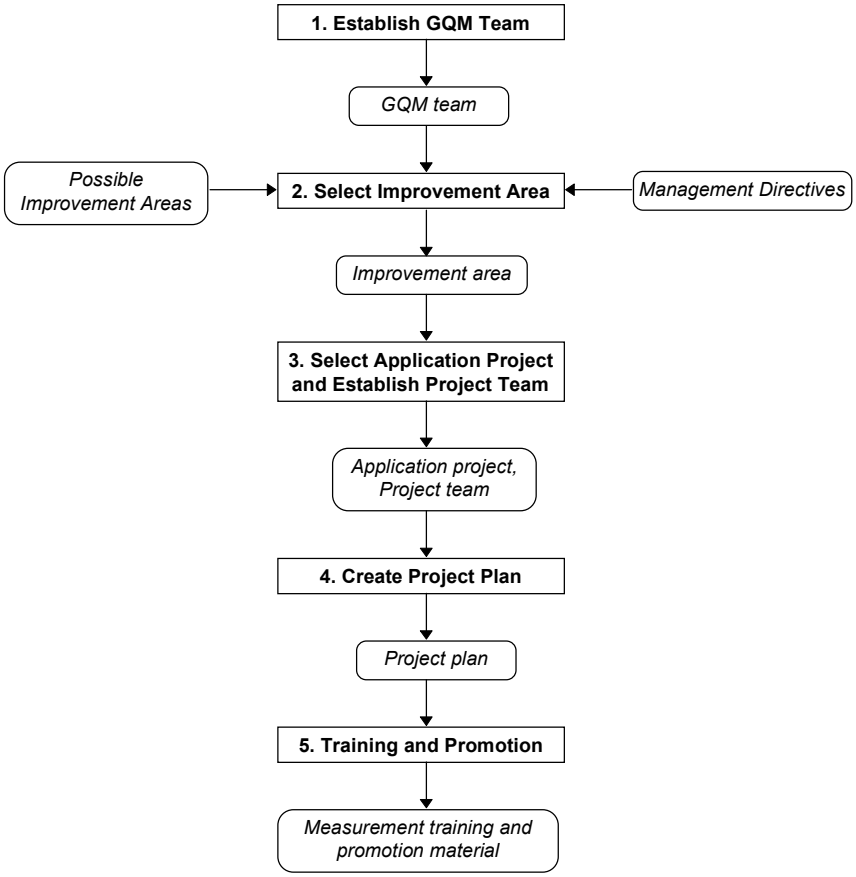


Figure 5-2: GQM planning procedures.

possible to incorporate its activities in for example a Quality Assurance department, Software Engineering Process Group, or SPI department. A GQM team should:

a. Be independent from project teams, and have no interest in measurement results.
b. Possess sufficient background knowledge on the objects of measurement.
c. Keep in mind that the project team 'owns' the improvement programme, because a project team is most knowledgeable on a project.
d. Be improvement oriented, which includes that it is willing to improve itself too.
e. Be enthusiastic, in order to motivate the project team.

An important requirement for the success of a measurement programme, is the level of mutual trust and cooperation between the GQM team and the project team (Solingen et al, 1997). Therefore, it is important that the GQM team has no interest in the data that the project team gathers. In order to be able to guide and support a measurement programme, the GQM team needs to have a certain level of background knowledge regarding the development processes and products the measurement programme focuses on. This is an important prerequisite, as the GQM team should be able to discuss interpretations with the project team.

Furthermore, the GQM team should regard itself as a facilitator of learning (Solingen et al, 1997) and have an improvement-oriented mind. The GQM team should show respect to the project team regarding the execution of the development tasks, as not always pre-defined procedures will, or can, be followed. The GQM team should have an open mind toward such issues, as in the end, the developers have all the knowledge on the processes and products and are eventually responsible for their project and its measurements.

The roles of the GQM team are 'manager' (responsible for continuation of the measurement programme), 'coach' (expert on GQM), and 'support engineer' (to support measurement activities). The main activities of the GQM team are:

- *Plan* measurement programmes within development projects.
- Carry out measurement *definition* activities and develop GQM deliverables.
- Check *data collection* by the project team and process available data.
- Prepare *interpretation* of the measurement data by organising feedback sessions.
- Moderate feedback sessions.
- Report progress to project team and management, and disseminate and package results.

Personal skills of GQM team members play a crucial role in the success of a measurement programme, particularly their ability to motivate others. When a GQM team lacks this ability, the whole measurement programme stands little success. A GQM team should therefore include reputable staff.

5.2.2 Step 2: Select improvement area

After initiation of a GQM team, the next step is the identification and selection of suitable product or process improvement areas. Examples are:

- Apparent problems that the organisation struggles with.
- Process improvement areas identified by means of an assessment.
- Product improvement areas based on high-level business goals.

The improvement areas should be operational refinements of corporate improvement objectives and business goals. Particularly corporate improvement objectives together with the four basic aspects: *cost*, *time*, *risk* and *quality*, should give an idea of improvement goals. Examples of such goals are 'increase customer satisfaction', and 'expand market share', however, they can also be derived from governmental directives, such as, safety or environmental issues.

Guidance during the refinement of high level business goals into improvement and measurement goals is difficult. However, SEI has issued a report which contains some steps to support this refinement. This report is publicly available and can be downloaded free of charge from the SEI home-page. The title is 'Goal-Driven Software Measurement: a Guidebook'. This report primarily focuses on the *definition* of software measurement.

Apparent process or product problems are usually indicated by departmental management or the developers themselves. Information on such issues can be retrieved during interview sessions with representatives that are directly involved, or responsible for handling those problems. Because these problems are usually straightforward, the people involved are normally well motivated to solve them and to apply measurement in doing so. If members of an organisation are unable to give a clear description of their problem, however, brainstorming sessions with departmental and group representatives can be held in order to create a better understanding of the issues. Such sessions typically start with defining the purpose of the organisation on a company-wide level, followed by focusing on a specific department, group, or team, and ending with describing particular problems or improvement areas (Solingen et al, 1995).

Another method to identify improvement areas is to conduct an assessment. An assessment is a review of a software development organisation to provide a clear and fact-based understanding of the organisation's development practice, to identify its major problems, and to initiate actions to make improvements (based on Humphrey, 1989).

Once suitable improvement areas have been identified, the following details have to be described:

- problem or improvement area;
- processes or products involved;
- environmental, legislative, organisational, and technological influences
- people involved;
- previous experience of these people with measurement and GQM.

These characterisations will be used as input for a subsequent phase of the GQM method. To support these characterisations, questionnaires can be used. However, questionnaires always have to be tailored to a specific organisation or project.

5.2.3 Step 3: Select application project and establish a project team

A project team consists of all the people that work on a particular software development project. Since the project team members are the people that will actually perform measurement activities and in the end may need to adopt their work processes, the success of a measurement programme heavily depends on their willingness, motivation, and enthusiasm. Therefore, effort should be spent to align measurement objectives and improvement ideas of a project team. This is a responsibility of a GQM team. They will continuously monitor and stimulate the dedication of a project team regarding the measurement activities.

Because an improvement area is selected first, even before a project team, a GQM team has a significant influence on the measurement area. However, a GQM team should never be made fully responsible for a measurement programme, since all measurements should be related to specific goals of a project team. The project team becomes the owner of a measurement programme and responsible for the results. If the project team decides to stop specific measurements, or to change the scope of measurements, this needs to be done. The GQM team is, of course, allowed to challenge a project team and to discuss their proposed changes. Possibly, the GQM team can convince the project team that their proposed changes are not appropriate. However, it is emphasised that an improvement initiative is the responsibility of the actual developers and not of a GQM team.

5.2.4 Step 4: Create project plan

Once a GQM team and project team have been established and improvement areas selected, a proposal for a measurement programme should be developed on which the measurement programme will be based. This proposal, ie the project plan, is primarily created by the GQM team based on input (and approach) from the project team, and should contain the following items:

- *Management abstract*, which presents the measurement programme in approximately 20 lines.
- *Introduction*, which presents the scope of the measurement programme, and the relation of the improvement objectives to the software development project goals.
- *Characterisation*, which describes the outcomes of the characterisations that were held within the programme on the organisational, project, project team and GQM team levels.
- *Schedule*, which presents the managerial issues for the measurement programme, such as a timeline, list of deliverables, resource allocation and a cost-benefit analysis.
- *Organisation*, which describes the relevant organisational, project team and GQM team structures of the measurement programme.
- *Management process*, which presents priorities, management reporting procedures and risk control activities.
- *Training and promotion*, which presents the activities planned for training the project team and for promotion and dissemination of the results over the organisation.

As indicated, the main part of the project plan is based on the information gathered in the first steps of the planning phase. However, the project plan should contain at least two sections that will contribute to the acceptance of the project by corporate management.

In the first place, the project plan should contain an *initial cost-benefit analysis* of the project. This analysis is based on the proposed effort of the GQM team and project team necessary to completely execute the measurement programme, and the expected benefits the improvement goals will offer once they have been achieved.

Furthermore, the GQM team should also make a *realistic planning* of the measurement programme, in which all steps to be taken in the subsequent procedures of the GQM definition, data collection, and interpretation phase of the project are scheduled in time. This planning should also indicate to what extent members of a project team will be involved in the particular steps of the project. Project milestones and deadlines, together with a list of the deliverables at each point, also need to be specified in a project plan.

A project plan will primarily serve as a proposal to management in order to receive management approval of, and commitment to, a project. Once a proposal is accepted, a project plan will be maintained by the GQM team. During a project additional information will become available and a project plan will probably evolve over time.

5.2.5 Step 5: Training and promotion

Crucial for the success of a measurement programme is that all people participating in the project are enthusiastic and motivated, and remain committed to the objectives of the project. To accomplish this, the GQM team should organise regular training and promotion sessions during which:

- a clear definition of proposed improvement goals are presented;
- benefits of the measurement programmes are explained;
- the impact of measurement on daily development activities is indicated;
- experiences from other organisations/projects are discussed.

If possible, all people participating in a measurement programme should be present during these sessions. Particularly important is that persons responsible for managing the project team are present at those sessions, as well as representatives from higher-level management.

An indication of the investment and the expected benefits needs to be given up-front. This should be done for two reasons. One, because the project manager needs to assign his people to the programme, so he should plan their effort. Two, because the effort spent by the project team is considered as an investment, which should not be undertaken if the expected revenues would not exceed the investment.

Once a project is approved, the project team members have to be trained according to their roles within the project. To accomplish this, the GQM team should organise training sessions during which:

- principles of measurement are explained;
- the GQM paradigm is explained;
- the GQM method is explained.

The main focus of the training sessions should be on the last point: explaining the GQM method. The GQM team should explain the particular process steps within a measurement programme to the project team members, and should indicate to them to what extent they will be involved in these steps. Next to the explanation of the method, the project time table should be presented to the project team members, so they know at what point in time effort is required from them. During such training sessions, the GQM team should be aware that not too much emphasis is put on the theoretical background of GQM. Rather, the project team is interested in hearing practical issues regarding the method, as it wants answers to questions such as:

- What measurement tasks should I perform?
- Why should I perform those tasks?
- How and when should I perform those tasks?
- How much effort is required by me to perform those tasks?
- Do the tasks influence my daily activities?
- What do I get back from this? What will I learn?

Minimally, one training session should be organised during which the above points are discussed. If necessary, more should be added. All people involved in the project should be trained before the actual project activities start.

5.3 Questions and assignments

5.3.1 Questions

1. Which are the five steps of the GQM planning phase?
2. Which items should be included in a project plan of a measurement programme?
3. Which typical practical questions of a project team should be addressed during briefing or training of the project team?
4. Which percentage of project effort should a project team plan for a GQM measurement programme?
5. Who selects the improvement areas? How can this be supported?

5.3.2 Assignments

1. Design a table of contents for a project plan for the GQM measurement programme of which you have defined goals in the assignment of Section 4.4.2.
2. Download the characterisation questionnaire for a GQM programme from 'http://www.iese.fhg.de/Services/Projects/Public-Projects/Cemp/GQM-Process-Model/pm_76.html'. Complete such a characterisation for one of your recent projects and try to derive improvement areas from this characterisation.

'The truth is that you always know the right thing to do.
The hard part is really doing it'
H. Norman Schwarzkopf

6 GQM definition phase

Checklist GQM definition phase		
No.	**Checklist item**	✓
6.1	GQM goals are defined	
6.2	Project team adopts GQM goals	
6.3	Process models that identify measurements are available	
6.4	GQM question definitions are available and consistent with the goals	
6.5	GQM metric definitions are available and consistent with the questions	
6.6	GQM metrics are checked on consistency with process model	
6.7	GQM plan is available	
6.8	Measurement plan is available	
6.9	Analysis plan is available	

6.1 Introduction

This chapter describes the definition phase of a GQM measurement programme. The definition phase is the second phase of the GQM process (see Figure 6-1), and concerns all activities that should be performed to formally define a measurement programme. During this phase three documents are produced:

- GQM plan;
- measurement plan;
- analysis plan.

These three plans contain all pertinent information regarding the measurement programme.

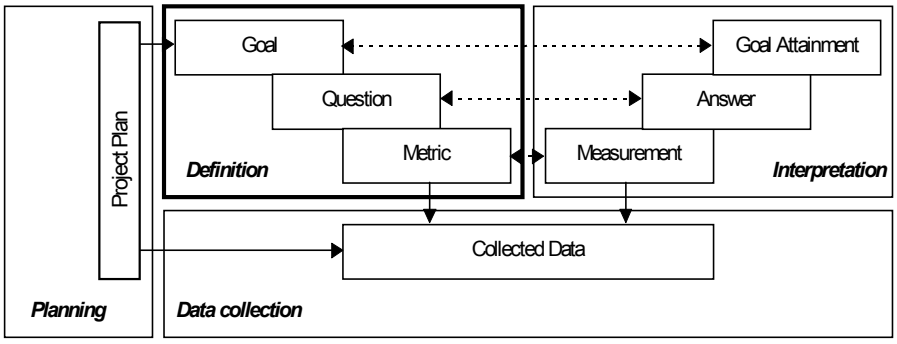


Figure 6-1: The definition phase of the GQM method.

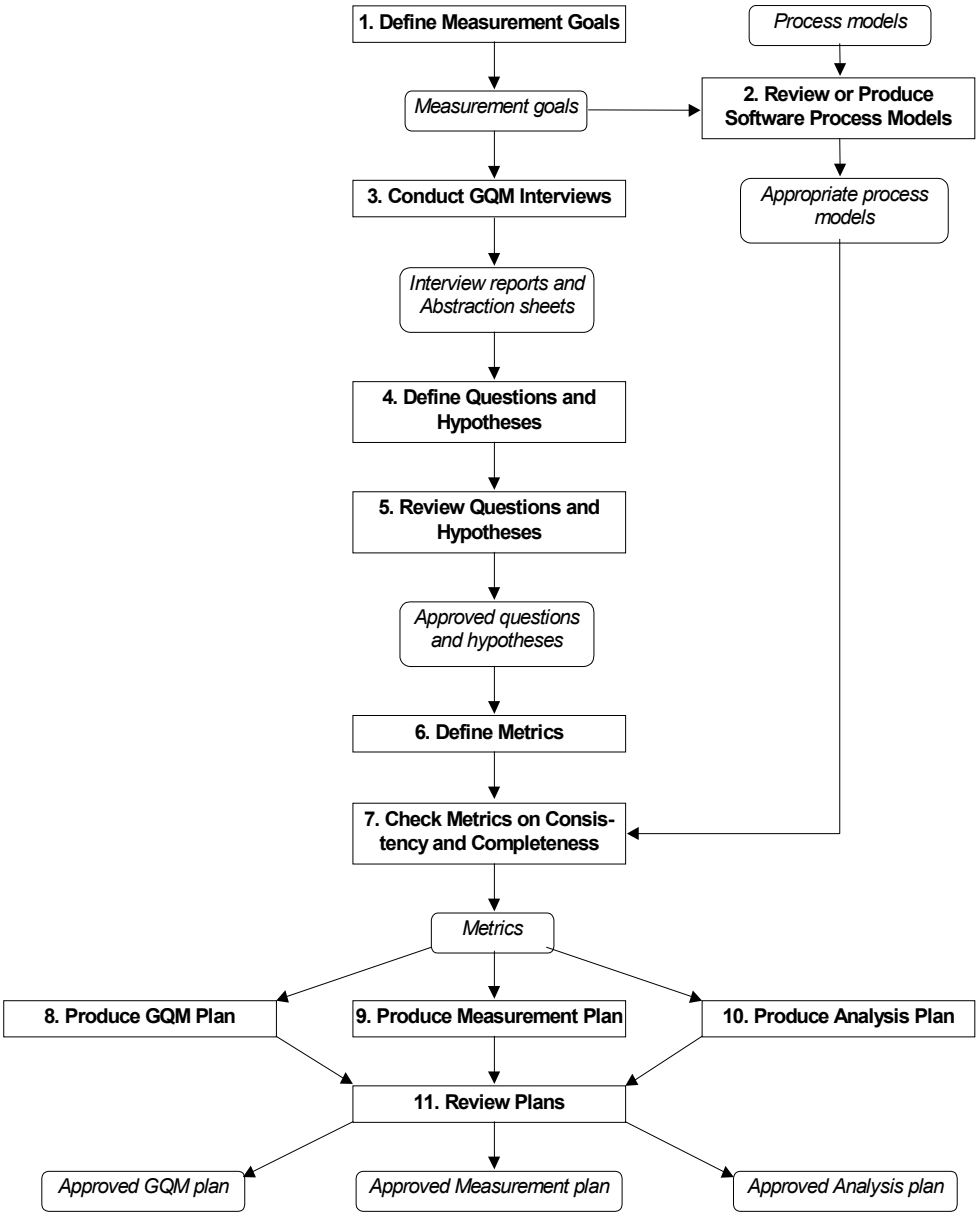


Figure 6-2: GQM Definition procedures.

6.2 Definition of procedures

To complete the definition phase, an eleven-step procedure is proposed, of which the flowchart is shown in Figure 6-2. The next sections describe in more detail the eleven steps of a GQM definition phase:

1. Define measurement goals
2. Review or produce software process models
3. Conduct GQM interviews
4. Define questions and hypotheses
5. Review questions and hypotheses
6. Define metrics
7. Check metrics on consistency and completeness
8. Produce GQM plan
9. Produce measurement plan
10. Produce analysis plan
11. Review plans

6.2.1 Step 1: Define measurement goals

The first step in the definition process is the definition of formal measurement goals. These measurement goals are derived from the improvement goals which were already identified in the preceding planning phase and are described in the project plan. All people participating in the measurement programme should be involved in the definition of measurement goals. Without this involvement, people's commitment to the measurement programme is at risk, as it may no longer be clear to them why measurement is applied. As already described in Chapter 3, the people participating in the measurement programme are the project team members, their manager and the GQM team members.

Measurement goals should be defined in an understandable way and should be clearly structured. For this purpose, templates are available that support the definition of measurement goals by specifying *purpose* (what object and why), *perspective* (what aspect and who), and *context* characteristics (Basili et al, 1994). This template is illustrated in Figure 6-3.

If a project team has little experience in GQM goal definition, the GQM team can refine the global improvement goals into GQM measurement goals, and have these goal definitions evaluated during a review session in which all project members participate. If a project team already has some experience with GQM goal definition, the GQM team should preferably define GQM goals in cooperation with all project members during a session. At the end of this session, all people should fully understand and agree upon the defined GQM

Analyse	the object under measurement
For the purpose of	understanding, controlling, or improving the object
With respect to	the quality focus of the object that the measurement focuses on
From the viewpoint of	the people that measure the object
In the context of	the environment in which measurement takes place

Figure 6-3: GQM goal definition template (Basili et al, 1994a).

measurement goals.

Within such a brainstorming session it is always possible to define many goals. However, these should be relevant to the business, represent strategic goals from management, and support high priority processes of the organisation. Sometimes it will be obvious what the measurement goals should be, for example an urgent problem that needs attention. However, often it will be difficult to select or prioritise measurement goals. In this case, multicriteria analyses are a possibility. A mechanism to support goal definition and selection in a meeting, is by asking the 'seven questions' stated below:

1. What are the strategic goals of your organisation?
2. What forces have an impact on your strategic goals?
3. How can you improve your performance?
4. What are your major concerns (problems)?
5. What are your improvement goals?
6. How can you reach your improvement goals?
7. What are possible measurement goals, and what are their priorities?

All members of a project team should be involved in the definition of the goals in order to establish their commitment. If they have selected the goals themselves, they will also be motivated to work on the measurement programme. Management involvement is also a critical success factor. Management should be able to show how measurement goals relate to business goals and support business improvement.

Deliverable 6.1: *List of GQM measurement goal specifications*

6.2.2 Step 2: Review or produce software process models

Models of the software development process are used to support the definition of a measurement programme by checking the set of metrics of forthcoming step 7 on completeness and consistency. Examples of software process models are given in section 6.3 on page 59.

If process models are already available in the organisation, they have to be reviewed, and, if necessary, improved to support definition of measurements. Possible review methods include formal reviews, brainstorming sessions, structured interviews, or presentations. There should be an agreement on the models by all people involved in the measurement programme. Make sure the process models describe in which way the work is really done and not the ideal way in which it should be done. As measurement is done during the actual work of the project, the process models must give a real picture of that actual way of working.

If no process or product models relevant to the defined measurement goals exist, the GQM team should develop them. As these models are supposed to support the definition of measurements, the application of a modelling technique that identifies measurements on the basis of processes is preferred. Once the GQM team has modelled all relevant processes, the project team should agree upon these newly defined process models.

For a more comprehensive description of suitable process modelling techniques, this chapter contains a section at the end on process modelling (Section 6.3).

Deliverable(s) 6.2: *Approved process models, suitable to identify measurements.*

6.2.3 Step 3: Conduct GQM interviews

The project team should always be closely involved in the development of the measurement programme, however, during the definition of goals, questions, metrics, and hypotheses, the input of the project team is of crucial importance. The team members are the experts with respect to the object under measurement. Therefore, this knowledge can only be extracted from them.

To extract the knowledge from the project team with respect to the defined measurement goals, the GQM team should conduct structured interviews with the individual members. The interviews aim at capturing the definitions, assumptions and models of the project team related to the measurement goals, and, therefore, the main purpose of these interviews is to make the implicit knowledge of the project members explicit.

Though it may seem more efficient to interview more than one team member in one interview, it is recommended to conduct individual interviews. As the purpose of the interview is knowledge acquisition, it is important to extract the knowledge from the people without the presence of factors that may influence their opinion. If more than one person is being interviewed during a single session, these people may influence each other's opinion, and not all available knowledge or honest opinions may be extracted from them. Also, the interviewer should not push the interviewee in any direction. The information acquired from different people during interviews will be combined in subsequent steps.

Using abstraction sheets

To support the communication between a GQM team and a project team during interviews, a GQM team uses so-called 'abstraction sheets' (Latum et al, 1998). The use of abstraction sheets during interviews provides a structured approach to focus on relevant issues regarding the goal, and prevents issues being overlooked. An abstraction sheet summarises the main issues and dependencies of a goal as described in a GQM plan and is discerned in four sections. The four sections of an abstraction sheet are (see Figure 6-4):

- *Quality focus*: what are possible metrics to measure an object of a goal, according to the project members?
- *Baseline hypothesis*: what is the project member's current knowledge with respect to these metrics? His or her expectations are documented as 'baseline hypotheses' of the metrics.
- *Variation factors*: which (environmental) factors does a project member expect to be of influence on the metrics?
- *Impact on baseline hypothesis*: how could these variation factors influence the actual measurements? What kind of dependencies between the metrics and influencing factors are assumed?

An example of an abstraction sheet is given in Figure 6-4. Hypotheses are grouped in two sections of the abstraction sheet, and are associated to the corresponding questions in the other sections. The four sections can be checked for consistency and completeness, because mutual relations between the sections exist. For example: for every Quality focus, there should be at least on Baseline hypothesis, and possibly some Variation factors. Also, for every Variation factor there should be at least on Impact on the hypothesis. The GQM team can use abstraction sheets in several ways:

- Fill in the abstraction sheet together with the project member, starting with discussing quality focus and baseline hypothesis, and after that variation factors and the corresponding impact. When all sections are filled in, one should follow this approach iteratively until the abstraction sheet is satisfactory completed.
- Train the project team in using abstraction sheets, and when they are familiar with the concept, let them fill in an abstraction sheet themselves. This approach requires some investment, since it is not easy to train people on the abstraction sheet concept.
- Fill in the abstraction sheet in advance before the interview. In this way the GQM team prepares the interview, and a kind of draft version becomes available. This approach must be handled carefully, since the interviewer is in fact representing his or her implicit models on the abstraction sheet. The interview is then some kind of validation of the draft version. To follow this approach, knowledge of the context and subject of the measurement goal is needed.
- Use the abstraction sheets as a guidance for analysis and interpretation of results during feedback sessions.

Abstraction sheets are a powerful tool that can be used during the set-up of the measurement programme: information from interviews can be organised in a structured way and copied from the abstraction sheets into the GQM plan. However, abstraction sheets can also be used for structuring the presentation and interpretation of measurement data during feedback sessions. In fact, an abstraction sheet is a one-page summary of a GQM plan. Not all direct measurements defined in a GQM plan are represented on abstraction sheets, only the basic ones that reflect the most important metrics. An example of an abstraction sheet is shown in Figure 6-4.

Deliverable(s) 6.3: *Set of interview reports and abstraction sheets.*

Object	**Purpose**	**Quality Focus**	**Viewpoint**
Delivered Product	Understanding	Reliability and its causes	Project Team

Quality Focus
Number of failures:
- by severity
- by detection group
- number of faults
- by module

Variation Factors

Level of reviewing

Baseline Hypotheses (estimates)
Distribution of failures:
- By severity:
  - Minor 60%
  - Major 30%
  - Fatal 10%

Impact of Variation Factors

The higher the level of reviewing, the less minor failures will be detected after release

Figure 6-4: Example of an abstraction sheet.

6.2.4 Step 4: Define questions and hypotheses

With respect to the measurement goals, questions should be defined to support data interpretation towards a measurement goal. As goals are defined on an abstract level, questions are refinements of goals to a more operational level, which is more suitable for interpretation. By answering the questions, one should be able to conclude whether a goal is reached. Therefore, during question definition, checks should be performed as to whether the defined questions have the ability to support conclusion of the goal in a satisfactory way.

If the questions are defined on a level that is still too abstract, data interpretation towards answering the questions provides difficulties as the relationship between the questions and the data is difficult to understand (see Figure 6-5). If, on the other hand, the questions are defined at too detailed a level, clear interpretation from the questions toward the goal will also not be possible. To support an optimal interpretation from collected data to answering questions to concluding on the goal, the questions should be defined at an intermediate level of abstraction between the metrics and the goals.

In order to come to the right level of abstraction for GQM questions, it is useful to document the questions formulated by the project team members in the interviews explicitly. Such interview questions will mostly be on the 'too detailed' level of abstraction, but they might also be too abstract. By grouping similar questions together it will normally be clear what the appropriate GQM question should be. Examples of working towards adequate GQM questions are give in the case studies of Part 3.

Subsequently, for each question, *expected* answers are formulated as *hypotheses*. Formulating hypotheses triggers the project team to think about the current situation and therefore stimulates a better understanding of the process and/or product. Furthermore, after measurement, during data interpretation, these hypotheses of measurement results will be compared with the actual measurement results. The purpose of this comparison should not be to evaluate a possible correctness of the hypotheses, but to encourage the project team to identify and analyse the underlying reasons that caused the actual results to deviate, or conform, from their expectations.

In other words, hypotheses are formulated to increase the learning effect from measure-

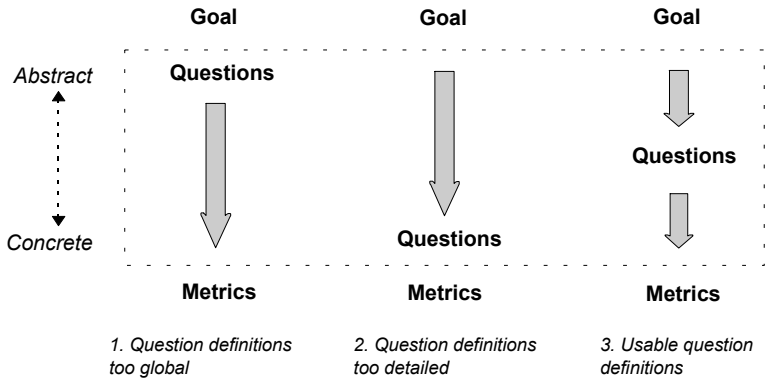


Figure 6-5: GQM question definition (Solingen, 1995).

ment. By formulating hypotheses, current knowledge of the project team is made explicit. Learning theory has shown that adults always learn against an existing frame of reference (Senge, 1990). People already have much knowledge and experience gathered throughout their lives, and learning requires that new knowledge is related to existing knowledge. Therefore, it is important to make knowledge people already possess explicit, before absorbing new information. This way, links can be created between new and existing knowledge, allowing the unfamiliar to be interpreted by the familiar. The previous knowledge of the project team with regard to the measured process, is captured by building the measurement programme on their knowledge of a development process, through close co-operation, structured interviews, and hypotheses formulation. Formulation of hypotheses also prevents ending up in a situation where people mistakenly think they knew the outcome along.

Deliverable(s) 6.4: *List of measurement questions and hypotheses, defined with respect to the measurement goals*.

6.2.5 Step 5: Review questions and hypotheses

To make sure that the right questions and hypotheses have been captured and correctly formulated, they should be reviewed. The questions are the basic translation from goals to metrics. When the actual data will be collected and presented to a project team, it should help in answering the questions of the project team. So, the questions take a central role, not only during definition, but also during interpretation. Therefore, it is important to make sure that these questions are correct. Also, that the questions have been reformulated from the input of the project team during the interviews. During that translation it is possible that mistakes are made or that the GQM team makes misinterpretations.

The hypotheses should be reviewed as well, because together with the questions, the hypotheses are used to define the metrics that will be established for data collection.

Deliverable(s) 6.5: *List of approved measurement questions and hypotheses*.

6.2.6 Step 6: Define metrics

Once goals are refined into a list of questions, *metrics* should be defined that provide all the quantitative information to answer the questions in a satisfactory way. Therefore, metrics are a refinement of questions into a quantitative process and/or product measurements. After all these metrics have been measured, sufficient information should be available to answer the questions.

Furthermore, *factors* that could possibly be of influence to the outcome of the metrics should also be identified. After all, factors that directly influence metrics, also influence the answers to the questions that the metrics are related to. If the influencing factors were not to be considered during definition of a measurement programme, some conclusions or interpretations of the collected data may not be correct. These influencing factors are usually also defined as metrics.

Deliverable(s) 6.6: *List of metrics suitable for supplying information to answer the questions.*

6.2.7 Step 7: Check on metric consistency and completeness

The defined goals, questions, and metrics must be consistent and complete with respect to models of the object under measurement (see Figure 6-6). To safeguard this, consistency and completeness checks have to be performed throughout the entire definition phase. Whenever, during these checks, definitions appear to be missing, incomplete, or inconsistent, either the definitions have to be adjusted to comply with the process models, or the process models have to be adjusted to comply with the goal, question, and metrics definitions.

If product metrics are defined these should be checked as to whether they are in fact possible and whether they can actually be measured at a specific moment in the development process.

Deliverable(s) 7.7: *Consistent and complete definitions of questions and metrics related to the measurement goals. Process models that are consistent and complete with the measurement goals, questions, and metrics.*

6.2.8 Step 8: Produce a GQM plan

A GQM plan is a document that contains the goals, questions, metrics and hypotheses for a measurement programme as defined in the previous steps. The GQM plan serves as a guideline for data interpretation, and provides the basis for the subsequently developed *measurement plan* and the *analysis plan*.

The GQM plan describes the refinement from the measurement goals into questions and subsequently from questions into metrics. As some of these metrics may be indirect metrics, it also describes all direct measurements that should be collected for each indirect metric. The GQM plan is used for the following purposes:

- Explicit description of goals, questions, metrics and hypotheses. This way, the GQM plan represents the formal documentation of the measurement programme.
- Guideline for interpretation of collected data. As the refinement from goals into questions and subsequently questions into metrics is described in the GQM plan, the

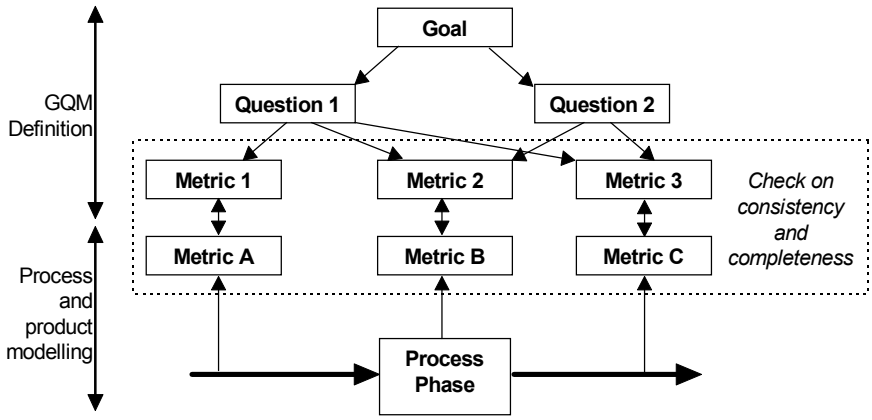


Figure 6-6: Checking on consistency and completeness (Solingen et al, 1995).

plan should also be used as a guideline for the interpretation of the measurement results by the project team. The measurement results should provide answers to the questions defined in the GQM plan, and once these questions have been answered, one should be able to draw conclusions regarding the GQM goals.

- Basis for definition of data collection procedures. As the GQM plan defines the metrics to be collected, it serves as the basis for the measurement plan, in which the data collection procedures are described.
- Basis for an analysis plan, that simulates and proposes data interpretation using graphics and tables before actual data has been collected.
- Basis for the measurement support system. The measurement support system that is described in the next chapter supports the GQM team in storing, maintaining, analysing and presenting measurement data. The GQM plan serves as the basis for the development of the part of the support system that analyses the collected data.

Deleverable 6.8: *Preliminary GQM plan*

6.2.9 Step 9: Produce measurement plan

A measurement plan describes the following aspects of each direct measurement that was identified in a GQM plan:

- It provides formal definitions of direct measurements.
- It provides textual descriptions of direct measurements.
- It defines all possible outcomes (values) of the direct measurements.
- It identifies a person that collects a particular direct measurement, ie a programmer, engineer, project manager, tester etc.
- It defines the particular moment in time when the person should collect the direct measurement.
- It defines by which medium (tool or form) that person should collect the direct measurement.

Furthermore, a measurement plan defines and describes both manual data collection forms and automated data collection tools. These forms and tools are described in more detail in Chapter 7 on page 65.

Deliverable 6.9: *Preliminary measurement plan.*

6.2.10 Step 10: Produce analysis plan

An analysis plan is a document that simulates data interpretation according to the GQM plan before actual measuring starts. Simulated outcomes of the metrics, graphs and tables are presented in this document that are related to the questions and goals as defined in the GQM plan. The analysis plan already gives an indication to the project team which charts they can expect to receive. Valuable adjustments in this material, but also in the GQM plan, can be expected from delivering such an analysis plan.

The main purpose of an analysis plan is to describe how the relevant measurement information is processed in such a way that it can be easily interpreted by the project team. An important starting point are the baseline hypotheses of the project team members which can be compared with the actual data. The analysis plan contains graphs and tables, which

validate the hypotheses and, furthermore, contains descriptions of how the corresponding numbers of the variation factors are going to be used in the results. The illustrations are developed according to the questions in the GQM plans.

The analysis of the data will consist of an analysis of the hypothetical data, the actual data and the data of the variation factors, which possibly could influence the metrics. In this way it can be analysed if the estimated impact of a variation factor is a major, minor, or insignificant factor for the goal under investigation. Furthermore, hypothetical and actual data can be compared, which preferably should be interpreted and explained by the persons who provided the hypotheses and the data for the metrics (top-down refinement and bottom-up interpretation).

Again it is emphasised that the data should be presented in such a way that interpretation by the project team is facilitated. It is important that the interpretations regarding the data, are only guided by the GQM team. By developing an analysis plan, a starting point for the development of feedback material is established, that can be presented to a project team to motivate them with respect to commitment to the measurement programme.

Deliverable 6.10: *Preliminary analysis plan.*

6.2.11 Step 11: Review plans

As the GQM plan, measurement plan and analysis plan represent the formal definitions of a measurement programme and describe all related data collection procedures, they should be reviewed and approved by a project team before data collection can actually begin. The review session should focus on:

- Do project members agree upon the defined goals, questions and metrics?
- Do project members identify any missing or unnecessary definitions?
- Do project members agree with the proposed definition of feedback material?

This means that review sessions focus more on the contents of the GQM plan and analysis plan, than on the measurement plan. The most important part of the measurement plan to be reviewed, is the part that describes the measurement tools and forms, because all project members should understand how to use these tools and forms. Preferably this is done, however, during the instruction sessions before the data collection and the trial data collection period (see Chapter 7). Once the three plans are approved by the project team and the GQM team the next phase can start: *the data collection phase*.

Deliverable(s) 6.11: *Approved GQM plan, measurement plan, and analysis plan.*

6.3 Modelling the software processes

6.3.1 Why model the development process?

The software development process is modelled to provide a more formal definition of the development of software. Such a model is essential to enable a structured development process and can also be used to identify and validate metrics. The development of a reference model is also emphasised in the quality improvement paradigm (Basili et al, 1994b).

A *task* in a software development process is defined as the lowest level of sub-processes in a software development process that creates a certain output. Even though a software development process is never the same, particular tasks are frequently executed, and often in the similar way. The software development process has the following characteristics:

1. The overall software development process consists of several sub-processes that each can be described as a software development process itself (recursion principle).
2. The lowest level of the software development process is a single task.
3. The order in which tasks are executed is not prescribed, and can not be defined beforehand.
4. Single tasks possess input and output products.
5. Single tasks in the software development process have certain properties.
6. Input and output products also have properties.
7. Properties of tasks, and input and output products, relate to certain metrics.

Through modelling the software development process to a level of many single tasks, a set of tasks becomes available with associated input and output products. These input and output products all possess particular measurable properties. Metrics can be associated with the input and output products. After modelling the entire software development process and all (intermediate) products, a comprehensive set of metrics is available for measuring the development of software.

Suitable techniques for modelling a software development process to a level suitable for measurement is the ETXM (Entrance criteria-Tasks-eXit criteria-Metrics) modelling technique (Humphrey, 1989), which is a refinement of the ETVX modelling technique.

6.3.2 The ETVX modelling technique

The ETVX model is used for modelling the phases and sub-phases of a software development process, and identifies four aspects in modelling an activity (Radice et al, 1985):

- A list of entrance criteria (E) that should be met before starting the activity.
- A set of tasks (T) that should be executed to perform the activity.
- A set of verification (V) tasks to ensure quality of the output of the tasks.
- A list of exit (X) criteria that should be met before the activity can be considered complete.

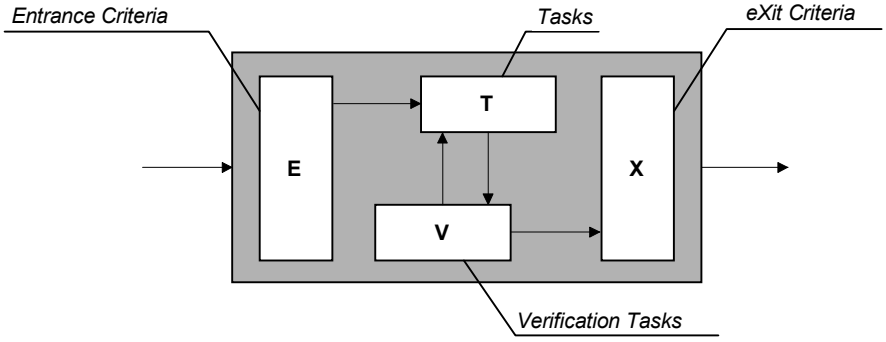


Figure 6-7: The ETVX modelling technique (Radice et al, 1985).

The ETVX modelling technique is illustrated in Figure 6-7. The principle idea is that a type of activity is defined by its entrance criteria, a task description and validation, and the exit criteria. The way the process is exactly executed cannot be defined beforehand, but the criteria that should be met can be defined. Input and output products of the development phase or activity are defined implicitly in the entrance and exit criteria. The sequential order of the tasks is not defined within the model. Changes in the process due to iterations, rework, work ahead, prototyping etc, can also be constructed with the ETVX modelling technique.

The ETVX model can be refined into any required level. A model can be made for the overall software development process, but also for phases, tasks, or sub-activities.

6.3.3 The ETXM modelling technique

At the lowest level of modelling a software development process, the ETXM model (instead of ETVX model) is used, because the verifications in ETVX are implied in the tasks that should be executed and measurements over the model are defined.

The ETXM modelling technique, based on the work of Humphrey, is used to model a single task (T) in an activity. It identifies entrance criteria (E) that have to be met before executing the task and exit (X) criteria that have to be met after execution of the task. Furthermore, ETXM identifies measurements on the entrance criteria, the exit criteria, and the task. This identification of measurements makes the modelling technique suitable for creating models to support the definition process of GQM-based measurement programmes. An example of applying ETXM to software development is shown in Figure 6-9.

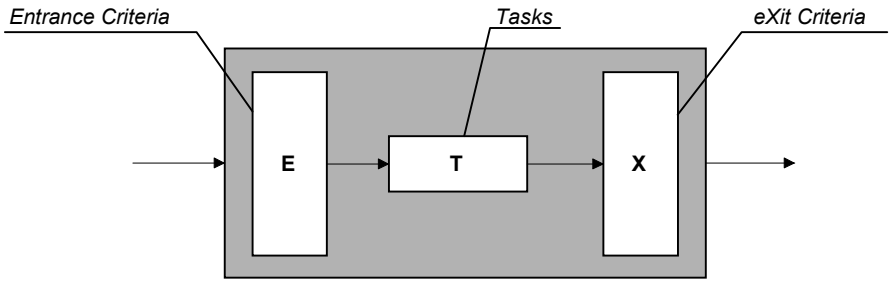


Figure 6-8: The ETXM modelling technique (based on Humphrey, 1989).

Software Program Implementation

Entrance criteria:
- Required detailed designs are available and approved
- Technical specification document is available and approved
- High level design document is available and approved.

Exit criteria:
- Code is available
- Code is written according to coding standards
- Code is written according to the procedures in the coding guidelines

Tasks:
- Coding
- Debugging
- Documenting code

Measurements:
- Size of Code (kB, LOC, number of Pages)
- Schedule time from start to finish (days, hours, start-date, finish-date)
- Resource time for coding (person-hours)
- Resource time for documenting code (person-hours)
- Number of revisions in the code
- Number of compile times
- Fault fixing data (time to locate a fault, time to fix a fault, cause of a fault, ...)
- Review data (person-hours, number of faults found, severity of faults found, amount of rework caused by review, number of people involved in review)
- (Part of) code reused (yes/no)
- (Part of) documentation reused (yes/no)

Figure 6-9: Example of ETXM modelling.

6.4 Questions and assignments

6.4.1 Questions

1. What are the 11 steps of a GQM definition phase?
2. What are the three deliverables of the GQM definition phase?
3. What are the five dimensions of a GQM goal template?
4. GQM measurement goals can be identified by answering ‘the seven questions’. Which are the seven questions? Are they always applicable?
5. What are the four sections of an abstraction sheet? How are they related?
6. What are the characteristics of a good GQM question?
7. What is the purpose of hypotheses?
8. What types of inconsistencies can be detected in step 7 of the GQM definitions phase? Also propose specific follow-up actions for each inconsistency type.
9. What can be used as a summary of a GQM plan?
10. What should be described for each metric in a GQM measurement plan?
11. In which case is the development of an analysis plan not necessary?
12. What is the purpose of reviewing a GQM measurement and analysis plan?

6.4.2 Assignments

Select one of the cases below. Take the one that suits your ideas best.

Case A: The objective of a GQM measurement programme is to identify a product's reliability before it is released on the market. In this case a payment terminal which is currently under development. The payment terminal contains 10 sub-systems which are concurrently developed in a period of one year.

Case B: The objective is to identify how you can save $3000 in one year. This will allow you to purchase a personal computer. However, it is not possible to earn additional money. Your income will be exactly the same as last year. You will have to find cost savings. Sources of data are, for example, your agenda, bank notes and salary receipts.

1. Define the GQM goal for this measurement programme according to the GQM goal template.
2. Define a set of GQM questions related to the measurement goal (maximum of 6 questions).
3. Check and improve these questions according to the criteria of good GQM questions given in step 4.
4. Refine your GQM Questions into metrics. Make a distinction between direct and indirect metrics.
5. If possible list your hypotheses regarding your questions (or metrics) and carry out some trial measurements from historical data to test the GQM tree you have established.

'Action may not always bring happiness, but there is no happiness without action'
Benjamin Disraeli

7 GQM data collection phase

Checklist GQM data collection phase		
No.	**Checklist item**	✓
7.1	Tools and forms are available	
7.2	Trial measurement period is held	
7.3	Tools and forms are updated	
7.4	Measurement kick-off session is held	
7.5	MSS metrics base is available	
7.6	MSS analysis sheets comply with GQM plan and analysis plan	
7.7	MSS presentation slides are created	
7.8	Data collection forms are completely and correctly filled in	
7.9	Validated measurement data are stored in MSS metrics base	

7.1 Introduction

This chapter describes the basic ideas of the data collection phase of a GQM measurement programme. It starts with data collection procedures and corresponding tools to perform the registration. Subsequently, a section is included at the actual start of collecting data which consists of a kick-off session, a trial period and training. The final section of this chapter contains details on the functionality and development of a measurement support system (MSS), which is required to efficiently run a measurement programme.

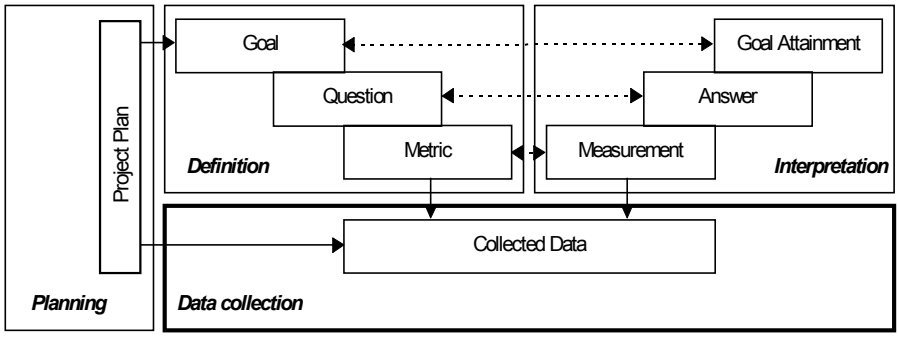


Figure 7-1: The data collection phase of the GQM method.

7.2 Data collection procedures

This section describes all details with respect to the data collection procedures of a GQM measurement programme. This includes the way in which procedures are defined, the way in which data collection forms are applied, and the way in which tools can support the data collection process. Examples of data collection procedures and form are given in the case studies of Chapter 9 to 12.

7.2.1 The need for data collection procedures

Data collection procedures define all the aspects that are necessary to carry out the data collection tasks. They are described in a measurement plan and concern the following aspects with respect to data collection:

- For a certain metric, which person should collect the data?
- When should the data be collected?
- How can the data be collected most efficiently and effectively?
- To whom should the collected data be delivered?

Based on the identification of these aspects for each measurement and the definition of corresponding procedures, tools such as paper forms or database applications are developed to support data collection.

7.2.2 Manual forms of data collection

Manual data collection forms are an easy, flexible and often applied tool in collecting GQM measurement data. This paragraph describes a number of prerequisites for developing such forms, and describes a way in which these forms can be used.

Manual data collection forms usually consist of a single page form on which data can be entered corresponding to the metrics defined in the GQM and measurement plan. To enable verification, a manual data collection form should always contain an entry field which identifies the originator of the measurement data. Furthermore, the form should be organised in such a way that they can be filled in sequentially, it should express all possible values of a measurement explicitly and it should contain a section in which the values are explained. Usually, a form also contains a section in which project team members can write down additional remarks, for instance, with respect to ease of use of a form.

The data collection forms should be filled in each time a particular event occurs that is being considered within the measurement programme. Frequently, preferably each day, the project team members submit their completed manual data collection forms to the GQM team, which enters the data into the measurement database.

Data collection forms are by no means static, but will evolve during the course of a measurement programme. Changes to a form, however, should always be supported by the project team. Changes should, of course, not only be processed on a form, but also in the GQM plan and the measurement plan.

7.2.3 Electronic forms of data collection

An efficient way of collecting data is by using electronic data collection forms which automatically handle the data entry activity. Several possibilities exist for electronic data collection forms of which we will discuss e-mails, web-pages and database applications.

Electronic forms require the same effort for project team members as manual forms, however, their advantage is that the data do not have to be re-typed into the measurement database. When on-line data collection is used, there is also the benefit that the measurement database is always up-to-date.

Electronic forms are an improvement over manual forms, because manual forms should be continuously available, distributed, updated, etc. While electronic forms can be centrally maintained, accessed by all members of the project team and transferred to the database more easily, project teams are often also used to sending e-mails and not to filling out forms.

E-mail data collection is normally easy to implement. An empty e-mail form should be created which can be filled in and be sent back to the GQM team. By having the data available in e-mail format there is the advantage for the GQM team that:

- tracking and checking of data collection are included in their daily process, since they will normally read their e-mail anyway;
- data do not have to be manually entered into the measurement database, because it can be entered using cut-and-paste functionality;
- implementation is not very time consuming, if the e-mail infrastructure is already available.

It is also possible to use spreadsheet forms for data collection, which can again be attached to e-mail. Transporting the data to the measurement database will then require even less effort than copying data from e-mail.

Another possibility is database driven data collection. This has as main benefit that the project team members store their data directly in the database which will be used for interpretation purposes. In this case, the GQM team spends less time in storing data. There is also the opportunity to provide on-line feedback material to the project team, because the data in the database are always up-to-date. However, there might be some problems with using a database tool, because in most cases it means that such a tool needs to be installed. Since this might include installing it on all computers the project team works with, this might be quite time consuming and will be more expensive. The time that project team members spend on data collection should also be limited.

Web-page data collection seems to combine the best of both e-mail and database data collection. On the one hand data collection is performed using an existing infrastructure that can be accessed by all members of the project team with already installed tools. On the other hand, data are stored directly into the database, which creates the benefits mentioned before. We advise you to set-up your data-collection based on this technology, which will also be quite beneficial within a distributed organisation. However, changes to electronic data collection forms are more time consuming than changes to manual data collection forms. Therefore it is advised to start with manual versions, or to use flexible database tools that can be changed easily.

7.2.4 Automated data collection tools

It is also possible to get support during the measurement programme from automated data collection tools, which calculate specific metrics according to some predefined algorithms. Examples are:

- Static analysers that can be used as automated data collection tools.
- Specific software metrics calculators. These tools calculate source-code metrics such as the number of code-lines, number of statements and possible paths in the structure of a programme.
- Other automated data collection tools, for example, CASE-tools, word processors, readability calculators, configuration management systems, workflow tools, or other automated calculators.

7.2.5 Restrictions to data collection tools

Use of automated data collection tools may be efficient, however, be careful and do not limit data collection to these tools. The most valuable information generally comes from people and not from tools. It is quite possible that project team members dislike data collection. They will then emphasise that only automated data collection tools should be used. Remember that goal attainment is the main purpose of each measurement programme, so whenever manual data collection is sufficient to answer a specific question, convince a project team that automatic data collection is unnecessary.

All data collection procedures, tools and forms should be defined and documented, and all people involved in the measurement programme should know how to apply them. Before data collection can actually start, people need to be instructed. This is done during the phase of data collection start-up and training, which is described in the next section.

7.3 Data collection start up and training

In this section the steps that need to be taken during the start-up of data collection and the associated training that needs to be organised, will be described. These steps are illustrated in Figure 7-2.

Before the actual data collection starts, a certain trial period should be held. After the improvements of the trial have been included in the data collection procedures, a kick-off session is organised in which both project-team and GQM-team start data collection. If necessary, additional training is organised on how to use the data collection forms or tools. In parallel with the actual data collection, the GQM team develops a Measurement Support System (MSS), which will become the basis of the interpretation phase. In the following sections our experiences regarding these activities will be described in more detail.

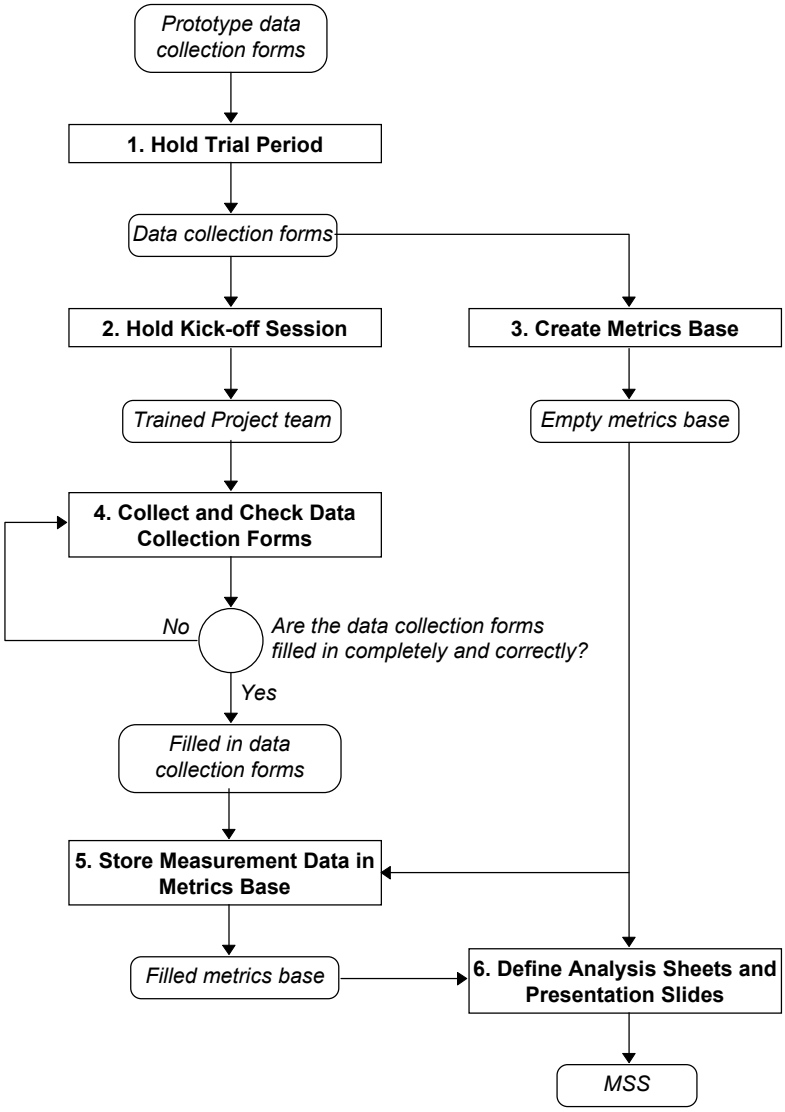


Figure 7-2: Data collection start-up procedures.

7.3.1 Trial period

Preceding the actual data collection period, a trial measurement period should be held during which the defined data collection procedures, tools and forms are tested. During this trial period, data collection is performed by only one or two people for only one or two days. It is advised to include at least one of the senior engineers, because a senior engineer can give more valuable suggestions and because the other engineers will be more confident

of the procedure if they know that one of their seniors has approved it. Based on these trial experiences, improvements will normally be necessary on the procedures, tools or forms.

Deliverable(s) 7.1: *Data collection forms.*

7.3.2 Kick-off session

Once all procedures, tools and forms are up-to-date, the GQM team organises a so-called 'kick-off session', during which all people participating in the measurement programme should be present. The main objective of this session is to get an agreement with the project team, that data collection will indeed start. During this session, once again the goals of the measurement programme are explained, and the project team members are instructed with respect to the data collection procedures, tools, and forms. At the end of the session, everybody has to agree that data collection can actually start.

Only if a large number of people participate in a project team, or if the procedures, tools, and forms seem complicated to the project team, may the GQM team decide to organise separate sessions with individual team members or small groups These smaller groups will then focus on training particular skills with respect to data collection.

Even when data collection has started, the GQM team should monitor the project team's use of the procedures, tools and forms. If any unfamiliarity's with, deviations from, or wrong usage of the procedures, tools and forms are observed, the GQM team should provide immediate additional support and training regarding those issues. Small problems tend to escalate during data collection.

Deliverable(s) 7.2: *Trained and motivated project team.*

7.3.3 Data collection activities

During the data collection period, filled-in data collection forms are gathered by, or delivered to, the GQM team on a frequent basis, preferably daily. The GQM team then checks the forms on correctness and consistently. If any mistakes are observed, corrective actions should immediately be taken. Completed and corrected forms are stored by the GQM team in a metrics base, that is created when the measurements start. As shown in Figure 7-2, the definition of a metrics base is the first step in the establishment of a measurement support system. The next section describes in detail how such a measurement support system (MSS) is built.

Deliverable(s) 7.3: *Completely filled in data collection forms.*

7.4 Building a measurement support system (MSS)

This section describes the development of a measurement support system. To enable the interpretation of the gathered data the measurement support system is an essential element in the measurement programme. It is advised to set up a proprietary MSS for every measurement programme[2]. Basis of an MSS will often be generic tools such as spreadsheets, statistical tools, database applications, and presentation tools. These tools

[2] One commercial GQM measurement tool is available. It is called 'MetriFlame', and information can be obtained from the internet page: http://www.ele.vtt.fi/docs/soh/metriflame/index.html.

normally provide sufficient flexibility to change measurement forms, charts, or tables. And that they will change during the course of a measurement programme is absolutely certain.

7.4.1 The need for an MSS

The measurement support system (MSS) supports all measurement activities. These activities are:

- Collecting measurement data, which means registration of the data by a project team.
- Storing measurement data, which means entering data into the measurement database.
- Maintaining measurement data, which means, for example, performing corrective changes.
- Processing measurement data, which means for example, combining, sorting, and dividing data to provide the required metrics.
- Presenting measurement data, which means the visualisation of metric data in an interpretable format, such as tables or charts.
- Packaging measurement data, which means the storage of data for expected presentation in the future. This can be during presentations, seminars, or conferences, but also for application in other measurement programmes with a similar measurement goal.

Particularly, flexibility and accessibility are important features of an MSS. Flexibility is required to adapt the MSS to the changing demands of the measurement programme without loosing earlier captured data. Accessibility will be crucial during the review sessions of the measurement data that will be organised later on in the measurement programme. The MSS will enable the project team members to 'play' with the measurement data and receive answers to any type of question regarding the measurement data.

In the next section the functionality of the MSS with respect to supporting these activities is described.

7.4.2 Development of an MSS

An MSS should preferably be built in parallel with the development of the measurement programme. However, as the basis of the MSS is provided by the GQM plan, the MSS will, in most cases, not be built earlier than data collection starts. The MSS is built by the GQM team because the resources spent by the project team on the measurement programme should be limited.

We always use a spreadsheet tool and a presentation tool. The spreadsheet tool is used to store, maintain and process the collected measurement data. The presentation tool is then used to prepare the presentation of the measurement results.

The reason for using a spreadsheet tool is its flexibility. Measurement programmes evolve over time, as measurement goals may be adjusted, added or removed. Spreadsheet tools provide the flexibility necessary to support these kinds of adjustments. The presentation tool should support both preparation of overhead slides and preparation of fully automated presentations using a data projector. Such an on-line data projector is essential to enable 'going into the data' during future feedback sessions.

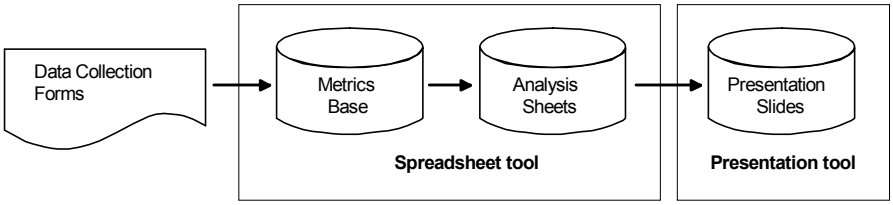


Figure 7-3: Building blocks of a measurement support system (based on Kooiman, 1996).

The MSS consists of three basic parts: a database for storage (metrics base), a data analysis part for abstracting data (analysis sheets), and a data presentation part for expressing information in charts and tables (presentation slides). These building blocks are illustrated in Figure 7-3 and will subsequently be described.

MSS metrics base

The metrics base can be implemented in a spreadsheet tool and contains all raw data as collected from the data collection forms. The data are entered into the metrics base by the GQM team. Checks need to be performed on correctness and completeness of the data. The hypotheses that were defined by the project team and described in the GQM plan, should also be incorporated in the metrics base. The metrics base should be established as soon as possible to support immediate processing of the first data by the GQM team.

MSS analysis sheets

The data in the metrics base are aggregated in analysis sheets. An analysis sheet is built in three layers with different levels of abstraction (Kooiman, 1996).

The lowest level of abstraction is found in the *raw data layer*, as it only contains raw data copied from the metrics base. In the *processed data layer,* data from the raw data layer is processed to a level of abstraction suitable for creating graphs and tables. For this purpose, processing data includes selecting relevant data, calculating data and sorting data. The results of processing the data are stored in the processed data layer and provide the basis for

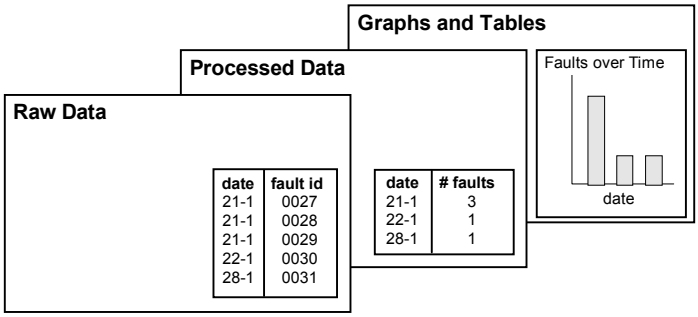


Figure 7-4: Three abstraction levels in an analysis sheet (Kooiman, 1996).

the *graphs and tables layer*. This graphs and tables layer transforms the results from the processed data layer into graphs and tables, suitable for presentation and interpretation.

The analysis sheets are created preceding the first feedback session. The basis for the implementation of analysis sheets is provided by the measurement programme's GQM plan. As a GQM plan defines the relationships between goals, questions and metrics, it also defines the framework for analysing the measurement data. To support data processing towards analysis and interpretation of measurement data, the analysis sheets should be structured according to the relationships defined in the GQM plan.

For each measurement goal, a separate spreadsheet-file is used to create an analysis sheet. It is not recommended to use one single spreadsheet file for processing data with respect to all the measurement goals, as processing time increases significantly if large amounts of data are concerned. By using single files for single goals, processing time will normally be within acceptable limits.

Each analysis sheet consists of the following worksheets:

- A *goal sheet* that describes the goal and the related questions, as stated in the GQM plan.
- A *data sheet*, that contains all data relevant to answering the questions with respect to the goal, including the hypotheses. These data are all copied directly from the metrics base into the analysis sheet.
- Several *question sheets*. For each question relevant to the goal, a separate question sheet is created. On the question sheets, the raw data is processed and transformed into graphs and tables. Therefore, the questions sheets represent the implementation of the processed data layer and the graphs and tables layer.

MSS analysis slides

After the analysis described in the previous section, the graphs and tables from the graphs and tables layer of the analysis sheets need to be copied to a presentation tool. In this file, the graphs and tables are prepared for subsequent presentation.

Consistency between the presentation tool and the graphs and tables in the analysis sheets can be maintained with automated links between the different files. This way, any change in the analysis sheets will automatically result in updating the presentation slides.

Data distribution

Data distribution concerns the distribution of interpreted material to all people involved in the measurement programme. As will be explained in the next chapter, we particularly encourage interactive sessions to discuss measurement results. However, it might be necessary to provide specific measurement information to be used in daily decision making to specific persons, for example, a project manager or a test engineer. On-line feedback of measurement data will then be necessary. Examples of questions that are supported by such a system are:

- Which sub-system should be tested first?
- What is the project progress if expressed in percentage inspected?

All parts of an MSS can be developed on different quality levels. It is advised to build it in an iterative way, allowing a low cost start and continuous improvement.

7.5 Questions and assignments

7.5.1 Questions

1. What are the six steps of the data collection start-up?
2. Which two types of data collection forms exist? What are their differences?
3. What is the purpose of the data collection trial period?
4. What is the purpose of a Measurement Support Systems (MSS)?
5. Which three typical levels exist in an analysis sheet of an MSS?
6. Which measurement activities are supported by an MSS?

7.5.2 Assignments

1. Design a data collection form of the GQM tree defined in your assignment of section 6.4.2.
2. Does your assignment contain possibilities for electronic data collection? If so, indicate how.
3. Make a list of advantages and disadvantages of collecting data electronically in general, and in your particular case.

'The important thing is to not stop questioning'
Albert Einstein

8 GQM interpretation phase

Checklist GQM interpretation phase		
No.	**Checklist item**	✓
8.1	Feedback material is consistent with GQM plan	
8.2	Presentation slides are correct and up-to-date	
8.3	Presentation handouts are distributed among project team	
8.4	All GQM and project team members are invited for feedback session	
8.5	Accommodation and equipment has been reserved	
8.6	Feedback session reports are distributed among participants	
8.7	Measurement results are reported	

8.1 Introduction

In this chapter the data interpretation phase is described. An essential phase, because in this phase we will try to find answers to the questions underlying the measurement programme. It is still remarkable for us to see how companies sometimes invest large amounts of money into a measurement programme without ever using the collected data.

In the previous chapter, the data collection phase was described. This phase included all activities required to actually gather, store and process the measurement data. In the data interpretation phase we will focus on drawing conclusions regarding the measurement programme. Of course, these conclusions will be specific for every measurement programme, however, the process and conditions that enable a productive interpretation process will be very similar. Therefore, we will focus on these issues in this chapter.

Results of a measurement programme are discussed in so called *feedback sessions*. It is our experience that the success of a measurement programme is primarily determined by the quality of these feedback sessions. Particularly, maintaining a constructive and goal driven attitude by all participants is a delicate element.

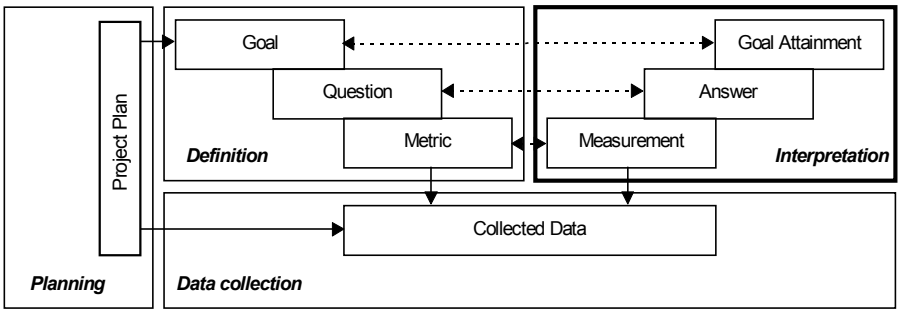


Figure 8-1: The interpretation phase of the GQM method.

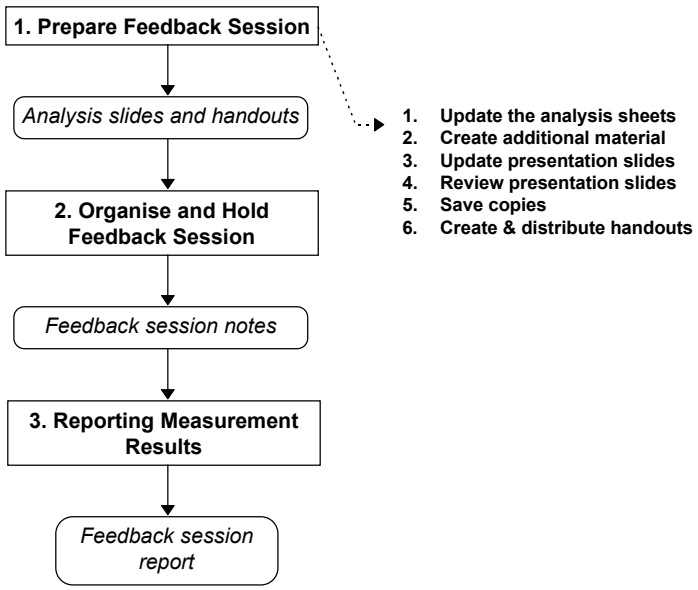


Figure 8-2: GQM interpretation procedures.

In the following sections, our suggestions for organising feedback are described. First, the activities for preparing a feedback session are described. Then, the activities for organising and holding feedback sessions are discussed. The last section in this chapter concerns reporting results of the measurement interpretations. This outline is illustrated in Figure 8-2.

8.2 Preparation of a feedback session

Preparing feedback sessions concerns processing the collected data into presentable and interpretable material. The GQM plan provides the basis for preparing feedback sessions: feedback material should support answering the questions as defined in the GQM plan, and based on these answers, one should be able to conclude whether the defined measurement goals are attained. The analysis should also be supported by the measurement support system (MSS).

The GQM team primarily does the preparation of feedback sessions. However, some involvement of (representatives of) the project team will often be beneficial. A guide to the first set-up of the MSS can be the analysis plan of the definition phase in which a first set-up of the data interpretation is documented.

A six step procedure, which is pretty straightforward, is used for the preparation of feedback sessions:

1. Update the analysis sheets in the MSS
2. Create additional feedback material
3. Update presentation slides
4. Review presentation slides
5. Save copies of slides and metrics base
6. Create and distribute handouts

These steps will subsequently be discussed.

8.2.1 Step 1: Update the analysis sheets of the MSS

Preceding the first feedback session in a measurement programme, usually a large number of feedback material has to be created, requiring a lot of effort from the GQM team. To minimise the effort necessary for preparing following feedback sessions, the analysis sheets in the measurement support system should support automatic updating of the feedback material. One should consider excluding measurements that significantly deviate from all other data. If possible, include the original GQM question in the slide that the particular chart purports to answer.

Deliverable(s) 8.1: *Updated analysis sheets.*

8.2.2 Step 2: Create additional feedback material

Feedback material should, of course, answer the GQM questions of the project, and address relevant issues. As questions and needs may change during a measurement programme, additional feedback material should sometimes be created. Creation of additional feedback material should, of course, never lead to abandoning the original goals and questions.

Deliverable(s) 8.2: *Additional analysis slides.*

8.2.3 Step 3: Update presentation slides

The graphs and tables of the presentation that have a link to the analysis sheet of the MSS can be updated automatically.

Deliverable(s) 8.3: *Updated analysis slides.*

8.2.4 Step 4: Review presentation slides

It is advised to present 15 to 20 slides at most in a single feedback session. Therefore, the selection of slides will often require significant effort. The slides that are selected should of course be free of errors to avoid incorrect interpretations. It is therefore advised to have the presentation reviewed by another GQM team member and a project team member.

Deliverable(s) 8.4: *Approved analysis slides.*

8.2.5 Step 5: Save copies of slides and metrics base

It is advised to store the measurement data and related feedback material for future use (not only those selected for interpretation). To prevent automatic updating, these copies should not contain any link to the analysis sheets.

Deliverable(s) 8.5: *Copy of metrics base and copy of analysis slides*.

8.2.6 Step 6: Create and distribute handouts

Handouts and large tables containing measurement data are hard-copied and distributed several days before a feedback session. This way, the people participating in the session can study the material before the actual session, resulting in a more effective and efficient feedback session.

Deliverable(s) 8.6: *Hard-copies of support material and hard-copies of feedback material.*

8.3 Holding a feedback session

Feedback sessions are meetings of all project team and GQM team members in which the measurement results are discussed. Feedback sessions are held approximately every six to eight weeks. They typically last about 1.5 to 2 hours, with a maximum of 3 hours. Any longer is normally experienced as counter-productive. This time is sufficient to discuss some 15 to 20 slides (containing graphs and tables).

Organising the session includes arranging a meeting room suitable for group sessions and all the required facilities necessary for presenting the feedback material, such as a computer, data projector and/or overhead projector. Furthermore, all people participating in the measurement programme have to be invited to the feedback session, and, if more than five people participate in the session, a moderator has to be appointed within the GQM team in advance, who will present the material and lead the group discussion. As well as the role of the moderator, there is a clear distinction between the role of the GQM team and the role of the project team during the feedback session.

In principal, a project team should run a feedback session alone. They analyse, interpret and draw conclusions regarding the measurements, and translate their conclusions into action points. After all, they are the experts with respect to the object under measurement. The project team should focus on:

- evaluating action points from earlier sessions;
- interpreting measurement data with respect to the questions and goals as defined in the GQM plan;
- translating interpretations into conclusions and action points.

The GQM team should avoid interpreting the data themselves, as they are not the experts. Their role is to challenge a project team, for example, by offering alternative interpretations (Solingen et al, 1997). Furthermore, the GQM team will make notes of the feedback sessions. These notes provide the basis for the meeting report that is created afterwards.

The moderator presents the slides to the project team using automated tools (computer with data projector) or an overhead projector, and leads the group discussions. During this discussion, the moderator should keep the attention focused on discussing the group's performance, and not the performance of individual group members. The moderator should also make sure that a common language is used during the feedback session, so the different attendants of the sessions all understand the issues involved. Feedback sessions generally have the following outline:

1. Opening by a member of the GQM team.
2. Overview of the GQM paradigm: a short presentation on the principles of GQM.
3. Detailed overview of the measurement object: the process or product under measurement and the related measurement goals are described.

4. Presentation of graphs and tables: graphs and tables relevant to answering the questions defined for the goals are presented to the project team for interpretation.
5. Evaluation: not only is it evaluated whether the project team can act upon the measurement results, ie formulation of action point, the measurement programme is itself evaluated for the purpose of improving all measurement related activities.

Deliverable(s) 8.7: *Session notes containing observations, interpretations, conclusions, and action points.*

Feedback sessions are a delicate phase in a measurement programme. Mutual trust among all participants is, for example, an essential element of a feedback session. Through focusing on identified goals, questions and metrics, the discussion will start on the basis of facts. It is the role of the moderator, and if necessary of the whole GQM team, to make sure that the discussion remains focused on facts and improvement actions. Make sure that measurement analysis does not become focused on individual performances. For more tips on data analysis we refer you to (Goodman, 1993).

8.4 Reporting interpretations of measurement results

After the feedback session, the GQM team writes a meeting report containing all relevant observations, interpretations, conclusions and action points that were formulated during the session. This report should, of course, be distributed to the project team. Be careful with submitting all measurement details or feedback session reports to higher management, because they often misinterpret data. It is advised to stay with the rule that the project team 'owns' the measurement data and therefore decides on distribution of both data and reports to management. When the GQM team wants to inform higher management, the GQM team only uses particular, often aggregated, results and asks permission to do so.

In order to reuse measurement results and experiences in future measurement programmes, the results of a measurement programme should be documented in such a way that they are easily accessible and understandable. It is difficult to perform effective packaging, because future needs for measurement information are usually not known in advance. However, some suggestions can be made on how organisations can deal with packaging measurement results.

Furthermore, the most important results of the measurement programmes should preferably be disseminated to the entire organisation to create awareness. This can be achieved by presenting the results of the measurement programmes on bulletin boards or posters in the organisation. Another approach is to make the measurement results available in an electronic way, such as an intranet.

Deliverable(s) 8.8: *Feedback session report and measurement programme packages.*

8.5 Cost and benefits analysis of a measurement programme

Goal attainment is, of course, an essential element of the success of a measurement programme, however, an analysis of costs and benefits should also be included in the final report of a measurement programme. In such a cost/benefit analysis, it should be evaluated whether the estimated benefits exceed overall costs.

Typical costs of a measurement programme are:

- time spent to prepare a measurement programme by GQM team (salary and overhead);
- time spent in meetings by project team;
- time spent to fill in data forms by project team;
- time spent to develop MSS;
- purchase of additional hardware and software to support measurement programme;
- time spent to process measurement data and prepare feedback sessions by GQM team.

Typical benefits are:

- additional sales because of improved quality;
- avoidance of a decrease in sales because of improved quality;
- time savings of software development effort because of a better understanding of development process;
- cost savings through better resource management;
- cost avoidance through better resource management.

A cost/benefit analysis is often experienced as difficult to perform, however, the contrary is true. The objective of a cost/benefit analysis is not to calculate the exact 'return on investment' of a measurement programme, but to identify whether the project was worthwhile from an economical perspective. Many people will claim that you can not value quality against cost. We strongly oppose this view. In most organisations, for example, the marketing department knows exactly how sensitive customers are to quality and what increases in turnover may be expected given additional ore improved product features.

Not including a cost/benefit analysis means that management and project team members will make their own implicit cost/benefit analyses. These analyses will often be based on less detailed information and your measurement programme might be abandoned because people think it is no longer worthwhile. Making a cost/benefit analysis is a responsibility of the GQM team.

Deliverable(s) 8.9: Cost/benefit analysis report.

8.6 Questions and assignments

8.6.1 Questions

1. Which six steps are discerned to prepare a feedback session?
2. Why is the interpretation phase the most important phase of a measurement programme?
3. What are the objectives of a feedback session?
4. What is a typical agenda of a feedback session?
5. Which typical cost and benefits should be considered for the cost/benefit analysis of a measurement programme?

8.6.2 Assignments

Feedback sessions are considered to be the most crucial part of a measurement programme. They are also the most expensive part (see Chapter 4). Try to identify alternatives for a feedback session that have the same effect, yet require less effort. An example of an alternative would be to place posters in your organisation. Do so by:

1. Identifying the effects that will occur during a feedback session in which measurement results are openly discussed.
2. Identifying alternative ways to communicate feedback material.
3. Plot effects and ways to communicate in a two-by-two matrix and evaluate the effects of particular ways to communicate.

PART 3

Cases

'For every complex problem,
there is a solution that is simple, neat, and wrong'
H L Mencken

9 Case A: reliability measurement

9.1 Description of project A

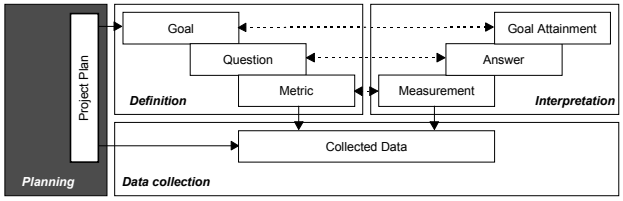
This chapter describes a project during which product reliability was measured. The impact of reuse on product reliability was also taken into account. This project, referred to as 'project A', was the first GQM based measurement programme for the particular organisation.

Project A aimed at developing both software and hardware for a real-time low-end cashing system. It's final product was designed for service station personnel and their managers giving access to site control and management information. This project was a second (incremental) release of the system and a considerable part of the software was reused from an earlier release as well as from other projects. At the end of the project, product A contained over 70,000 source lines of C-code. The project team consisted of a project leader, two hardware engineers and two software engineers. This project had a duration of 2 years.

The results presented in this chapter are based on the whole measurement programme. Therefore, results consist of measurement data, conclusions drawn by the project team during feedback sessions, and actions taken which were based on the measurements. In this chapter the four GQM phases of project A will subsequently be described, being, planning, definition, data collection, and interpretation.

This chapter also includes documentation of the particular measurement programme, such as GQM plan, data collection forms and feedback charts. This documentation can be reapplied to other measurement programmes with a similar goal.

9.2 Planning



This particular measurement programme started with the establishment of a GQM team. Head of the team was the Quality Assurance manager, assisted by a quality engineer. Their objective was to start project measurement in a structured way, to support projects with information to manage product quality. Goal-oriented measurement had to be used, because earlier measurement experiences had failed, probably caused by a lack of clear goals.

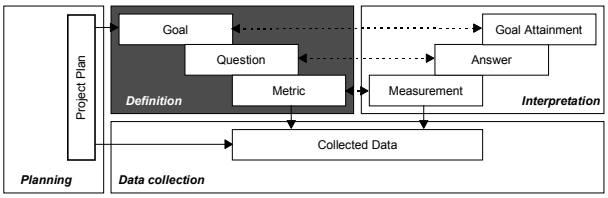
Based on the current status of product quality in the organisation, it was decided to focus on reliability measurement. This was done for several reasons:

- 'Reliability' was the most important quality requirement for that system.
- Practical relevance of reliability measurement was clear to the project team.
- Comparison of results with other projects was possible because of the international setting of this measurement programme.
- This project was also a try-out of GQM measurement. The GQM team also had to learn about such programmes, therefore the improvement area selected was not the most ambitious one. Success of this initial programme was therefore more likely to be achieved, which supports in spreading measurement further through the company.

This particular measurement programme was introduced by applying the QIP paradigm, because the GQM method presented in part II of this book did not yet exist at that time. A project plan was established that consisted of the six steps of QIP, starting with 'characterisation' up to 'packaging'. Both GQM team and project team received training on GQM concepts. Corporate management was frequently informed and formulated the initial objective to start with measurement in a goal-oriented way.

Both the organisation and the project were characterised by filling out a characterisation questionnaire. This questionnaire included both quantitative and qualitative questions. The results of the characterisation were used to create a project plan for this measurement programme. The characterisation questionnaire was also applied during the selection of improvement areas and measurement goals.

9.3 Definition



Based on the Project plan, the measurement programme was officially started. During several meetings with the Project manager a selection was made from the possible measurement goals based on the project objectives and the results of the characterisation. The definition process of the reliability measurements is presented in this section. Two measurement goals were defined for this programme.

Reliability:

Analyse the:	delivered product and development process
for the purpose of:	understanding
with respect to:	reliability and its causes
from the viewpoint of:	the project team
in the following context:	project A

Reuse:

Analyse the:	delivered product
for the purpose of:	understanding
with respect to:	effectiveness of reuse
from the viewpoint of:	the project team
in the following context:	project A

Achieving those measurement goals would yield a better understanding of reliability and reuse. All members of the project team were consulted for definition of these goals and they

were highly interested in reaching them. Management was involved as well and illustrated the relation of the measurement goals to the business goals.

The most important activity of the definition phase was the refinement of the selected goals into questions and metrics. This refinement was divided into two parts. First, a knowledge acquisition part, aimed at understanding the current knowledge within the project team and representing this knowledge in quantitative or qualitative models. Second, a measurement planning part, which involved documenting the Goal/Question/Metric refinement and corresponding measurement procedures.

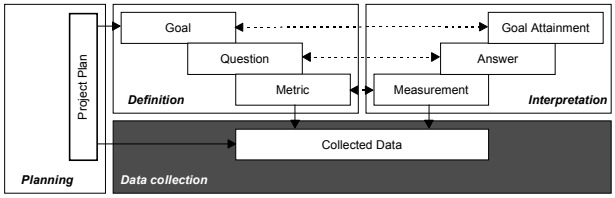
Knowledge was acquired through structured interviews as was advised in Chapter 5. The main purpose of these interviews was to make knowledge of the project members explicit. During the definition phase of this measurement programme interviews were held with all project team members. A report was written of every interview and returned to the interviewee for review. This interview report already contained a first abstraction sheet. As the project team was familiar with the abstraction sheet concept, this made it easier for them to review the interview report.

The interviews were also intended to motivate project team members for the measurement programme. The team members gave valuable input and realised that the data to be collected was related to issues raised by themselves. The result of the definition phase was a well-defined measurement programme, documented in the following three deliverables:

- The GQM plan which defined in a top-down way what would be measured. It defined the goals and the refinement into questions and metrics. The GQM plan of this measurement programme is attached in Section 9.6.
- The measurement plan which described how the measurements should be done. It described metrics and procedures and media to report, collect and validate the data.
- The analysis plan that defined how to close the feedback-loop by providing verified and validated measurement results to the project team on a proper abstraction level. Being a basic guideline to support feedback of measurement information to the project team.

The GQM team, based on input from the project team, wrote all three plans. The plans were reviewed by members of the project team, mostly by the project manager or a senior engineer. The GQM plan was reviewed thoroughly to obtain a standard terminology and to eliminate obscurities and ambiguities in the plan. We used several short follow-up interviews, and the entire team pursued an extensive review. It was essential to communicate that the GQM plan was going to be *the* basis for analysis and interpretation: insufficiencies here would lead to problems with drawing conclusions from the data.

9.4 Data collection



Data collection started with a ‘kick-off session’ in which the data collection forms were explained. The metrics defined in the measurement plan were largely covered by an existing defect tracking tool that the project team already used. A manual paper form was designed to record several additional metrics on fault handling.

The measurement data was collected according to the procedures defined in the measurement plan. Before measurement data could be stored in the measurement support system, it has to be thoroughly verified and validated (see Section 7.4 on page 70 for details of an MSS). In this measurement programme the GQM team performed the following checks (in cooperation with the project team):

- completeness of the data;
- timeliness of the data;
- accuracy of the data;
- whether the data is within range;
- correctness of classifications.

No procedures were defined for the further handling of the data by the GQM team. In practice, this handling was supported by the MSS. This tool was used by the GQM team for saving, maintaining and processing data of measurement programmes. The MSS was implemented in a spreadsheet tool, because of its flexibility. Experience had shown that measurement programmes evolve over time and analyses were not static. A flexible MSS appeared to be an important requirement, which caused spreadsheets to be highly applicable.

For this measurement programme a customised MSS was implemented. The implementation is divided over several files. Consistency between the files is accomplished by using automatic links between the different files. Two files were created for this measurement programme: a metrics base and analysis base (Figure 9-1). The analysis sheets that are stored in the analysis base were printed and photocopied onto transparencies for presentation on an overhead projector.

Data collected by the project team was entered into a database, the 'metrics base', by the GQM team. While entering the data into the metrics base a manual check on correctness and completeness was performed. Hypotheses were also included in the metrics base (stated in the GQM plan). The metrics base was created before actual measurement started. This way, the gathered data could be stored in the metrics base as soon as measuring started. The analysis sheets of the data were implemented in a separate analysis base. The analysis sheets were created preceding the first feedback session.

Two analysis databases were created in association with the two identified measurement goals. The analysis sheets were linked to the metrics base to assure consistency between the files (changes in the metrics base are automatically updated in the analysis sheets). One analysis sheet consisted of different worksheets for the questions that relate to the goal of

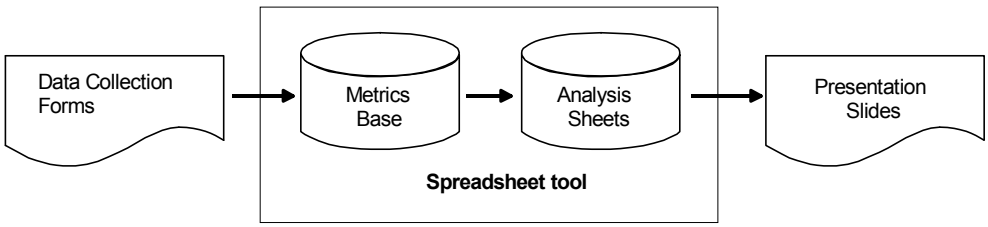


Figure 9-1: Processing data collection into presentation slides via the metrics base and analysis sheets.

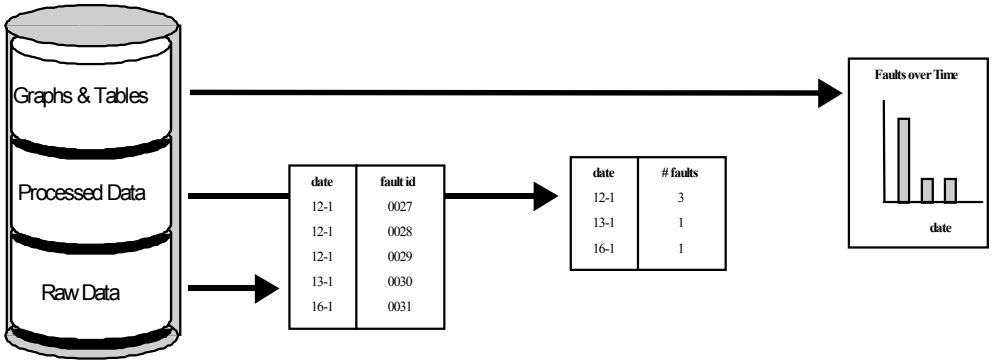


Figure 9-2:Three layers within an analysis sheet.

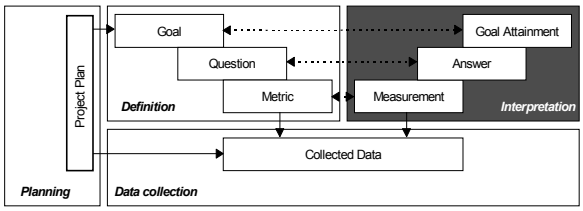
the analysis sheet (question worksheets). Within an analysis sheet, three different levels of abstraction for the data were distinguished. These levels of abstraction were called layers. Each analysis sheet was built up from three layers (Figure 9-2):

- raw data layer;
- processed data layer;
- graphs & tables layer.

The raw data layer contained links to the metrics base and, therefore, like the metrics base itself, contained raw data only. To be able to create graphs and tables it was often necessary to process the data in one way or another to get a suitable result. Processing data included, for instance, selecting particular data, calculating data and sorting data. The results of processing the raw data was kept in the processed data layer. Finally, the last layer contained the data as they were presented. These graphs & tables data layer also contained the questions from the GQM plan, which the project team members intended to answer.

Through connecting the processed data layer with the graphs & tables layer, this layer was updated automatically. Processing raw data into graphs and tables was carried out with the GQM plan as a guideline: the graphs and tables should support answering the GQM questions. This was already partially described in the analysis plan, however, additional analysis sheets appeared to be necessary during the course of the measurement programme.

9.5 Interpretation



In this section the measurement results of project A are presented. During project A's measurement programme more than 4000 data points were collected. In total, eleven feedback sessions were held. The final feedback session for this measurement programme took place two years after starting the measurement programme. In the final meeting few

new results were presented. However, the original measurement goals were officially attained by the project team during this final feedback session.

The two goals of the measurement activities were to *understand* product and process reliability, and to *understand* the effects of reuse. The results of the measurements were not only presentations of empirical data regarding these subjects, but also conclusions drawn or actions taken by project team members based on measurement information.

A selection of conclusions drawn and action taken by the project members based on the measurement results is presented here:

- Modules with high user interaction were expected to produce much more failures whenever a single fault was detected and were, therefore, reviewed quite intensively.
- Novice-user-tests appeared to be very effective and should be held more often. Novice-user-tests were also immediately introduced in another project.
- Complexity of a module appeared to be a cause of faults. This conclusion was based on the correlation between fault density of modules and cyclomatic complexity (McCabe's complexity metric). The measurement programme also showed a relation between fault density of modules and length of a module.
- Engineers found 55% of all failures. This percentage was expected to decrease when more field releases were done and engineers stopped testing.
- The test group found 25% of all failures and half of all minor failures. The project team concluded that the test group was stricter while testing and that this promoted continuing independent testing. Engineers appeared not to test on minor bugs.
- The project team identified that most fatal failures detected during testing were not likely to be found in a structured review. This is in contrast with literature on this subject that claims that approximately 60% of all faults can be found during reviews (Humphrey, 1989). Identifying the effectiveness of reviews was however NOT a measurement goal, so one should be really careful in drawing such conclusions. The opinion might be caused by an unclear view on that topic.
- Paper proposals for enhancement request appeared to cause most requirement faults, so this was not the right communication medium. No empirical results of the measurements reflected that this was based on the communication medium. Therefore, the possibility remained that the enhancement request had simply not been correctly written down. The project team already had the opinion that enhancement requests were often not clear, and repeated their statement supported by the measurements.
- Availability of the right test equipment increased detection of faults before delivery. This conclusion was based on an identification by the project team that most failures found after release were caused by a different configuration than available during test. Whether it was possible to create a test installation that covered all possible field configurations needed further investigation.

At the end of project A, conclusions towards the goal were drawn. The project team reached understanding on the reliability of the product, but since this understanding was required all the time, measurement was continued. However, the goal could be changed from 'understanding reliability' to 'control reliability'. The contribution of GQM in this measurement result was significant, because the focused measurements provided specific information that could easily be translated into practical improvements.

GQM was introduced by strict application of the QIP, which caused some minor problems because particularities of the development process of project A were not taken into account during the definition phase. If software development models would have been

used as a reference (as illustrated in Figure 3-4), the measurement programme could have been even more beneficial. Now the testing phase was initially not taken into account and the impact of testing on the reliability goal was not considered. Afterwards it was concluded that, therefore, the causes of fault detection could not be analysed fully.

The remainder of this chapter presents feedback material which supported answering questions, and goal attainment.

9.5.1 Product overview

In presenting the measurement results, the first two slides served an introductory purpose. This first slide in the feedback session offered a detailed overview of the modules in which faults were detected (Figure 9-3). A slide like this gives an overview of the entire product on unit level: the most important attributes for the stated goals were presented in one table. It answers several questions from the GQM plan like:

- Q.1: What is the composition of the product?
- Q.3: What is the complexity of the delivered software
- Q.9: What is the distribution of faults after delivery
- Q.13: What percentage of the delivered code was covered by peer reviews?
- Q.21: What is the distribution of modules among reuse modification classes?

It was found to be an important aid in tracking measurement data back to their cause: trends and distributions in graphs and tables can often be explained by considering other metrics. The overview acts as a reference in this consideration.

Module	# Faults	Fault ranking	Modification faults	Original faults	Reuse	Review	Size (KSLOC)	Complexity
Module 1	17	1	10	7	More than 20%	yes	4.56	624
Module 2	13	2	8	5	More than 20%	yes	3.10	560
Module 3	12	3	4	8	More than 20%	yes	4.10	624
Module 4	8	4		8	No reuse	no	2.10	207
Module 5	8	4		8	No reuse	no	0.68	210
Module 6	7	6		7	No reuse	yes	2.10	245
Module 7	6	7		6	No reuse	yes	0.50	70
Module 8	5	8	3	2	More than 20%	yes	4.68	420
Module 9	4	9		4	No reuse	yes	0.46	55
Module 10	4	9		4	No reuse	yes	0.45	8
Module 11	4	9		4	Less than 20%	no	0.39	39
Module 12	3	12	1	2	Less than 20%	yes	1.80	303
Module 13	3	12		3	No reuse	no	2.24	309
Module 14	3	12		3	No reuse	no	1.20	150
Module 15	3	12		3	No reuse	yes	1.20	132

Figure 9-3: First feedback slide with an overview on all data.

Total product size = 60.9 KSLOC
System fault density = 1.9 faults/ksloc

Data collection was carried out during 22 months

Trouble reports	Number	Real	Hypothesis
Failures	133	81%	50%
Change requests	31	19%	50%
Total	164	100%	100%

Number of detected failures = 133

Number of repaired faults = 118

Total effort for locating faults = 183 hours
Total effort for fixing faults = 138.4 hours
Effort per fault = 2.7 hours

Figure 9-4: Second slide in feedback session: global overview.

9.5.2 Overview on data collection

The next step in presenting the measurement data, was to give an overall impression of the product under measurement. The slide in Figure 9-4 presented a global overview of the product, including total product size, system fault density and number of analysed trouble reports. Often, project members have so much inside knowledge of the project that it becomes difficult for them to maintain a general overview of the product. A slide like this offered an overview.

9.5.3 Q.6: What is the distribution of failures after delivery?

Figure 9-5 illustrates the amount of failures reported by project A. The number of failure reports on the product approached zero. This chart did not only reflect reliability of the product, but also detection events that were executed during the development, for example in April of the first year a Novice User Test had been executed and therefore the number of failure reports was much higher. Another increase was visible in November, where the first field release was done. During the integration phase several failures were detected. The relatively low number of failures during July and August of year 1 reflect the summer holidays during which no detection activities were executed. Please note that this emphasised the need to let the project team interpret measurement data, because an outsider might have concluded that the product was becoming reliable.

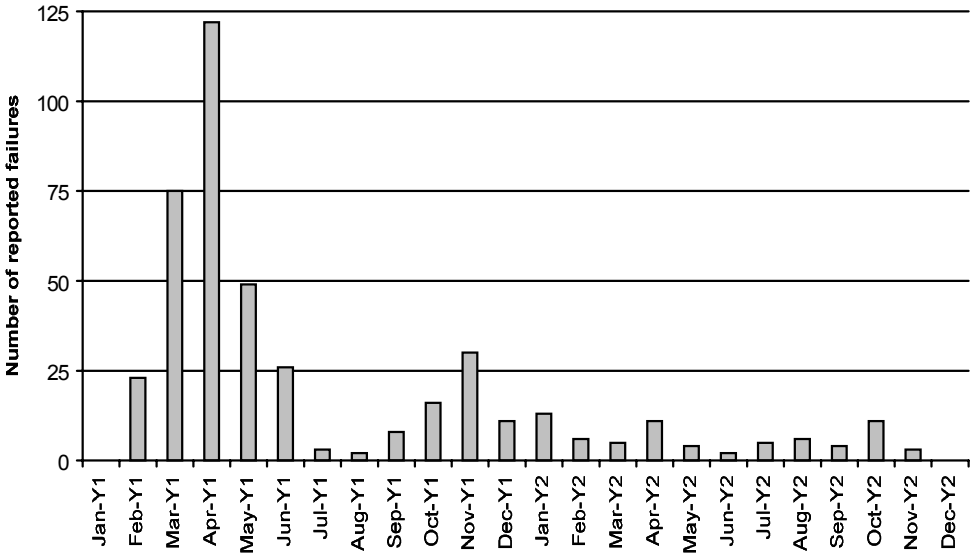


Figure 9-5: Number of failure reports on product under development.

Drawing conclusions on the current reliability of the product only based on Figure 9-5 was not possible, because no information was included on the amount of effort spent on finding failures. Because this major aspect of reliability was not identified during the definition process of the GQM plan, conclusions regarding this issue could not be drawn. This again illustrated the need for process models as references during the definition phase. The version of the GQM approach which was applied for this project did not include process models as described in Section 6.3. Otherwise, the absence of measurements on testing were certainly detected, because testing (detection) is one of the major techniques to detect failures.

9.5.4 Q.7: What is the distribution of failures over severity classes?

Severity of a failure was considered an important aspect of reliability. For example, after release of a system to the field it is not acceptable to find Fatal failures, while some minor failures were acceptable (for project A). Therefore, the GQM plan contained a 'severity' metric which defined three classes for severity of a failure:

- Fatal failures. For example, system failed, system executed wrong transactions, system lost transactions.
- Major failures. For example, system refused legitimate transactions, or system produced redundant outputs with small impact on performance.
- Minor failures. For example, aesthetic problems such as misspelling or output formatting problems.

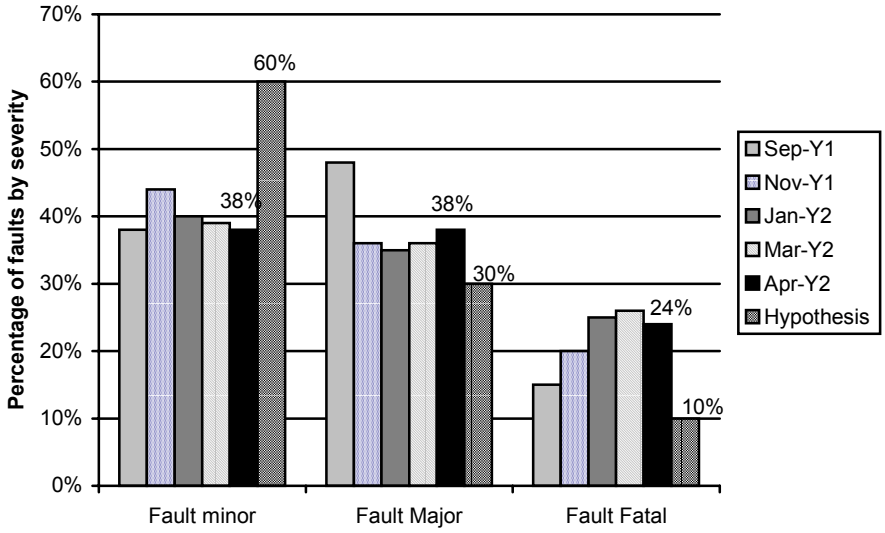


Figure 9-6: Number of faults per severity category.

Figure 9-6 illustrates the distribution of faults found over time per severity category. The hypotheses as the project team stated them in February year 1, are also presented in this chart.

The percentage of fatal failures was stabilising at around 25% while the expected number of fatal failures was 10%. The project team had now learned how failures were distributed by severity, and this knowledge could be used in future projects. These numbers could also be used in future projects to plan failure repair.

9.5.5 Q.9: What is the distribution of faults after delivery?

Historic overviews were popular in feedback sessions. A successful presentation of historic data was found at module level: the slide in Figure 9-7 presents the top 8 of fault-containing modules, and the accumulated number of faults detected over time. In this slide, both a graph and a table were presented. Experience showed that different people have different preferences when analysing data: some people preferred graphs, others preferred tables. Therefore, in most cases, both graphs and tables should be presented together. Although this might not always be possible.

9.5.6 Q.22: What was the relation between module reuse and reliability?

One of the measurement goals was to understand the effects of reusing software in project A. Please note that productivity increase was *not* taken into account during this measurement programme, because the project team decided to reuse software for reliability reasons only, not for productivity reasons. The software that was developed in this project contained a large amount of reuse. Figure 9-8, shows that the fault density (number of faults

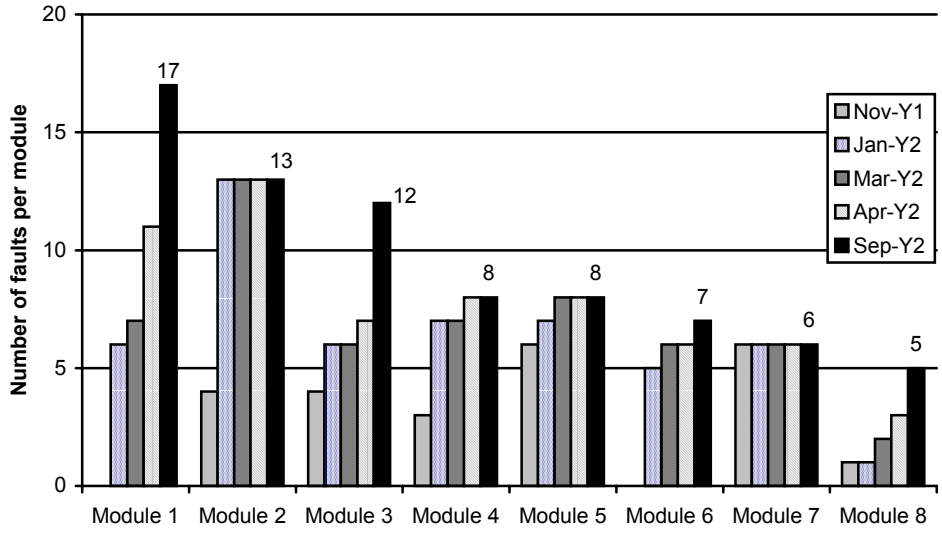


Figure 9-7: Project A top 8 fault containing modules.

per thousand source lines of code) was linearly decreasing as the amount of reuse increases. Also the amount of newly introduced faults decreased when the amount of reuse increases. When more than 20% of the code was reused this resulted in five time fewer new faults compared to complete new development. In all modules that were completely reused in the product, no faults were identified. The project team learned that it was beneficial to reuse (parts of) modules during development.

When reusing modules it was also possible to reuse faults that were already present in the code, so reuse could also be a cause for faults (whether this is negative was not clear, because it was also beneficial to identify faults in code that was already installed in the field). The project team concluded that reuse was not only a development approach, it was also a fault detection method! The relation between fault density and amount of reuse was almost linear in our measurement programme. However, there was also a relation between functionality and reuse, but this will be described in the next section.

The project team defined action to increase the amount of reuse in new development, and when a module was developed from scratch it was developed in such a way that it was more easily reusable. A concrete example that showed that the project team actually learnt from the measurements was that the project manager recently provided another department with certain modules they could reuse during their own new development. In the opinion of the project manager he would not have done that without the knowledge obtained in the measurement programme.

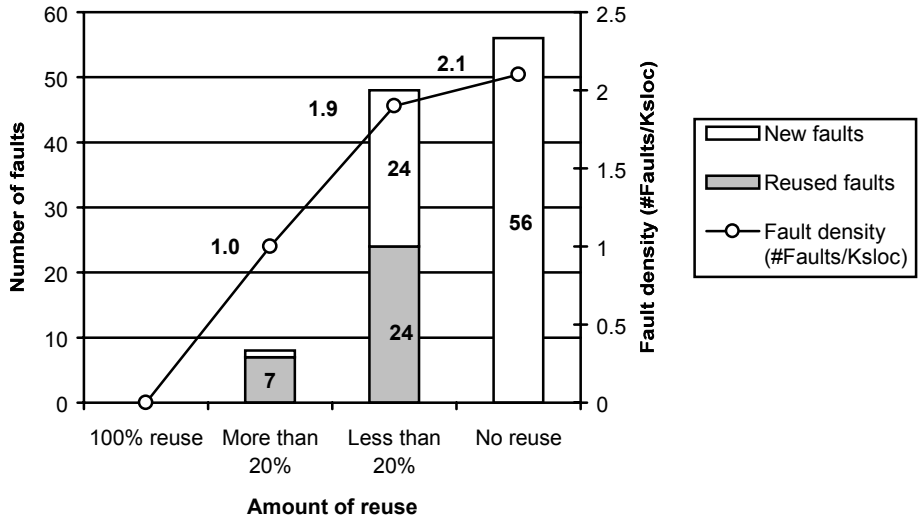


Figure 9-8: Fault densities of modules categorised by the amount of reuse.

Conclusions on reuse drawn by the project team:

- Reuse was a useably method for fault prevention, and detection.
- Most of the faults in partially reused modules were detected before release, which was caused by a lower confidence of engineers in partially reused modules than in personally developed modules.
- Reuse resulted in lower fault density.

These empirical results can be used to convince managers and engineers of the effectiveness of reuse, and can also be used to promote the reusability of code in the organisation. A new measurement programme could consider the productivity increase that is caused by reuse. The decrease of fault density caused by reuse was already identified in this measurement programme.

9.5.7 Q.12: What is the relation between module complexity and reliability?

Figure 9-9 is a complex chart. It shows at the x-axis the functionality categories of the product (as they are defined by the project team). The left vertical axis shows the size of the according software in SLOC (*Source* Lines Of Code). The amount of code is in the bars and also visualised by amount of change (reflecting in reverse the amount of reuse). For example the largest amount of 100% reused code (Unchanged) is in the console.

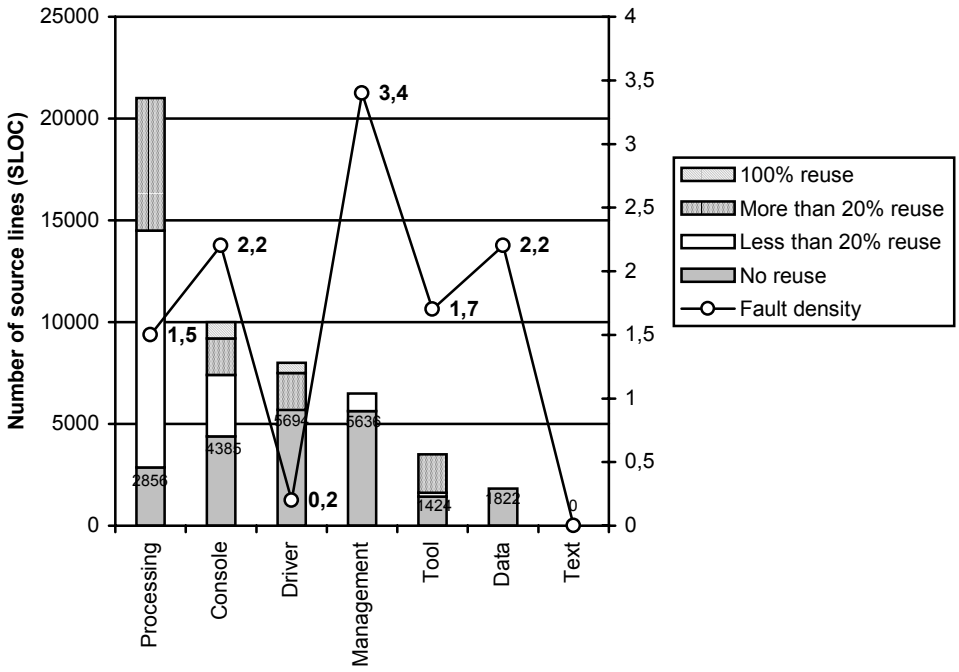


Figure 9-9: Software size per functionality and according fault density.

The right vertical axis reflects the fault density (in number of faults per KSLOC). The fault density appears to be related to the amount of new developed code (no reuse), looking at the processing, console and management functionality's. Those categories reflect the three major parts of the product. The drivers are different from this trend, because drivers are very well documented and therefore contain far fewer faults than average. The white numbers in the bars are the exact number of KSLOC for the no-reused modules.

Often, results are presented that seem to have important implications. It is important not to draw conclusions too easily. Such a premature conclusion may be drawn when considering Figure 9-10: this slide presents a number of metrics grouped by functionality. The code of the management functionality seems to be very prone to errors when considering the fault density in the last column. A closer look reveals the true cause of its high fault density: the relatively small size of the functionality causes the fault density to grow rapidly with every fault found. Also, the management functionality has a high user interaction, which causes a failure to be much more visible, and to be reported sooner. Closely analysing data and underlying data appears to be important in gaining valid interpretations. And again: this interpretation can only be done fully by the members of the project team.

Functionality	# Faults	% Faults	Size (KSLOC)	% Size	Fault Density (Faults/KSLOC)
Processing	47	43%	26.7	44%	1.8
Management	24	22%	7	11%	3.4
Console	18	16%	10.1	17%	1.8
Tool	8	7%	3.8	6%	2.1
Driver	8	7%	11.3	19%	0.7
Data	5	5%	1.9	3%	2.6
Text	0	0%	0	0%	0.0
TOTAL	110	100%	60.8	100%	
Average	16		8.7		1.8

Figure 9-10: Faults found classified according to functionality.

9.5.8 Q.8: What is the distribution of failures over detection mechanism?

Figure 9-11 reflects the percentage of failures detected by each detection mechanism. The hypotheses as they were put to the project team in February of the first year are also presented in this chart, creating the possibility for the project team to learn from fault detection.

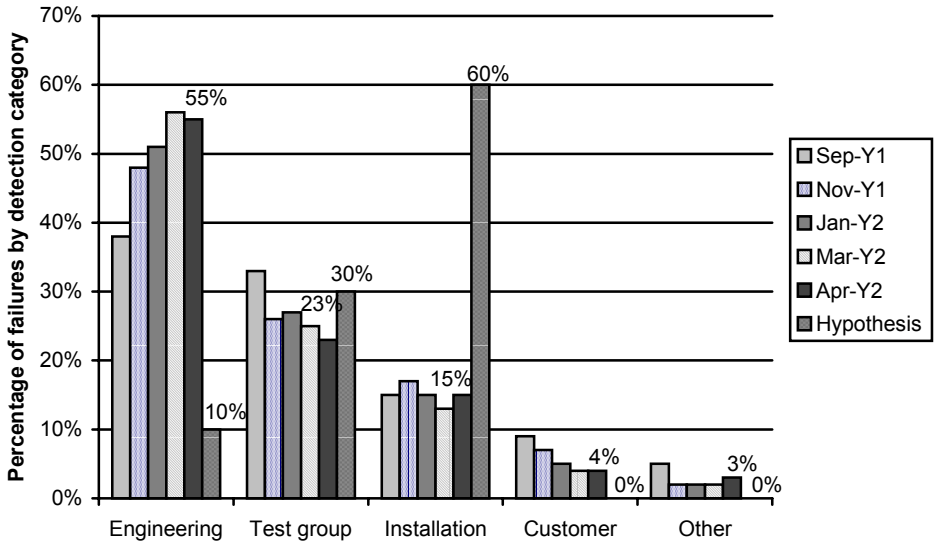


Figure 9-11: Detection efficiency of detection mechanisms (percentage of failures per category).

As visible in Figure 9-11 the amount of failures found by the installation department is still much lower than expected. The project team expects this number to increase when more systems are being installed. Most of the expected failures will be change requests. Conclusions and observations from the project team:

- Engineers find 55% of all failures.
- Test group finds 25% of all failures, but 45% of all minor failures (test group is more strict while testing).
- External testing is much more efficient than testing by engineers.
- Most fatal errors that were detected during testing were not likely to be detected in a review.

9.5.9 Types of severity by detection mechanism

Figure 9-12 reflects the percentage of types of severity of failures. For each detection mechanism it was presented how many of their detected failures were fatal, major or minor failures. As visualised, the test group found most minor failures. This was very positive since it reflected that the test group were more critical testers. Customer and other appeared to find only major failures and fatal failures, but as was already visualised in Figure 9-11 the amount of failures found was low. Note that customers tend to report only fatal and majors failures.

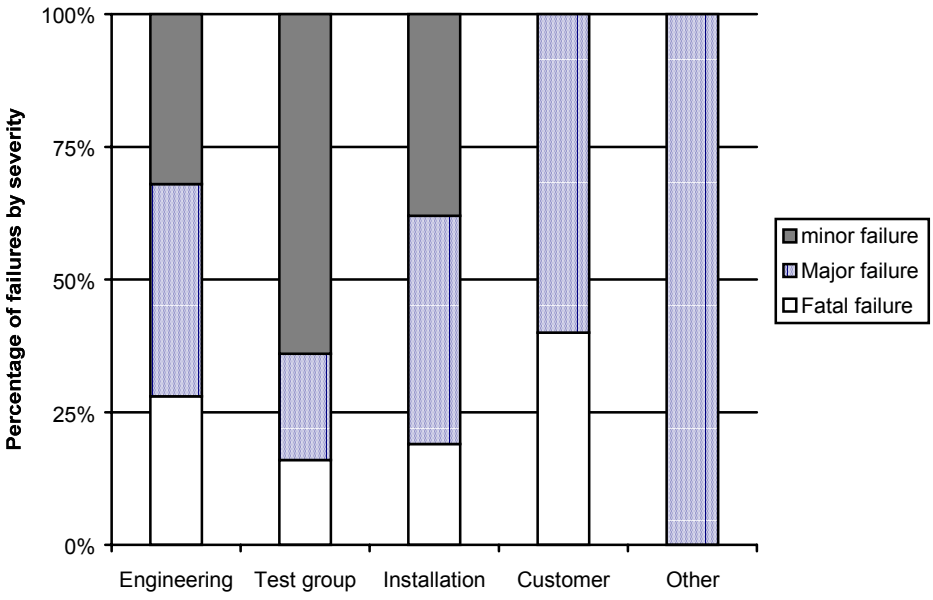


Figure 9-12: Percentage of failure severity found by detection mechanism.

9.5.10 Q.17: What is the distribution of failure handling effort?

Figure 9-13 shows the average effort for failure repair per development phase of introduction of the fault. It appeared that faults that were introduced in the requirement phase costed three times the average time to repair, while faults introduced during detailed design and implementation required 0.6 of average time to repair. No faults were introduced during evaluation and release. Better formulation of requirements and early customer involvement was concluded to be valuable with to respect to repair costs.

As Figure 9-13 shows, the effort needed to repair requirement faults is more than nine hours per fault. This number was used to promote the introduction of inspections during the requirements phase, since only two faults needed to be detected in an inspection of four hours with four people, in order to be beneficial. Additional research on the severity of faults during development phases, and effectiveness of inspections was also necessary, however, results in literature already identified high potential for increasing the efficiency of the development process by introduction of inspections.

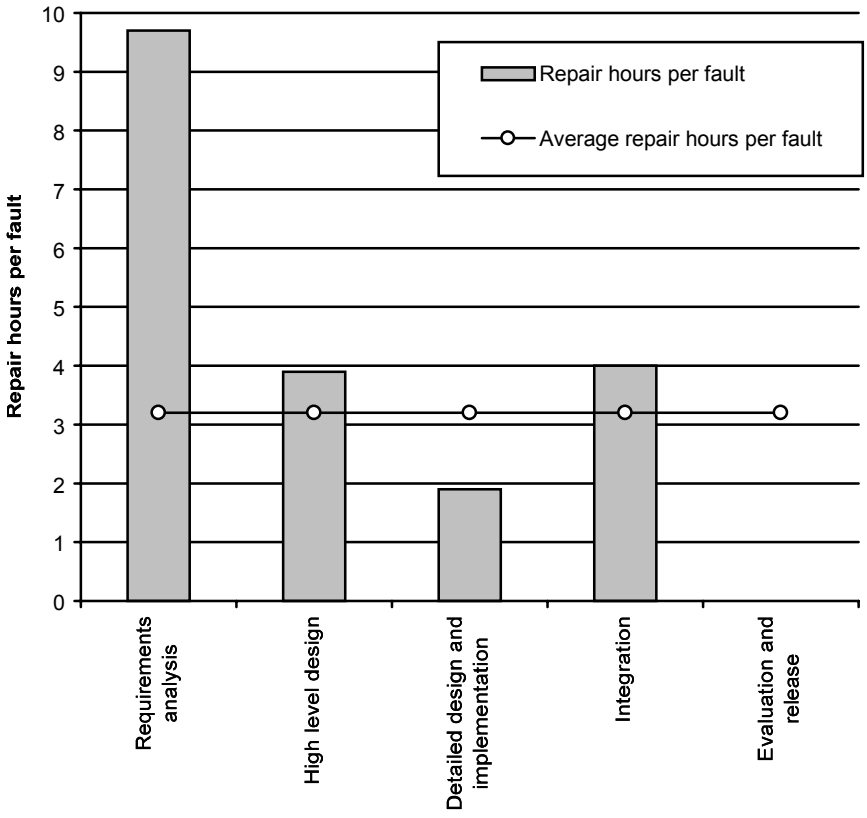


Figure 9-13: Average effort on repairing a failure, per phase of introduction.

Conclusions drawn and action taken by the project team with respect to repair effort:

- Requirement faults were expensive. This was caused by the fact that several requirements were not available in the original requirements document, and therefore had to be included later, when problems occurred.
- Early detection of requirement faults would pay itself back (requirements should be intensively reviewed). This was already concluded by the project team, identifying their readiness for detection methods on, for example, requirements and specification documents.
- Customers should be involved earlier during development. This conclusion of the project team was based on the fact that the marketing department was currently formulating the requirements. Early involvement of customers might cause other problems like unrealistic or non-profitable requirements. Further research should identify which approach could best solve these problems. A possible approach might be developing a prototype in order to identify missing, or incorrect requirements.
- The marketing department should be involved more frequently during development. This conclusion was related to the previous one. A project team was not always able to identify the requirements of customers.
- Finding requirement and high level design faults took less time than repairing them afterwards. This was a clear observation on the measurements, caused by the fact that missing functionality, or wrong interpreted specifications were directly clear as soon as the failure was reported, while implementation errors really needed to be recovered.
- Total effort spent on repairing requirement faults was equal to total effort spent on repairing implementation faults, however, there were six times more implementation faults.

Based on these numbers the project team decided to take several actions. They concluded that requirement faults took too much time, which caused planning delays. Therefore, the decision was made that all requirement faults would be considered to be ‘change requests’, which needed to be included in the planning of a next release. A major benefit from this was also that marketing was then responsible for the planning, so only those requirement faults would be repaired that had commercial impact.

9.5.11 Correlation of complexity versus size

When metrics showed a clear relation, the analysis could be limited to one of both metrics. In the case of project A, many metrics were related to size and cyclomatic complexity (see for example Figure 9-14). By showing that there was a relation between the two, analysis could be limited relative to either size or complexity.

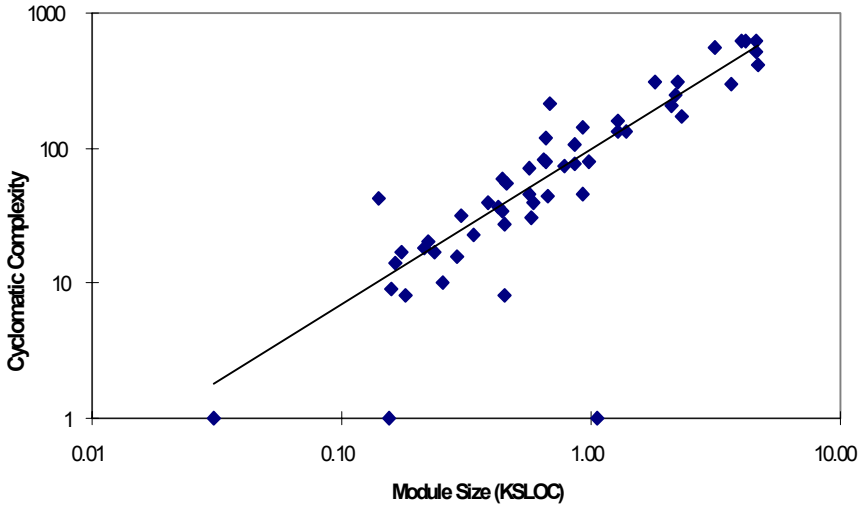


Figure 9-14: Relation between module size and cyclomatic complexity in project A: opportunity to limit the analysis.

9.6 Documentation of project A

This section contains a number of essential documents of the project A measurement programme. Examples are the GQM plan and a feedback session report. These documents are given as an example. Please note that project A was the first measurement programme undertaken. Therefore, the documentation also includes mistakes, that resulted in some recommendations included in part II of this book.

The feedback session report presented in this section presents the results of the final feedback session of the project A measurement programme.

9.6.1 Project A GQM plan

Introduction

Background

The *M*easurement *I*nformation *SYS*tem (MISYS) is a partly automated system, that contains a metric base, a database, historical base and procedures for data processing. The metrics, the data and the procedures for reporting, collecting and verifying data were defined according to the GQM paradigm. This section describes the GQM plans and the metric plan for the pilot project for measurement, project A.

Selected application project

To set up a GQM plan, the members of the project team of project A were interviewed as well as the software development leader of a project which was going to reuse some of the software developed by project A. Note that the hardware development leader and the SW development leader also worked as engineers.

Measurement goals

It was defined that two GQM goals would be incorporated: one for analysing the reliability of products and one for analysing the development processes with respect to reuse.

Goal 'reliability':	
Analyse the	delivered product and development process
for the purpose of	better understanding
with respect to	reliability and its causes
from the viewpoint(s) of	the software development team (SW project leader and software engineer)
in the following context:	project A

Goal 'reuse':	
Analyse the	delivered product
for the purpose of	understanding
with respect to	effectiveness of reuse
from the viewpoint(s) of	the software development team (SW project leader of a project developing for reuse and SW engineer of another project reusing software)
in the following context:	project A

Goal 'reliability'

GQM abstraction sheets

With respect to the 'reliability' goal, two interviews were made, one with the software development leader of project A and one with the software engineer of the project.

First, the summarised information was presented in a GQM plan abstraction sheet, which was derived from the abstraction sheets, which represented the two interviews. In the interviews the implicit knowledge of engineers and managers was made explicit. Therefore, questions were asked regarding their definitions of the quality focus, and the classifications within those definitions. Then questions about quantification of the expectations regarding those definitions were asked, the baseline hypothesis. Followed by questions regarding factors concerning the process and the product itself, that could influence the expected numbers from the baseline hypotheses, the variation factors. And the way those factors could impact the baseline hypotheses. The abstraction sheet is given in Figure 9-15.

Object: product and process	**Purpose:** better understan-ding	**Quality focus** reliability and its causes	**Viewpoint:** SW development team	**Environment** Project A	**External** GQM plan Represen-tation

Quality Focus

# of Failures
- Overall
- By Severity (Minor, Major, Fatal, Other)
- By Detection (Engineer, Test Group...)

# of Faults
- Overall
- By Life-Cycle Phase
- By Modules

Cost: Effort in Hours
- Overall
- Cost By Module
- Cost By Activity
- Locating vs. Fixing

Variation Factors

Process Conformance:
- Are the reviews done as prescribed in the process model:

Code Reviews - 100%,
Code Inspection <100%
- Deadlines

Domain Conformance:
- Experience Level Engineers

Attributes:
- Adherence to Coding Standards
- Complexity
- Consistency of Documents

Baseline Hypothesis

Distribution of Failures by Severity:
- Minor: 30%
- Major: 15%
- Fatal: 5%
- Change Request: 50%

By Detection (Pre, After release, Customer)

• Engineer	80%	10%	10%
• Test Group	20%	30%	nap
• Installation	nap	60%	30%
• Customer	nap	nap	60%

Top 6 fault containing modules (ranked):
- Module 1, module 2, module 3, module 4, module 5, ….

Modules with most effort for fixing failures (ranked):
- Module A, module B, module C, module D, …..

Distribution of effort for fixing faults after delivery per introduced activities:

	# of faults	effort
• Req's analysis and spec:	10%	35%
• High level design:	5%	6%
• Design and implementation	60%	35%
• Integration	10%	6%
• Evaluation and release:	15%	18%

Distribution of Effort between Locating and Fixing a Fault:
- Locating 60%, Fixing 40%

Impact on Baseline Hypothesis
- Better Process Control results in:
  - fewer failures,
  - less faults slipped through code review,
  - % of coding faults is reduced
- Too severe deadlines will increase the # of failures
- Higher Experience SW Engineers results in less faults introduced
- Better adherence to coding standards results in:
  - less faults in general
  - less effort for locating and fixing faults
- Complex modules have more faults
- Not updated documents could result in more errors in a later phase of the project

Figure 9-15: Example of abstraction sheet project A.

GQM plan for product goal

The following goal was established regarding the particular software product:

Goal 'reliability product':	
Analyse the	delivered product and development process
for the purpose of	characterising
with respect to	reliability and its causes
from the viewpoint(s) of	the software development team (SW project leader and software engineer)
in the following context:	project A

In the subsequent paragraphs this goal will be decomposed into measurement questions and metrics.

Product definition

Attributes measured here were the composition of the product, the consistency of documents with the delivered product, adherence to the coding standards, and the size of the software.

Q.1 What is the composition of the product?
- M.1.1 List of life cycle documents.
- M.1.1 List of subsystems.
- M.1.1 List of modules.

A module is a part of a subsystem (SW system) with a defined functionality.

Q.2 Is the source code in accordance with the coding standards?
- M.2.1 Adherence to coding standards per module (high, low).

Q.3 What is the complexity of the delivered software?
- M.3.1 Overall size of software (KSLOC).
- M.3.2 Size of software per module (KSLOC).
- M.3.3 McCabe's cyclomatic complexity per module.
- M.3.4 Level of user interaction per module (high, medium, low).
- M.3.5 Change rate of the module (categories or numbers)?
- M.3.6 Personal assessment of the overall complexity per module (high, medium, low)?
- M.3.7 Combined complexity based on the individual complexity metrics per module.

We may later define a combined complexity metrics based on the relative importance of the individual complexity metrics.

- M.3.8 Combined complexity per functionality of the system (eg, dispenser).

Quality model 'failures after delivery'

Reliability of a product was modelled on the number and severity of failures detected after delivery, ie after installation of a pilot release. In the case of project A, a release implied:

- shipment of all up to date modules to the customisation group(s), which added the country specific software modules;
- delivery to Installation;
- commercial release of complete product to customer;
- delivery to customer.

To get more insight, and to identify improvement opportunities, a distinction was made between, faults, failures and defects. Defects was the most general term that was used to denote anything that went wrong. According to the IEEE standard definition, *errors* were defects in the human thought process made while trying to understand given information, to solve problems, or to use methods and tools. *Faults* were concrete manifestations of errors within the software, ie one error might cause several faults and various errors might cause identical faults. *Failures* were departures of the operational software system behaviour from user expected requirements. Thus a particular failure might be caused by several faults and some faults might never cause a failure.

To investigate failures further, numbers from the trouble reports were used, reported failures were separated from change requests, and each failure was tracked back to faults. Failures were classified according to severity and according to the detection mechanism. Faults were classified according to the life-cycle phase where the fault was introduced, according to subsystems where they were found, and according to the type of fault. Costs were broken down into costs per module, costs per activity (documents that were updated), and costs for fault locating versus costs for fault fixing. Model definition:

Q.4 How long did data collection take place?
- M.4.1 Number of months after delivery was used for trouble data collection.

Q.5 What was the percentage of failures reported in the trouble reports?
- M.5.1 Number of trouble reports after delivery.
- M.5.2 Percentage of trouble reports, not being change requests.

It was assumed that trouble reports addressed both failures and change requests. With respect to reliability the change requests were to be ignored.
Hypothesis: 50% of trouble reports addressed change requests.

Q.6 What was the distribution of failures after delivery?
- M.6.1 Overall number of detected failures for software system after delivery.

Q.7 What was the distribution of failures after delivery over severity classes?
- M.7.1 For each detected failure: classification by severity (minor, major, fatal, other).

Q.8 What was the distribution of failures after delivery, over detection mechanisms?
- M.8.1 For each detected failure: classification by detection mechanism (engineer (provide name), test group (provide name), acceptance test group (provide name), Installation (provide country), customer (provide name), other).

Q.9 What was the distribution of faults after delivery?
- M.9.1 Overall number of software system faults detected after delivery.
- M.9.2 For each fault detected after delivery of the software system: life-cycle phase the fault was introduced (requirements analysis, specification/high level design, detailed design/implementation, integration, evaluation/ release.

Note that the information with respect to life-cycle phases had to be treated carefully. Since the document frequently was not updated when requirements changed during the development process, all changes were implemented in the software code. Therefore, it happened that, for example, design faults were perceived as code faults.
Hypothesis: see abstraction sheet.

M.9.3 For each fault detected after delivery of the software system: name of the module the fault was located in.

Hypothesis: top 6 modules containing faults were module 1, module 2, module 3, module 4, module 5, module 6.

M.9.4 For each fault detected after delivery of the software system: name of the functionality the fault was located in.

Q.10 What was the distribution of failure handling effort after delivery?

M.10.1 Total effort for handling failures after delivery (locating faults and fixing them) in staff-hours.

M.10.2 For each failure detected in the software system after delivery: effort in hours for locating and fixing the fault(s) by module.

M.10.3 For each failure detected in the software system after delivery: effort in hours spent on activities related to various life-cycle phases (FEASIB, REQANAL, SPEC,. HLDES, DDES, IMPL, INTEGR, UDOC, EVALREL).

The idea was that if, for example, the high-level design document was to be updated because the failure was due to a design fault, effort was spent on an activity associated to the phase HLDES (high level design).

Q.11 What was the effort spent on fault locating vs. the effort spent on fault fixing?

M.11.1 For each fault detected in the software system after delivery: effort in hours for locating the fault.

M.11.2 For each fault detected in the software system after delivery: effort in hours for fixing the fault.

Q.12 What was the relationship between module complexity and reliability?

M.12.1 For all modules: what were the average complexity metrics.

M.12.2 For top faulty modules: what were the complexity metrics.

M.12.3 For top non-faulty modules: what were the complexity metrics.

M.12.4 For all modules: what was the fault density.

M.12.5 For top simple modules: what was the fault density.

M.12.6 For top complex modules: what was the fault density.

The complexity metrics included all the complexity metrics defined earlier in the product definition. The fault density could be replaced with another metric defining reliability if some other metrics was more descriptive for project A.

Process definition

Process conformance:

Q.13 What percentage of the delivered code was covered by peer reviews and inspections (with respect to number of modules and KSLOC)?

This question also applied to updates of the product, then the amount of modified source lines of code was taken into account (ΔKSLOC).

M.13.1 Total number of modules in operational software system.

M.13.2 Size of operational SW system (KSLOC).

M.13.3 Number of modules in the operational SW system whose code was peer-reviewed.

M.13.4 Aggregated size of peer reviewed modules in the SW (KSLOC).

M.13.5 Number of modules in the operational SW system whose code was inspected.

M.13.6 Aggregated size of inspected modules in the SW (KSLOC).

Q.14 What is the relationship between code reviews and reliability?

M.14.1 For all modules: what is the fault density.

M.14.2 For code reviewed modules: what is the fault density.

M.14.3 For non-reviewed modules: what is the fault density.

M.14.4 For all modules: what is the percentage that has been reviewed.

M.14.5 For top faulty modules: what is the percentage that has been reviewed.

M.14.6 For top non-faulty modules: what is the percentage that has been reviewed.

Domain conformance:

Q.15 What is the experience of the SW development team? (percentage of SW team members for each of the experience classes)

M.15.1 SW development team: classification of experience with application domain (high, average, low).

M.15.2 For each SW development team member: classification of experience with earlier phases of the same project (participation in directly preceding phase, participation in earlier phase but not the directly preceding one, participation in all preceding phases, no participation in earlier phases, no earlier phases of project existing). These are assumed to be variation factors. Hypothesis: higher experience results in fewer faults and failures.

Quality model 'failures during development'

To understand the causes for reliability the failures that are detected during development, ie in the life-cycle phases up to and including Evaluation and Release are investigated and tracked down to faults.

The issues addressed by the questions are the roughly the same as in the GQM for goal 'reliability1.product'. The difference is that here the failures detected during development are considered, and not the failures that are detected after delivery of the system.

Model definition:

Q.16 What is the distribution of detected failures before delivery?

M.16.1 Overall number of SW system failures detected before delivery.

M.16.2 For each detected failure before delivery: classification by severity (minor, major, fatal, other).

M.16.3 For each detected failure before delivery: classification by detection mechanism (engineer (provide name), test group, acceptance test group, other).

Q.17 What is the distribution of failure handling effort?

M.17.1 Total effort for handling failures detected (locating faults and fixing them) in staff-hours.

M.17.2 For each failure detected in the SW system: effort in hours for locating and fixing the fault(s) by module.

M.17.3 For each failure detected in the SW system: effort in hours spent on activities related to various life-cycle phases (FEASIB, REQANAL, SPEC,. HLDES, DDES, IMPL, INTEGR, UDOC, EVALREL).

The idea is that if, for example, the high-level design document is to be updated because the failure was due to a design fault, effort is spent on an activity associated to the phase HLDES.

Q.18 What is the effort spent on fault locating vs. the effort spent on fault fixing?

M.18.1 For each fault detected in the SW system before delivery: effort in hours for locating the fault.

M.18.2 For each fault detected in the SW system before delivery: effort in hours for fixing the fault.

Goal 'Reuse'

GQM abstraction sheets

With respect to the 'reuse' goal, two interviews were made, one with the software development leader of project A (who works as an software engineer as well) and one with the software development leader of a related project which is to reuse software developed by project A.

First, the summarised information is presented in a GQM plan abstraction sheet, which is derived from the abstraction sheets, which represented the two interviews. Note that the abstraction sheet in Figure 9-15 was filled out only partially, because the project team had no experience with reuse to provide sound expectations (hypotheses).

Object: delivered product	**Purpose:** understanding	**Quality focus** effectiveness of reuse	**Viewpoint:** software development team	**Environment** Project A	**External** GQM plan represen-tation

Quality Focus

Reuse of Software, any kind of document or code (core development only)

- Degree of modification:
  (unchanged, part of document deleted, part of document changed (up to 20%), part of document changed (more than 20%) which should be an exception)
- Reused artefacts: Life Cycle documents or code
- Defects and reuse:
  - Faults in reused artefact vs. faults introduced when modifying artefact
  - Faults in reused modules vs. faults in non-reused modules
- Cost of reuse:
  Overall, per component, vs. cost for new development

Variation Factors

Process Conformance:

- 1st or later phase of a project (if the project is phased)

Domain Conformance:

- Experience of the development team with reuse or the reused software
- Is developing for reuse done, for example appropriate structures, coupling and object oriented techniques

Figure 9-16: Summarised GQM plan abstraction sheet project A.

GQM plan for 'reuse goal'

The following goal was formulated regarding reuse of code:

Goal 'reuse':	
Analyse the	delivered product
for the purpose of	understanding
with respect to	effectiveness of reuse
from the viewpoint(s) of	the software development team (SW project leader of a project developing for reuse and SW engineer of another project reusing software)
in the following context:	project A

In the subsequent paragraphs this goal will be broken down into measurement questions and metrics.

Quality model 'software reuse'

Here the reuse of software for core development is investigated; The questions are related to the following issues: document types that are reused (code or documents) and their origin, amount of reuse overall and for certain subsystems of the developed system, degree of modification when reusing components, defects in reused components, cost of reuse.

Model definition:

Q.19 What is the percentage of modules that were not developed from scratch, ie some kind of document or code is reused (overall and per subsystem)?

- M.19.1 For each software module: developed from scratch (Yes, No). Hypothesis: 70 %.
- M.19.2 For each software module: name of subsystem the module belongs to.

Q.20 What is the percentage of reused code (overall and per subsystem)?

- M.20.1 For each SW module: code reused (Yes, No).

Hypothesis: 70 % of modules some degree of reuse

- M.20.2 For each SW module: name of subsystem the module belongs to.

Q.21 For SW modules where code is reused: What is the distribution of modules among reuse modification classes (overall and per subsystem)?

- M.21.1 For each SW module where code was reused: degree of code modification (code is unchanged, mainly deletions, less 20% of lines changed, more than 20% of lines changed).

  Hypothesis: 30% of reused modules are unchanged, 10% of reused modules require deletions only, 30% of reused modules require changes in up to 20% of KSLOC, 30% of reused modules require changes in more than 20% of KSLOC.

  This classification can be calculated with the configuration management tool. Therefore 2 metrics must be added to determine the versions of the original and the released modules.
- M.21.2 For each SW module where code was reused: name and version of the original reused module.
- M.21.3 For each SW module where code was reused: name and version of the released reusing module.
- M.21.4 For each SW module where code was reused: name of subsystem the module belongs to.

Q.22 What is the relationship between module reuse and reliability?
- M.22.1 For all modules: what is the level of reuse?
- M.22.2 For top faulty modules: what is the level of reuse?
- M.22.3 For top non-faulty modules: what is the level of reuse?
- M.22.4 For all modules: what is the fault density?
- M.22.5 For reused modules: what is the fault density?
- M.22.6 For non-reused modules: what is the fault density?

9.6.2 Project A feedback session report

In this section project A's final feedback session report is given.

FEEDBACK SESSION REPORT			
Project Name: Project A	Project No: QAxxxxxx	Date:	Rev: 00.02
Meeting Date:		Present: xxxxx	
Authors: xxxxx		Absent: xxxxx	
Subject: Final feedback session of the GQM based measurement results for project A			

Introduction

In February year 1 a measurement programme was started at the R&D department. The goals of the measurement programme were defined as:

Analyse:	the delivered product and development process
for the purpose of :	understanding
with respect to:	reliability and its causes
from the viewpoint of:	the software management team
in the context of:	project A

Analyse:	the delivered products
for the purpose of :	understanding
with respect to:	effectiveness of reuse
from the viewpoint of:	the software management team
in the context of:	project A

This is a report of the final meeting during which the results of the measurement programme were presented by the GQM team and the results were interpreted by the project team and GQM team according to the questions and goals as stated in the GQM plan.

It appeared that the GQM plan as available is a design document of the measurement programme, and difficult to apply as a basis for interpretation. The GQM team has therefore interpreted the questions in the GQM plan in a way that the feedback material supports reaching the Goals.

Results of feedback session on 'reliability'

The meeting started with an overview of the project A software product:

- 52 modules;
- 60 KSLOC;
- system fault density: 1.9 Faults/KSLOC;
- average repair time: 2.7 Hours/Fault.

The average time needed for repairs is not used to schedule repairs. Instead, about 70% of time is scheduled for normal tasks and 30% is reserved for other work like fault finding and fixing.

The distribution of trouble reports was 81% failure reports and 19% change requests. The hypothesis made was that this distribution would be 50-50%. The reason for this deviation of results from hypothesis was explained by the fact that change requests are handled in a different way than originally expected. Change requests with a high priority and/or a short time to implement were included in Trouble Log (TLOG) whereas other change requests were sent to the Marketing Department to be planned as additional functionality. Previously, all change requests were reported in TLOG. The criteria for including a change request into TLOG are:

- Is the specification of the CR clear enough to be directly implemented?
- Is the effort required for implementing the CR relatively low?
- Is the risk that the CR will not succeed within this time low?

The measurement programme created a better understanding of failure detection by engineers. The amount of failures found by engineers was much higher than originally expected. Causes for this were:

- The installation tests found less failures than expected. This was very important: if failures are not found within R&D, it is more likely that those failures will be found in a pilot than in an Installation test.
- Another reason is that no test group exists anymore (unfortunately).
- Before a release all new or changed modules are tested by the engineers.
- By reusing modules, faults in existing code are detected by the engineers.

With respect to the original goals of understanding product reliability and the influence of the development process, it can be stated that:

- The goal is reached, it is understood how to identify product reliability.
- A measurement programme is a source of information that gives some visibility to the reliability of the product.
- The most important identification of product reliability is that no failures are reported from the field. Keeping good contact with pilots in the field is therefore very important.
- Combining field information with a measurement programme gives better insight to the reliability of the product.
- The measurement programme would give more information on product reliability if it would include testing effort and system use.

Without information on how much effort is put into failure detection, the number of failures found doesn't give enough information. Not finding failures gives less visibility on the reliability, than finding failures. The fact is that when many failures are detected, it means that the product is not reliable. The opposite, not finding failures, however, does not mean the product is reliable.

Results of feedback session on 'reuse'

Few modules in the Driver functionality have been reused. The reason for this is that no reusable modules were available. It is expected that those modules will be reused in the future.

Module X has been developed by reusing Module Y. The decision for reusing has been based on the measurement data, because it has been identified that reuse results in lower fault density. From the 8 largest modules, only one has been developed from scratch.

With respect to the reuse goal it was stated that:

- The goal is reached, it is understood what the effects of reuse are.
- It would be interesting also to look at the influence of reusing modules on development effort.
- It is not possible to draw general conclusions whether new development or reuse is absolutely better, since only one of the options is completed. Developing a module both from scratch and with reuse would create the possibility to compare, but this is not done.
- Reusing large parts of a module that is already fixed, results in significantly lower faults than developed from scratch
- Faults in reused modules are detected more before delivery, because reused modules are reviewed and tested more intensively than new developed modules.
- The results on the effects of reuse must be shown to other parts of the organisation, in order to show the benefits of reuse.

Conclusions

The project A measurement programme was successfully completed, all goals were attained. The project did not only create more insight to the product and the development process, but also supported the project team in particular decisions. Another benefit was the increase of experience in how to set up a measurement programme.

Project A will get continuous support of measurement by defining a new measurement programme on testing, and similar measurements on module X.

Action list

The following action list was established in the final feedback session:

- Define measurement programme on test goal.
- Define measurement programme on module X.
- Post mortal analysis on project A measurement programme.
- Present some major results of project A over the organisation.

It was concluded that subsequent meetings needed to be organised for defining a new measurement programme on a test goal. It was expected that a large amount of such a new measurement programme could be based on the measurement programmes project A and RITME. Also module X should be supported by some measurements similar to the project A measurement programme.

9.7 Questions and assignments

9.7.1 Questions

1. Which aspect of reuse, would you consider most apparently missing in the reuse measurement goal of this chapter?
2. What do you consider the main purpose of Figure 9-3?
3. Explain why a chart such as Figure 9-5 should only be interpreted by a project team?
4. What are the main differences between *fatal*, *major* and *minor* failures?
5. Should failure severity classification (of fatal, major, minor), be interpreted from the perspective of a developer, or from the viewpoint of an end-user?
6. Give examples of interpretations derived from Figure 9-7 that are contradictory, but would, however, be supported by Figure 9-7.
7. What does Figure 9-5 imply about the effects of reuse on product reliability?
8. Is cyclomatic complexity an adequate metric for measuring software complexity according to the data of Figure 9-14?
9. Which two main topics do you consider missing in the feedback session report of project A in Section 9.6.2?

9.7.2 Assignments

The GQM plan of the case study presented in this chapter was the first GQM plan developed by that particular organisation. Therefore, it does not comply to the criteria set in Chapter 6. Review the GQM plan and identify the mistakes in this plan. Use the validation mechanisms introduced in Chapter 6 as a reference.

'When the comments and the code differ, go with the code'
Source unknown

10 Case B: review and inspection measurement

10.1 Description of project B

This chapter describes the results of the RITME measurement programme. RITME was the acronym for 'Reviews and Inspections: The Measurement of their Effects'. The RITME measurement programme investigated the aspects of fault detecting and learning effects of reviewing. A review was defined as a *cross reading of a document by another person than the writer of the document*. Reviewed were deliverables such as designs and programming code of project B.

A review involves two people: the developer of the designs or programming code, and a reviewer. The developer decided whether a document requires a review and assigns it to a reviewer. The reviewer read the document and made notes upon detected faults or obscurities. Following, the reviewer and developer discussed the reviewer's notes. Finally, the developer removed the detected faults from the document.

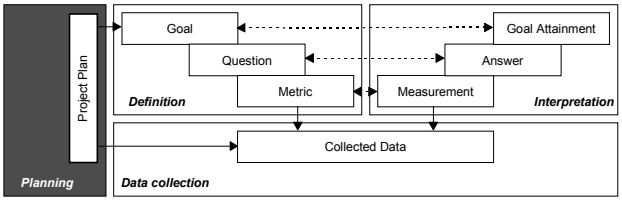
Reviews were measured for two purposes. The first purpose was finding faults, '*... a brief inspection by competent co-workers invariably turns up mistakes the programmers would not have found by themselves*' (Humphrey, 1989). The second purpose was to train new people: reading documents not only teaches about the contents of the document, but also about what lay-out or coding standards were used.

In the RITME project a high-end cashing and fuel station management system was developed. Software code was primarily written in the C^{++} programming language, with additional application of component libraries. The total duration of the project was approximately 6 years of which 3 years was spent on the first increment of the product. In total 20 software engineers worked on this product. Also, several customised versions for specific countries or customers were developed.

The results presented in this chapter are based on the complete measurement programme. Results are not only the measurement data, but also conclusions drawn by the project team, or action taken based on the measurements. The RITME measurement programme will be described by the four phases of the GQM method: planning, definition, data collection and interpretation.

This chapter also includes a lot of documentation of this measurement programme, such as the GQM plan, data collection forms, and feedback charts. Documentation that one can copy for similar measurement programmes.

10.2 Planning



An independent GQM team already existed in the department. This team discussed with the project team what the role could be of the measurement programme in the context of the organisation's improvement programme. Due to successful application of goal-oriented measurement in foregoing projects, it was decided to apply the GQM method towards an operational improvement goal relevant to the project. The necessary training for the project team was given by the GQM team, which had experience of the practical application of the GQM method. The planning phase delivered the RITME Project plan.

Based on the current status of the project and the changes in the department it was decided to consider reviews and inspections as the measurement topic. The two reasons for this were:

- The main development work was shifting from initial development to changes and expansion of functionality. Testing therefore became less efficient, because minor changes would need large testing efforts. Reviewing those changes or additional developments was expected to be more effective and efficient. System testing was still performed, but only in the process of a new version release, and by a separate test group.
- The department was recently expanded with five new engineers, who were educated by 'training-on-the-job'. Reviews were considered a powerful tool during this training process in two ways. Firstly, new engineers could review deliverables of senior engineers to reach understanding of the system. Secondly, senior engineers could review deliverables of the new engineers, to check conformance to the departments coding and quality standards.

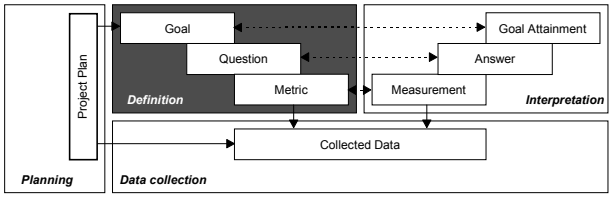
So, the purpose of reviews was not just the detection of faults or failures, but also a major tool for teaching (new) team members on parts of the product. This educational function of reviews was also included in the measurement programme. Measurements were also conducted to identify the effects of reviews on learning capability.

The effort of the project team for reviewing was limited to 4 hours maximum per week per project member. This effort was already planned to be spent on reviews for training new engineers. The overhead of the measurement programme for the project team needed to be extremely low due to time constraints of the project. The GQM team gave as much support as possible, and executed most of the operational tasks of the measurement programme.

The general expectation was that the time spent on reviewing and the supporting measurement programme, would result in better quality of the product. Therefore, less time was expected to be spent on repairing faults in later stages of the project.

The project supported during RITME, was Project B. The Project B team consists of 9 engineers and a project manager.

10.3 Definition



The improvement goal defined in this project was already quite operational. Reviews would be introduced in the department and held structurally. Because of the two main purposes of these reviews: fault detection and training, the measurement goal could be formulated directly from the improvement goal.

Reviews:

Analyse the:	effectiveness of structured reviews
for the purpose of:	understanding
with respect to:	- the detection of faults - the learning ability of the technique
from the viewpoint of:	the project team
in the following context:	project B

Achieving this measurement goal would yield a better understanding of reviews, focused on the fault detection capabilities and training effectiveness. The project team supported this measurement goal, even though the project manager initially identified it.

The definition steps used in the RITME measurement programme were similar to those presented in Chapter 6. The main difference of the definition process of RITME compared to that in project A (Chapter 9), was that the complete GQM method was used as described in this book, instead of the QIP-based approach. The main reason for this was that experiences from project A had provided knowledge as to the importance of the several steps. Because of the efficiency requirements of RITME it was decided to perform only the critical steps of the GQM approach. Furthermore, process modelling was done for the review process, which was a learning point from earlier GQM application.

Knowledge acquisition was supported by structured interviews. The main purpose of the structured interviews was to make implicit knowledge of the Project members explicit. Developing a documented description of the review process was executed during structured interviews with two representatives of the project team and the project manager. The ETXM model that was developed based on the interviews is included in Figure 10-1.

The definition of questions and metrics was also provided by structured interviews. Documenting the question, metrics, and hypotheses in abstraction sheets, checking completeness and consistency between abstraction sheet and the process model and writing the GQM and Measurement plan was executed by the GQM team. The GQM plan of RITME is attached in Section 10.6.

Review
Entrance criteria: Product to be reviewed is available Reviewer(s) has been assigned Exit criteria: Review report is available and approved Tasks: Review the product Fill in review report Transfer the review report to the author Correct the product based on the faults noted in the review report Approve the review report if all faults have been addressed Measurements: Number of faults detected during review Size of reviewed product Complexity of reviewed product Estimated resource time for review Resource time for review Schedule time from start to finish for review Severity of faults detected during review

Figure 10-1: ETXM model of a review.

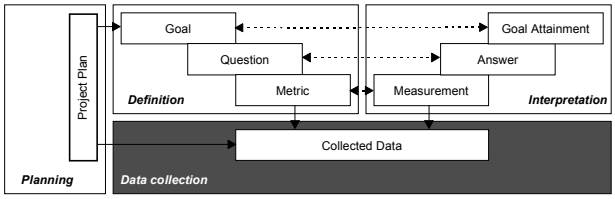
The questions to be answered by the RITME measurement programme were:

- Question 1: What are the baselines (rules of thumb) for executing reviews?
- Question 2: What are the characteristics of a document, to get optimal effects of a review?
- Question 3: What are the acceptance criteria for the quality of the document, after a review?
- Question 4: Are reviews a useful and efficient method for training new engineers?

Additionally, an analysis plan was produced to present the interpretation phase beforehand. The analysis plan was created on a number of virtual data and contained graphs and tables, based on these data, that were related to the stated questions and goals. In this way, a starting point was created for the creation of future feedback material.

Reviews of the GQM plan, measurement plan and analysis plan was conducted in a meeting. During this meeting, the GQM team repeated the concept of the GQM method to the project team. The GQM team also presented some results from previous measurement programmes to motivate the project team for measurement. Finally, the GQM team presented the RITME measurement programme: the goal template of the measurement programme was introduced along with the refinement into questions and metrics. All material was discussed by all attendants of the meeting, ambiguities were solved, and a number of remarks were noted for improvement of the plans.

10.4 Data collection



Data collection started with a 'kick-off session' in which the data collection forms were explained. The metrics defined in the measurement plan were collected by one manual paper form. A number of metrics that had to be filled in on the 'Review Form' (size, cyclomatic complexity and number of comment lines) were calculated using an automated tool, provided to the project team via the network.

Data correctness was checked when importing the data into the metrics base and during reviews of the feedback material.

10.4.1 The RITME measurement support systems (MSS)

The goal of the measurement programme was twofold: the fault finding effects and the learning effects of reviewing were investigated. As these aspects were heavily intertwined, only one analysis sheet was created. To improve the possibility of analysing historic data, a new implementation was created. The links between metrics base and analysis sheet were not implemented. Instead, the data from the metrics base, that were to be analysed, had to be copied into the analysis sheet. Automatic updating was implemented within the analysis sheet as far as supported by the spreadsheet tool (Figure 10-2).

Another change included the exclusion of hard-copy transparent slides for the presentation of measurement results. Instead, the graphs and tables that were to be presented, were linked to an on-line presentation tool and directly projected by means of an LCD display, connected to a computer. This implementation supports automatic updating of graphs and tables and gives additional support for selecting and editing presentation material (useful for highlighting interesting results). Furthermore, it provides a better appearance of the slides. The main advantage of a LCD display is that the original data can be accessed on-line. Once specific questions occur during the feedback session, the underlying data can be consulted immediately.

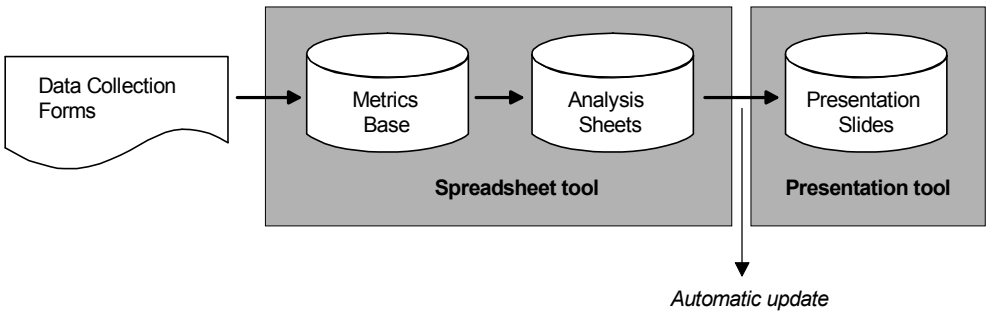


Figure 10-2: MSS of RITME, included with linking analysis sheets to analysis slides.

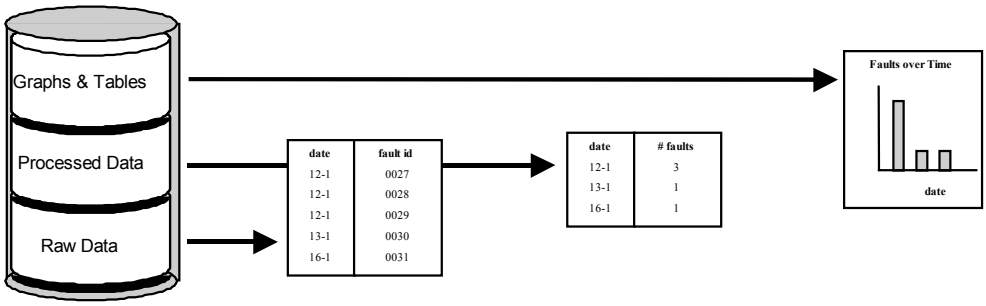


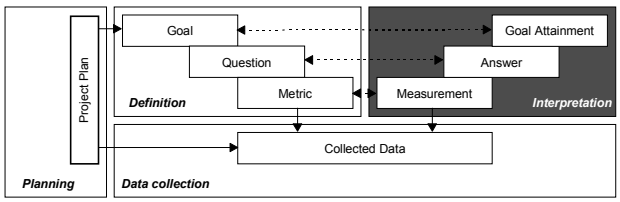
Figure 10-3: Three layers within an analysis sheet.

The MSS was set-up with the concepts presented in Section 7.4. Each analysis sheet consisted of particular worksheets for the questions that relate to a goal of the analysis sheet (question worksheets). Each analysis sheet was built up from three layers (Figure 10-3):

- raw data layer;
- processed data layer;
- graphs & tables layer.

The raw data layer contains the links to the metrics base and, therefore, like the metrics base itself, contains raw data only. The results of processing the raw data are kept in the processed data layer. Finally, the graphs & tables data layer contained the questions from the GQM plan, which they intended to answer, added to the relevant charts or tables.

10.5 Interpretation



This section presents interpretation results of the RITME project. Due to a relative slow delivery of data points (around 3 reviews were conducted per week), feedback sessions were held every four months. During the RITME measurement programme three feedback sessions were held. However, reviewing was not expected to end when the measurement goal was achieved. Therefore reviews were included in the normal process after the measurement programme had ended.

The results of the measurements are not only presentations of empirical data on the different subjects, but also conclusions drawn or actions taken based on the measurements. A selection of conclusions drawn and actions taken by the project members based on the measurement results is presented:

- Reviews are an efficient aid to detect faults.
- Reviews improve uniformity and maintainability of code.
- Reviews increase communication, synergy and understanding within a development team.
- Reviews detect more than 1 Fatal/Major per hour.
- Reviews detect 7.5 faults per hour.
- Reviews should be done on documents smaller than 50 pages.
- Reviews are more effective on documents of maximum 20 pages.
- Reviews are at least four times more efficient than testing.
- Reviews should be done for at least one hour a week per engineer.
- New engineers learn most from reviewing code from experienced engineers
- Training of new engineers should subsequently consist of:
  - courses to provide relevant theory and general overview;
  - reviews to learn system details;
  - on-the-job training to provide practical experience.

Feedback was provided to the project team in feedback sessions. All project B members were involved in the interpretation phase. However, compared to the project A measurement programme, improvements were made in the way feedback was given.

- Firstly, in theory, feedback sessions should be held often and the quantity of material presented should be low to achieve learning in many small steps. This implementation of feedback in a business environment would, however, create a number of practical problems: first, feedback sessions would present few new data points, and second, it is already hard to get an entire project team to attend a meeting every eight weeks. Therefore, it was decided to provide some feedback on posters every two to three weeks. In this way, feedback would be provided more frequently. However, these posters had little effect compared to feedback sessions.
- Secondly, to improve data correctness, the presentation material was reviewed internally before the feedback session: one GQM team member prepared the material and another would review the material. Simultaneously, the reviewing GQM team member could evaluate whether the created graphs and tables were understandable and whether any other related metrics should be included in the graphs (or tables).
- Thirdly, it was decided to conclude every feedback session with an evaluation of the results accomplished in the session, in relation to the stated goal.
- Finally, to create a framework for the interpretation, it was decided to relate measurement results to the questions they were intended to answer by presenting the questions along with the measurement results. Therefore, each session followed the questions in the GQM plan: each time, a slide with GQM questions was presented preceding the graphs and/or tables that contributed to answers to these questions (Figure 10-4).

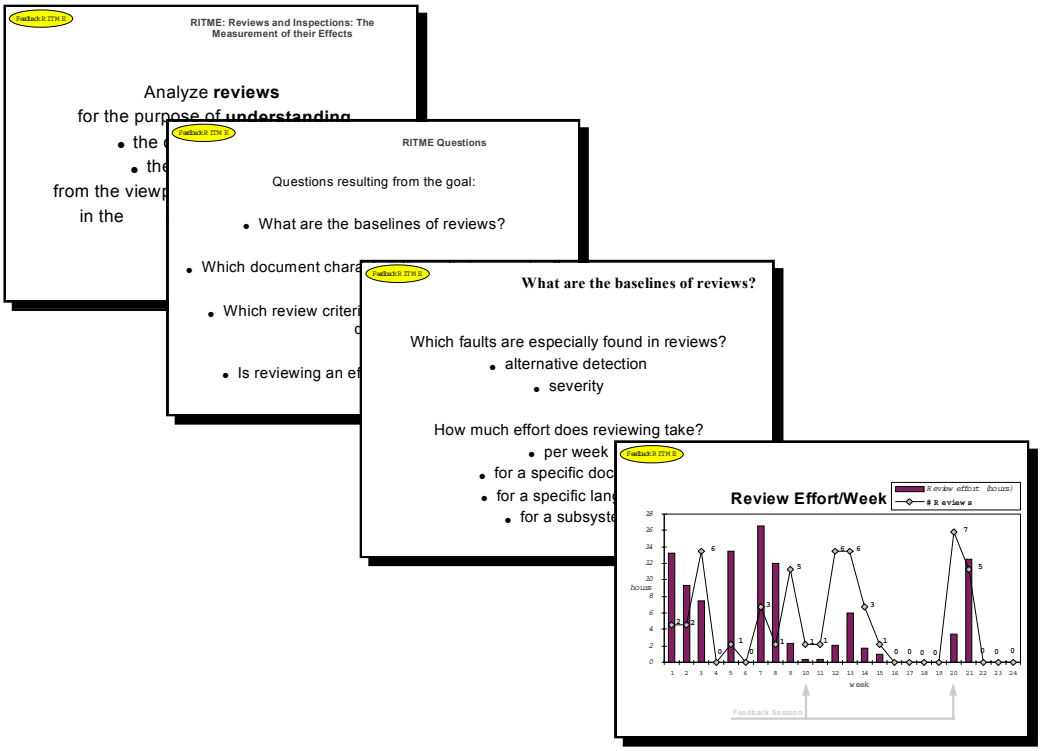


Figure 10-4: Presentation of results in relation to goals and questions.

Another aspect that was learned through the experience in providing feedback for RITME concerns scales. Intuitively, all graphs should have scales. Just as important as having scales in graphs, is using equal scales in different graphs to make it easier to compare different graphs.

In RITME, problems occurred through the definition of classes: faults found were to be classified according to their expected alternative detection (ie how they would have been detected if not by reviewing). During the definition of the measurement programme, it was decided to include both a class ‘never’ and a class ‘unknown’. In practice, it appeared that these classes were used too easily: many faults were rated to belong to these classes, and none to the ‘field test’ and ‘installation’ classes (Figure 10-5), whereas in reality faults were also detected in field tests and by installation’s (remark of the project manager). During the feedback sessions, it was recognised by the project team that the ‘unknown’ and ‘never’ faults were probably better rated as ‘field test’ or ‘installation’. Therefore it was decided to delete the ‘unknown’ and ‘never’ classes from the Review Form.

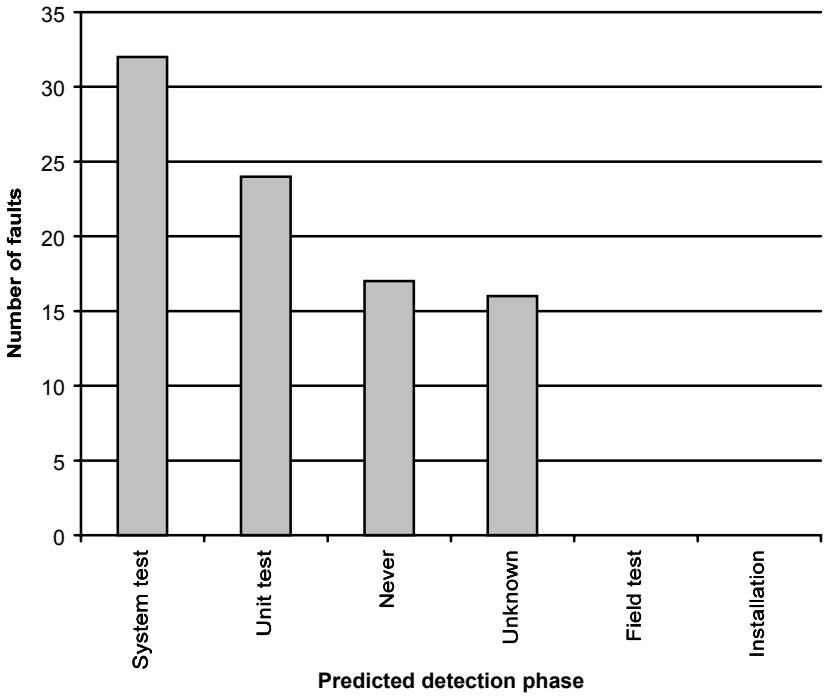


Figure 10-5: Fault classification according to alternative means of detection: bad use of classes.

The second feedback session started with a thorough evaluation of the actions defined in the first session. In the first session (week 15), it was for instance decided to limit the maximum length of a document for a review, initiated to find faults, to 50 pages (Figure 10-6). During the second feedback session it was evaluated whether this 50 pages limit was held. Only one review exceeded the 50 pages.

In providing feedback for RITME, a background was introduced to the slides. The background was introduced to create a recognisable look for the RITME feedback slides and posters. When doing so, the background should not be too fancy: the first background for the RITME feedback material was experienced as being too disturbing.

The results of the measurements will be presented for both the faultfinding capabilities and the learning capabilities.

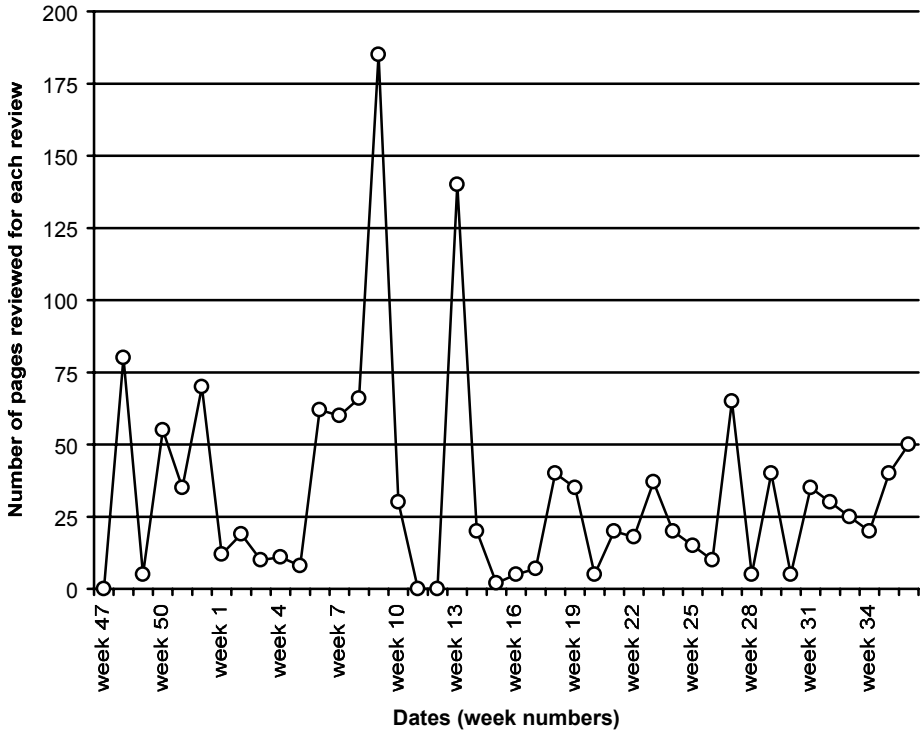


Figure 10-6: Evaluation of action points.

10.5.1 Fault finding capabilities of reviews

The faultfinding results of reviews are summarised in Figure 10-7. Considering the number of detected faults per page, the number of faults detected by reviewing Project B was 0.4 faults per page equalling 24 SLOC[3](Figure 10-7). Grady reports that inspections[4] resulted in 0.3 faults per page (Grady, 1992).

[3] The number of 24 SLOC per page was based on a sample of documents containing programming code of the Project B systems.

[4] Grady does not define inspections beyond them being reviews of requirements models, designs, code and other work products.

Faults found by reviewing documents							
Total effort for reviewing:				43 hours			
Total effort including meetings and repair:				74 hours			
Total number of pages:				998 pages			
Total number of lines:				24 KSLOC			
		Distribution		Faults/hour		Faults/page	
	# Faults	hypothesis	real	hypothesis	real	hypothesis	real
minor	292	70%	77%		6,8		0,3
major	57	25%	15%		1,3		0,1
fatal	29	5%	8%		0,7		-
TOTAL	378			20-65	8,7	3-7	0,4

Figure 10-7: Overall results of RITME.

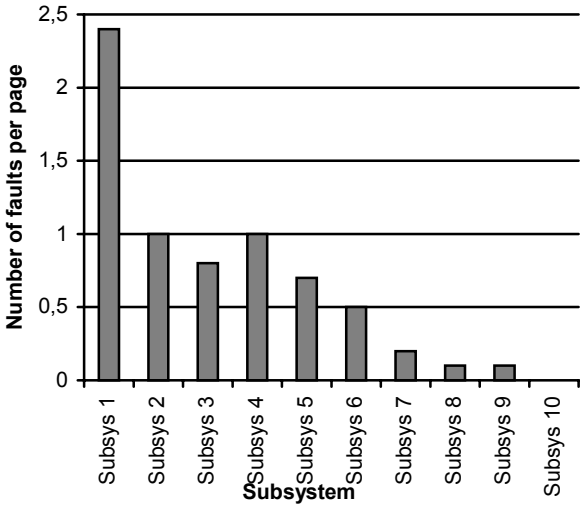
	Review	Acceptance test	Novice user test
Number of faults	378	111	49
Hours spent	43	322	32
Faults per hour			
minor	6,76	0,09	1,2
major	1,32	0,15	0,3
fatal	0,67	0,11	0,1
TOTAL	8,75	0,34	1,5

Figure 10-8: Comparing faults found by reviews, an acceptance test and a novice user test.

Comparing reviews to tests showed large differences in fault-detecting performance (# detected faults per hour). During the execution of RITME, two tests were executed: an acceptance test and a monkey test (novice user test). In the acceptance test, 0.34 faults were found per hour testing, in the monkey test 1.5, while reviews produced a number of 8.7 faults per hour reviewing (Figure 10-8): an important indication for the effectiveness of reviewing. The effectiveness of reviewing over the acceptance test differs a factor 25; over the monkey test it differs a factor 5. AT&T reported inspections[5] to be a factor 20 more effective (Humphrey, 1989). No subsystem was identified in which significantly more faults were found than in other subsystems[6] (Figure 10-9).

[5] Reviews as performed in Project B comply more to Humphrey's definition of walkthroughs than inspections.

[6] The high faults/page value of the Sybsys 1 subsystem is caused by the very low number of reviewed pages.



Subsystem	Number of faults	Number of pages	Faults/page
Subsys 1	12	5	2,4
Subsys 2	41	42	1,0
Subsys 3	130	158	0,8
Subsys 4	1	1	1,0
Subsys 5	4	6	0,7
Subsys 6	80	152	0,5
Subsys 7	71	302	0,2
Subsys 8	37	255	0,1
Subsys 9	3	38	0,1
Subsys 10	0	455	-

Figure 10-9: Detected faults/page by reviewing per subsystem.

Under the assumption that the major and fatal faults found by reviewing, would have been detected during the monkey test if no reviewing was performed, reviewing resulted in 86 prevented test defects: twice the number of actual test defects of the monkey test. A prevented test defect is a defect that would have been detected through testing if not through another means of fault detection (Rooijmans et al, 1996). In their study of inspections[7], Rooijmans et al found the numbers of test defects prevented through inspections ranged from 1.4 to 3.9 in comparison to actual test defects (Rooijmans et al, 1996).

[7] Note that the investigated inspections concerned inspections of requirements and design documents, not programming code as in the investigated reviews.

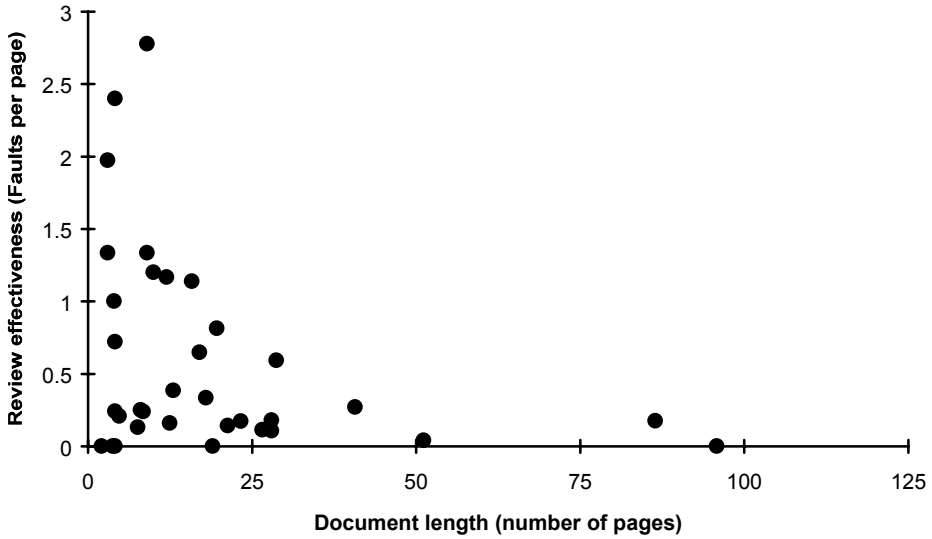


Figure 10-10: Decreasing number of detected faults/page with increasing document length.

In RITME, it appeared that the number of detected faults per page decreased with increasing length of the document under review (Figure 10-10). The cause of this phenomenon might be identified by a decreased effort spent per page (Figure 10-11). Wenneson found that inspections of FORTRAN code took 3 to 10 hours per KLOC (Humphrey, 1989). Reading KLOC as KSLOC[8], this translates to 4.3 to 14 minutes per page. These numbers are significantly higher than the numbers found in RITME. No clear cause for this deviation could be identified. Possible causes include the difference between an informal review and a formal inspection, differences in code complexity, and difference in programming language.

[8] KLOC - kilo lines of code (1000 lines of code), KSLOC - kilo source lines of code (1000 source lines of code, excluding comment).

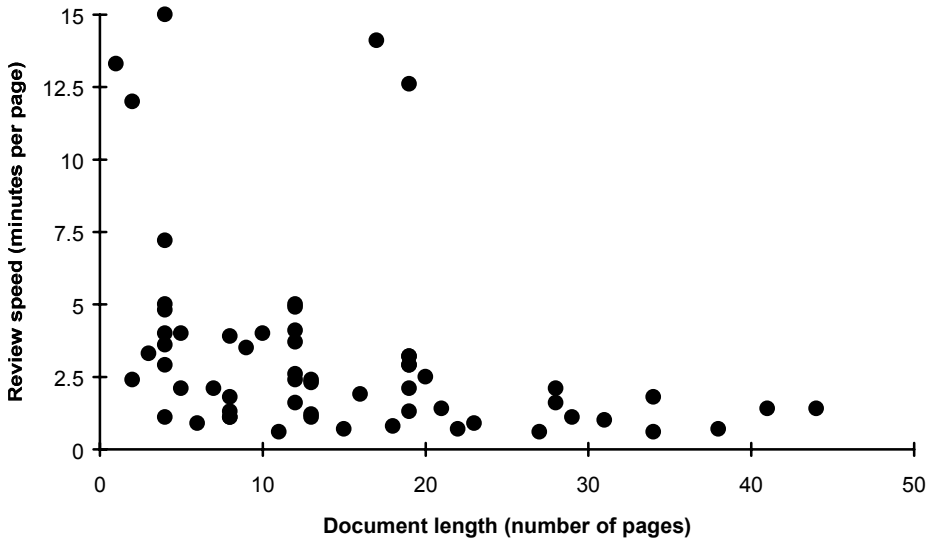
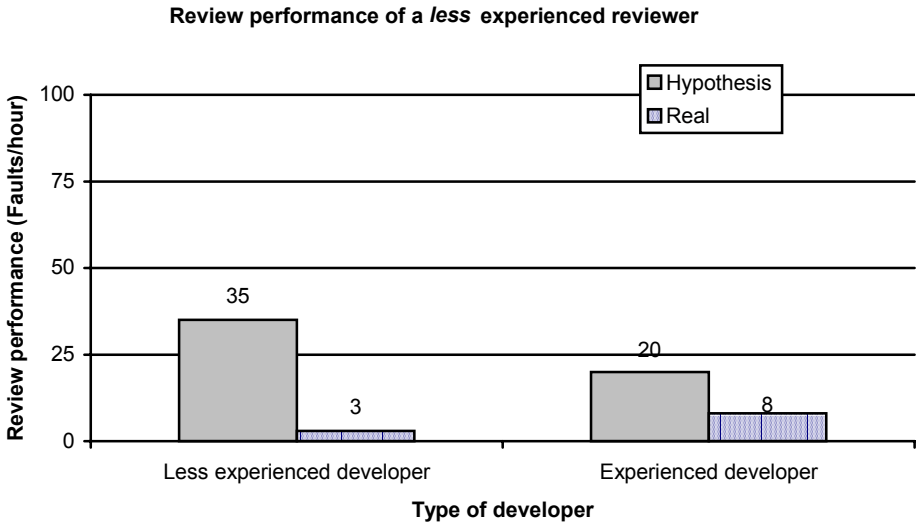


Figure 10-11: Decreasing effort per reviewed page with increasing document length

Resulting from both results, a maximum number of pages for a review was defined: the maximum number of pages for a review was not to exceed 50. This result was tracked throughout the measurement programme. The result of this tracking is depicted in Figure 10-6; the maximum was defined in the first feedback session, held March 8. Thereafter, no reviews exceeded the 50-pages limit.

Besides a maximum document length for reviews, RITME has caused a standardisation of reviews: before the measurement programme started, reviews were carried out in an unstructured, informal manner. As a result of RITME, existing review checklists were updated and re-introduced for the execution of reviews directly after goal attainment and formal ‘end’ of the measurement programme.



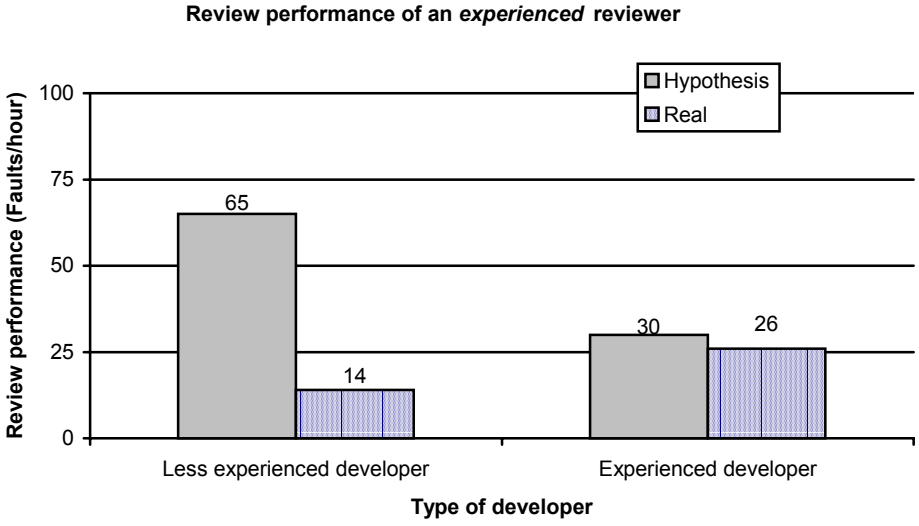


Figure 10-12: Review performance and experience of engineers.

Other results of RITME include (Figure 10-12):

- Experienced reviewers detect over three times more faults per hour than less experienced reviewers.
- Experienced reviewers detect two times more faults per page than less experienced reviewers.
- More faults are detected per hour in code developed by experienced engineers compared to code developed by less experienced engineers.
- Less faults are detected per page in code developed by experienced engineers compared to code developed by less experienced engineers.

10.5.2 Learning capabilities of reviews

In project B, reviewing is not only performed to detect faults, but also to train new people. This method of training was also measured by RITME as the learning effect of performing a review. The main result from this aspect of RITME is a good insight into how new people can be trained optimally. RITME indicated that reviewing is an important element in training new people, but should be part of a larger training programme. Besides reviews to learn system details, training should include courses to provide relevant theory and general overview and on-the-job training to provide practical experience.

The change in learning effect over time was remarkable. Figure 10-18 shows the results on learning effects during the first half year of the measurement programme. Most learning effects occur when a less experienced engineer reviews the material of an experienced one. During the second half year of the measurement programme this picture changed totally.

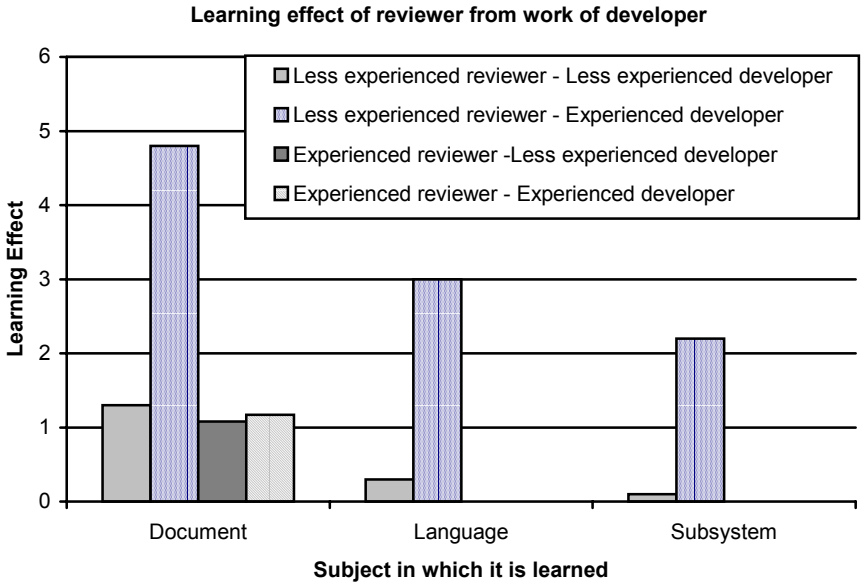


Figure 10-13: Learning effect of reviewing.

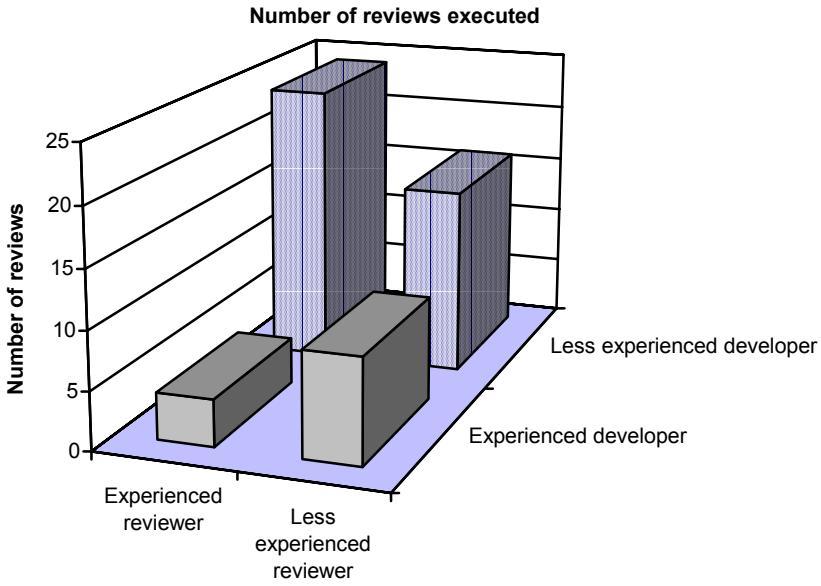


Figure 10-14: Executed reviews, divided according to experience level of developer or reviewer.

The less experienced became much more experienced and started to learn much more from each other than from experienced engineers. This lead to the conclusion by the project team that six months is about the time needed to train people to work in their development group.

Considering these results, the distribution of reviews among (in)experienced reviewers is remarkable (Figure 10-14): while the learning effect of a review is the largest when an less experienced reviewer reviews a document of an experienced developer, the number of reviews executed in this category is relatively small. A possible explanation could be that the emphasis of reviews is on fault finding.

10.6 Documentation of RITME project

This section contains:

- RITME GQM plan and measurement plan;
- applied review form;
- example of a feedback session report.

These documents will subsequently be described.

10.6.1 RITME GQM and measurement plan

Introduction

Software measurement programmes are started in the R&D department. The methodology used for introducing and defining a software measurement plan is the GQM method. This method emphasises specific use of measurement data. Cooperation from, and feedback to, the project team is therefore considered to be the main success factor of this approach. Due to the success of current experiences, management has decided that such project support by a measurement programme must be introduced in all development projects.

This is the GQM plan for the RITME measurement programme. Measurement will be introduced for a better understanding of current processes, to provide information on the performance and in that way identify areas of improvement. This document will be used for three purposes:

1. Feedback to the project B team members by the GQM-team.
2. Definition of data collection, since all direct measurements are defined in this document.
3. Analyses and feedback of collected data, because the GQM plan identifies which questions need to be addressed.

RITME stands for: Reviews & Inspections: The Measurement of their Effects. The Project B team will start investigating the effects of reviews. In a meeting with the group leader and two team members this goal has been identified. Since then interviews have been held with the same people and, based on these interviews, questions have been formulated and metrics have been defined with associated hypotheses.

In this document a representation is given of the information that has been gathered during those sessions. For each goal the following aspects are described:

- goal description and the refinement to questions;
- refinement from questions to metrics and direct measurements;
- abstracted direct measurements that have to be collected for the goal;
- hypotheses to the direct measurements given by the project team members.

The GQM plan will be subject to changes during the measurement period due to new information and experience gained in the feedback sessions.

Reviews

A review is an activity that will cause improvements in a document, by:

1. detecting and fixing of faults;
2. defining (quality) improvement areas in the document;
3. increased understanding of the product by engineers (learning).

A review is organised by the following steps:

1. Preparation phase, in which the document is identified, assigned to a reviewer, and the necessary relevant documentation is studied and understood.
2. Review phase, in which the actual cross-reading of the document is done, and detected faults and questions found are reported.

3. Review meeting, in which the review is discussed, open questions from the reviewer are answered, and after which the final review report can be finished.
4. Follow-up phase, in which actions are taken based on the review. It will be examined, to which extent faults found in a review are also really corrected.

The follow-up phase is included in this process in order to identify what the real result of a review was. This is also a check for identifying whether required changes are really implemented.

From goal to questions

The goal that will be addressed is:

Analyse:	the effectiveness of structured reviews
for the purpose of:	understanding
with respect to:	- the detection of faults - the learning ability of the technique
from the viewpoint of:	the project team
in the following context:	project B

This Goal will be attained when answers can be given to the following questions:

- Q.1 What are the baselines (rules of thumb) for the execution and results of reviews? This question focuses on identifying objective, quantifiable numbers that are related to the review and is answered when the relation between all factors involved in a review are clear and values are available that visualise the baselines and rules of thumb, that are related to reviews, objectively.
- Q.2 What are the characteristics of a document, in order to obtain optimal effect of a review? This question focuses on the definition of criteria that define whether a review will obtain the required effects and will give an answer to which characteristics a certain document should posses in order to result in a specified result of a review.
- Q.3 What are the acceptance criteria for the quality of the document, after a review? This question focuses on the possibility to identify if a reviewed document has an acceptable quality level. Therefore it should be investigated whether criteria can be defined to which the performance of a review must be met in order to identify an acceptable quality level. This question is answered when the possibilities for such criteria are understood, and when possible values to those criteria are known.
- Q.4 Are reviews a useful and efficient method for training new engineers? This question is answered when all learning abilities have been identified, and their effects can be quantified.

From questions to metrics

Question 1 was **'**what are the baselines (rules of thumb) for the execution and results of reviews'? This question focuses on identifying objective, quantifiable numbers that are related to reviews. This question is answered when the relation between all factors involved in a review are clear and values are available that visualise the baselines and rules of thumb, that are related to reviews, objectively.

Detailed questions focusing on practical experience by the project team are:

- Q.1.1 Which kind of faults are likely to be found in reviews?
- Q.1.2 How much effort is spent on reviews per week?
- Q.1.3 What is the average time required to review a document of a specified size?
- Q.1.4 What is the average time required to review a document in a specific language?
- Q.1.5 What is the average time required to review a document of a specified complexity?
- Q.1.6 What is the average fault detection performance of reviewing documents?
- Q.1.7 Does the average performance of reviews change over time?
- Q.1.8 How effective is it to review a certain document for the second time?
- Q.1.9 What is the effort I can estimate for a review on a document of which the characteristics are known?
- Q.1.10 What is the difference in review performance between subsystems of the product?

The following metrics have been associated to these questions:

- subsystem of the document;
- document size;
- document specification language;
- document complexity;
- effort for a review;
- effort per page for a document;
- number of faults found for a review;
- other ways in which the faults could have been detected for a review;
- distribution of faults over severity for a review;
- number of faults per page for a document;
- date on which the review was held.

The above metrics resulted in the following direct measurements:

- M.01 Document name for each review.
- M.02 Document subsystem for each review.
- M.03 Document length for each review.
- M.04 Document cyclomatic complexity for each review.
- M.05 Number of necessary related documents for each review.
- M.06 Document (programming) language for each review.
- M.07 Effort spent on each review.
- M.08 Date for each review.
- M.10 Severity for each fault for each review.
- M.11 Other detection mechanism for each fault for each review.
- M.12 Name of the reviewer.

Question 2 was ‘what are the characteristics of a document, in order to obtain optimal effect of a review’? This question focuses on the definition of criteria that define whether a review will obtain the required effects. The question will give an answer to which characteristics a certain document should posses in order to result in a specified result of a review.

Detailed questions focusing on practical experience by project team:

- Q.2.1 What is the influence of document size on the review performance and effectiveness?
- Q.2.2 What is the influence of document complexity on the review performance and effectiveness?
- Q.2.3 What is the influence of a number of related documents on the review performance and effectiveness?
- Q.2.4 What is the influence of specification language on the review performance and effectiveness?
- Q.2.5 What is the influence of product subsystem on the review performance and effectiveness?

The following metrics have been associated to these questions:

- document name;
- subsystem to which the document belongs;
- document size;
- document specification language;
- document complexity;
- effort for a review;
- average number of faults found;
- average distribution of faults over severity;
- number of faults per page for a document;
- number of faults detected in the document after the review.

The above metrics resulted in the following direct measurements:

- M.01 Document name for each review.
- M.02 Document subsystem for each review.
- M.03 Document length for each review.
- M.04 Document cyclomatic complexity for each review.
- M.05 Number of necessary related documents for each review.
- M.06 Document (programming) language for each review.
- M.07 Effort spent on each review.
- M.10 Severity for each fault for each review.
- M.99 Document name for each fault found later on in the process (this metric will be very important to answer the question, however, at this stage it is not clear how this will be measured).

Question 3 was ‘what are the acceptance criteria for the quality of the document, after a review’? This question focuses on the possibility to identify if a reviewed document has an acceptable quality level. Therefore it should be investigated whether criteria can be defined to which the performance of a review must be met in order to identify an acceptable quality level.

This question is answered when the possibilities for such criteria are understood, and when possible values to those criteria are known.

Detailed questions focusing on practical experience by project team:

- Q.3.1 Is the number of faults a usable criteria for identifying whether a second review is necessary or not?
- Q.3.2 What is the amount of failures to be found on average, in order to claim that a document is of high quality?
- Q.3.3 What is the effect of a review on the improvement of a document?
- Q.3.4 Is the effort spent on repairing failures a usable indicator of product quality?

Metrics:

- document name;
- subsystem to which the document belongs;
- document size;
- document specification language;
- document complexity;
- effort for a review;
- average number of faults found;
- average number of faults repaired;
- effort for repairing faults found;
- average balance between faults found and faults repaired;
- average distribution of faults over severity;
- number of faults per page for the document;
- number of open questions per page for the document;
- another review necessary for the document;
- quality improvement due to the review;
- number of faults detected in the document after the review.

Direct measurements:

- M.01 Document name for each review.
- M.02 Document subsystem for each review.
- M.03 Document length for each review.
- M.04 Document cyclomatic complexity for each review.
- M.05 Number of necessary related documents for each review.
- M.06 Document (programming) language for each review.
- M.13 Number of faults transported to TLOG.
- M.14 Another review of the document required (Yes/No) for each review.
- M.07 Effort spent on each review.
- M.15 Effort spent on repairing failures.
- M.10 Severity for each fault for each review.
- M.16 Repaired/Not Repaired for each fault found in the review.
- M.17 Number of open questions for each review.
- M.99 Document name for each fault found later on in the process (this metric will be very important to answer the question, however, at this stage it is not clear how this will be measured).

Question 4 was ‘are reviews a useful and efficient method for training new engineers’? This question is answered when all learning abilities have been identified, and their effects can be quantified. What is the difference in review performance between new and experienced engineers?

- Q.4.1 Detailed questions focusing on practical experience by project team
- Q.4.2 Does a new engineer learn much from reviewing?
- Q.4.3 What is the learning effect of asking questions based on a review?
- Q.4.4 Does knowledge on a document improve quickly, by reviewing it?

Metrics:

- knowledge of reviewer on the document before the review;
- knowledge of reviewer on the specification language before the review;
- knowledge of reviewer on the subsystem before the review;
- knowledge of reviewer on the document after the review;
- knowledge of reviewer on the specification language after the review;
- knowledge of reviewer on the subsystem after the review;
- amount learned from the review;
- experience in software engineering for the reviewer;
- experience of the reviewer;
- number of open questions for the review;
- number of necessary questions during the review;
- number of questions resolved after the review meeting;
- subsystem to which the document belongs;
- document specification language;
- effort for a review;
- effort for review meeting;
- another review necessary for the document.

Direct measurements:

- M.18 Knowledge of reviewer on the document before the review.
- M.20 Knowledge of reviewer on the specification language before the review.
- M.22 Knowledge of reviewer on the subsystem before the review.
- M.19 Knowledge of reviewer on the document after the review.
- M.21 Knowledge of reviewer on the specification language after the review.
- M.23 Knowledge of reviewer on the subsystem after the review.
- M.24 Amount learned for each review.
- M.17 Number of open questions for each review.
- M.25 Number of questions resolved after the review meeting for each review meeting.
- M.02 Document subsystem for each review.
- M.06 Document (programming) language for each review.
- M.07 Effort spent on each review.
- M.26 Effort spent on each review meeting.
- M.14 Another review of the document required (Yes/No) for each review.

Hypotheses regarding direct measurements

During the interviews hypotheses have been collected regarding the expected outcome of the measurements. The expectations were combined, and when their hypothesis did not differ too much, an average was calculated. The following hypotheses are defined:

- The balance between minor, major and fatal faults respectively is about 70%, 25%, 5%.
- If a document is created by an inexperienced engineer, an experienced engineer will find 65 faults per hour in that document, if reviewed by an inexperienced engineer, 35 faults per hour will be found.
- If a document is created by an experienced engineer, an experienced engineer will find 30 faults per hour in that document, if reviewed by an inexperienced engineer, 20 faults per hour will be found.
- An experienced reviewer will have 2.5 open questions after reviewing a page.
- An inexperienced reviewer will have 5 open questions after reviewing a page.
- If a document is created by an experienced engineer, 3 faults will be found per page.
- If a document is created by an inexperienced engineer, 7 faults will be found per page.

Distribution of faults among the categories in which the faults would have been detected if not reviewed is given in the table below.

A number of other, more detailed, hypotheses were defined as well. These will be described in the forthcoming analysis phase.

Measurement plan

The measurement plan included the following measurements:

- M.01 Document name for each review. This metric gives the name of the document that will be reviewed.
- M.02 Document subsystem for each review. This metric gives the name of the subsystem of the product to which the document belongs.
- M.03 Document length for each review. This metric describes the length of the document in lines of code. The scale will be KSLOC, which are all lines minus blank lines and comment lines.
- M.04 Document cyclomatic complexity for each review. This metric describes

	Experienced engineer	Inexperienced engineer
unit test	40%	12%
system test	10%	3%
Installation test	20%	6%
field test (pilot release), customer use	10%	3%
never because is exceptional situation	10%	3%
never because is lay-out fault (coding standard)	10%	73%

Figure 10-15: Executed reviews, divided according to experience level of developer or reviewer.

McCabes cyclomatic complexity which gives the number of possible programme paths in the code.

- M.05 Number of necessary related documents for each review. This metric describes the number of documents that is referred to in the document for the review. In the case of programming code this means the number of other files that are necessary to review that specific part of the code.
- M.06 Document (programming) language for each review. This metric describes the language in which the document is written. Possible values are natural language, specification language, assembler, plain C, C^{++}, Sycero, etc.
- M.07 Effort spent on each review. This metric gives the amount of time that is spent on reviewing the document. This effort can be given in minutes or hours.
- M.08 Date for each review. This metric gives the date on which the review is executed.
- M.09 Effort estimated for each review. This metric gives the amount of time that is estimated in advance, for the effort that will be spent on reviewing the document.
- M.10 Severity for each fault for each review. This metric describes the level to which the fault may influence future performance of the product. The categories are minor, major and fatal.
- M.11 Other detection mechanism for each fault for each review. This metric describes in which fault detection event a fault would be found, if it was not found during the review. Possible values are: unit test, system test, field test, installation test, never and unknown.
- M.12 Name of the reviewer. This metric defines the name of the person who has reviewed the document.
- M.13 Number of faults transported to TLOG. This metric describes the number of faults found that could not be solved during the repair phase, but which were serious enough to be solved at a later stage.
- M.14 Another review of the document required (Yes/No) for each review. This metric gives advice as to whether it is necessary to review the document again, after the faults have been repaired. This metric defines a kind of trust (reliability) in the document. Possible values are Yes or No.
- M.15 Effort spent on repairing failures. This metric describes the amount of time that is used for repairing the faults that were found in the review.
- M.16 Repaired/Not Repaired for each fault found in the review. This metric describes each fault found in the review whether it has been repaired or not. If a fault is not fixed, it is possible to enter a reason for that on the form.
- M.17 Number of open questions for each review. This metric defines the number of open questions that are raised, by reviewing the document.
- M.18 Reviewer's knowledge of the document before the review. This metric describes the reviewer's knowledge of the document before the review is started. This level of knowledge is given on a scale from 0 to 10.
- M.19 Reviewer's knowledge of the document after the review. This metric describes the reviewer's knowledge of the document after the review has been done. This level of knowledge is given on a scale from 0 to 10.
- M.20 Reviewer's knowledge of the specification language before the review. This metric describes the reviewer's knowledge of the specification language before the review is started. This level of knowledge is given on a scale from 0 to10.

- M.21 Reviewer's knowledge of the specification language after the review. This metric describes the reviewer's knowledge of the specification language after the review has been done. This level of knowledge is given on a scale from 0 to 10.
- M.22 Reviewer's knowledge of the subsystem before the review. This metric describes the reviewer's knowledge of the subsystem before the review is started. This level of knowledge is given on a scale from 0 to 10.
- M.23 Reviewer's knowledge of the subsystem after the review. This metric describes the reviewer's knowledge of the subsystem after the review has been done. This level of knowledge is given on a scale from 0 to 10.
- M.24 Amount learned for each review. This metric scales the amount that has been learnt by reviewing the document. This level of learning is given on a scale from 0 to 10.
- M.25 Number of questions resolved after each review meeting. This metric describes the number of open questions of the reviewer that were resolved after a review meeting.
- M.26 Effort spent on each review meeting. This metric identifies the amount of time spent on a review meeting.
- M.99 Document name for each fault found later on in the process (this metric will be very important to answer the stated question, however, at this stage it is not clear how this will be measured).

10.6.2 Review form of the RITME project

Explanation of the review form

The review form of the RIME project is illustrated on the next two pages. General guidelines for filling in the form:

- The form is divided according to phases and/or persons.
- Wherever input is expected, a line is depicted: .....
- When a selection is required, characters are typed using italic, underlined capitals: *YES / NO.*

1. Defining the review:

Document name: .....
Subsystem: ..... Date: ...../...../.....
Current release.: .....
(Progr.) Language: ..... Estimated effort for the Review: *MINUTES/HOURS*
Developer: .....
Reviewer: ..... Primary review focus: *fault finding/learning*

2. Review preparation:

	None				Low		Medium				High	Super
Knowledge of the reviewer of the document:	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	
Knowledge of the reviewer of the subsystem:	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	
Experience of the reviewer of the progr. language:	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	

Document Length: .....LOC Cycl.Compl.(VG2): ..... # Comm. lines: .....

These values can be calculated by executing the COUNT program. If a path is available to U:\UTIL\ it is OK to run it from the source directory. Syntax: **TELMET filename.ext ext** (So CPP or C has to be typed again)

3. Executing the review:

Review Date: ...../...../.....

Number of questions asked during the review: .....questions
Number of Faults:
Minors: ..... Majors: ..... Fatals: .....

Where would the **major** and **fatal** faults be found if this document was not reviewed?:
Unit test:..... Acceptance test:..... Integration test:..... Installation:.... Field(test):.....

Effort spent on the review: *MINUTES / HOURS*
Number of questions to ask in the review meeting: ..... questions

4. Review meeting:

Review meeting held with: .....
Effort spent on review meeting: *MINUTES / HOURS*
Number of questions solved: .....questions

If all faults found are repaired, is a second review then necessary? *YES / NO*

	None				Low		Medium				High	Super
Knowledge of the reviewer of the document:	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	
Knowledge of the reviewer of the subsystem:	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	
Experience of the reviewer of the (progr.) language:	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	
How much did you learn from this review?:	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	

5. Review fault repair:

Number of faults repaired:
Minors: ..... Majors: ..... Fatals: .....

Number of faults reported in TLOG: .....
Effort spent on repair: *MINUTES / HOURS*

6. Comments:

No.	Page	Line	m	M	F	Description	Repaired
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							
32							
33							
34							

Guidelines for specific elements:

- 2. Review preparation:
  Document length: the length of the reviewed part of the document.
- 3. Executing the review:
  - *Number of questions during the review*: the number of questions asked during execution of the review in order to be able to continue reviewing.
  - *Number of faults*: all faults found during the review; when the same fault is found more than once, every occurrence should be counted.
  - *Number of questions to ask in the review meeting*: the number of questions the reviewer has on the reviewed document after reviewing the document.
- 4. Review meeting:
  Number of questions solved: those questions resulting from the review (as recorded as 'Number of questions to ask in the review meeting' in point 3) that were sufficiently answered in the meeting.
- 6. Comment:
  If desirable, the reviewer or developer can add comment to the items on the form.

Definition tables:

- For fault severity the following table was used.

Minor	**Major**	**Fatal**
Faults that would affect only the non-functional aspects of the software element	*Faults that would result in failure of the software element(s) or an observable departure from specification*	*Fault that results in the complete inability of a system or component to function*
Aesthetic problems such as: • misspelling • output formatting problem	• system refuses legitimate transactions • system produces redundant outputs with small impact on performance	• system fails • system executes wrong transactions • system loses transactions

- For alternative detection the following table was used.

Unit test	Separate testing of a (separately testable) component
Acceptance test	testing to determine whether a system satisfies its acceptance criteria
Integration test	testing of which components are combined and tested to evaluate the interaction between them
Installation	customisation and installation of the system
Field (test)	operational use of the system

CODE REVIEW CHECKLIST

1. Are the comments added to the modules and procedures sufficient?	
2. Are the comments useful? Do they agree with the code?	
3. Does the product match the coding standard?	
4. Is the code clear and easy to understand?	
5. Are meaningful names used that won't get confused?	
6. Are there redundant code segments?	
7. Are all variables and parameters being used?	
8. Was 'Check programme' used to check the code?	
9. Does the code use local rather than global variables whenever possible?	
10. Is all memory allocated (via 'new') freed again (via 'delete')?	
11. Is all memory allocated(via 'malloc') freed again (via 'free')?	
12. Does each CASE statement contain a 'default' switch?	
13. Are all variables initialised before use?	
14. Are constants used whenever possible?	
15. Is the branch taken the correct way on logical tests?	
16. Will the code work correctly under the loop boundary conditions?	
17. Were any EXIT commands used?	

What other items should be included in this checklist? Which should be eliminated?

10.6.3 Example feedback session report of the RITME project

This section contains the final feedback session report of RITME.

FEEDBACK SESSION REPORT		
Project Name: Project B	Project No: QAxxxxxx	Date:
Meeting Date:	Present: xxxxxxx	
Author: xxxxx	Absent: xxxxxxx	
Subject: Final feedback session for the RITME measurement programme		

Introduction

This is the report of the final feedback session of the RITME measurement programme. It was the objective of this session to try to answer all original GQM questions and by it reach the goal. A clear learning process had been visible within the group on reviews and their effects. This report had two purposes:

1. report on the final feedback session;
2. capture the learning points of the reviewing programme, to be spread to other engineering groups whenever they are interested in the results.

The structure of this report equals the structure of the original GQM plan. First the goal of the measurement programme is presented again:

Analyse:	the effectiveness of structured reviews
for the purpose of:	understanding
with respect to:	- the detection of faults - the learning ability of the technique
from the viewpoint of:	the project team
in the following context:	project B

To give some more explanation to this goal, what the characteristics of reviews are related to their detection capability is studied. At the same time the way in which reviewing contributes to group-learning and individual learning is also studied. Finally, it is added that a feedback process as implemented within this programme also contributed to establishing a team-spirit. It also created self-critique within the team, which resulted in a self-evaluation of the team during a feedback session.

General conclusions on measuring reviews

The reviews during the development process turned out to:

- be an efficient aid to detect faults;
- improve the uniformity and maintainability of code;
- increase communication and understanding within a development team;
- require five minutes per page;
- detect more than one fatal/major fault per hour;
- detect 7.5 faults per hour;
- should be done on documents smaller than 50 pages;
- are more effective on documents of a maximum of 20 pages;

- are four times more efficient compared to testing;
- should be done for one hour a week per engineer;
- should be supported by measurement to maintain continuation.

Conclusions regarding Q.1: What are the baselines (rules of thumb) for executing reviews?

The associated effort of executing reviews was stated as follows:

- Reviewing detects faults with a speed of 7.5 faults per hour.
- Divided over severity this is:
  - fatal and major: 1.25 faults per hour;
  - minors: 6.25 faults per hour.
- Many minors are detected in a review, which improve readability, maintainability, and create a generic coding style.
- Several types of minors are detected. Classification of those could be helpful to show detailed effectiveness of reviews.
- Reviews are useful to make sure that coding standards are applied, and maintained.
- When reviewing speed is faster than two minutes per page, results get reasonably lower.

Alternative detection:

- Comparing reviewing to testing, the technique appears to be four times more efficient, considering fatals and majors.
- Comparing reviewing to testing, the technique is over six times more efficient for minor faults.
- Alternative detections for faults found in reviewing are mainly 'unit test' and 'system test'. Although these findings were concluded to be incorrect during feedback sessions, since installation and field-test *do* find faults. It was noted that a 'never' category should be added.

Planning reviews:

- Reviewing takes on average 20 minutes per week per engineer.
- This depends on development phase. During weeks with intensive reviewing each engineer spends 1.5 hours on reviewing.
- Each hour of reviewing needs 15 minutes to repair the faults found.
- On average, reviewing takes three minutes per page.
- Cyclomatic complexity did not give a better indication for estimating review length, but this was also caused by problems with the calculation tool.

Document characteristics:

- Reviewing design documents in text processor format takes twice as long: six minutes per page.
- But also twice as much faults are found per page. For code 0.2 faults are found per page, for designs 0.4 faults are found per page. (Note: literature describes a rule of thumb of 10 faults per page, when 15 minutes per page are spent on reviewing).

Influences from subsystems:

- Subsystems appear to give differences in reviewing effort. It is expected that this is caused by characteristics of that sub-system. Also, object orientation requires more time than sequential code. More detailed analyses have not been performed.
- The subsystem characteristics should be examined, as should their relation to review effort.

Conclusions regarding Q.2: What are the characteristics of a document, to get optimal effect from a review?

Size:

- Documents with more than 50 pages are unsuitable for reviews.
- Results show a drop in review performance for documents larger than 20 pages.

Language:

- Both text processor and CPP documents find 4.0 faults per hour.
- However, text processor documents require twice as much effort per page, and twice as many faults per page are detected for text processor documents.

Subsystem:

- Results of subsystems differ, but underlying causes are not clear yet.
- Further analyses should be done, based on more detailed subsystem characteristics.

Complexity:

- Results of comparing cyclomatic complexity were disappointing, mainly caused by trouble with the calculation tool.

Conclusions regarding Q.3: What are the acceptance criteria for the quality of the document, after a review?

This question appeared to be a little over-ambitious. During this project we learned that statistical acceptance criteria for documents after reviewing seem unrealistic. Both reviewer and developer have enough insight to accept or reject a document based on a subjective quality assessment.

Conclusions regarding Q.4: Are reviews a useful and efficient method for training new engineers?

Regarding the learning curve of new engineers:

- New engineers learn most from reviewing code from experienced engineers.
- Level of understanding after a review is sufficient (score: 6), not depending on whether the reviewer or developer is experienced or not.
- During the first six months most reviews were carried out by experienced engineers that reviewed documents from new engineers. After six months this changed, then the new engineers were reviewing each others documents.
- New engineers learn more from other new engineers after six months of reviewing.

- Training should subsequently consist of:
  - courses to provide relevant theory and general overview;
  - reviews to learn system details;
  - on-the-job training providing practical experience.

Learning effect of holding review meeting:

- No data was available on this subject, however, review meetings were experienced as an important means for learning and communication.

Difference between new and experienced engineers:

- Experienced engineers review two times faster than new engineers.
- Experienced engineers find twice the number of faults as new engineers.

Action points resulting from the measurement programme

The following action points were formulated during the measurement programme:

1. Define a new measurement programme focused on project management, including time spent and fault management. This measurement programme should also cover reviews.
2. Study subsystems in more detail, to clarify the various relations stipulated in this document.
3. Promote (this kind of) reviewing to other engineering departments, preferably in department '…'.

10.7 Questions and assignments

10.7.1 Questions

1. What are the two most important purposes of reviews in the RITME measurement programme?
2. Which metrics of the GQM plan, are not included in Figure 10-1, however, could well have been included?
3. What is the main difference between the measurement support system of Chapter 9 and of Chapter 10?
4. How many defects are detected per hour on average in software reviews?
5. Figure 10-8 contains a comparison between reviewing of software and two tests of software. Which mistake can be made in comparing these data regarding defects found, considering that reviews and tests were performed on the same product?
6. What can be concluded regarding review effectiveness and document size, considering Figure 10-10 and Figure 10-11?
7. Figure 10-13 illustrates learning effects. What was concluded from this figure and how does this conflict with what happened six months later?

10.7.2 Assignments

The GQM plan of RITME also contained more detailed questions raised by project team members.

1. Do you consider that the abstraction to higher level questions was done appropriately? Do the questions, for example, meet the criteria given in Chapter 6.
2. Take the final feedback session report and identify which questions in the GQM plan can be considered over-ambitious, also explain why?

'There cannot be a crisis next week. My schedule is already full'
Henry Kissinger

11 Case C: interrupt measurement

11.1 Description of project C

This chapter describes the results of an interrupt measurement programme. This measurement programme investigated the causes and effects of interrupts during software development work. An interrupt is defined as a: *distortion that makes a developer stop his actual activity to respond to the initiator*. Even though many of the interrupts were necessary, they still caused problems for the interrupted engineer. An interrupt does not only stop an engineer from working, but also implies additional (often not planned) work. Eventually, this may cause project planning problems, stress, quality problems, or delay of delivery dates.

The project team of this measurement programme was dedicated to develop and maintain the central control unit in a fuel dispenser of a petrol station. This control unit was termed a 'calculator' and controls all fuel transactions, from activating the fuel pump and valves, to authorisation requesting and transaction processing to the cashing or payment device. Such calculators consist of about 50,000 lines of source code, about equally divided over the programming languages C and assembler. The project team developed and maintained both hardware and software in parallel and consisted of two hardware engineers, two software engineers and their manager. At the time of the interrupt measurement programme the team was maintaining a new (lower cost) increment of a calculator, which was developed in a period of six months.

This measurement programme was started because the development group experienced problems with interrupts, mainly reflected by planning problems and project delays. The aim of the programme was to achieve basic understanding of the frequency of interrupt occurrence and interrupt causes. Based on that level of understanding it was expected that interrupts could be planned, causal analysis could be performed, and corrective actions could be taken.

Most software measurement literature warns against, or even forbids, measuring people (Goodman, 1993). Although one should always consider the delicate nature of measuring people factors, in this case it is not possible to avoid measuring individuals. Perry reports

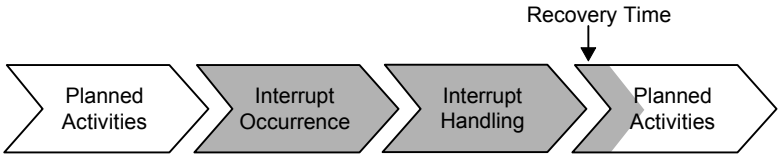


Figure 11-1: Model of an 'interrupt'.

Aspects of Interrupts	
Positive Aspects	**Negative Aspects**
• Part of developer's task	• Not a part of developer's task
• Supports problem solving	• Does not prevent problems
• Stimulates social interaction	• Requires extra 'interrupt' effort
• Random communication essential for development work	• Causes projects to delay
• Provides overview of current status development work	• Disturbs concentration of developers
Aspects of Measuring Interrupts	
Positive Aspects	**Negative Aspects**
• Creation of interrupt awareness	• Discourages 'appropriate' interrupts
• Establishment of 'improvement' culture	• Requires additional effort (but profits can be gained)
• Visualisation of interrupt causes and effects	• Measuring human aspects is 'not allowed'

Figure 11-2: Example of positive and negative aspects of interrupt measurement

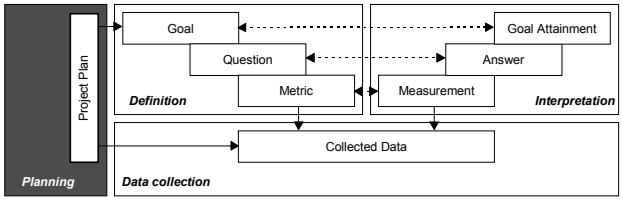
from his research in development organisations that '*people are willing to be observed and measured if you take proper precautions'* (Perry et al, 1994).
Some positive and negative aspects of interrupts learned from this measurement programme are described in Figure 11-2.

No consensus exists on whether interrupts are in the end positive or negative. Although many developers consider interrupts part of development work, interrupts are not always positive. One of the strengths of interrupts is their ability to solve development problems. However, this is also a weakness since problems must be prevented, which is not done when an organisation relies on the interrupt mechanism. Measuring interrupts results in awareness of the negative aspects of interrupts. Negative aspects appear to decrease when this interrupt awareness increases, because initiators of interrupts start considering the importance of each interrupt.

The results presented in this chapter are based on the complete measurement programme. Results are not only the measurement data, but also conclusions drawn by the project team, or action taken based on the measurements. The Interrupt measurement programme will be described by the four phases of the GQM method: planning, definition, data collection, and interpretation.

This chapter also includes the practical deliverables of this programme, such as the GQM plan, data collection forms and feedback charts. Deliverables that might be reused in similar measurement programmes are included in the last section of this chapter. More information on interrupt measurement can be found in the article 'Interrupts: just a minute never is' (Solingen et al, 1998).

11.2 Planning



An independent GQM team already existed in the department. This team discussed with the project team what the role could be of the measurement programme in the context of the organisation's improvement programme. By means of a refinement method from business goals into operational improvement goals, the measurement goal was selected. It was decided to focus on a problem that the project team struggled with: Interrupts. The necessary training for the project team was given by the GQM team, which already had experience with the practical application of GQM. The planning phase delivered a project plan.

Before measurement goals could be specified, it was identified which improvement areas, problems, or organisational (business) goals were relevant. We used a structured brainstorming session to identify the improvement areas. In this brainstorming session all the people involved in the measurement programme were present. It was started at a company wide level and concentrated on what the purpose of the organisation is, focused towards the specific team, and finalised by ending on a level of specific problems or improvements. We applied the 'seven questions' approach for Goal-oriented measurement (based on Parker et al, 1988) that proved to be applicable for such sessions:

1. What are the strategic goals of your organisation?
2. What forces have impact on our strategic goals?
3. What are your strategies to reach the strategic goals of your organisation?
4. A. How can you improve your performance?
   B. What are your major concerns (problems)?
5. What are your improvement goals?
6. How will you reach your improvement goals?
7. A. What are possible measurement goals
   B. Which measurement goals will you take?

We customised these 'seven questions' to the situation of this particular project team. An improvement goal definition session was held to identify the need for a measurement programme, to instruct on the application of the software measurement programme, and to define the improvement goals. The (summarised) answers that were formulated by the project team are:

1. What is the objective of the project team?
   - The group is responsible for all functionality that is available between the dispenser and the shop.
   - Development, maintenance, knowledge acquisition, training and support.
   - The group must anticipate the market (customer) and technology.
   - The group has a central position between all departments.

2. What is the function of (software/hardware) development in the project team?
   - The realisation and implementation of required functionality.

- Optimisation in terms of reliability (serviceability), efficiency (manufacturability) and extendibility (maintainability).
- The realisation of significant cost reduction

3. What are the strategic goals of the project team?
   - Visibility in the development process.
   - To guarantee the reliability (quality) of products
   - Cost reduction.
   - Analyse and optimise the distribution of functionality in products.
   - To implement connectivity to the different devices

4. What are possible performance improvements?
   - Improve effectiveness of testing (eg more testing time, external testing and end user testing).
   - Reduction of failures in products after release.
   - Improvement of specifications.
   - Improvement of communication and information.
   - Management of time and in particular of interrupts.
   - Management of responsibilities.
   - Reduce the impact of organisational changes.

5. What are the major concerns (problems)?
   - The project members identified that the major problems were already defined in question 4.

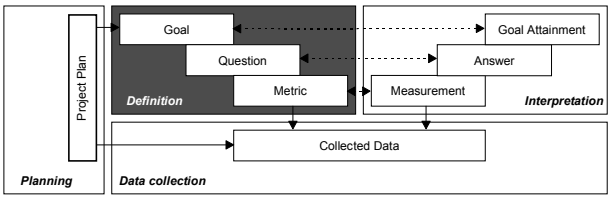
6. What are possible measurement goals?
   - Effect of interrupts on work and planning, types of interrupts.
   - Classifying interrupts and their effects.
   - Origin of failures after release and the reasons for not detecting failures before release.
   - Cost reduction as a cause for faults.

7. Which measurement goals will be chosen?
   - Understanding interrupts.

The project was characterised by filling out a characterisation questionnaire. This questionnaire included both quantitative and qualitative questions. The results of the characterisation were used to create the project plan for this specific measurement programme. The characterisation was carried out prior to the session in which the improvement areas and measurement goals were selected.

This measurement programme did not support a specific project, but supported a department. The people in this department are, however, referred to as ‘the project team’. This project team develops components consisting of hardware (electronics) and software. However, the project team also gives support to the national representatives on new and old products.

11.3 Definition



During the meeting on defining the improvement goal, the measurement goal was defined as well. Restated in a goal template the measurement goal was:

Analyse:	interrupts and their effects
for the purpose of:	understanding
with respect to:	- impact on schedule - the cost/benefit of interrupts
from the viewpoint of:	project team
in the following context:	project C

The refinement of the measurement goal into question and metrics was done using the knowledge acquisition techniques described in Chapter 4. However, in this case no individual interviews were held, but the project team was divided in groups of two to three persons for the definition of questions, metrics and hypotheses. Since the project team consisted of hardware and software engineers, the groups were so organised that at least one hardware engineer and one software engineer were in each session. The question session with the project manager was held separately since he might have a different viewpoint than the engineers.

As described in Section 6.2.4, questions should be described on an intermediate level between the goal, and the metrics. The questions that were formulated during these question sessions appeared to be of a practical but too detailed level. It was found difficult to define questions on a more abstract level. The GQM team decided to perform this abstraction, and let the project team define the practical questions they found relevant. Each session was started by repeating the concepts of GQM, summarising the activities that were already performed, and explaining the activities that are planned. Also the purpose and structure of each session was explained. After each question session two documents were made:

1. Preliminary GQM plan. Describing the goal to question refinement, and already abstracting some of the questions to a more abstract level. An initial abstraction sheet was also included.
2. Report on the session. This report was distributed to each project team member that was present during a session. Abstracting some of the questions to a more general level, was already done in these reports, creating the possibility for the project team to respond to these changes.

Abstracting all the questions to a general level was necessary because the questions were formulated as metrics (for example: ‘How many interrupts are caused by documentation issues?’). The part of the GQM plan that describes the goals, and the refinement to question for this measurement programme is included in Section 11.6.1. The question definition sessions were effective for defining the measurement questions. Following the question

definition sessions, the preliminary GQM plan was developed, that described all questions and their relation to the measurement goal. These questions are:

- What is the distribution of interrupts over particular factors, related with the occurrence of an interrupt?
- What factors influence treatment and effort for handling an interrupt?
- Is prevention of interrupts possible?
- Should we change our reserved effort for interrupts?
- What is the influence of interrupts on the work that was interrupted?
- What are the total costs of interrupts?

The next step could now be performed: defining metrics that should provide answers to these questions.

Metrics had to be defined that provide the information that is needed to answer each question. Also hypotheses were needed from the project team on expected measurement results. During the sessions, the project team supported the definition of the metrics and in according classifications, and stated their hypotheses. These sessions were also supported by on-line display on the overhead projector, creating an interactive group-working environment that was suitable to formulate definitions by a group. The hypotheses were given by the project team members on a numerical scale (eg 0 to 10, 1 to 5 or 1 to 100) and those numbers were recalculated to percentages. This was done to give all project members the possibility to define hypotheses in their personally preferred way. Recalculating these hypotheses to percentages gave the possibility to compare them between the group members.

Refining the questions to metrics was not very difficult, because the questions were abstracted from 'metric-like' questions, so the process of defining metrics was already carried out implicitly. The definition of influencing factors and hypotheses regarding the metrics was done in these sessions as well. After the goal formulating sessions, question sessions and metric sessions, the GQM plan was completed, describing the refinement from goals to questions to metrics. For every question it was defined which metrics had to be collected in order to answer the particular question. Both direct metrics (measuring only one attribute), and indirect metrics were defined. An additional layer was added to the GQM plan defining all direct metrics that had to be collected to construct all indirect metrics. For the direct measurements, the following issues had to be identified:

- When they should be measured?
- What possible values their outcomes could have?
- What they would actually measure (textual description)?
- What medium would be used for data collection?

These issues were addressed in the measurement plan. The measurement plan also presented the data collection media (forms, procedures and tools). When the measurement plan and GQM plan were constructed and approved, measurement started.

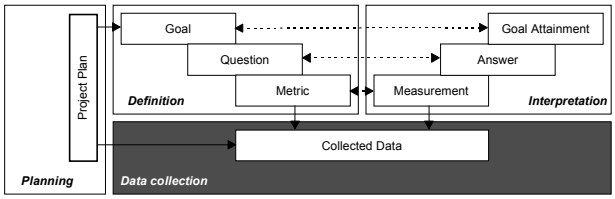
As described in Section 6.3, the GQM approach applies software development process and product models for checking the GQM metric definitions. The availability and development of these models in parallel with the definition of goals, questions and metrics appeared to be very useful. Through a model on interrupts it became clear, how interrupts could occur and how they could be handled. It was also assumed that the availability of the models prevented many problems, because the data collection forms and procedures could

be cross-checked. The availability of these models on software development offer the possibility to apply a second source of relevant information. The first sources are the participating project members. However, when the project members were focusing on a certain problem (or goal), it was difficult to extract all their knowledge, and at the same time maintain a global overview. By applying formal models of software development processes and products, this missing overview was compensated for. The models should be suitable for defining metrics, and should, therefore, focus on both processes and products.

Checking the GQM plan against the software development process models was continuously done during the definition of the measurement programme. This provided the possibility to discuss possible inconsistencies immediately with the project team during each session. The models also provided a framework to the GQM team when they were preparing the various sessions.

During the definition of the GQM plan there was a continuous focus on possible problems with the analyses of the data. The GQM plan should describe how to analyse measurement data, in order to answer questions of the project team, and/or to conclude on a measurement goal.

11.4 Data collection



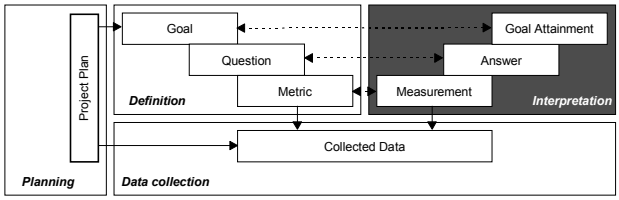
The data collection forms were developed in parallel with the development of the measurement plan, and iteratively enhanced by the GQM team. After some adjustments to the forms, they were agreed upon, and a measurement kick-off-meeting was held. During this meeting the GQM plan, the defined models, and the data collection forms and procedures were explained and presented to the project team. It was decided to start interrupt measurement the week after.

The data collection resulted in a significant amount of data (more than 1000 data points were collected in the first week).

The MSS for handling the data was developed using a spreadsheet tool. Manual data collection forms were used by the project team, which were manually entered into the metrics base by a member of the GQM team. The data collection form was designed in such a way that the registration of an interrupt by a project member would only take a couple of seconds, already in the perspective that many interrupts were expected.

While entering the data into the metrics base, data correctness and completeness was checked manually. The MSS was developed in parallel with the first measurement period. Once the MSS was completed, the first feedback session could be held. No linking features were included in this MSS, because no knowledge or experience existed at that time.

11.5 Interpretation



This section describes the results of the measurement programme as discussed in the feedback sessions. The measurement programme was started to identify the influence of interrupts on schedules, the cost of interrupts, and the causes for interrupts. The possibilities to manage, decrease and plan interrupts were also investigated.

After six weeks of measurement it already became clear that the number of interrupts doubled the expected amount, and a large percentage of them were caused by personal visits (55%). Personal visits require more effort compared to telephone or e-mail. The average effort per interrupt for personal visits were 13 minutes, for telephone 10 minutes, and for e-mail 8 minutes. The project team defined three action points:

- Reduce the number of personal visits.
- Promote applying e-mail, at least for less urgent interrupts.
- Create interrupt awareness among the initiators in order to make clear to initiators that interrupts are sometimes a burden.

The department spent approximately 20% of its total time on handling interrupts. Measurement results identified that 25% of all interrupts were about products or tasks that the department was not responsible for.

It appeared that due to the measurement activities, project members and initiators of interrupts became more aware of the impact of interrupts. This already caused some improvements in the occurrence of interrupts in the department. Initiators started to balance the importance and urgency of an interrupt before disturbing a project member.

The first feedback session was a bit disappointing since no real conclusions were drawn, or discussions were started. The project team appeared to be a bit passive in interpreting the analysis material. In the second feedback session the project team was challenged significantly. A list of observations and statements on the data analysis slides were created and handed over to the team members a week before the feedback session. Together with the statement and observation list each team member received:

- a set of personal data graphics;
- a one page overview of his personal interrupt data;
- a set of personal interrupt data overviews (one page each) of the other team members;
- a set of data graphics and a one page interrupt data overview of the whole group.

Instead of showing a lot of data graphics, the list of observations and statements was discussed to structure the feedback session. In this meeting there was much more discussion, and several clear action (improvement) points were defined. The general impression of all attendees about the meeting and the delivered data was positive. Remarks on the observations and statements in the feedback sessions were:

- Interrupts were perceived as being useful and not negative. Interrupts provided the possibility to remain informed about other activities and were necessary to run the business. Also interrupts on products or tasks the dispenser electronics group were not responsible for, were often regarded as useful.
- It was stated that the measurement programme influences the behaviour of people. Several examples were given of people that became cautious about interrupting team members. Therefore, informing the rest of the organisation about the results of the measurement programme and explaining the effects of the interrupts, would be a clear action point for the next period.
- Persons visiting caused a majority of the interrupts (55%). The idea was to reduce this number by using other ways of interrupting such as electronic mail. Other ideas would be gathered during the next period. The fact that 55% of all interrupts were visits, did not imply that those interrupts could have been done by e-mail or phone, however, the project team estimated that a certain part of the interrupts could be made in a less disturbing way.
- The rule of thumb used in the department is that 20% of the total available time is spent on miscellaneous issues (including interrupts). Due to the measurements it was identified that already 20% of the total time was spent on interrupts. The idea of the project team was that an additional 10% is spent on other miscellaneous issues than interrupts. When 20% is included in the planning this implies that 10% of the project teams work must be done in spare time (1 hour per day additional effort would be required).
- The maximum level of interrupts is expected to be 10 per day. At higher rates the time between interrupts cannot be used efficiently anymore. Proposals for handling this were for instance: use a ‘do not disturb’ sign, closing the door of the office, or promoting making appointments.
- Level of detail of information contained in the interrupt appeared to have no influence on the handling of the interrupt. This was derived from the measurement data, and the project team confirmed this observation.
- A new, more simplified interrupt registration form was suggested and it was agreed to also use the form to add ideas for corrective actions on specific interrupts.

Three practical conclusions from the interrupt measurements are listed:

1. If more than ten interrupts occur during a day, the time between the interrupts is too short for development work. This is in line with the results from Perry, which show that developers perform work in blocks of two hours (Perry et al, 1994).
2. Project plans used to include 20% slack for unexpected activities. It appeared that more than 15% is already spent on interrupts, which could be an explanation for planning delays. One possible solution is to increase slack in planning, or to reduce the number of interrupts.
3. Interrupts during meetings should be reduced to zero. Such interrupts turned out to be very expensive because discussions are blocked for so many people.

Improvements were achieved regarding interrupts. For example, the number of interrupts during meetings decreased to almost zero. For the remaining interrupts during meetings, the effort was reduced by 50%. Also, the number of interrupts on documentation issues was reduced by 80%, mainly by updating and distributing documentation.

To influence the initiators of interrupts, a focus was created to establish a company-wide 'interrupt awareness'. For this purpose, the measurement results were distributed over the organisation, through presentations, posters and on-line availability of relevant information. This interrupt-awareness should contribute to a more efficient delivering and handling of interrupts, from the point of view of both initiator and developer.

The hypotheses showed that the project team expected that a lack of information details would cause an interrupt to take longer. They assumed that 50% of all interrupts had insufficient details. However, measurement showed that 90% of all interrupts were formulated with sufficient details.

The feedback session material will be discussed here according to the following questions on interrupts:

- What is the workload of interrupts?
- Through which medium do interrupts occur?
- Who causes interrupts?
- What is the recovery time after interrupts?
- How are interrupts handled?
- What are the underlying reasons for interrupts?
- Which actions can be taken to solve the interrupt problem?
- To which products are interrupts related?

What is the workload of interrupts?

Looking into the effort spent on interrupts, we found that interrupts require an average effort of 15 minutes for the initiator to sufficiently handle an interrupt. Developers spent around 1.5 hours per day on interrupt handling, consuming 20% of total time available for development work. This was similar to the percentage found in other empirical work (Perry et al, 1994).

The project manager always included 20% slack in the planning for unexpected activities, including organisational communication, training, social talks and interrupts. However, the measurements showed that this slack was already fully consumed by the interrupts. This indicated a possible cause for planning delays. Action was taken to increase interrupt awareness, and to try to decrease the number of interrupts in the department.

Through which medium do interrupts occur?

Figure 11-3 shows the distribution of interrupts over the three possible media: personal visits, telephone, or e-mail. Of all interrupts, 95% were delivered by personal visits or telephone calls, whereas the rest of the interrupts are delivered by e-mail. Please note that interrupts delivered by personal visits or telephone calls occur to the developer in random order with 'number 1' priority (whereas only 25% of all interrupts are classified as urgent)! This meant that the initiator of the interrupt determines the moment of interruption, and that the developer has to respond immediately to satisfy the initiator. However, e-mails could be handled at a later moment suitable to the developer. Furthermore, personal visits and telephone calls needed more handling effort from the developer than e-mails. We found the reason for this to be that interrupts delivered by e-mail are usually well formulated by the initiator.

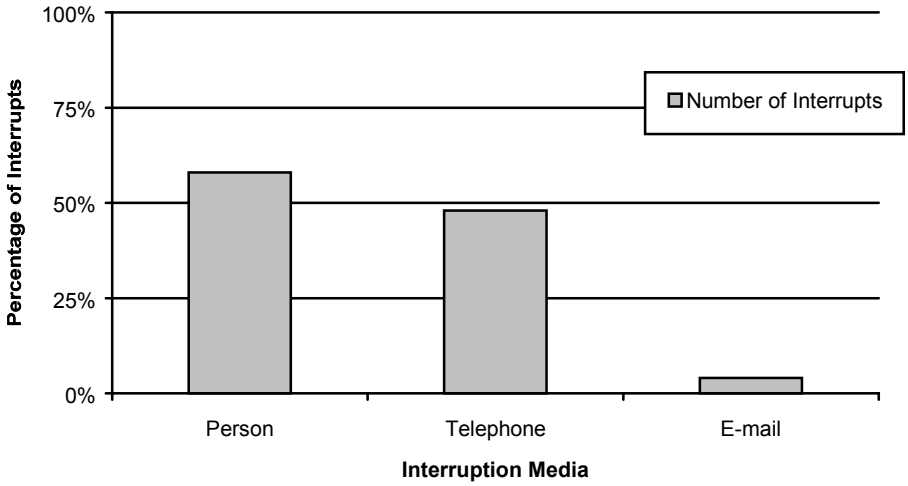


Figure 11-3: Interrupt occurrence by medium.

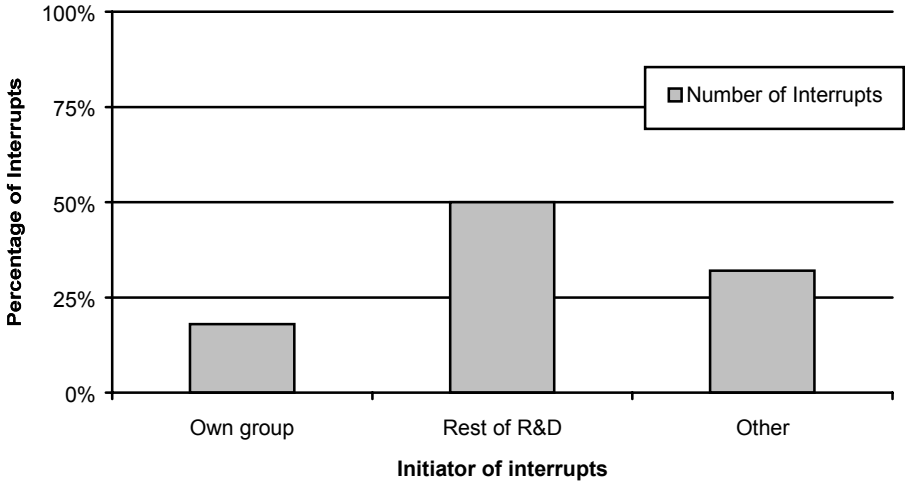


Figure 11-4: Initiators of interrupts.

Who causes interrupts?

Figure 11-4 shows the distribution of interrupts over the types of initiators: own group, other R&D groups and other groups outside R&D. The measurements showed that the R&D department (including their own group) were the 'number one' initiator, contributing to 70% of the total amount of interrupts. The developers found this quite remarkable, as they expected that other departments would be the main source of interrupts. This indicated that

interrupts were a inherent R&D problem, and that solutions for this problem should primarily be found within the R&D departments.

What is the recovery time after interrupts?

In the measurement programmes we initially included a metric on the '*recovery time'* after an interrupt. Unfortunately, it appeared to be too difficult to measure this and after the first measurement period we omitted this measurement from the programme. However, we could observe that the recovery time was primarily a problem when an interrupt occurred during actual programming work, and less when it occurred, for instance, during documenting or meetings. It was assumed that the level of concentration that was required for programming caused this.

Other sources in literature reported that the recovery time after a phone call is at least 15 minutes (DeMarco and Lister, 1987). Even though we could not measure this exactly, we had the same impression. One of the conclusions from the measurements was that when more than ten interrupts occur during a day, the remaining time for planned work became insufficient. Taking these ten interrupts a day, and the measurement that an interrupt required approximately 20 minutes to handle, it then seemed reasonable that the recovery time from an interrupt was roughly 15 minutes, explaining why the time left was useless for development work.

How are interrupts handled?

Figure 11-3, on interrupt media, indicated that interrupts occurred for 95% in such a way that it was difficult for developers to avoid the interrupt *occurrence*. However, it would have been possible to postpone the *handling* of an interrupt. Of course this depended on the type of interrupt. Most interrupts were minor questions or requests which developers preferred to handle immediately. Also, some interrupts could not be postponed because of the urgency of the initiator.

During the measurement programme it was measured whether interrupts were handled immediately, or whether handling was postponed. It appeared that 90% of the interrupts were handled immediately and that only 10% were postponed. The interrupts that were postponed needed three times more effort to handle. This might also very well be the reason why they were postponed in the first place.

It was expected that interrupts were also postponed because of a lack of information details. Both organisations assumed at the beginning of the measurement programme that 50% of all interrupts would have insufficient details. However, the measurements showed that 90% of all interrupts were formulated with sufficient information.

What are the underlying reasons for interrupts?

Figure 11-5 illustrates the reasons of interrupts over a number of classes. Discerned were: knowledge or experience exchange, social issues, documentation, organisational issues and other.

Issues related to knowledge or experience caused approximately 30% of all interrupts. Furthermore, 20% were caused by what was called documentation issues, such as inadequate documentation, non-existing documentation, and review of documentation. About 10% of all interrupts were caused by organisational shortcomings, as initiators were unaware of the exact responsibilities of the particular developers.

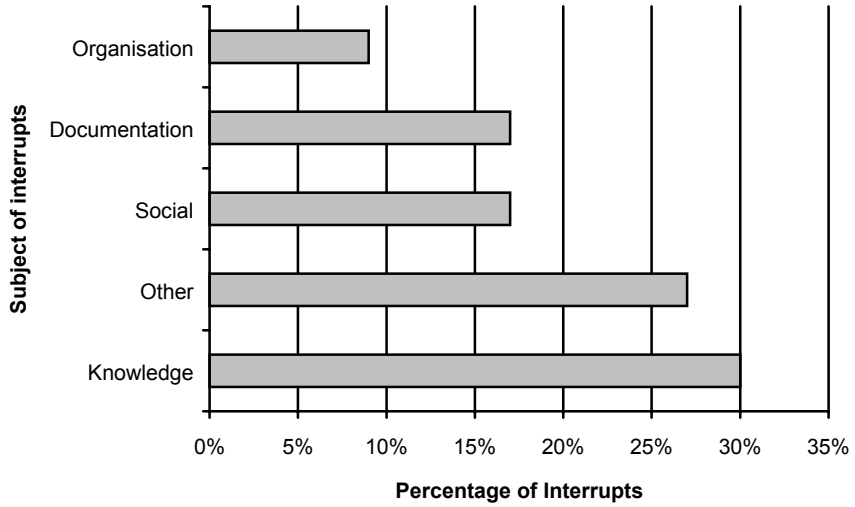


Figure 11-5: Underlying reasons for interrupts.

Which actions can be taken to solve the interrupt problem?

The developers considered interrupts to be 'inappropriate', if an interrupt was delivered to the wrong person, or if the initiator himself could have handled the interrupt. It appeared that approximately 30% of all interrupts were experienced as 'inappropriate'. To improve 'interrupt-efficiency' it was first tried to minimise these 'inappropriate interrupts'. It appeared that many inappropriate interrupts were caused by unclear organisational responsibilities. Therefore, more effort was spent on communicating responsibilities of employees throughout the organisation.

To influence initiators of interrupts, a company-wide 'interrupt awareness programme' was established. For this purpose, the measurement results were spread throughout the organisation, by presentations, posters and on-line availability of relevant information.

Based on the notion that developers handled interrupts delivered by e-mail more efficiently than interrupts delivered by personal visits or telephone calls, an e-mail policy was established, especially for less urgent interrupts within the department. Urgent interrupts could be delivered to the developer by telephone, however, preferably not by personal visits, because such visits proved to be quite inefficient and would also distract other people besides the developer.

To which products are interrupts related?

Figure 11-6 shows the relation between the products that the dispenser electronics department is responsible for and the number of interrupts. The actual number of interrupts was the total number in the first six weeks of measurement. Because the measurements were done for a relatively short period of time the total number of interrupts could be shown in this chart. When larger periods of time are considered, it is probably better to present the total number per time unit (for example, total number of interrupts per week). The number of interrupts concerning old products (released to the field and in maintenance phase)

appeared to be quite high. The average effort per interrupt for the two calculators (central control unit in a dispenser) appeared to be twice as high as average.

11.6 Documentation of interrupt measurement programme

In this last section of Chapter 11, the GQM plan and the data collection form are described. This section contains no new information on the case of the interrupt measurement programme.

11.6.1 GQM plan

Introduction

In this subsection the GQM plan for the interrupt measurement programme is described. The measurement programme was executed within the department of Project C. Measurement was introduced for a better understanding of current processes, to provide information on the performance and to identify areas of improvement. Subsequently, the measurement goal, questions, metrics and hypotheses will be described. This GQM plan would be used for three purposes:

1. Feedback to the project team members by the GQM team.
2. Definition of data collection, since all direct measurements were defined in the GQM plan.
3. Analyses and feedback of collected data, because the GQM plan identified which questions should be answered.

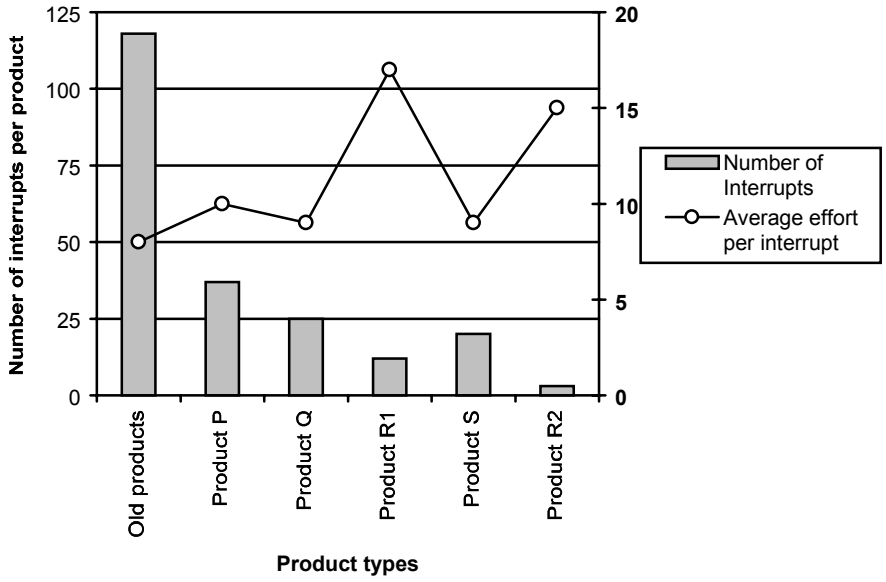


Figure 11-6: Relation between products and interrupts.

Interrupts

An interrupt is an event that disturbs a person during the activity that person is working on. An interrupt exist of three phases:

1. Occurrence of interrupt including a quick decision on whether the interrupt must be handled at that certain moment, or can be postponed/planned.
2. Handling of the interrupt, where the initiator of the interrupt is helped onto a satisfactory level. After handling the interrupt, the interrupt is officially finished.
3. Follow-up of interrupt where the activities are executed that are necessarily involved with the interrupt, but that are not related to the initiator of the interrupt. For example update in a document.

From goal to questions

The goal of the measurement programme was as follows:

Analyse:	interrupts and their effects
for the purpose of:	understanding
with respect to:	- impact on schedule - the cost/benefit of interrupts
from the viewpoint of:	project team
in the following context:	project C

This Goal will be reached when answers can be given to the following six questions.

Q.1 What is the distribution of interrupts over factors, related to the occurrence of an interrupt?

This question focuses on the occurrence of an interrupt. The question will give an answer to the event of the interrupt and the factors that relate to that event. Therefore the question will investigate the relation between the initiator of the interrupt, the subject the interrupt is about, the cause of the interrupt, the treatment of the interrupt and the interruption medium. This question is answered when the relation between all factors involved with the occurrence of an interrupt are clear and values are available that visualise the distribution of interrupts over these factors.

Q.2 What factors influence treatment and effort for handling an interrupt?

This question focuses on the reasons why an interrupt is handled right away or included in a planning (postponed), and the reasons why an interrupt requires more effort. The question will give an answer to the amount of time that is related to the types of influencing factors and the treatment of these specific interrupts. This question is answered when the relations between all these factors and the treatment/effort for the interrupt are known and values are available that visualise these relations

Q.3 Is prevention of interrupts possible?

This question focuses on the possibility to eliminate interrupts that are unnecessary. It should therefore be investigated whether such interrupts exist and, if so, how their occurrence can be prevented. This question is answered when the possibilities for interruption prevention are known and when the amount and effort of interrupts that should not have occurred is visualised by values.

Q.4 Should we change our reserved effort for interrupts?

This question is answered when the amount of time estimated for handling interrupts can be compared to the actual effort spent on handling interrupts.

Q.5 What is the influence of interrupts on the work that was interrupted?

This question focuses on the work that is being interrupted. This question is primarily answered when the overall effort spent on interrupts is known. Values need to be available regarding the occurrences of interrupts and the associated decrease in concentration.

Q.6 What are the total costs of interrupts?

This question investigates the costs of interrupts in relation to the 'normal' work of the project team. This question is answered when the effort spent on different types of interrupts is known, together with other costs that are related to these interrupts.

From questions to metrics

The six questions of the measurement programme are restated below, together with their associated metrics.

Q.1 What is the distribution of interrupts over factors, related with the occurrence of an interrupt?

Detailed questions focusing on practical experience by project team:

- What is the number of interrupts that are not related to the current project (or current work)?
- What is the number of interrupts that have informal contacts as initiator?
- How many interrupts can be scheduled or must be treated immediately?
- How is the occurrence of interrupts distributed over time (during the day and week)?
- What is the relation between interrupts and their initiator?
- What is the number of interrupts that are caused by (unclear or uncompleted) documentation?
- What is the object (product) that the interrupt is about?
- What is the nature of an interrupt?

Metrics:

- number of interrupts for current work;
- number of interrupts for not current work;
- number of interrupts that are treated immediately;
- number of interrupts that are planned;
- number of interrupts per day;
- number of interrupts per initiator;
- number of interrupts per subject of interrupt;
- number of interrupts per interruption medium;
- number of interrupts due to documentation;
- number of interrupts per product;

- number of interrupts per subject;
- number of interrupts due to unclearness of organisation;
- number of interrupts per cause.

Direct Measurements:

- initiator of each interrupt;
- subject of each interrupt;
- cause of each interrupt;
- object (product) of each product related interrupt;
- effort spent on each interrupt;
- interruption medium for each interrupt;
- treatment for each interrupt (immediately/planned);
- number of interrupts for each day of the week.

Q.2 What factors influence treatment and effort for handling an interrupt?

Detailed questions focusing on practical experience by project team:

- What is the impact of (status of) initiator on the way the interrupt is treated?
- What is the effort spent on interrupts per initiator?
- What is the impact of the communication medium (eg fax, letter, e-mail, personal visit or other) on the way the interrupt is handled and the required effort?
- What is the relation between communication medium and the effort needed for collecting the interrupt information?
- What is the impact of the level of detail in which the interrupt is described on the effort spent on the interrupt?
- What is the efficiency gain, when interrupts are presented with more details?
- What is the impact of the associated object's characteristics (eg age of the product, complexity, documentation) on the way the interrupt is treated and the effort required?

Metrics:

- effort spent on interrupts per initiator;
- effort spent on interrupts per interruption medium;
- effort spent on interrupts by level of detail of interrupt description;
- effort spent on interrupts per subject of interrupt;
- effort spent on interrupts per cause of interrupt;
- treatment of interrupts per initiator;
- treatment of interrupts per interruption medium;
- treatment of interrupts by level of detail of interrupt description;
- treatment of interrupts per subject of interrupt;
- treatment of interrupts per cause of interrupt;
- level of detail of interrupt description per interruption medium;
- level of detail of interrupt description per interrupt initiator.

Direct Measurements:

- initiator of each interrupt;
- interruption medium for each interrupt;

- level of detail of each interrupt description;
- subject of each interrupt;
- cause of each interrupt;
- effort spent on each interrupt;
- treatment for each interrupt (imediately/panned).

Q.3 Is prevention of interrupts possible?

Detailed questions focusing on practical experience by project team:

- What is the amount of (and effort on) interrupts that are caused by (un)clearness of organisation by the initiator?
- How could interrupts have been prevented?
- How many interrupts could have been prevented?

Metrics:

- number of interrupts due to unclearness of organisation;
- effort spent on interrupts due to unclearness of organisation;
- number of interrupts due to unclearness of organisation per initiator;
- number of interrupts due to non existing documentation;
- effort spent on interrupts due to non existing documentation;
- number of interrupts due to incomplete/unclear documentation;
- effort spent on interrupts due to incomplete/unclear documentation;
- number of interrupts due to badly executed follow-up;
- effort spent on interrupts due to badly executed follow-up;
- number of repetitive interrupts;
- effort spent on repetitive interrupts.

Direct measurements:

- Subject of each interrupt
- Cause of each interrupt
- Effort spent on each interrupt

Q.4 Should we change our reserved effort for interrupts?

Detailed questions focusing on practical experience by project team:

- What is the difference between total reserved effort and actual effort for handling interrupts?
- What is the difference between planned and actual time for activities due to interrupts?

Metrics:

- total number of interrupts;
- total effort spent on interrupts;
- average effort per interrupt;
- estimated number of interrupts;
- estimated effort spent on interrupts;
- estimated effort per interrupt;

- planned effort for interrupts;
- actual delay due to interrupts.

Direct measurements:

- effort spent on each interrupt;
- estimated effort for each interrupt;
- date for each interrupt;
- treatment of each interrupt (immediately or planned);
- planned effort for handling interrupts.

Q.5 What is the influence of interrupts on the work that was interrupted?

Detailed questions focusing on practical experience by project team:

- What is the additional effort needed for planned activities due to the fact that these activities have been interrupted?
- What is the level of concentration for the activities that are interrupted?
- How many times are activities interrupted?
- How many interrupts are really inconvenient?

Metrics:

- number of interrupts per activity;
- number of interrupts and associated level of concentration of present activity;
- number of interrupts classified as inconvenient.

Direct measurements:

- work per interrupt;
- level of concentration per activity.

It should be mentioned here that a number of members of the project team proposed to omit this question due to the expectation that measurement would be too complicated.

Q.6 What are the total costs of interrupts?

Detailed questions focusing on practical experience by project team:

- What is the effort spent on handling interrupts?
- What is the cost of interrupts that could have been prevented?
- What is the cost of repetitive interrupts?
- What is the cost of interrupts that should have been handled by people outside the project team?
- What is the cost/impact of not following up and completing an interrupt?

Metrics:

- effort spent on interrupts that could have been prevented:
  - effort spent on interrupts due to unclearness of organisation;
  - effort spent on interrupts due to not existing documentation;

  - effort spent on interrupts due to incomplete/unclear documentation;
  - effort spent on repetitive interrupts;
- total effort spent on interrupts;
- estimated effort spent on follow-up;
- effort spent on follow-up;
- interrupt(s) causing follow-up;
- effort spent on interrupts per treatment.

Direct measurements:

- subject of each interrupt;
- cause of each interrupt;
- effort spent on each interrupt;
- treatment for each interrupt (immediately/planned);
- estimated effort for each follow-up;
- effort spent on each follow-up;
- interrupt(s) causing each follow-up.

Direct measurements

The direct measurements that have been defined for the stated goal, and that will provide enough information to answer the questions are:

- Interrupts:
  - initiator of each interrupt;
  - subject of each interrupt;
  - cause of each interrupt;
  - effort spent on each interrupt;
  - interruption medium for each interrupt;
  - treatment for each interrupt (immediately/planned);
  - estimated effort for each interrupt;
  - date for each interrupt;
  - activity working on per interrupt;
  - level of detail of each interrupt description;
  - object (product) of each product related interrupt.
- Follow-up:
  - estimated effort for each follow-up;
  - effort spent on each follow-up;
  - interrupt(s) causing each follow-up.
- Level of concentration per activity.
- Planned effort for handling interrupts.

Hypotheses regarding the direct measurements

The project leader and project team members suggested different hypotheses for particular measurements. The hypotheses of the hardware project team and the software project team, were roughly the same. When the hypotheses of the project team are combined, the following hypotheses are formulated:

- Production and the project team are the largest initiators of interrupts, together responsible for 30% of the interrupts.
- Students (trainees) are expected to cause 10% of the interrupts.
- Most interrupts refer to operational problems of the product that is currently being worked on (19%), failures (14%), consultancy (13%) and old products (12%).
- The reason that somebody is interrupted (member of project team) are problems with documentation (31%), and his/her knowledge and experience (25%).
- More than 50% of the interrupts require an effort of less than 5 minutes to handle.
- 75% of the interrupts require less than 30 minutes to handle.
- The largest numbers of interrupts are caused by telephone calls (43%) and persons visiting (27%), these two sources together cause 70% of the interrupts.

11.6.2 Measurement plan of interrupts

In this subsection the measurement plan is described. Subsequently, the metrics will be defined regarding:

- occurrence of an interrupt;
- handling of an interrupt;
- follow-up of an interrupt.

All 16 metrics have already been identified in the previous subsection on direct measurements and are only formally defined in this subsection.

Occurrence of an interrupt

A number of metrics were identified regarding the occurrence of an interrupt, including a quick decision whether the interrupt should be handled immediately, or should be postponed/planned.

M.1 Initiator of each interrupt
Classification of persons initiating interrupts.

Values:

MT:	Management
OW:	Own project team
PE:	Product engineering (all mechanical and hydraulic engineers)
MK:	Marketing (marketing people)
IN:	Installation engineering and service
MA:	Manufacturing
QA:	Quality
RD:	R&D
SA:	Sales
ST:	Students
GV:	Governmental institutes
EX:	External (persons outside the company)

Data collection medium:
Interrupt registration form

M.2 Interruption medium for each interrupt
Classification of media that interrupts are done with

Values:

Telephone	Telephone
Person	Direct visit by initiator
E-mail	Computer electronic mail
Fax/letter	Written material

Data collection medium:
Interrupt registration form

M.3 Treatment for each interrupt (immediately/planned)
Classification of the way in which the interrupt is treated

Values:

Immediately	The interrupt is handled at the moment that it occurred
Planned	Handling the interrupt is postponed/planned to another moment

Data collection medium:
Interrupt registration form

M.4 Date for each interrupt
Day, Month, Year, day of the week at which the interrupt occurred

Values:
DD/MM/YY
Day of the week (Mon, Tue, Wed, Thu, Fri, Sat, Sun)

Data collection medium:
Interrupt registration form

M.5 Estimated effort for each interrupt
Effort estimated to be spent on handling the interrupt. Collected only when an interrupt is planned/postponed.

Values:
Estimated effort in minutes

Data collection medium:
Interrupt registration form, estimated time is filled in the planned/postponed cell of that interrupt

M.6 Level of detail of each interrupt description
Classification of levels of detail of the description of interrupts.

Values:
Sufficient information available
Not sufficient information available

Data collection medium:
Interrupt registration form

M.7 Level of concentration per activity
Level of concentration per activity defines if an activity can be interrupted or not.

Values:
Strictly related to the interrupted activity:

Meeting:	High concentration and very inconvenient to be interrupted.
Development:	High concentration
Documentation:	Low concentration

Others: Rest, low concentration

Data collection medium:
Interrupt registration form

M.8 Activity working on per interrupt
Activity that was interrupted

Values:
Name of the activity
Meeting: High concentration and very inconvenient to be interrupted in
Development: High concentration development activity
Documentation: Low concentration writing activity
Others: Rest, low concentration

Data collection medium:
Interrupt registration form

Handling of an interrupt

Handling of an interrupt, where the initiator of an interrupt is helped onto a satisfactory level. After handling an interrupt, it is officially completed.

M.9 Subject of each interrupt
Classification of subjects the interrupt is about

Values:
Existing functionality
New functionality
Products
Consultancy
Organisational subjects
Social

Data collection medium:
Interrupt registration form
Product abbreviations must be entered, when interrupt is about a product

M.10 Cause of each interrupt
Classification of causes that the interrupt is done by the interrupted person

Values:
Documentation
System failures
Wrong system usage
Knowledge and experience
Organisational shortcomings
Job (function) related
Social
Repetitive

Data collection medium:
Interrupt Registration Form

M.11 Effort spent on each interrupt
Effort spent on handling the interrupt, so that the initiator is helped

Values:
Effort in minutes/hours

Data collection medium:
Interrupt registration form

M.12 Object (product) of each product related interrupt
Product relation of interrupt

Values:
Name of the product

Data collection medium:
Interrupt Registration From
Product abbreviation is entered at the subject category (M.9)

M.13 Interrupt(s) causing each follow-up
Each follow-up is caused by (one or more) interrupts

Values:
Interrupt numbers: Date, number

Data collection medium:
To do list
Follow-up must be entered on To do list.

M.14 Estimated effort for follow-up on interrupt
Estimated effort required for executing follow-up if necessary

Values:
Effort in minutes/hours

Data collection medium:
To Do List

Follow-up of the interrupt

Follow-up of an interrupt where activities are executed that are necessarily involved with the interrupt, but that are not related to the initiator of the interrupt. An example is an update of the product documentation.

M.15 Effort spent on each follow-up
Effort spent on executing the follow-up

Values:
Actual effort: Minutes, hours

Data collection medium:
To do list

M.16 Planned effort for handling interrupts
Total effort planned for handling interrupts during the project

Values:
Planned effort: Hours, days

Data collection medium:
Project plan

11.6.3 Interrupt data collection form

The form below illustrates the interrupt data collection form which was used in the interrupt measurement programme. This form is not exactly in line with the GQM plan included before, because it is a second version and some measurements were already omitted from the programme for various reasons.

Name : Date: /

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Interruption Medium	Enter "1" for urgent interrupts (should this interrupt have occured now ?)													
Telephone														
Person														
E-mail														
Fax/Letter														
Initiator of the Interrupt														
Effort for Interrupt														
Immediately handled effort														
Planned: Estimated effort for handling														
Actual effort for handling														
Interrupted Activity														
Meeting (enter number of people present)														
Development (high concentration)														
Others														
Subject of the Interrupt														
Maintenance Team														
Development Team														
Documentation problems														
Products (enter: abbreviation of product)														
Organizational subjects														
Others														

Initiator of interrupts:

MK: Marketing	**PE**: Product Engineering	**RD**: R & D
O : Own project team	**ST**: Students	**IN** : Installation
EX: External	**QA**: Quality	**MA**: Manufacturing
	OT: Others	

Figure 11-7: Relation between products and interrupts.

11.7 Questions and assignments

11.7.1 Questions

1. What is the definition of an interrupt?
2. Which effects in Figure 11-2, are experienced as positive and negative at the same time? Explain why.
3. What do you consider the most important property of a data collection form for interrupt measurement?
4. What is the most apparent difference between phone and visits as interrupt medium, and e-mail?
5. What are the advantages and disadvantages of an 'e-mail only' policy for interrupts?
6. How much effort did the developers spend on interruptions per day? How many hours is that per week?
7. Who was the initiator of the majority of the interruptions?
8. How many minutes is the recovery time from an interruption?
9. Compare the measurement plan of this case with the one of case B in Chapter 10. Which one is better, and why?

11.7.2 Assignments

1. Take the GQM plan of this chapter and make a new chart or table that you could include in the analysis plan for each question.
2. Compare the charts and tables of the previous assignment with those provided in this chapter and analyse the completeness and validity of the results presented in this chapter.
3. The GQM plan of this case contains no abstraction sheet on the interrupt goal. Create this abstraction sheet based on the information in the GQM plan.

'The goal of software engineering is to build something that will last at least until we have finished building it'
Source unknown

12 Case D: effort measurement

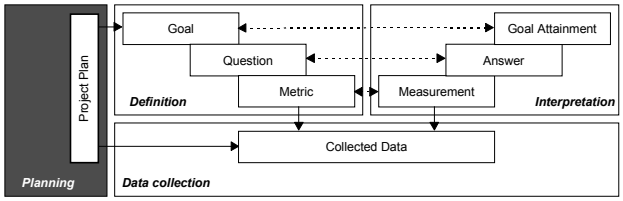
12.1 Description of project D

This chapter describes the results of a measurement programme to support a project team during a reorganisation. This project team had to change from development to maintenance work. To support a project team in its learning process from development to maintenance work, we applied GQM to measure how the change impacted on the workload of the team. This measurement programme was called the SUPSYS project.

The project team that worked on the SUPSYS project contained the same members as in Chapter 10. The project team had already released several versions and increments of the product, and was facing a phase in which support on local customisation should be provided instead of development.

The SUPSYS measurement programme will be described again following the four phases of the GQM method: planning, definition, data collection and interpretation. This chapter also includes the practical deliverables of SUPSYS, such as the GQM plan, data collection forms and feedback charts. These documents can be useful as example for similar measurement programmes.

12.2 Planning



An independent GQM team already existed in the targeted department. This team discussed with the project team what the role could be of the measurement programme in the context of the project team's improvement programme.

The project team used to develop a system for a global market. However, this product is also customised to national requirements in separate distributed teams. Those teams integrated, for example, company logos, languages, national currencies, etc into the system. Major customisations were also done by the national teams, for example, integration with other systems, or connections of the system to national bank cards, air-miles or governmental standards.

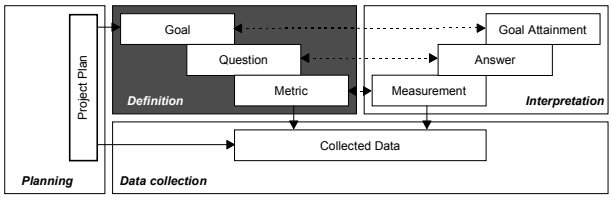
After developing the basic product it was released to those national customisation teams. However, those customisation teams needed support from the original developers. Relevant customisation projects in one country might also be relevant to another country. After the second release of the product, management decided to change the objectives of the team

that originally developed the product. Their focus on development was changed to maintenance, which included giving support to the customisation teams and integrating customisation functionality into the basic product.

This organisational change implied that the team could no longer plan their activities as they used to do. They also became dependent on questions and demands from the customisation teams. In order to support this project team in their learning process from development to maintenance, we applied the GQM method to measure effort spent to learn how the change impacted on the teamwork.

It was not necessary to train this project team on the GQM method, because it was the same team that worked on the RITME measurement programme, which we described in Chapter 10. For efficiency reasons, the project plan was not documented separately, but immediately integrated into the GQM plan.

12.3 Definition



This measurement programme was again defined according to the steps described in Chapter 6. Measurement goals were defined in a session with the project team. Two goals were identified for this measurement programme, these two goals are depicted below.

Analyse the:	time spent on support activities
for the purpose of:	understanding
with respect to:	- the ongoing projects in the national customisation teams - the support given by the team
from the viewpoint of:	the project team
in the following context:	project D

Analyse the:	quality of support
for the purpose of:	understanding
with respect to:	support given by the central department
from the viewpoint of:	national customisation team-engineering
in the following context:	project D

It was decided to focus the attention of the measurement programme on the first goal. The second goal was put on hold and would be addressed later, because the support role was new, so some time had to be spent on support before national customisation teams could be asked for their opinion.

During the first measurement period we intended to focus only on support the activities of the project team. However, it became clear that the team could not make a clear distinction between ‘support’ activities and ‘normal’ activities. Therefore the team decided to broaden the initial measurement goal to effort spent on all activities. This was experienced as a strong point of a GQM measurement programme: as the project team learned from their activities and the measurement programme could be enhanced.

This measurement programme differed from the ones described in the previous three chapters, because the shift in the department also caused the developers not to know what would happen. As the GQM definition phase is particularly based on ‘knowledge acquisition’, we had to adjust this approach. We did hold the interviews to find out the most

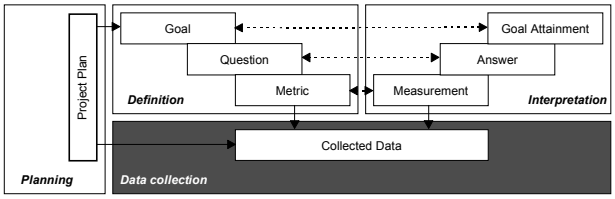
important questions of the engineers and to identify which metrics were necessary. The questions that were defined are:

- What are the main characteristics of the support activities?
- Which projects are ongoing to be executed by the national customisation teams from the viewpoint of the central department?

However, problems arose regarding the identification of influencing factors and especially regarding the definition of hypotheses and process modelling. We realised that this specific measurement programme was started to find out new knowledge, and that we could expect some rigorous changes on the way. Therefore, we did not make a process model for support, nor did we ask for hypotheses, because these would become available during the measurement programme itself.

The definition phase was documented in a GQM and measurement plan, which is detailed under Section 12.6.1.

12.4 Data collection



The data collection and analysis and the different programmes that were used have been illustrated in the figure of the measurement support system (Figure 12-1). A paper data collection form was used to quickly register some notes when the support was requested. At the end of the day the support-requests were entered in the database. Based on this database several queries were defined based on the GQM plan. Those queries were expressed in charts, that were automatically updated by the MSS. Only the final link to the presentation tool could, unfortunately, not be updated automatically. This had to be done by copying the charts to the presentation tool manually.

The database tool was used to gather the data of the support activities. It was possible for members of the project team to enter the data directly into the database. It was also easy to link the data in the database with other applications such as a spreadsheet tool or presentation tool.

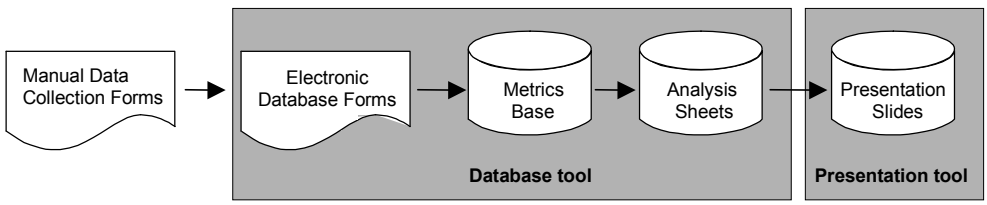


Figure 12-1: Data collection procedures

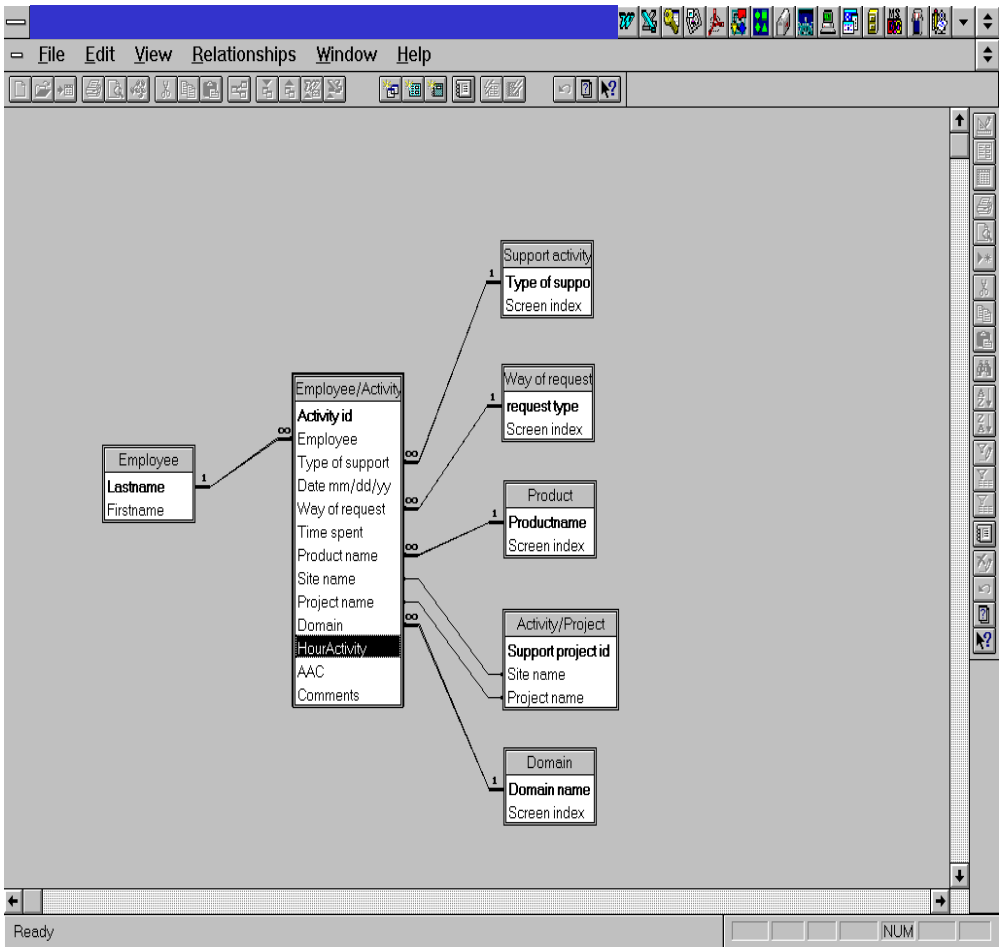


Figure 12-2: ERD model.

An ERD model was designed from the measurements. This ERD model is presented in Figure 12-2. The electronic form that was used to enter data should be easy and quick. It was developed with these two things in mind. The form was just big enough to be displayed on a full-screen. So no scrolling was needed to enter the data. The database and form was improved several times. The electronic data entry form is displayed in Figure 12-3.

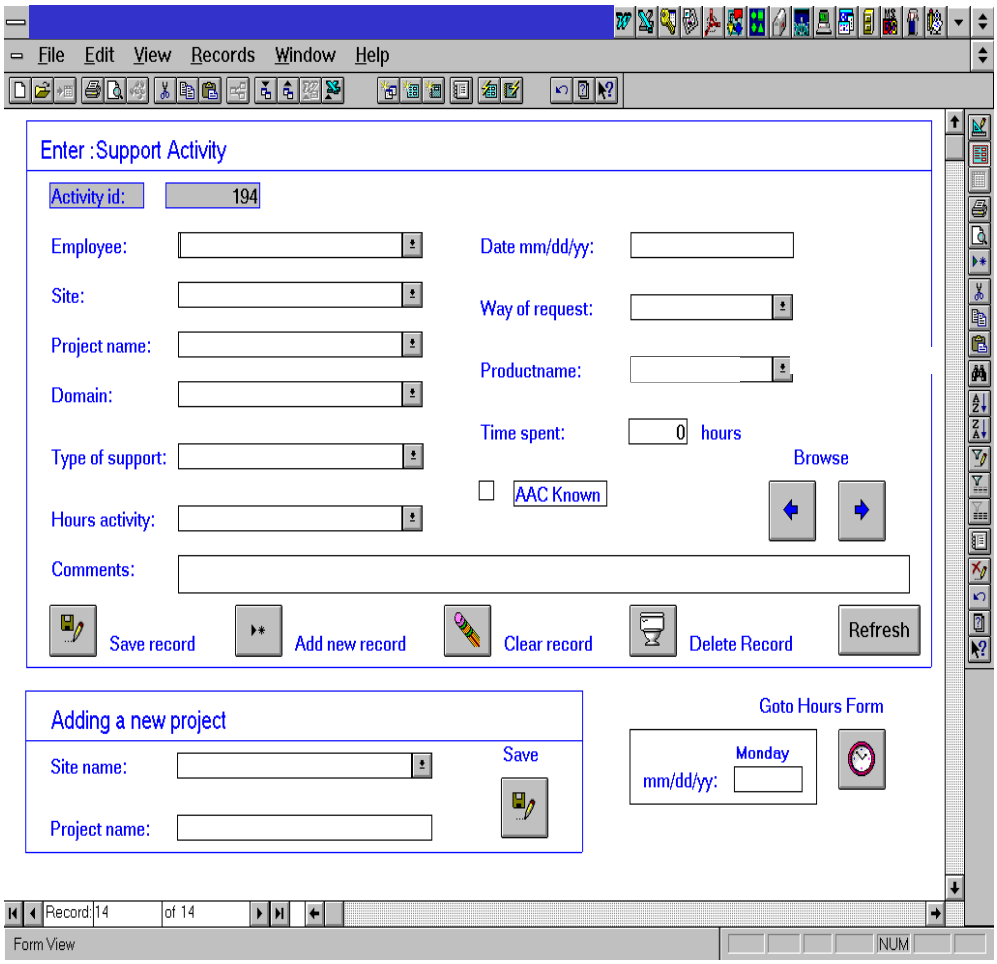
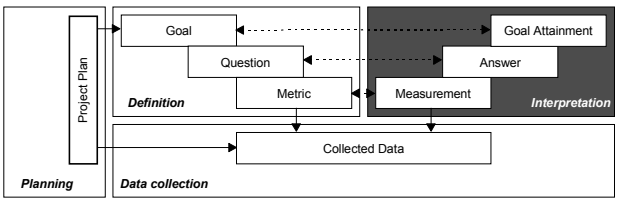


Figure 12-3: Electronic data entry form.

It was attempted to establish a link between the new support database and the database of the existing hour registration system, so the employees would not have to enter their data twice. This link was, however, not established. Therefore, it was decided that a report would be retrieved from the MSS, which would then be manually imported into the existing hour registration system by the GQM team.

12.5 Interpretation



This section describes the results of the measurement programme. The results are based on two measurement periods of two months each. This section contains the four main topics which were analysed during the feedback sessions:

- support requests to the team members;
- effort spent on the different maintenance activities;
- effort spent on product subsystems;
- effort spent for national customisation teams.

What were the baselines of support requests to the team members?

This section describes the nature of the support requests that occurred to the team members during the measurement period. The project team concluded that most numbers of requests came by telephone. Although these calls took on average only half an hour, they could still be very inconvenient when they quickly followed each other or exceeded a particular number per day (see also Chapter 11). The project team concluded that the number of requests per engineer differed and that the number of telephone calls were according to the expectations. The other media differed from their expectations (see Figure 12-4).

According to the data, a request by the project manager needed, on average, most effort, followed by meetings/visits by other persons. Some engineers expected newsgroups would be used to request support, however, the data showed no requests at all by this medium.

We also studied the total number of request per week. A representation of the data is shown in Figure 12-5. The low amount of requests in the second week is caused by a holiday period, so only a few engineers were available. From the data it was concluded that roughly 80 support requests a week could be expected.

Medium of Request	Actual	Hypothesis	Number	Time per request (hours)
Telephone	34%	33%	119	0.5
Meetings/Visits	27%	21%	95	2
E-mail	22%	33%	79	1.4
Project Manager	17%	6%	58	2.2
Newsgroup	0%	7%	0	0

Figure 12-4: Effort expenditure by medium.

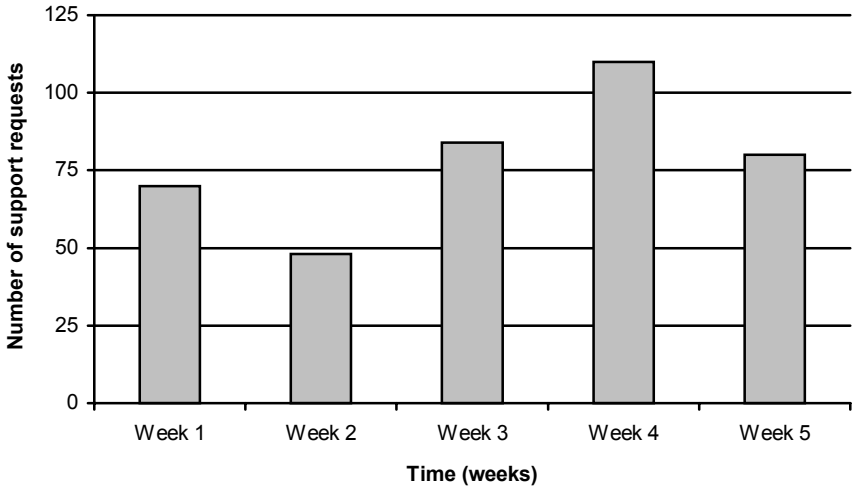


Figure 12-5: Total number of support requests per week.

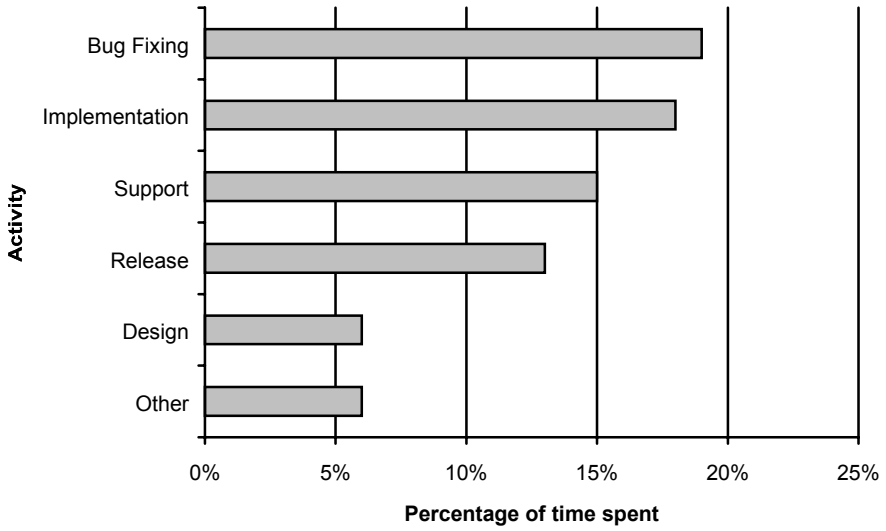


Figure 12-6: Percentage of time spent per type of activity

What is the effort spent on maintenance activities?

Figure 12-6 illustrates the distribution of effort over the top six activities. The project team concluded that they were still spending 20% of their time on implementation, even though the organisational change intended that they should be fully available for support activities. Design and review should be done in cooperation with the project team and the national customisation teams should implement. The project team could now conclude that this was not the case.

The 'review' category that was not in the top six of Figure 12-6 required 4% of the overall effort. However, the recent quality assurance audit resulted in a 'non-compliance' because no review reports were available. The project team defined an action point to make a review report for each review.

What is the effort spent on sub-systems?

Figure 12-7 shows the average daily time spent on support for each subsystem. From this figure the project team concluded that for the I/O functions subsystem one engineer should be available. Two other engineers should always be available to manage the support requests for the other subsystems. The project team explained that the high percentage of 'other' was caused by the fact that it was difficult to identify for which subsystem several activities were executed. In other words, sometimes activities were performed for more subsystems or for no subsystem at all.

One of the reasons that so much effort was spent on the I/O functions, was that many versions of this subsystem were available. Action was taken by the project team to reduce this amount of versions.

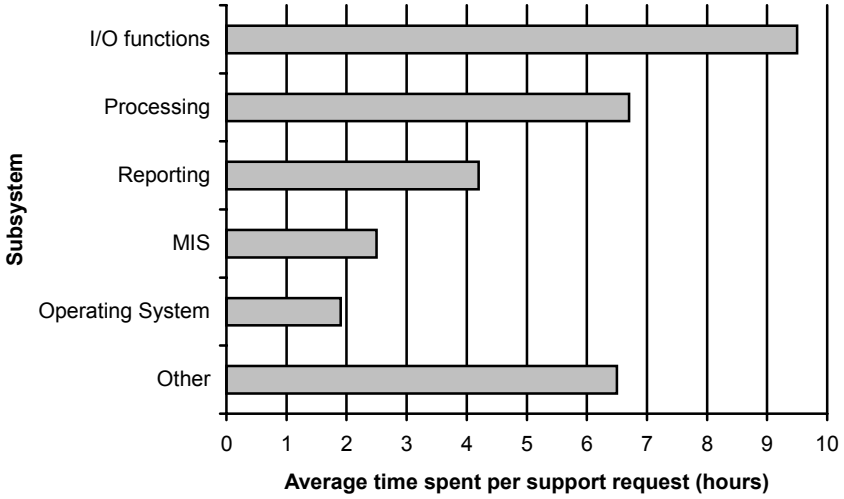


Figure 12-7: Average time spent daily per subsystem.

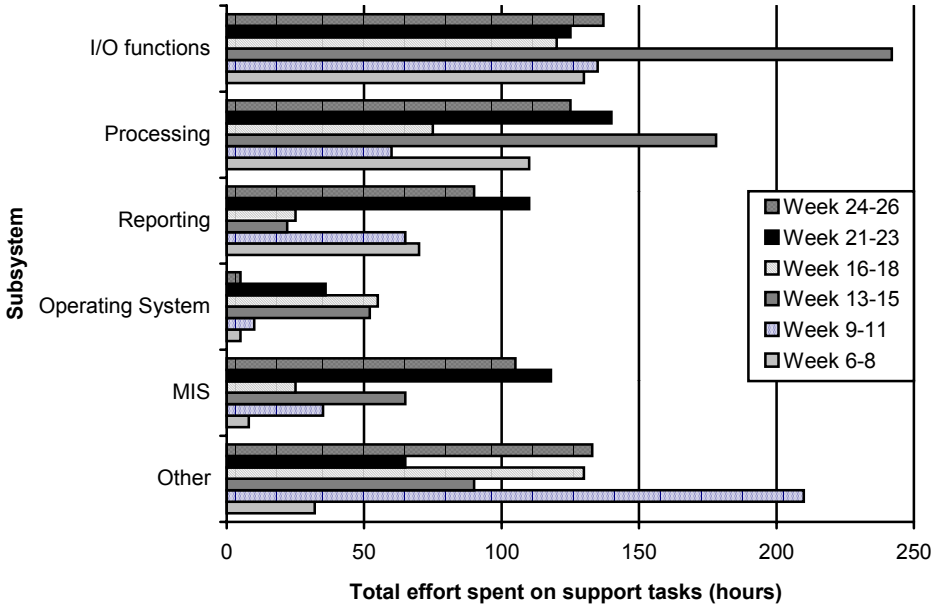


Figure 12-8: Total time spent per period per subsystem.

Figure 12-8 shows the trend in total amount of time that is spent on the subsystems per period of three weeks. The 'other' category was again high because the activities could be for more subsystems at once, or it was not clear for what subsystem the activity was performed. The project team concluded that the peak in the third period for the I/O subsystem was caused by the release of a new version of that subsystem. The peak in the third period for the subsystem 'cashing functions', was caused by a project that had high implementation priority according to the project team.

What is the effort spending for national customisation teams?

Figure 12-9 shows the percentage of time that was spent per national customisation team. The miscellaneous category was high, because often it was not clear for which site the activities were performed. An improvement action was defined to improve communication with the national project teams.

Within the category of requests of the own group a part could also be for other sites although a member of their own team requested them. The percentage for site 1 is high because there was urgency to performing these activities. For the other areas only a limited amount of effort was spent.

Which actions were taken based on the measurements?

Goal-oriented measurement implies that measurement results in interpretations, conclusions or actions. Interpretations and conclusions were described before, together with the feedback charts. In this section we list some of the actions taken by the team during this measurement programme.

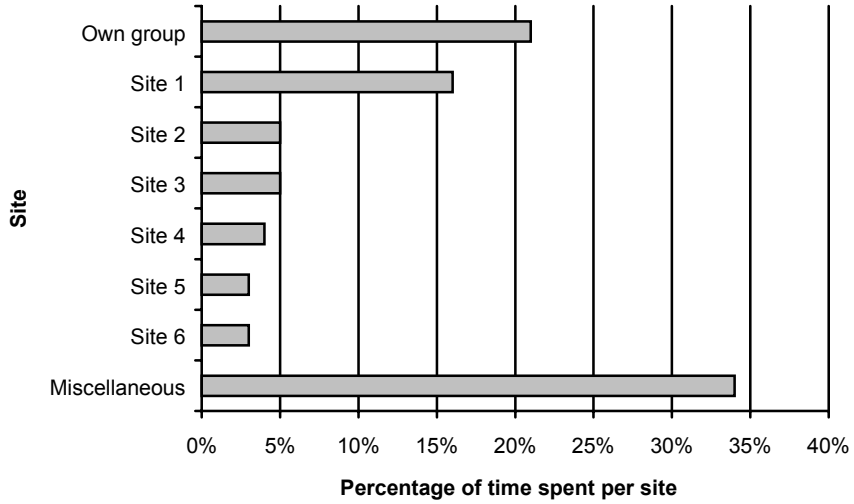


Figure 12-9: Percentage of time spent per site.

- *Show support performance to management.*
  The project manager sent parts of the measurement data to management, to show the support tasks performed by the department.
- *Communicate national projects to the project team.*
  Based on the measurements it became clear that the project team lacked information on projects going on in the national sites. Action was taken to keep all project team members informed by the project manager.
- *Report reviews to quality department.*
  The measurements showed that quite some effort was spent on reviewing deliverables. However, the recent project audit had signalled that no review reports were available. It appeared that the team members did review, but neglected reporting this. Action was defined to improve review reporting which was successfully done.
- *Plan support effort.*
  Measurement showed that the effort spent on direct support to the national sites consumed about 25% of the total effort of the team. Based on this number, the planning of the team included this amount of space for support.
- *Integrate measurement classes with new hour reporting system.*
  A new hour reporting system was set up for the whole business unit. The experiences of the measurement programme were used as input to define the necessary classes, such as activities, sites, projects and domains. By the time this new hour reporting system got fully operational, the measurement programme attained its goal.

12.6 Documentation of SUPSYS measurement programme

In this final section the GQM and measurement plan, and the data collection forms are described. This section contains no new information on the case of the SUPSYS project and is added only to illustrate both documents.

12.6.1 GQM plan

In this section the GQM plan is described. The GQM plan was subject to changes during the measurement period due to new information and experience gained in the feedback sessions.

The GQM plan consisted of three parts. The first part contained the goal that were formulated for this GQM plan. The second section described the questions that were derived from the goals. The third section described the metrics derived from the questions. For each question the necessary measurements were also given.

Goals of the GQM plan

Two goals were identified for the measurement programme, these two goals are depicted below. The attention was focused on the first goal of the measurement programme. The second goal was put on hold.

First goal:	
Analyse:	time spent on support activities
for the purpose of:	understanding
with respect to:	- the ongoing projects in the national customisation teams - the support given by the team
from the viewpoint of:	the project team
in the following context:	project D
Second goal:	
Analyse:	the quality of support
for the purpose of:	understanding
with respect to:	national customisation team-support given by the project team
from the viewpoint of:	national customisation team-engineering
in the following context:	project D

Below the associated questions are formulated.

From goal to questions

Two questions were formulated to reach the first goal described in the section above. The first question focused on the baseline characteristics of the support activities being performed at the central department.

- Q.1: What are the main characteristics of the support activities? This question focused on identifying objective and quantifiable aspects that were related to support activities. The question would give an answer of the properties of support activities.
  The second question was focused on the visibility of the project that were currently going on at the national customisation teams from the viewpoint of the central department.
- Q.2: Which projects are ongoing in the national customisation teams from the viewpoint of the central department? This question focused on identifying the projects

that were ongoing in the national customisation teams from the viewpoint of the central department. This question could be used to answer questions about the visibility of running projects at the national customisation teams in the central department.

From questions to metrics

This section describes the metrics that were formulated to answer the questions. The two questions were worked out into sub-questions that were formulated based on the practical experience of the project team. This experience was captured during interviews with the members of the project team. The sub-questions were worked out in metrics and these metrics were again reformulated in direct measurements.

Sub-questions for Q.1: 'What are the main characteristics of the support activities?':

- Q.1.1 In which way is the support requested (phone, e-mail, etc)?
- Q.1.2 On which date is the support requested?
- Q.1.3 For which project is the support requested?
- Q.1.4 By which national customisation team is the support requested?
- Q.1.5 For which domain is the support requested?
- Q.1.6 Which employee is going to give support?
- Q.1.7 How long is the support going to take?
- Q.1.8 How long did the support take?
- Q.1.9 What kind of support activity was it?
- Q.1.10 Is the project authorisation number for the support known by the department?
- Q.1.11 How is the difference between support for old and new products?

Metrics:

- How are the support activities distributed over the national customisation teams?
- How are the support activities distributed over the domains?
- How is the distribution of the way the request are made?
- What is the average time of a support activity?
- What is the average time of a specific support activity?
- What is the percentage of support activities of which a project authorisation number is known at the central department?

Direct measurements:

- M.01 Type of support activity.
- M.02 Domain of the support activity.
- M.03 Date of the support request.
- M.04 Way the request reached the domain manager.
- M.05 National customisation team that requested the support.
- M.06 Project the support is requested for.
- M.07 Project authorisation number known (yes/no).
- M.08 Name employee.
- M.09 Product.
- M.10 Time spent.

Sub-questions for Q.2: 'Which projects are ongoing in the national customisation teams from the viewpoint of the central department?'

- Q.2.1 Which national customisation team is the project running?
- Q.2.2 How is the project indicated?
- Q.2.4 Which domain manager is assigned?

Metrics:

- How many of the projects are known at the central department?
- What is the distribution of the activities for which a project is known?

Direct measurements:

- M.01 Project the support is requested for?
- M.02 National customisation team the support is requested by?
- M.03 Name domain manager?

Hypotheses on the direct measurements

During the interviews hypotheses had to be established. However, these appeared to be difficult to define, because the project team was unfamiliar with their new role. Some 'educated guesses' were formulated:

- The numbers of support requests expected per day are between 7 and 12.
- Average time spent on a support request is expected to be 0.25 to 1.0 hours.

Other hypotheses could not be agreed upon.

12.6.2 Measurement plan

In this section the measurement plan that was used to collect the data is described. The following metrics were defined:

- M.01 Type of support activity. This measurement described the kind of activity that was performed by the employee (reviewing, coding, etc).
- M.02 Domain of the support activity. This measurement gave the name of the domain for which the support activity was requested.
- M.03 Date of the support request. This measurement gave the date at which the support activity was requested.
- M.04 Way the request reached the domain manager. This measurement gave the medium that was used to reach the domain manager or the domain specialist.
- M.05 Site which requested the support. This measurement gave the name of the site that requested support.
- M.06 Project the support is requested for. This measurement gave the name of the project for which the support activity was requested.
- M.07 ACC known (yes/no). This measurement indicated if at the moment of the request the central department knew about a project authorisation number for the project, that the support was requested for.
- M.08 Name employee. This measurement gave the name of the employee of the central department that received the request for the support activity.

- M.09 Product. This measurement gave the name of the product the support activity was requested for.
- M.10 Time spent. This measurement gave the time that was spent to complete the support request.
- M.12 Project the support was requested for. This measurement gave the name of the project the support is requested for.
- M.13 National customisation team the support was requested by. This measurement gave the name of the site at which the project is running.
- M.14 Name domain manager. This measurement gave the name of the domain manager the support is about.

12.6.3 Data collection form

The applied data collection form is given in Figure 12-10. For the activities stated in this form, only the begin and end time had to be filled in. Furthermore, the following data needed to be filled in:

Activity Registration Form								
Name:								
No.	Date	Start	End	Activity Description				
				Medium	Domain	Site	PAN	Activity
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								
29								
30								

Figure 12-10: Data collection form project D.

- medium of request;
- domain for which the activity was executed;
- site for which the activity was executed;
- Yes/no if the project was known or a project authorisation number was known;
- type of activity that was performed.

The GQM team collected the forms and put the data in a database. It was preferred to use one of the terms in the table below, but it was not compulsory. Improvements could also be made.

Activity description
1. Activity
Medium of request
Telephone E-mail Visit/Person Newsgroup None
Domain
Domain 1 Domain 2 Domain 3 ……
Site
Central department-site 1 Central department-site 2 Central department-site 3 National customisation team 1 National customisation team 2 National customisation team 3 National customisation team .... Test group
Type of activity
Advice/Assistance Design Implementation Meeting Merging/Integration Release Requirements analysis Reviewing Testing Training Trouble log solution
2. Other activity
Network supervision Travel Legal absence Training Assist marketing Assist service Assist manufacturing Technology watch Hardware platform development

Figure 12-11: Data collection form project D.

12.7 Questions and assignments

12.7.1 Questions

1. What kind of tasks does ‘support’ include?
2. What was the reason to start the SUPSYS measurement programme?
3. What is wrong with the second measurement goal of this measurement programme?
4. What is new in the way the measurement support system was set up for this measurement programme compared to the previous cases?
5. Which problems occurred during definition of the measurement programme regarding the formulation of hypotheses? What were the reasons for this problem and what could be possible solutions?
6. Figure 12-4 contains data that is similar to data used in the previous Chapter. Given this similarity, what is, however, the main difference between the support measurement programme and the interrupt measurement programme?
7. What is remarkable in Figure 12-6, considering the new objectives of the project team?
8. What are the advantages and disadvantages with the inclusion of an ‘other’ category (see for example, Figure 12-7 or Figure 12-9)?
9. What were the actions taken by the project team based on the measurement data?
10. Which way of documenting a measurement plan is better: the measurement plan of Chapter 11 or the one in Chapter 12?

12.7.2 Assignments

Design your own GQM tree for your personal interrupts, looking at the questions in both Chapter 11and Chapter 12.

1. Refine your questions (maximum of six) into metrics and design a data collection form you can use.
2. Use this form to collect your personal interrupt measurement data during a period of 1 to 4 weeks.
3. Store your data in a spreadsheet and perform data analysis in order to answer your questions stated in the GQM tree. And remember: analysing these data is the most important part of measurement. Without analysing your measurement data, you could have spent your effort and energy better!

References

Basili, V.R., D.M. Weiss, 'A methodology for collecting valid software engineering data', *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 6, 1984.

Basili, V.R., C. Caldiera, H.D. Rombach, 'Goal Question Metric Paradigm', *Encyclopedia of Software Engineering* (Marciniak, J.J., editor), Volume 1, John Wiley & Sons, 1994a, pp. 528-532.

Basili, V.R., C. Caldiera, H.D. Rombach, 'Experience Factory', *Encyclopedia of Software Engineering* (Marciniak, J.J., editor), Volume 1, John Wiley & Sons, 1994b, pp. 469-476.

Bemelmans T.M.A., *Bestuurlijke Informatiesystemen en Automatisering* (Management Information Systems and Automation), Stenfert Kroese, The Netherlands, 1991.

Berghout, E.W., D.S.J. Remenyi (editors), *Evaluation of information technology*, Proceedings of the fourth European Conference on the Evaluation of IT, Delft, Delft University Press, 1997.

Birk, A., R. van Solingen, J. Järvinen, 'Business impact, benefit and cost of applying GQM in industry: an in-depth, long-term investigation at Schlumberger RPS', *Proceedings of the 5th International Symposium on Software Metrics (Metrics'98)*, Bethesda Maryland, November 19-21, 1998.

Boehm B.W., *Characteristics of software quality*, New York, 1978.

Cavano J.P., J.A. McCall, 'A framework for the measurement of software quality', *Proceedings of the software quality and assurance workshop*, 1978.

Curtis, B., W.E. Hefley, S. Miller, *People capability maturity model (P-CMM)*, CMU/SEI-95-MM-02, Software Engineering Institute, 1995.

DeMarco, T., T. Lister, *Peopleware: productive projects and teams*, Dorset House Publishing, 1987.

Fenton, N.E., S.L. Pfleeger, *Software Metrics, a rigorous and practical approach*, Thomson Computer Press, 1996.

Gibbs, W.W., 'Software's chronic crisis', *Scientific American*, September 1994, pp. 86-95.

Glass, R.L., *Software creativity*, Prentice Hall, 1995.

Goodman, P., *Practical implementation of software metrics*, London, McGraw-Hill, 1993.

Grady, R.B., *Practical software metrics for project management and process improvement*, Prentice-Hall, Englewood Cliffs, 1992.

Heemstra, F.J., R.J.Kusters, R.Nijhuis, Th.M.J. van Rijn, *Dealing with risk: beyond gut feeling: an approach to risk management in software engineering*, Maastricht, Shaker Publishing, The Netherlands, 1998.

Humphrey, W.S., *Managing the software process*, SEI Series in Software Engineering, Addison-Wesley, 1989.

Humphrey, W.S., *A discipline for software engineering*, SEI Series in Software Engineering, Addison-Wesley, 1995.

IEEE, *Software Engineering*, IEEE Standards Collection, Institute of Electrical and Electronic Engineers, 1994.

ISO9000-3, *Guidelines for the application of ISO9001 to the development, supply and maintenance of software*, International Standard: Quality Management and Quality Assurance Standards - Part 3:, International Organization for Standardization, 1991.

ISO9001, *Quality Systems - Model for quality assurance in design/development, production, installation and servicing*, International Organisation for Standardisation, 1994.

ISO/IES 9126, *Information Technology - Software Product Evaluation - Quality Characteristics and guidelines for their use*, International Organisation for Standardisation, 1991.

ISO12207, 'Software life cycle process standard 12207', *Online presentation at: http://home.sc7.ctisinc.com/sc7/ustag/12207/ sld001.htm.*, Issaquah (WA), Software Engineering Process Technology, February, 1995.

Karjalainen, J., M. Makarainen, S. Komi-Sirvio, V. Seppanen, 'Practical process improvement for embedded real-time software', *Quality Engineering*, Vol. 8, No. 4, 1996, pp. 565-573.

Kooiman, E., *Towards a feedback theory*, Masters Thesis, Delft, Delft University of Technology, August 1996.

Kuvaja, P., et al, *Software process assessment and improvement: The BOOTSTRAP approach*, Oxford, Blackwell Publishers, 1994.

Latum, F. van, R. van Solingen, M. Oivo, B. Hoisl, D. Rombach, G. Ruhe, 'Adopting GQM-based measurement in an industrial environment', *IEEE Software*, January/ February, 1998, pp 78-86.

Looijen, M., *Information systems: management, control and maintenance*, Kluwer Bedrijfsinformatie, 1998.

McCall, J.A., P.K. Richards, G.F. Walters, *Factors in software quality*, Vol. I, II, III, US Rome Air Development Center Reports, 1977.

Möller, K.H., D.J. Paulisch, *Software Metrics: A practitioner's guide to improved product development*, London, Chapman & Hall, 1993.

Parker, M.M., R.J. Benson, H.E. Trainor, *Information Economics: linking business performance to information technology*, Englewood Cliffs, Prentice-Hall, 1988.

Paulk, M.C., B. Curtiss, M.B. Chrissis, C.V. Weber, *Capability Maturity Model for software*, Version 1.1, Pittsburgh, Software Engineering Institute, 1993.

Perry, D.E., N.A. Staudenmayer, L.G. Votta, 'People, organizations, and process improvement', *IEEE Software*, July 1994, pp. 36-45.

Pfleeger, S.L., *Software engineering: the production of quality software*, New York, Macmillan Publishing Company, 1991.

Pulford, J., *AMI: A quantitative approach to software management*, ESPRIT Report 5494, 1992.

Radice, R.A., N.K. Roth, A.C. O'Hara Jr, W.A. Ciarfella, 'A programming process architecture', *IBM Systems Journal*, Vol. 24, No. 2, 1985, pp. 79-90.

Rooijmans, J., H. Aerts, M. van Genuchten, 'Software quality in consumer electronic products', *IEEE Software*, January, 1996.

Senge, P.M., *The fifth discipline: the art and practice of the learning organisation*, New York, Doubleday, 1990.

Solingen, R. van, *Goal-oriented software measurement in practice*, Master thesis, Delft, Schlumberger RPS and Delft University of Technology, 1995.

Solingen, R. van, F. van Latum, M. Oivo, E.W. Berghout, 'Application of software measurement at Schlumberger RPS: towards enhancing GQM', *Proceedings of the 6th*

European Software Control and Metrics (ESCOM) conference, The Netherlands, May 17-19, 1995.

Solingen, R. van, E.W. Berghout, E. Kooiman, 'Assessing feedback of measurement data: Schlumberger practices with reflection to theory', *Proceedings of the 4th International Symposium on Software Metrics (Metrics'97)*, Albuquerque, New Mexico, USA, IEEE Computer Society Press, pp 152-164, November, 1997.

Solingen, R. van, E. Berghout, F. van Latum, 'Interrupts: just a minute never is', *IEEE Software*, September/October, 1998.

SPICE 1997, *Introduction to SPICE*, Software Engineering Institute, Online documentation at http://www-sqi.cit.gu.edu.au/spice/, July, 1997.

Index

H

I

K

L

M

N

O

P

Q

R

S

T

V

W