MASTER

Prodoct focused software process improvement
integrating SPI and SPQ approaches into a quality improvement method for RPS

van Uijtregt, Arnim P.M.

Award date:
1998

Product Focused Software Process Improvement

. . . .

Integrating SPI and SPQ approaches into a quality improvement method for RPS

Graduation Report by:
Arnim van Uijtregt
Bladel, December 1998

EINDHOVEN UNIVERSITY OF TECHNOLOGY
FACULTY OF TECHNOLOGY MANAGEMENT

Graduation Information

Speaker:	A.P.M. van Uijtregt
Title:	Product Focused Software Process Improvement
Date:	16 December, 1998
Time:	10.00 hours
Place:	Room C9, Faculty of Technology Management Pavilion, Eindhoven University of Technology
Examination board:	Dr. Ir. J.J.M. Trienekens
	Prof. Dr. Ir. A.C. Brombacher
	Dr. R.J. Kusters
Project mentor:	Ir. D.M. van Solingen
Company:	Retail Petroleum Systems
	Industrieweg 5, NL-5531 AD Bladel, The Netherlands
	Tel. +31 (0)497-389555
	Fax. +31 (0)497-381950

Preface

This is the report of my graduation project at the faculty of Technology Management, of Eindhoven University of Technology. The project is carried out in the quality assurance department of Retail Petroleum Systems in Bladel. The project is done in the scheme of three quality improvement projects Retail Petroleum Systems participated in: SPIRITS, SPACE-UFO and PROFES.

The research in this report deals with the integration of software quality improvement activities as developed in the above mentioned projects. These activities can be divided in process related quality improvement activities and product related quality improvement activities. The goal of this thesis is to integrate the best parts of each of the projects into one quality improvement method for RPS.

This master thesis was executed with support of Ir. D.M. van Solingen, Quality Engineer at the Quality Assurance department of Retail Petroleum Systems, and Ing. E. Rodenbach, Quality Manager Special Projects of Retail Petroleum Systems. Support and supervision from Eindhoven University of Technology came from Dr. Ir. J.J.M. Trienekens, Prof. Dr. Ir. A.C. Brombacher and Dr. R.J. Kusters. I want to thank all of them for their time and valuable advice. Also I want to thank all the people I worked with at Retail Petroleum Systems in Bladel, Schwelm and Montrouge for their co-operation and support during the project.

Finally I want to thank my family and friends; especially Koen van Horssen who helped me with the design of the front cover.

Arnim van Uijtregt
December 10, 1998
Bladel, The Netherlands

Summary

Retail Petroleum Systems (RPS) has been active in software quality improvement activities by participating in three quality improvement projects: SPIRITS, SPACE-UFO and PROFES. The SPIRITS project (Software Process ImpRovement of embedded IT environmentS) focused on the improvement of the development process with respect to product reliability. The SPACE-UFO project (Software Product Advanced Certification and Evaluation - User FOcus) focused on the user perspective to improve and evaluate product quality. The PROFES project (PROduct Focused improvement for Embedded Software processes) focused on product oriented improvement of software processes.

All three projects have developed their own method for quality improvement. Each of these methods has its own specific characteristics and emphasises on either improving the product or improving the software process. For RPS it is of course important to have one method for quality improvement. Therefore the best parts of each of the methods are taken and integrated by means of a model. This model should integrate Software Process Improvement related activities with Software Product Quality related activities in order to focus Software Process Improvement on the product quality needs.

This master assignment included a research with the goal of integrating product and process related improvement activities of the three projects into a RPS method for software quality improvement. The end result of this research is the Product Oriented Process Improvement model, which identifies four distinct phases in improving software quality. The first phase in improving product quality, is identifying the product quality needs, because it is important to know what quality is needed. This is not enough, however, because it is also necessary to know the quality already reached. Therefore the second phase contains a method for estimating product quality you will likely get with the current software process. The results of the first two phases are analysed in phase 3. For product quality areas where the current process is not likely to deliver the quality needs, actions will have to be implemented. In order to check whether the quality improvement actions have the required effect, the product quality is evaluated and GQM measurement programmes are started. Feedback is tended to by means of a knowledge base.

The Product Oriented Process Improvement method has been applied in RPS in two case studies: the OPT case study and the Omega case study. The results of these case studies were promising and even before the complete cycle has been finished the benefits outweigh the cost. When the fourth phase will be completed the benefits will become even clearer and therefore the cost/benefits balance will become even more positive. Also the method is developed to be repeated over time. Results of these case studies are valuable input for future projects. The continuation of the application of the method in the RPS development process is therefore definitely recommended. The ISO 14598-5 risk scale that is being used for the product quality profiles has some serious drawbacks, however. The scale should be replaced with a new scale that addresses these drawbacks. A scale that addresses some of these drawbacks is proposed in this thesis.

Table of Contents

Table of Figures

1. Introduction

The number of different approaches for software quality improvement at Retail Petroleum Systems, causes the need for integration of these approaches into one useful quality improvement method. This situation lead to the master thesis project. This chapter consists of a description of the needs that were felt at the organisation at the start of the project. Those needs lead to the assignment which is the basis of this report. Furthermore a description of the organisation in which the assignment is conducted will be presented.

1.1. Problem definition

Retail Petroleum Systems (RPS) is one of the main suppliers in the petroleum retail market. Their product range varies from fuel dispensers to high-tech embedded products like Outdoor Payment Terminals and Retail Automation Systems including a Point of Sales and a retail management information systems. The impact of software in RPS products is increasing. As RPS' strategy is to sell high quality products, the company is continuously active in improving the quality of its products.
Therefore RPS participated in three quality improvement projects:

- the SPIRITS project (Software Process ImpRovement in embedded Information Technology environmentS) focused on the improvement of the development process with the aim to improve product reliability;
- the SPACE-UFO project (Software Product Advanced Certification and Evaluation - User FOcus) focused on the user perspective to improve and evaluate product quality;
- the PROFES project (PROduct Focussed improvement for Embedded Software processes) focused on product oriented improvement of software processes.

In these projects several methods have been developed with accompanying tools and procedures to assist the software development process in making better quality products.

However, for RPS it is now important to start using these methods, with accompanying tools and procedures in daily practice. By using them, RPS will gain experience. Along with the increased knowledge on the subject this will lead to adapted wishes for the software development process assistance.
Furthermore for RPS it will be more useful to have one method to assist the software development process. Therefore the different methods developed in the various projects have to be integrated.

1.2. Graduation assignment

The assignment consists of a practical and a theoretical part:

Theory: Investigate the integration of the three different methods developed in the projects RPS participated in, in order to come to an integration of these methods by means of a model. Analyse the model with respect to the distinction in product oriented quality improvement and process oriented quality improvement in order to improve the model.

Practice: Use a development project as a pilot project to investigate the effectiveness of the methods and tools developed, by following the model from quality requirements engineering to process improvement action selection and measurement. Make improvements to the methods, the tool-kit, and the integration between them based on the experience gained during the pilot project.

1.3. Graduation approach

The following tasks can be identified for the theoretical part:
- investigate the similarities and differences between the SPACE-UFO methodology, the SPIRITS methodology, and the PROFES methodology and their instruments;
- investigate the possibilities for integrating these methodologies by means of a model;
- analyse the model with respect to the two different views on software quality improvement: product oriented quality improvement and process oriented quality improvement;
- make the model more applicable to RPS' requirements by means of:
  - formulating rules of thumb for (some of) the transformation processes;
  - analysing and if possible improving the scales in which the product quality profiles are presented;
  - analysing the metrics on Fenton & Pfleeger's segmentation in product, process and resource metrics;
- analyse the cost and benefits of applying the model in the RPS software development process;
- analyse possible alternatives for the model.

The following tasks can be identified for the practical part:
- use questionnaires as a preparation for conducting interviews;
- apply the Multi-Party Chain approach to identify the RPS environment, the parties and roles involved in specifying quality requirements;
- keep interviews with the involved parties to determine their quality wishes;
- determine the product quality profile and the quality requirements;
- use process assessment results to estimate product quality;
- select and implement process actions based on the quality profile and the estimated product quality;
- set-up a measurement program to support (some of) these process actions;

1.4. Retail Petroleum Systems: from Schlumberger to Tokheim

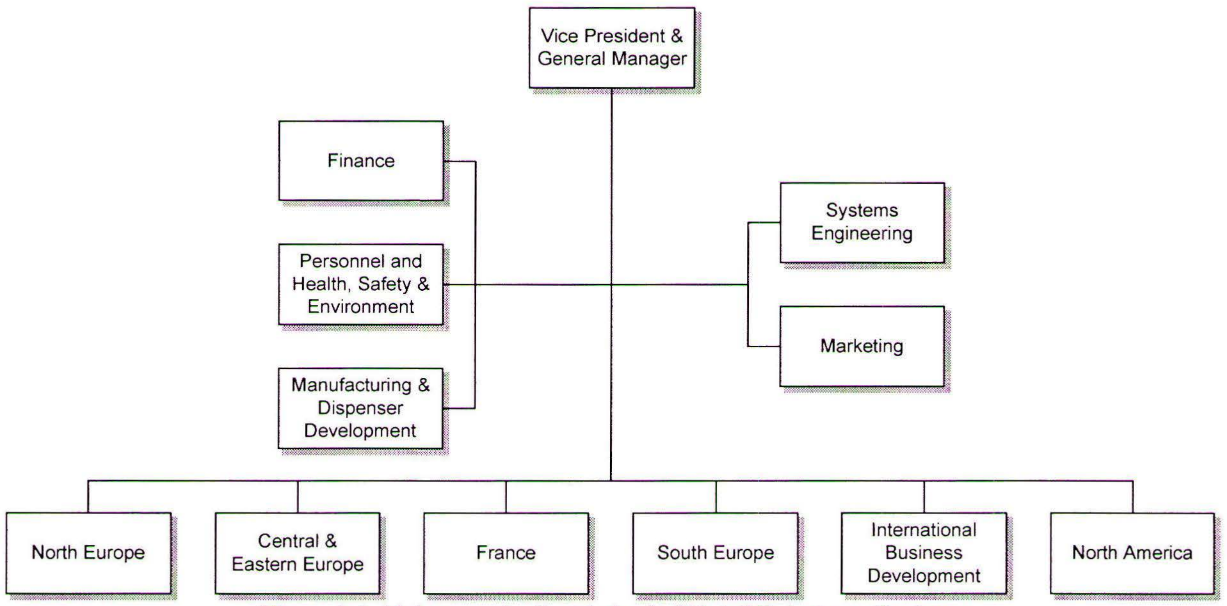
Retail Petroleum Systems is a subdivision of the Test & Transactions division of Schlumberger Ltd. Schlumberger is an industrial company, comprised of about 65,000 employees in over 100 nationalities.
Schlumberger is the leading supplier of services and technology to the petroleum industry. The company provides virtually every type of exploration, production and completion service needed in the petroleum industry including oil-rigs, drilling equipment, pumping services, but also equipment for seismic data acquisition, processing and interpretation. Next to services for the petroleum industry, Schlumberger also develops measurement systems for gas, water and electricity distribution, and manufactures test equipment for semi-conductors. Finally, Schlumberger is also active in the area of Communications and Information Technology. Here it provides solutions for wide- and local area networks, Internet and Intranet applications for the energy exploration and production sector.
Until recently, Schlumberger was also a supplier of systems for the petroleum retail market with its Retail Petroleum Systems subdivision, next to their services for the production of

petroleum. The Retail Petroleum Systems division was created by Schlumberger to serve the major fuel station networks and offer the major oil companies and independent retailers alike products for dispensing fuel and transferring funds. The product range of Retail Petroleum Systems includes dispensers, outdoor payment facilities, electronic cash registers and electronic payment services, training and maintenance programs, and 24-hour customer service. The customers of Retail Petroleum Systems include most of the large oil companies (like Shell, Esso, BP, Texaco).

As of October 1, 1998, Retail Petroleum Systems, was sold to one of Schlumberger's competitors in this market: the Tokheim Corporation.
The Tokheim Corporation is based in the United States. As such, their market was mainly situated in North America. As Retail Petroleum Systems, operates mainly in the European market, Tokheim now becomes a real global company and the world's market leader in petroleum retail equipment, with an expected yearly turnover of 735 million dollar.
Currently projects are started for the reorganisation of the Tokheim company. At the moment the organisational structure of Retail Petroleum Systems (see section 1.4.1) has not changed yet, but as the reorganisation continues, changes are inevitable. In the following I will only refer to the company with the name Retail Petroleum Systems (RPS).

1.4.1. Retail Petroleum Systems

RPS employs approximately 1650 people. Its matrix organisational structure consists of on the one hand five traditional departments being Finance, Personnel, Health, Safety & Environment, Manufacturing, Systems Engineering, and Marketing and other the other hand six regional Sales & Service Departments, which next to sales and service, also perform installation and training, and implement national customisations for the products (see figure 1-1).


Vice President & General Manager
Finance
Personnel and Health, Safety & Environment
Manufacturing & Dispenser Development
Systems Engineering
Marketing
North Europe
Central & Eastern Europe
France
South Europe
International Business Development
North America
Figure 1-1: Main organisational chart of Retail Petroleum Systems

The master thesis project was carried out at the Quality Assurance department. The Quality Assurance department is part of the Systems Engineering department, responsible for the development of the embedded systems. In the organisational structure of the Systems Engineering department the same division into regions can be found. However, Bladel is the centre for its development activities. Customisations and client specific maintenance activities lay at the regional departments (overview in figure 1-2).

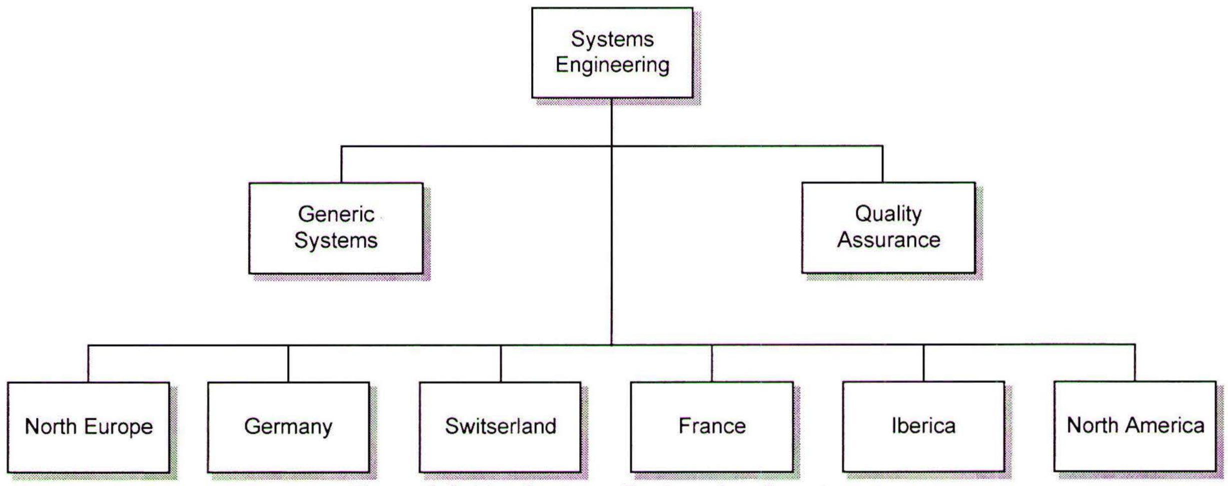


Figure 1-2: The Systems Engineering department

The software development activities in the Systems Engineering department are carried out by software engineers. The engineers develop all deliverables including requirements documents, high level designs, programming code, and user documentation.

The development activities are carried out according to the Software Development Process Model. The Software Development Process Model describes how to perform each development activity, the entry and exit criteria for each activity, and the defined information inputs and outputs for each activity [Murez, 1994]. The five phases of the software development process model can be found in frame 1-1.

Frame 1-1: The five phases of the Software Development Process model

1. **Requirements Definition**. An idea for a product is adopted and, after a systematic evaluation of customer needs, competitive products and market potential, the requirements are specified and prioritised. Upon completion of this phase, the user requirements are formalised for the product and the business plan is completed.
2. **Feasibility**. After a thorough analysis of available techniques, components and tools, choices for technical solutions are made and documented in a technical specification and architectural design. A project plan and a test plan are developed. Upon completion of this phase, the acceptance criteria and the release date for the product are settled.
3. **Design and Implementation**. The detailed product designs are made, implemented and integrated (hardware, software, mechanical and hydraulic). At the completion of this phase, a (preliminary) product as well as a (preliminary) technical product documentation are available, and are then validated.
4. **Process and Product Validation**. The (preliminary) technical product documentation is formalised. A pilot series of the product is produced to evaluate the production process. The product is tested and validated in the field. Upon completion of this phase, the product and the production process are ready for commercial production.
5. **Production and Life Cycle**. The production is started based on customer orders. Small updates, corrections and customisations to the product take place in this phase. Strategic updates are defined as new projects and restart a new cycle at the Requirements Definition phase.

Most software development activities are organised as projects and carried out under supervision of a project manager.

1.5. Structure of the report

This report consists of seven chapters. In chapter 2 two different views on improving the quality of software products are presented: process oriented and product oriented. Chapter 3 contains a description of the three projects in which RPS participated. The strengths and weaknesses of the three methods developed in the projects will be analysed and integrated into one overall model. After that, two case studies addressing this model will be presented in chapter 4. In chapter 5 the scales used for the product quality profiles are being analysed in order to improve them. Then will be returned to the two different views of software quality, process oriented and product oriented, to analyse the model in chapter 6. Finally, chapter 7 contains conclusions and recommendations.

2. Quality of embedded software products

RPS produces embedded systems for the petroleum retail market. This chapter deals with software quality improvement of embedded software products. Therefore first will be explained what embedded products are. After that some widely accepted software quality improvement methods will be presented. Basically the methods can be divided in two different views, the product oriented view and the process oriented view. For software quality improvement activities to succeed it is very important that the processes and the products are being measured. Therefore this chapter finishes with software measurement.

2.1. Embedded software products

Today we are very familiar with the widespread use of personal computers in almost every field of engineering and commerce. It is therefore tempting to assume that the use of microcomputers is limited to such applications. In reality, these represent only a small fraction of the uses to which microcomputers are put, and account for a small part of the overall market.

Most microcomputers are used within embedded products. The range of these embedded products is vast and includes all industrial and commercial sectors. Examples include computers within: engine management systems, microwave ovens, robot controllers, lift control systems, telephone answering machines, and also retail petroleum systems.

In order to avoid confusion first a clear distinction is made between an *embedded product* and *embedded software*. Basically one can say that embedded software is the software that makes a product an embedded product. To define embedded software I would like to refer to Van Solingen and Rodenbach [van Solingen & Rodenbach, 1996]. They describe embedded products and embedded software as follows.

> *"An embedded product is an autonomous physical unit, consisting of hardware (electronics) with software embedded in it, between which there is interaction by way of a specific interface."*

In order to illustrate the importance of quality for embedded products, below some characteristics of embedded products are listed, which have serious impact on its software quality needs and reaching these quality needs. The list is not complete, but is intended to give some insight in the quality issues manufacturers of embedded products face.
First, embedded products usually require a long lasting operation time. A lifespan of over ten years is common [van Solingen & van Uijtregt, 1997]. This makes important demands on the product's maintainability, because during the lifespan requirements may change significantly. Also hardware parts may be renewed and/or replaced by more modern versions during those years.
As the software in embedded products is embedded in the hardware, and it is difficult to replace this software with a new version, reliability of the software is crucial. The software should be failure free, because fixing problems in thousands of shipped products is very expensive [van Solingen & van Uijtregt, 1997]. In business software fixing failures after release is often easier, especially considering benefits of the use of the Internet. Nowadays

suppliers release service patches, in which bugs in their software are fixed, and make them available via the Internet. For embedded products this possibility is rarely available.
Embedded software has often very stringent time restrictions on the rate at which data has to be processed [Downes & Goldsack, 1982]. They are real-time systems, systems that must be able to receive continuously changing data from external sources and process that data sufficiently rapid to enable timely decision-making. This results in extra demands on the software efficiency, the time behaviour of the system and the use of processor capacity.
Furthermore the development of embedded products consists of simultaneous development of different technologies [PROFES consortium, 1997a]. Next to the development of software, other activities take place in order to simultaneously develop electronics, and hydraulics. It is clear that at the end of the development process these various technologies have to fit together, which introduces more problems.

2.2. Quality improvement approaches

Just like industrial manufacturers look for ways to assure and improve the quality of their products, software developers have to find methods to assure and improve the software's quality. IEEE provides two definitions for quality [IEEE Standards Collection, 1994]. First, quality is the degree to which a system, component, or process meets specified requirements. This definition, however, is not very practical for software engineering. One of the problems in the software industry is the difficulties software engineers face during requirements specification. Users usually only know the general needs for a computer system. Specific detailed requirements often have to be provided by the software engineers themselves. Furthermore, checking whether a complex software system with thousands of variables, meets the specified requirements, is nearly impossible. Therefore the second definition is used. In this definition the term 'specified requirements' is replaced with 'user needs or expectations'.

"Quality is the degree to which a system, component, or process meets customer or user needs or expectations."

Traditionally there have been two different in improving this quality: the process oriented, and the product oriented approach. The product oriented approach tries to guide quality improvement by making product quality explicit, whereas the process oriented approach tries to improve product quality indirectly, by controlling and improving the software development process. In the following I will present some well-known quality improvement methods in both areas.

2.2.1. Software Process Improvement

The process oriented quality improvement approach, Software Process Improvement (SPI), aims at improving the software development process. By creating a better development process it is assumed that the software product quality improves. In many companies, like Motorola, Raytheon and Hughes Aircraft, this SPI approach has proved to be beneficial [Diaz & Sligo, 1997], [Dion, 1993], [Humphrey et al, 1991].

In order to define Software Process Improvement the definition of the European Software Institute is used [European Software Institute, 1998].

> *"The implementation of an appropriate culture and organisational procedure designed to support and enable the continual improvement of development processes, based on the results of measurement data and assessment results."*

Well-known Software Process Improvement methods like CMM [Humphrey, 1992] and BOOTSTRAP [BOOTSTRAP Institute, 1997], are based on the Shewhart improvement cycle (see figure 2-1). This improvement cycle was introduced by Deming [Deming, 1982] and was named after Walter Shewhart, who laid the foundation for statistical quality control in the 1930s.

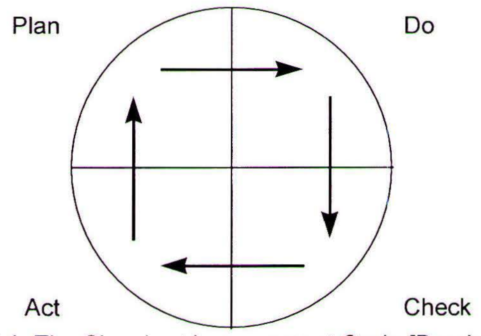


Figure 2-1: The Shewhart Improvement Cycle [Deming, 1982]

As shown in figure 2-1, it defines four steps for a general improvement process. The cycle begins with a plan for improving an activity. Once the improvement plan is completed, the plan is implemented, results are checked, and actions taken to correct deviations. The cycle is then repeated.

Based on these principles the Software Engineering Institute's Capability Maturity Model (CMM) was designed to provide a graduated improvement framework of software capabilities. Each level progressively adds further enhancements that software organisations typically master as they improve.

The improvement cycle the Software Engineering Institute uses is adapted to correspond with the SPI terminology and SPI specific needs. The steps organisations have to take, to improve their software process are the following [Humphrey, 1989].

1. Understand the current status of the development process.
2. Develop a vision of the desired process.
3. Establish a list of required process improvement actions in order of priority.
4. Produce a plan to accomplish the required actions.
5. Commit the resources to execute the plan.
6. Start over at step 1.

The cycle starts with finding out what the current status of the software development process is. For this step the processes are assessed. The objective of a process assessment is to provide a clear and factual understanding of the organisation's state of software practice. This state is expressed in the maturity level of the software development process. Once the current state of the software development process has been identified, it is important to determine which development process is desired. Without a clear objective the improvement activities may lead to the wrong results. Once we know the current status and the desired status of the software process, a list of process improvement actions can be made. Accordingly a planning is made to accomplish these process improvement actions and the plan is implemented. After the implementation it is important to check the effect of the improvement actions. Therefore we return to step 1 to determine the status of the software process.

The Capability Maturity Model distinguishes five maturity levels. These maturity levels are shown in figure 2-2. The levels are designed in such a way that the capabilities at the lower levels provide a progressively stronger foundation on which to build the upper levels. At each level, the organisation has a distinct process capability. By moving up these levels, the organisation's capability is consistently improved.

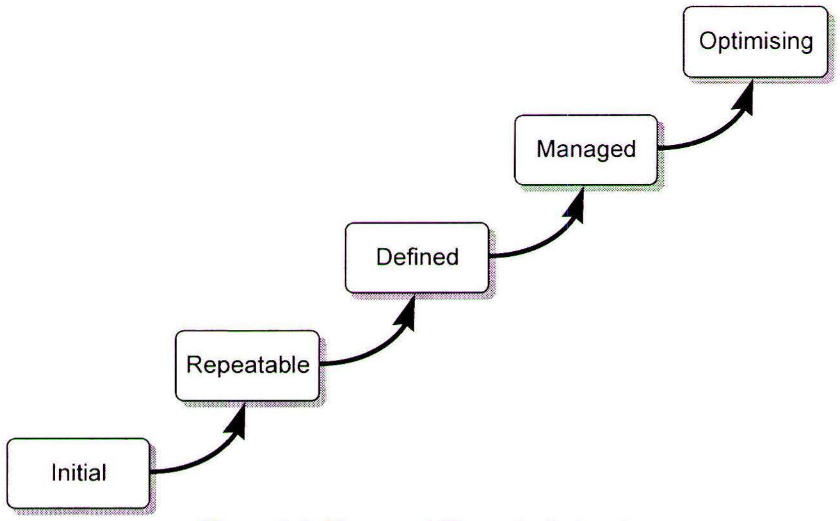


Figure 2-2: The capability maturity levels

The five maturity levels are:

1. *Initial:* the software process is characterised as ad hoc, and occasionally even chaotic. Few processes are defined and success depends on individual effort.
2. *Repeatable:* basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earliest successes on projects with similar applications.
3. *Defined:* the software process for both management and engineering activities is documented, standardised and integrated into a standard software process for the organisation. All projects use an approved, tailored version of the organisation's standard software process for developing and maintaining software.
4. *Managed:* detailed measurements of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
5. *Optimising:* Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

2.2.2. Software Product Quality

The process oriented approaches, as described in section 2.2.1, can be considered as indirect methods to achieve quality improvement of software products. They focus mainly on the definition, the structure and the improvement of software processes. Despite the interesting results of Software Process Improvement, it still evokes questions like:

- How can product quality be made explicit?
- How can product quality be measured?
- Which language is used between customers and developers to discuss quality?

In order to be able to answer such questions, a product approach of software quality is needed. A product approach reflects the idea that software quality can be gained by identification and specification of quality needs for software products, followed by assessment and evaluation of these quality needs. Customers of software products are no longer satisfied with software products that 'only' meet the functional specifications, delivered in time and at reasonable costs. Nowadays the customer also demands clear, and quantifiable software quality. Therefore the software industry has to focus on software product quality, next to Software Process Improvement.

The International Organisation for Standardisation established a practical way to make software product quality more explicit, by refinement of software product quality into a set of characteristics and sub-characteristics. It provides a model which defines six categories of software quality [ISO/IEC, 1996].

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

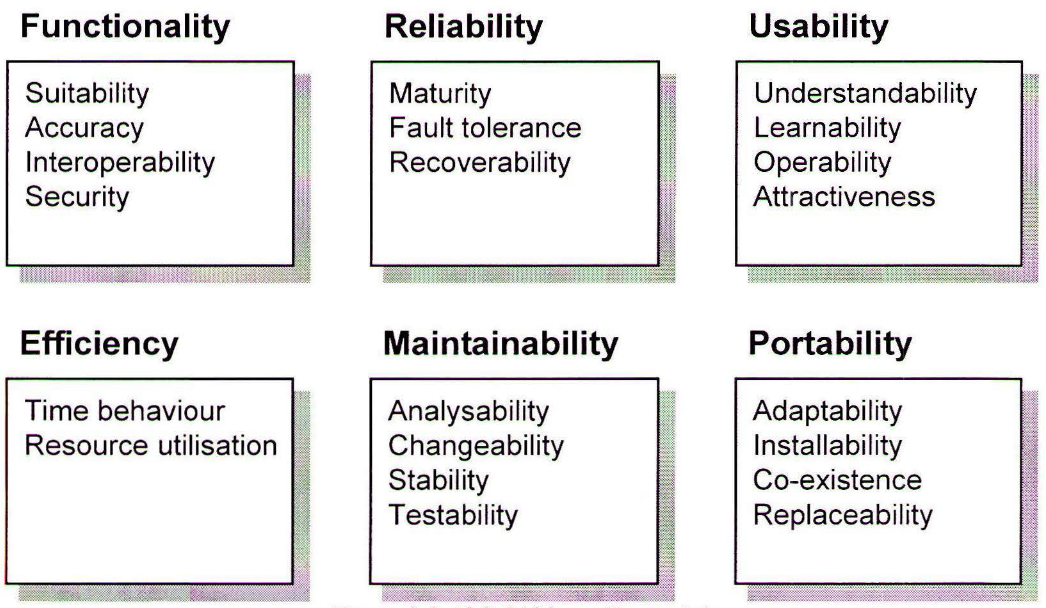


Figure 2-3: ISO 9126 quality model

These six product quality characteristics are refined into sub-characteristics (see figure 2-3). The definitions for the product quality characteristics and sub-characteristics are listed in appendix A. For each of the subcharacteristics metrics are defined in part 2 and 3 of the ISO 9126 standard. These metrics enable quantification of product quality and the possibility to evaluate software products on quality.

2.2.3. Software Process Improvement or Software Product Quality?

In sections 2.2.1 and 2.2.2 two different approaches have been presented to improve software quality, the process approach and the product approach. The process approach tries to improve the product quality indirectly by improving the process and the product oriented approach tries to make product quality explicit in order to guide the improvement activities.

The Software Process Improvement approach assumes a positive correlation between process improvement and product quality. Companies sell products, not processes. For the company it is important to make the best quality products for the lowest cost. When quality improvement activities focus too much on the process without being clear about its effect on product quality, it is very well possible that effort is invested in activities that barely effect product quality. Also it is possible that the process improvement activities have effect on the wrong quality areas, where the product quality is already according to user/customer needs.
It is therefore important to invest in process improvement activities that improve product quality where needed, and in the process improvement activities that are expected to have the most effect. The 'traditional' Software Process Improvement approaches lack this focus on product quality to a great extent. Therefore it is necessary that Software Process Improvement embraces a product focus.

The Software Product Quality approach uses the idea that software quality can be gained by identification and specification of product quality needs, followed by assessment and evaluation of these needs. Identifying and specifying quality needs is indeed the first step in creating better quality products, but after this step the software process will have to create the software product that meets these requirements. It is therefore necessary to integrate Software Process Improvement activities that support the creation of better quality products.

Summarising, it seems that both process improvement approaches and product improvement approaches are essential for companies that want to improve their software product quality. Now only the question remains whether the process improvement activities and the product improvement activities can be combined in an effective and efficient way.

2.3. Software Measurement

In section 2.2.1 as well as in section 2.2.2 the importance of software measurement is already implicitly referred to. Software measurement is important for three basic activities: understanding, control and improvement [Fenton & Pfleeger, 1997].
First, software measurement helps to understand the current situation of the software development process and the software product. The measurement visualises aspects of the process and the product, and therefore triggers taking actions accordingly and determining of goals for the future.

Second, software measurement enables control of the software development process. If information is available on products and processes, corrective actions can be taken to adjust the process in order to overcome problems.
Third, software measurement can promote improvement activities. Based on information obtained from measurement new possibilities for improvement are identified. For example, insight in the reliability of the software product may lead to starting code reviews.

Software measurement is conducted at three different entity classes [Fenton & Pfleeger, 1997]:
1. **Processes** are collections of software related activities.
2. **Products** are any artefacts, deliverables or documents that result from a process activity.
3. **Resources** are entities required by a process activity.

Software measurement should be performed towards an explicitly stated goal, because it is necessary to know what you want to gain from the measurement. Then it is also possible to structure the data collection and the data analysis according to this goal in order to make sure that the measurement is beneficial. The Goal Question Metric (GQM) approach is the most well-known way of goal oriented measurement [Basili et al, 1994]. The GQM approach provides a method to define metrics top-down using three steps.
1. list the major goals of the project;
2. derive from each goal questions that must be answered to determine if the goals are being met;
3. decide what must be measured in order to be able to answer the questions adequately.

The interpretation of the data is then done bottom-up. The data found for the metrics is used to answer the questions. With the answers on the questions the goal will be met. The GQM paradigm is presented graphically in figure 2-4.

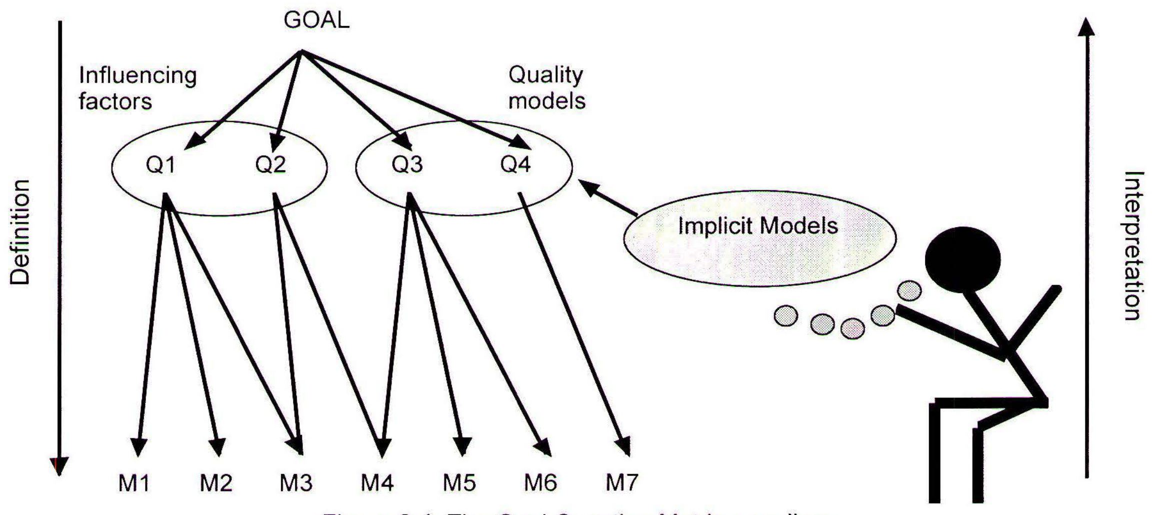


Figure 2-4: The Goal Question Metric paradigm

3. Integration of methods for product oriented process improvement

In this chapter the results of the quality improvement projects RPS participated in, SPIRITS, SPACE-UFO and PROFES, are presented. After that an analysis of the three projects will follow to identify their strengths and weaknesses. The purpose of this analysis is to come to one integrated method for software quality improvement, useful for RPS. This method will be presented by means of the Product Oriented Process Improvement model.

3.1. SPIRITS

The first project RPS participated in, was the SPIRITS project. SPIRITS is an acronym for Software Process ImpRovement in embedded IT environmentS. This joint project of RPS and the Frits Philips Institute for Quality Management from Eindhoven University of Technology was subsidised by SENTER and ran from January 1996 through February 1998.

The SPIRITS project focused on improving the software process, by taking into account the quality of software products. This is best illustrated by figure 3-1.

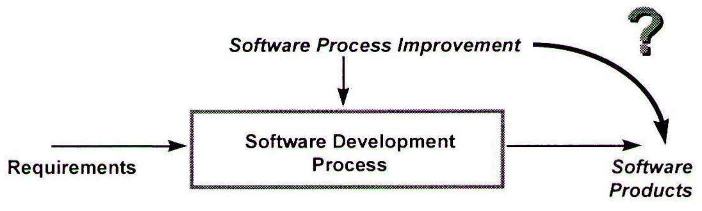


Figure 3-1: Relation between SPI and Product Quality

RPS develops embedded products. The software in these products is developed according to a software development process. Many methods and techniques can be used to improve this development process. However, due to the specific characteristics of embedded products, it is not clear whether the process improvement methods, are applicable in the embedded domain. Therefore it is necessary to evaluate whether a certain process improvement action results in an improved product. This evaluation can be done by executing measurement programmes. The goal of these measurement programmes is to find out what effect the process improvement action has on product quality.

SPIRITS focused at the development of concepts and methods for effective process improvement. Effectiveness had to be expressed in quantifiable (quality) characteristics of embedded products. The project has developed concepts and methods for process improvement to accomplish high reliability of embedded products. The main objective was to establish:

- 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 SPIRITS method

The basic result of the SPIRITS project is the SPIRITS method. This method can be presented with a flowchart depicted in figure 3-2.

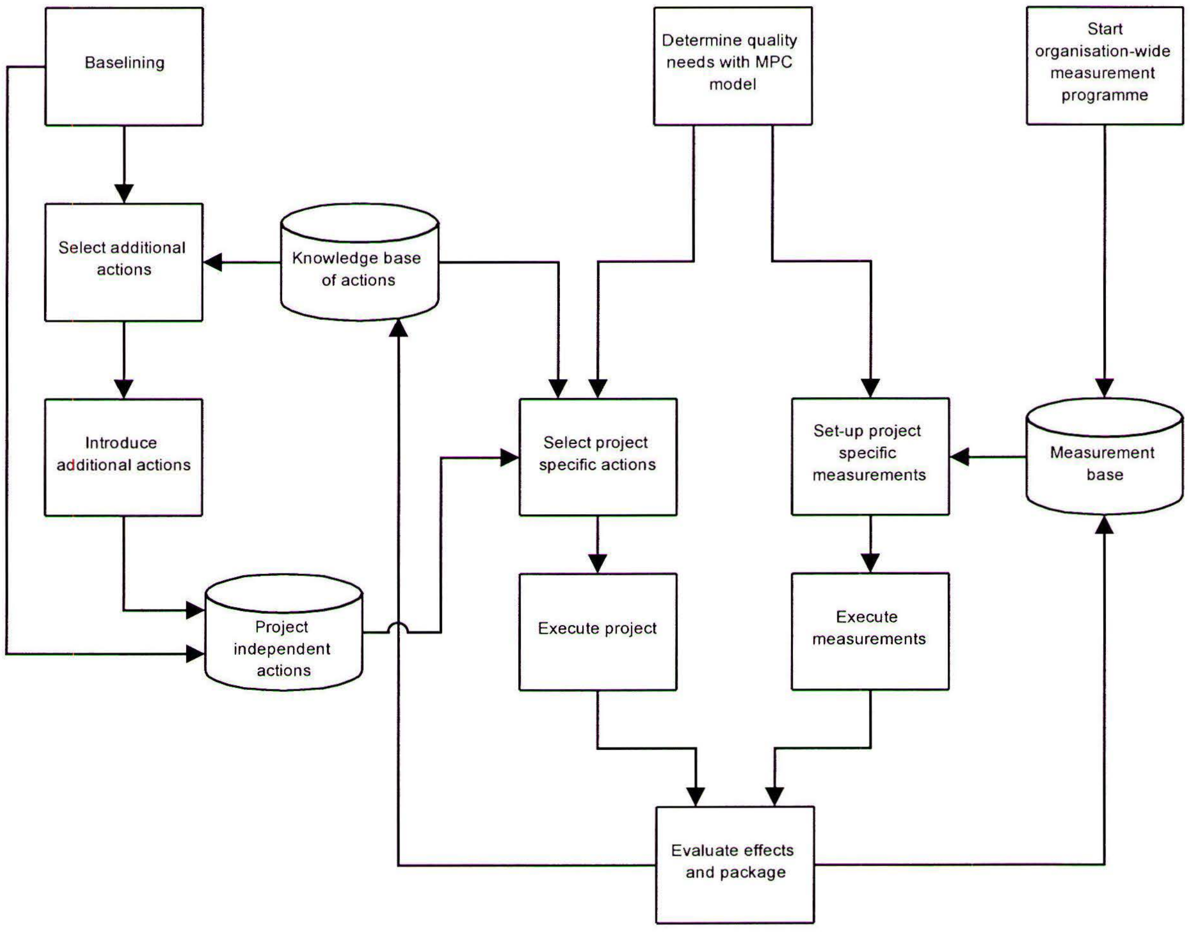

Baselining
Determine quality needs with MPC model
Start organisation-wide measurement programme
Select additional actions
Knowledge base of actions
Introduce additional actions
Select project specific actions
Set-up project specific measurements
Measurement base
Project independent actions
Execute project
Execute measurements
Evaluate effects and package

Figure 3-2: SPIRITS method [Rodenbach & van Solingen, 1998]

The three basic streams of the SPIRITS method are:

- baselining and increasing the capabilities of the organisation;
- performing a project specific improvement project;
- maintaining an organisation wide measurement programme.

This method is described in high detail in a procedure binder. The procedures of the SPIRITS method explain all steps to be taken in an improvement programme that applies the SPIRITS approach.

3.1.1. Baselining

Before it is possible to tailor the software development process to the needs of a specific project, the organisational capabilities have to be known. The baseline capability of the organisation consists of a list of actions, which are part of the capability of the engineers. The baseline capability of the organisation is stored in the project independent actions database.

Over time the baseline can be extended by adding new actions to the capability of the software engineers. These actions are expected to have a positive effect on product quality.

This expectation is usually based on literature. The information on the effect of actions on product quality is stored in the knowledge base of actions. After new actions are selected to extend the company baseline, the actions are introduced to the complete organisation.

3.1.2. Project specific improvement project

A project specific improvement project starts with determining the product quality needs. These needs are determined using the Multi-Party Chain approach.
The first step in this approach is specifying the Multi-Party Chain. The Multi-Party Chain flowchart is used to facilitate the identification of all parties involved and of their relations concerning product requirements. This is a 'chain' chart, because a number of customer-supplier relations exist between the different parties. This flowchart can be used in a specific situation to identify concerns and to facilitate communication among the parties involved.
The quality needs are then determined by gathering the product quality requirements. Representatives of each of the parties, identified in the Multi-Party Chain chart, are interviewed to find their wishes according to product quality. The result of this product quality requirements definition phase is a documentation of the users' quality requirements in non-technical terms. These requirements are translated into measurable objectives according to the ISO 9126-2 standard. Also a quality profile is derived, based on the requirements of all parties.

After the product quality needs have been specified, the software development process is tailored to these needs. Basic input for this step is the quality profile identified at the end of the Multi-Party Chain approach and the knowledge base of actions. In this base the impact of certain actions on product quality is predicted. By choosing actions wisely, a development process is specified which accommodates the quality needs.

3.1.3. Measurement

Tailoring the project to the product quality needs, as described above, is only possible when the knowledge base of actions contains information on the impact of certain process actions on product quality. A good way of acquiring this information is setting up GQM based measurement programmes throughout the company. For the SPIRITS method GQM measurement programmes are used to gather information about impact of process actions on product quality.
The GQM approach is a systematic way to define measurement programmes towards a specific goal. By stating specific goals the measurement programme is given a clear context. The goal is then refined into a set of questions, which reflect the implicit models of the software developers. Each question is again refined into a set of metrics, which serve to answer the question [van Latum et al, 1998].
The data collected for the GQM measurement programmes is stored in the measurement base. The data in the measurement base then has to be analysed. For this analysis feedback is required from the software engineers. Therefore feedback sessions are held with the project team on a regular basis. The feedback sessions' main objectives are to discuss the measurement programme results, to interpret the data and to define actions for the next measurement period.
The conclusions drawn during these sessions are packaged in two ways. First, the information about the impact of software process actions are packaged using the knowledge base. Second, results and experiences of the GQM measurement programme itself are packaged for re-use in future measurement programmes. Therefore the measurement base contains all information about past measurement programmes.

3.1.4. Involvement in the SPIRITS project

At the start of the graduation assignment, the SPIRITS project was already practically finished, so my work for SPIRITS is not a result of the graduation project as such. My involvement in SPIRITS was a practical training to design and develop a generic GQM measurement tool within the framework of SPIRITS [van Uijtregt, 1998b]. This tool was not only needed to store data, but also for generating analysing material in the form of graphs.

3.2. SPACE-UFO

From July 1996 until September 1998 RPS participated in the ESPRIT project SPACE-UFO. SPACE-UFO stands for Software Product Advanced Certification and Evaluation - User FOcus. The SPACE-UFO project was subsidised by the European Union and consisted of nine partners, Brameur, Etnoteam, IMK, Italtel, KEMA, London City University, Philips, RPS, and SMC international, from four different European countries, France, Italy, The Netherlands, and the United Kingdom.

The concept of the SPACE-UFO method can best be described by the SPACE-UFO reference model. This reference model is depicted in figure 3-3. The picture consists of four levels:

- the main objective level;
- the high level;
- the conceptual level;
- the instrumental and technical level.

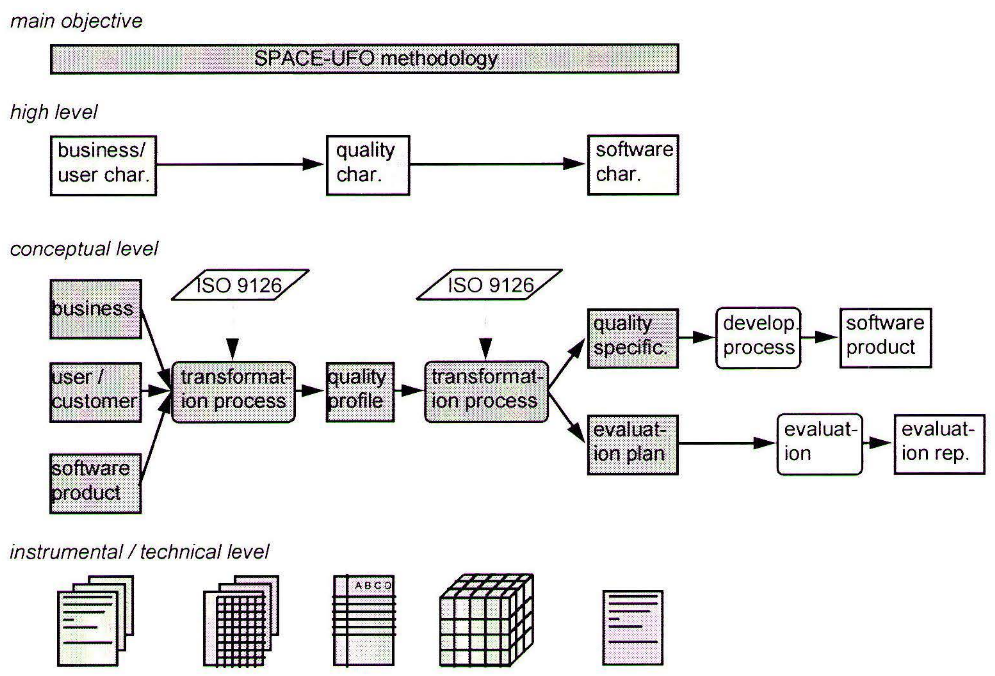


Figure 3-3: The SPACE-UFO reference model [SPACE-UFO Consortium, 1998b]

3.2.1. The main objective level

SPACE-UFO aims to be a user oriented methodology for specifying software quality requirements and evaluating software product quality, based on software quality concepts that explicitly take the needs of users and their businesses into account.
The main objective of the SPACE-UFO project was to develop a method:

- to specify the quality requirements to be set to a software product
- to assess the quality of the software product.

3.2.2. The high level

At a high level, the basic idea of SPACE-UFO can be illustrated by the following line of thought. Business processes that have to be supported by the software product and user needs (or expectations) require a certain quality level of the software. This quality level is expressed by one or several (software) quality characteristics. These characteristics have to meet certain requirements that are set, so as to ensure that the user needs are fulfilled. The quality characteristics can be met by the characteristics of the software itself.

Not all software is used to support business processes. For example, software in embedded products often is only required to perform a specific function. Next to this, there are some issues related to the product such as applicable laws, standards and operational environment.

The high level of the SPACE-UFO reference model is indistinct in the way that it only describes in general terms that software characteristics have to be derived from quality characteristics, and quality characteristics from business characteristics. In the conceptual level details are provided that make the high level clearer.

3.2.3. The conceptual level

The conceptual level gives a more detailed description of the SPACE-UFO method. Some aspects are described into more detail and the transformation processes between these aspects are made explicit.

The software quality specification and assessment process starts with a description of the business processes that have to be supported by the software product, the needs (or expectations) of the user/customer and the characteristics of the software product itself. Related to the product are issues such as applicable laws, standards and conditions necessary for use of certain hardware or technologies.

In the first transformation process the software quality level is derived based on the business processes, user expectations and the software product itself. This transformation process applies the refinement of software quality in quality characteristics and quality subcharacteristics from the ISO 9126 standard, as described in section 2.2.2. It addresses which (sub)characteristics of ISO 9126 are important (also relatively against each other).

The software product quality level is described by means of a product quality profile. Such a quality profile, which is the output of the transformation process, describes the different aspects and requirements. The importance of the distinguished aspects is reflected by so-called importance levels.

The second transformation process uses the product quality profile as input to translate it to a quality specification and/or an evaluation plan. This transformation process describes:

- which quality specifications will be set to the software to be developed (the quality specification)
- which standards, methods, techniques and tools will be used to evaluate the software with respect to the aspects that are described in the quality profile.

The quality specification is a description of the quality characteristics that have to be fulfilled by the software to be developed. This specification is input to the software development process. The development process results in a software product.

The evaluation plan is the formal description of the way the software will actually be evaluated. The actual evaluation does not lie within the scope of SPACE-UFO. The end result of the evaluation is the evaluation report.

3.2.4. The instrumental and technical level

The instrumental and technical level addresses the operational aspects of the methodology and consists of techniques and tools to support these techniques, based on the concepts of the conceptual level. These techniques are used to carry out the various processes and describe the entities mentioned at the conceptual level.

The first technique is a questionnaire which is used to determine product quality characteristics. The questionnaire contains questions about business, user/customer and software product characteristics. Because of the different nature of embedded systems and non-embedded systems two different questionnaires have been created for each of the environments. For RPS the embedded systems questionnaire is only relevant. This questionnaire is described in [SPACE-UFO Consortium, 1998a].

The questionnaire has to be used together with the second technique, a relation matrix which translates the product quality characteristics obtained with the questionnaire into a product quality profile. This technique is created to support the first transformation process of the reference model. The quality profile is presented by means of an evaluation level, according to the ISO 14598-5 standard, for each of the ISO 9126 (sub)characteristics.

For the non-embedded environment these two techniques are supported with an automated tool. For the embedded environment this tool is not suited. The use of a simple spreadsheet, however, produces the same results with little effort.

The second transformation process consists of two branches. The first branch translates the quality needs, as described in the quality profile to a quality measure specification and the second branch translates the quality profile into an evaluation plan.
A set of guidelines and a standard template are the only techniques that have been created for the quality measure specification branch. For the evaluation plan branch also a multi decision criteria tool and a knowledge base tool have been developed. The lack of available evaluation modules in this knowledge base makes it difficult to use these tools properly, however.

3.2.5. Involvement in the SPACE-UFO project

At the time the graduation project started, the SPACE-UFO project was already running for approximately 18 months. At that time the SPACE-UFO method was already developed on the higher levels. On the instrumental level, some activities still had to be developed. As such I participated in the construction of the embedded systems questionnaire and the embedded systems relation matrix, together with the other partners KEMA and Philips.
The most important task that had to be done by RPS was the validation of the SPACE-UFO method. Therefore several case studies have been done with respect to the OPT, Omega and WWC projects to check the usefulness of the method. Results of these case studies can be found in chapter 4.

3.3. PROFES

In January 1997 RPS took part in the PROFES project. PROFES is an acronym for PROduct Focused improvement of Embedded Software processes. The PROFES project is still running and will be finished in June 1999. PROFES is another European Union funded project which consists of the seven partners Dräger, Ericsson, Etnoteam, Fraunhofer IESE, RPS, University of Oulu and VTT Electronics from the countries Finland, France, Germany, Italy and The Netherlands.

The objective of the PROFES project is to support the embedded systems industry with a tailored improvement methodology that:

- Focuses improvement actions on those parts and characteristics of the software development process that contribute most to the critical customer-oriented product quality factors.
- Combines and enhances the strengths of goal-oriented measurement (GQM), process assessment (BOOTSTRAP), product and process modelling and experience factory.
- Is validated through case studies in three industrial organisations.

The baseline of the PROFES method is constructed by reusing best practices and results from other projects and established methods. This is primarily related to the basic contributions used within PROFES: process assessment, product and process modelling, goal-oriented measurement, and experience factory. Secondly, it is related to the overall method paradigm for systematic quality improvement and experimentation [PROFES Consortium, 1997b].

3.3.1. PROFES improvement steps

The PROFES improvement method uses a modified version of the Quality Improvement Paradigm framework in order to meet the goals for product driven process improvement. To illustrate and emphasise the importance of the product as a driver for improvement, it is placed in the middle of the PROFES improvement circle (see figure 3-4). The product is the starting point for any improvement activity, starting with the identification of the product quality needs and the determination of the preliminary product quality goals. Product-Process Dependency (PPD) is the linking element between the product and the product development processes. PPDs are used to find and determine the required process changes to achieve stated product quality improvement goals.

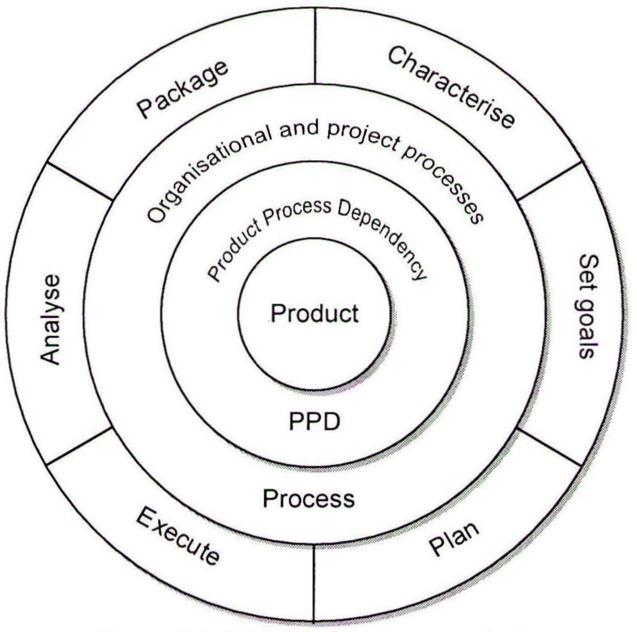


Figure 3-4: PROFES improvement circle

The PROFES method consists of six main phases. These phases are described shortly below.

Characterise
Product driven process improvement starts with an identification or review of product quality needs that are gathered from customer surveys, market research or from other sources. Based on the product quality needs preliminary product quality goals are set and aligned with high-level goals, e.g. business goals. Characterising includes organisational and project-level assessment and modelling to find out what is the current status of the processes. With assessment and modelling activities candidate process changes are identified. In addition to the process assessment, a product assessment can provide key information in such a product driven process improvement approach. If product related measurement is feasible product quality characterisation can be done using Goal/Question/Metric (GQM) based measurement, or some other available method. The results of process and product characterisation are the starting point for improvement goal setting.

Set goals
After the assessments, the improvement goals are set based on the current status of both process and product and the product quality needs that have to be met. Based on the product quality improvement goals a set of candidate Product-Process Dependency Experience Packages (PPD-EP) are identified. The experience packages are available through an experience base that provides the PPD-models. If there is no such experience base, an analysis of project history may provide preliminary PPD-model candidates for experience packaging. By modelling and applying these PPD-models the experience base can grow through each consecutive project cycle. The PPD-EPs are verified PPD-models. The PPD-EPs contain suggestions of process improvements that are further studied from the viewpoint of the process assessment results and a selection is made based on criteria such as feasibility. Product improvement goal(s) are transformed into measurement goals. The measurement goals are refined into measures via questions according to the GQM method. The main purpose of the defined measures is to follow and continuously analyse the product improvement and process change realisation and to draw final conclusions.

Plan
Improvement is planned properly before project execution. In this phase needed process changes are selected, described and modelled. A measurement plan that is based on measures defined in the GQM Plan is determined. The measurement plan includes specifics about the measurement process, measurement frequency, information sources and all additional information to completely measure the goals defined in the GQM plan.

Execute
The defined improvement actions are implemented in the development process. This means implementing a process with prescriptive process models that intends achievement of the desired product quality. Measurement activities are established based on the GQM and measurement plan. Measurements are collected and product quality is continuously analysed in GQM feedback sessions during project execution. Measurement actions may include using a measurement system that enables continuous measurement and facilitates on-line assessment.

Analyse
The purpose of the analysis phase is to analyse if product quality has improved as assumed with the changes made to the process. Based on measurements conducted in the execute phase, results are analysed, and corrective actions may be defined. In the analysis phase the released product data and process data and findings are analysed and interpreted thoroughly possibly leading to an update of relevant models. The differences between product and process related plans and realisation are analysed and root causes of deviations are identified. Also the new process models taken in use are evaluated. This may require a re-assessment of the changed processes.
This phase emphasises gathering the lessons learned during improvement actions. With organisational mechanism for re-use it is assured that organisational wide learning is enabled.

Package
The purpose of the package phase is to store all experiences gained in the project regarding product-process dependencies. This includes also rejection and modifications of PPD-EPs, which have been concluded in the analysis phase. The storage of the experience is necessary for later reuse in forthcoming projects.
Findings from the project evaluation are documented in a reusable format. Project and organisation specific terms are removed and the context situation in which the PPD-EPs are supposed to be reused are defined. The experience gained in the development project is stored in the Experience Base for each PPD-EP.

The six phases of the PROFES improvement circle are divided into 12 smaller steps. The relation between phases and steps is presented in table 3-1.

Table 3-1: Relation between phases and procedure steps

PROFES phases	Steps in the main procedure
Characterise	1. Identify product quality needs 2. Gain commitment 3. Identify current product quality 4. Identify current process status
Set Goals	5. Set product improvement goals 6. Determine needed process changes using PPD 6.1. Build PPD 6.2. Use PPD
Plan	7. Describe process changes 8. Set metrics for the processes and product 9. Plan improvement actions
Execute	10. Execute and monitor project
Analyse	11. Evaluate results
Package	12. Update experience base

3.3.2. Involvement in the PROFES project

I became involved in the PROFES project in September 1998. At this time, the master thesis project entered its final phase. For the PROFES method the exact activities for each step in the characterisation phase were still unclear. For this phase much of the work that will be presented in section 3.4 has been used to fill in this gap. The way in which RPS performed these steps was presented to the PROFES consortium. The PROFES consortium decided to use this way of working to perform the activities for the characterisation phase and included it in the first draft of the user manual.

3.4. Product Oriented Process Improvement

In this section the methods described in the previous three sections are analysed with the purpose of integrating them into one method, suited to the needs of RPS, the Product Oriented Process Improvement method.
The first conclusion that can be drawn from the three methods developed within the SPIRITS, SPACE-UFO and PROFES projects is that they are compatible. All three methods incorporate the general idea that in order to make better embedded products, one first has to identify the quality needs for the product, followed by taking the appropriate actions in the software development process to accommodate these quality needs. The effect of these actions are then evaluated and stored in some sort of knowledge base to help decision taking in future projects. The PROFES method makes also a fourth overall phase explicit: estimating product quality based on the status of the current process. This phase helps in comparing needed quality with expected quality, so decisions can be made more well-founded. In figure 3-5 the above described basic line of thought is presented.

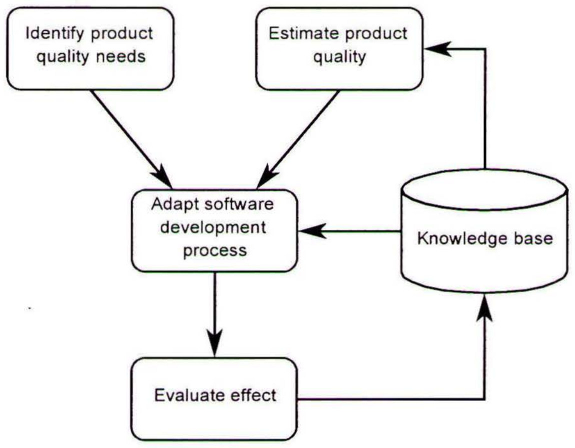

Identify product quality needs
Estimate product quality
Adapt software development process
Knowledge base
Evaluate effect

Figure 3-5: The general phases for product oriented process improvement

SPIRITS, SPACE-UFO and SPIRITS prescribe the same phases for quality improvement. The difference in the three methods, is the specific way in which these general phases have to be done and which instruments are to be used. Each of the methods emphasises on certain phases and now the objective is to take the best pieces from the respective projects. Also the language used in the three methods differs. Especially the terminology with respect to adapting the software development process is quite confusing. Three different terms (measure, process improvement action, and Product Process Dependency) are used to express approximately the same thing. The terms measure and process improvement action can be viewed as synonyms, an action that is expected to influence the product quality in a certain area. The term Product Process Dependency, used in PROFES, is used to indicate the influence of an action on a certain product quality area itself.
Because of the foregoing it is important to select the most useful steps, techniques and tools and integrate these tools into one (RPS) method. Below the analysis is done for each general phase.

3.4.1. Identify product quality needs

Both SPACE-UFO and SPIRITS have their own techniques for the identification of the product quality needs. SPACE-UFO uses the embedded software questionnaire [SPACE-UFO Consortium, 1998a]. The advantage of this questionnaire is that it can be used quickly to get insight in the quality needs. The disadvantage is that, due to the nature of the questions, general questions about characteristics of the product, the user, and the system, the questionnaire gives little specific information about needed quality. Also the nature of the questions about characteristics like 'the number of users of the product' or 'the size of the software code' makes it senseless to retrieve information from more than one person. A second person would provide the same answers.

SPIRITS uses the Multi-Party Chain approach for this phase. The Multi-Party Chain chart is constructed to get insight in the organisational structure for the product. This chain chart is then used to select interviewees representing each of the stakeholders. In an open interview the quality requirements are gathered. For each of the requirements metrics are formulated to make the requirements measurable. The requirements can be analysed using the requirements engineering database tool. The product quality profile resulting from the wishes of the interviewees is indicated with 'wanted value'.

The feasibility of the requirements is discussed with the project manager. Some of the requirements are rejected; no effort will be spend in order to reach the requirement. The product quality profile is modified accordingly. This second product quality profile is indicated with 'target value'.
The disadvantage of this approach is that you interview without much guidance, which means that the end-result of the interview is mainly based on the skills of the interviewer. The advantage of this approach, however, is that you get much more detailed information about the quality needs and that you get information from every stakeholder's point of view.
A disadvantage of both approaches is that it is not clear whether the translation of the answers from interviewees into levels for ISO 9126 (sub)characteristics is correct.

PROFES is non-descriptive about which method to use, to identify the product quality needs. It suggest either a global survey of the needs, which could be executed with the SPACE-UFO questionnaire, or an extensive survey, which could be executed with the Multi-Party Chain approach. PROFES does however make an additional step explicit that is very important: gaining commitment from the project team. What is awkward however, is that this step is done after the identification of the quality needs. As if there is no need for commitment on the side of the interviewees.

Taking into account the advantages and disadvantages of the methods mentioned above, the 'identify product quality needs'-phase should be done as presented in the flowchart of figure 3-6.

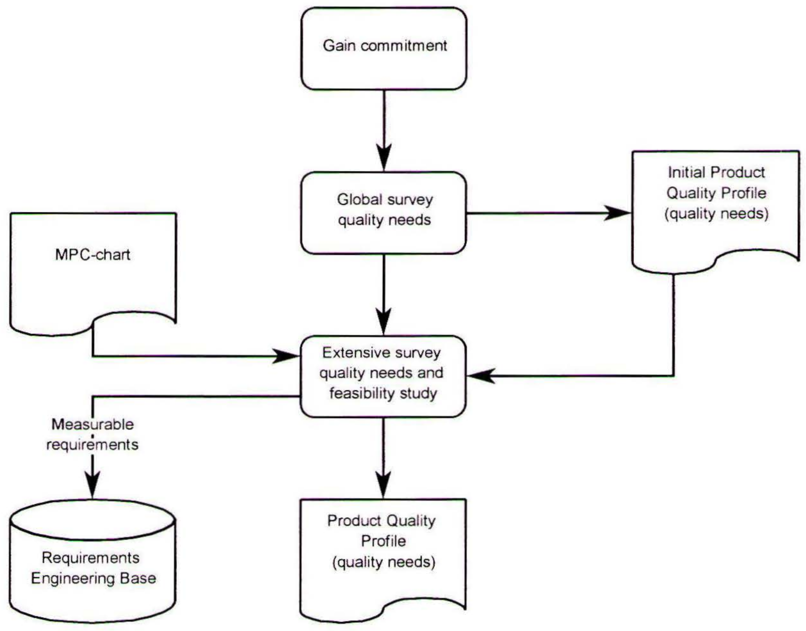

Gain commitment
Global survey quality needs
Initial Product Quality Profile (quality needs)
MPC-chart
Extensive survey quality needs and feasibility study
Measurable requirements
Requirements Engineering Base
Product Quality Profile (quality needs)

Figure 3-6: Identify product quality needs phase

The first step is to gain commitment. When commitment from the project team is lacking, there will be no support for the investigation of the quality needs, or the adaptation of the software development process. Next a global survey of the quality needs should take place, using the SPACE-UFO questionnaire. An interview with the project manager is held, because the project manager has the best knowledge about the product. This results in an initial product quality profile. This quality profile is input for the extensive survey of the quality needs. This survey is needed, because the global survey offers too little detailed information.

Open interviews are held with the various stakeholders, as defined in the Multi-Party Chain chart according to the SPIRITS approach. The initial quality profile helps to guide the interview to a certain extent, because the interviewer already has some insight in the needed product quality. During the interviews the quality requirements are gathered and stored in the requirements engineering base. Metrics are determined for the requirements, so it is possible to check whether the product meets the requirements. Later these metrics are used in the final phase; 'evaluate effect'. Two different product quality profiles are determined indicated with 'wanted value' and 'target value', based on the feasibility analysis of the requirements. Only the 'target value' will be used in subsequent steps.

3.4.2. Estimate product quality

The PROFES method uses BOOTSTRAP assessment results to estimate product quality. Part of the assessment is the investigation and identification of the current way of working, i.e. how the software development process is at this moment. In the assessment report it is therefore possible to identify the processes and activities in the development process. By applying the PPD experience packages to estimate the contribution to product quality for each specific process or activity, it is possible to estimate the product quality for each of the ISO 9126 (sub)characteristics. A product quality profile can then be constructed.
Of course the estimation is nothing more than an educated guess and the accuracy of the estimation is highly dependent on the person who performs the estimation. Nevertheless I believe that it will reveal the areas of product quality that the current software development process is insufficiently suited to.
The SPIRITS and SPACE-UFO methods do not use process assessment results to estimate product quality explicitly, but this does not mean that this aspect is not taken into account. When new actions are introduced to the software development process, automatically the actions that have been done already are taken into account. In the Product Oriented Process Improvement model the estimation process will be included as presented in figure 3-7. The model does not prescribe which assessment method has to be used. Only the assessment report is used as input for the estimation.

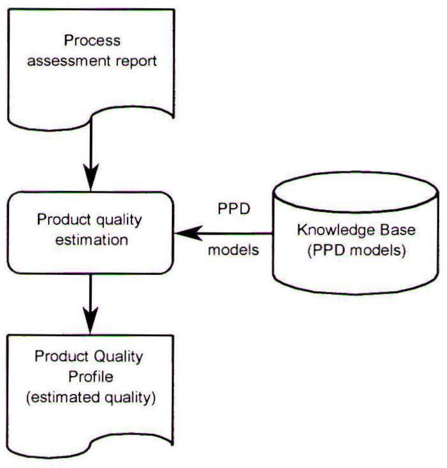

Process assessment report
Product quality estimation
PPD
models
Knowledge Base (PPD models)
Product Quality Profile (estimated quality)

Figure 3-7: Estimate product quality phase

3.4.3. Adapt software development process

The results of the first two phases are now input for the 'adapt software development process'-phase. Based on analysis of the two product quality profiles problem areas are

identified. Then decisions have to be taken on how to adapt the software development process, in order to come to a software development process that is capable of fulfilling the needed product quality.
As already stated in section 3.4.2, SPIRITS and SPACE-UFO do not explicitly estimate product quality based on the current software development process, but consider the current process in this phase.
A further analysis of the SPACE-UFO method provides the insight that SPACE-UFO actually only states that measures (or actions) have to be implemented in order to accommodate the quality profile. The way to select these measures is not provided.
SPIRITS is more elaborate on the steps to take in this phase. Based on the product quality profile and the information in the knowledge base for actions, an analysis is done to tailor the software development process to the quality needs. A new process is 'build' from available process actions. Then the project is started with this new process. The whole process is supported with a tool called 'knowledge base for process actions'.
PROFES uses a different entity, called Product Process Dependency (PPD) models. PROFES also makes a distinction between the identification of candidate PPD models to be used and the actual selection of PPD models. The new process is described in the process improvement plan.
Based on the foregoing the 'adapt software development process'-phase should be done as presented in figure 3-8. It starts with an analysis of both product quality profiles from the previous phases, in order to identify certain problem areas for which the software development process is not sufficiently suited to meet the quality needs. For these problem areas candidate improvement actions are identified, based on the information in the knowledge base. The SPIRITS tool, especially suited to the RPS needs, will be used as this knowledge base. Process improvement actions will be selected after consultation with the project team. In this negotiation it may prove to be impossible to reach the quality needs. The target will then have to be adjusted accordingly.
Finally, the Process Improvement Plan will be created and implemented in the project.

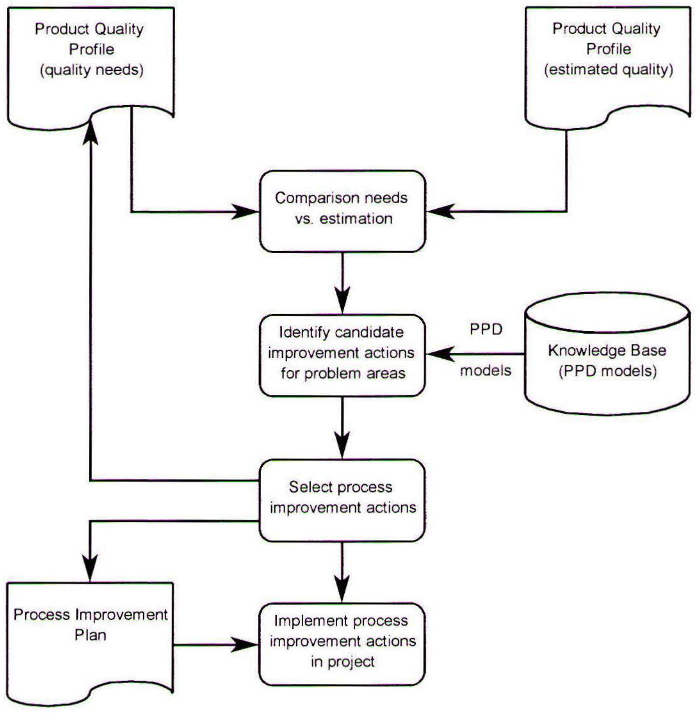

Product Quality Profile (quality needs)
Product Quality Profile (estimated quality)
Comparison needs vs. estimation
Identify candidate improvement actions for problem areas
PPD models
Knowledge Base (PPD models)
Select process improvement actions
Process Improvement Plan
Implement process improvement actions in project

Figure 3-8: Adapt software development process phase

3.4.4. Evaluate effect

After the software development process definition and the start of the project, the final phase still has to be completed. During this phase the effect of the project on the product quality is evaluated. This supports decision-taking in future projects, because then the results of this project are known. The evaluation is done in two ways. First, the evaluation of product quality. This evaluation answers the question whether the final product meets the quality needs. Second, the evaluation of impact of certain process improvement actions on product quality.

SPACE-UFO is quite elaborate on the way to select an evaluation module for a given product, to identify whether a certain product meets the quality needs. Because SPACE-UFO does not encompass the actual definition of the evaluation modules itself, the result is that there is a good way of selecting an evaluation technique, but there are little actual techniques available to select from. SPIRITS also created a way to evaluate the product quality at the end of the project. It uses the quality requirements and the accompanying metrics, which were defined during the first phase. It consists of an easy check whether the target is met within the final product. Also the difference between current quality (at the start of the project) and final quality (at the end of the project) helps in finding out whether the process improvement actions have had any effect.

PROFES does not mention a way to evaluate product quality, but is quite extensive on the way to evaluate the PPD models. For the PPD models GQM measurement programmes are started to monitor the results and the influence on product quality. After the measurement data has been collected, the data is analysed during feedback sessions with the project team. The conclusions are packaged and the PPD repository is updated with this new information. SPIRITS uses the same way of evaluating the effects of process actions on product quality; the use of GQM measurement programmes and feedback sessions to evaluate results. PROFES, however uses the tools Metriflame and GQM Aspect, and SPIRITS uses the MEFSYS system to support the GQM programmes. As the MEFSYS system is designed for the use within RPS it is better suited to the needs of RPS and therefore this tool is being used. SPACE-UFO does not mention a method to evaluate the effects of measures on product quality.

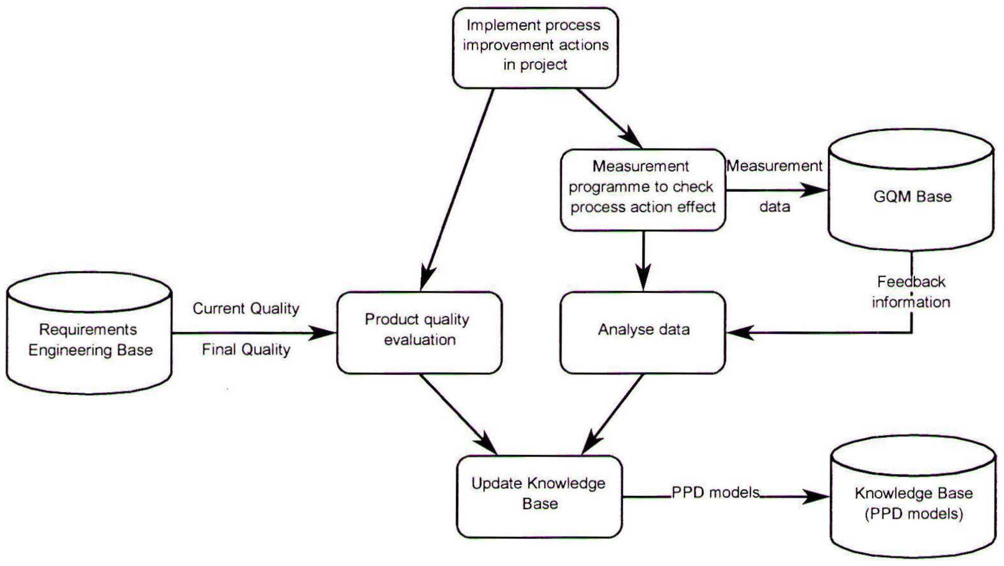

Implement process improvement actions in project
Measurement programme to check process action effect
Measurement data
GQM Base
Feedback information
Requirements Engineering Base
Current Quality
Final Quality
Product quality evaluation
Analyse data
Update Knowledge Base
PPD models
Knowledge Base (PPD models)

Figure 3-9: Evaluate effect phase

3.4.5. Product Oriented Process Improvement model

The Product Oriented Process Improvement method combines SPI activities with SPQ activities. The method focuses the SPI activities in order to improve the software development process in such a way that product quality improves in the areas where the software development process is not able to meet the product quality needs. The PPD models provide the expected impact of process improvement actions on product quality, and are therefore essential for the product focus. Usually SPI misses such a product focus, in that it suggests process improvement actions in all areas; also process improvement actions which have no expected impact on product quality characteristics where quality improvement is needed.

The Product Oriented Process Improvement method also adopts the cyclic approach of improvement. For each project new process actions have to be implemented in order to improve the product. Feedback is provided in the evaluation phase.

Chapter 1 states that, based on the results of the case studies, improvements have to be made to the model. In fact these improvements have been made, but are already processed in this section. To indicate the differences between the initial and the final model, the initial model is included in appendix B.

The main differences between these two models are the details in the 'adapt software development process'-phase. The initial model only mentions one step and does not provide any details. Based on insights gained in the case studies the model has been modified as can be seen in the complete Product Oriented Process Improvement model (see figure 3-10).

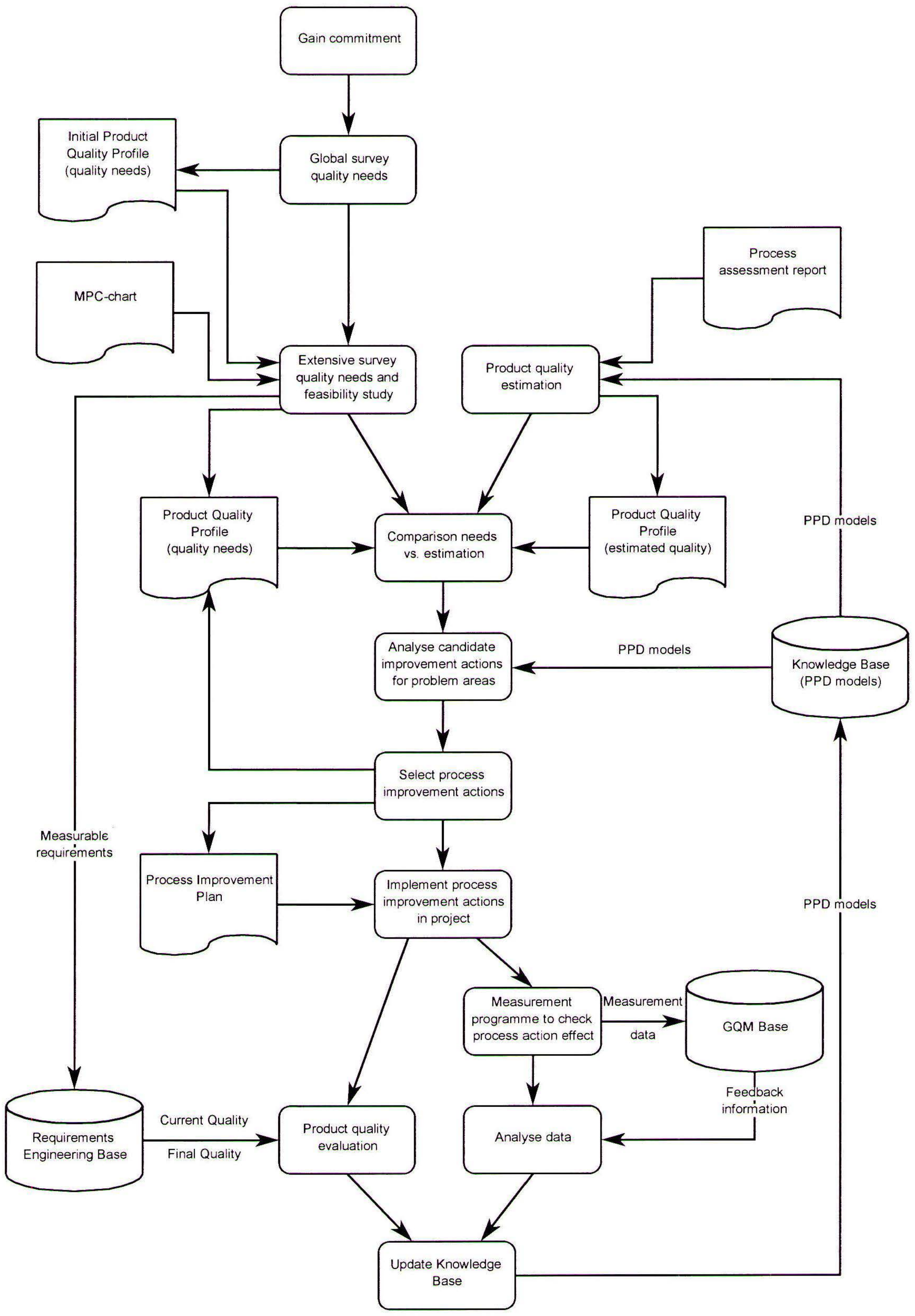

Gain commitment
Global survey quality needs
Initial Product Quality Profile (quality needs)
MPC-chart
Process assessment report
Extensive survey quality needs and feasibility study
Product quality estimation
Product Quality Profile (quality needs)
Comparison needs vs. estimation
Product Quality Profile (estimated quality)
PPD models
Analyse candidate improvement actions for problem areas
PPD models
Knowledge Base (PPD models)
Select process improvement actions
Measurable requirements
Process Improvement Plan
Implement process improvement actions in project
PPD models
Measurement programme to check process action effect
Measurement data
GQM Base
Feedback information
Requirements Engineering Base
Current Quality
Final Quality
Product quality evaluation
Analyse data
Update Knowledge Base

Figure 3-10: Product Oriented Process Improvement model

4. Case studies

The Product Oriented Process Improvement model, presented in chapter 3, has been applied in two RPS case studies: OPT and Omega. In this chapter the results of these case studies are presented. An analysis is done with respect to the cost of these case studies and the benefits gained during these case studies at the end of the chapter.

4.1. Outdoor Payment Terminal case study

In this section the results of the OPT case study in RPS are presented. In the OPT project the Outdoor Payment Terminal is developed. More information about the Outdoor Payment Terminal and the OPT project can be found in appendix C.
For the OPT project the first three phases of the Product Oriented Process Improvement model have been completed and the fourth phase is currently being worked on. Below the steps taken in the quality investigation are summarised. For a more detailed description of this case study I refer to [van Uijtregt & van Solingen, 1998], [van Uijtregt, 1998a].

4.1.1. Identify product quality needs

The first phase in the Product Oriented Process Improvement model is the identification of the quality needs. What first has to be done, however, before actually starting with this phase is gaining commitment. Without commitment there is no co-operation of the project team and improvement actions will definitely fail. Therefore the plans were discussed with the project manager, the software engineers and the marketing manager responsible for the OPT.
After this initial step, the embedded software questionnaire [SPACE-UFO consortium, 1998a] was used in the global survey of the quality needs, to get an initial product quality profile. An interview was held with the project manager, because he is most capable of answering all the questions. In the resulting product quality profile five ISO 9126 quality sub-characteristics (suitability, accuracy, interoperability, security, and time-behaviour) got the score C.
Because this global survey did not provide sufficient detailed information the extensive survey of the quality needs was started. Therefore the stakeholders in identifying quality requirements had to be defined using the Multi-Party Chain chart. This MPC-chart and the initial product quality profile were used as input for the extensive survey. In this survey interviews were held with representatives of the various parties. The goal for these interviews was basically, to gather the quality requirements for the OPT (example in frame 4-1). Based on the requirements the parties indicated, the final product quality profile was specified. A major difference between this final profile and the initial profile from the embedded systems questionnaire is the rating of reliability. Reliability was now rated C, instead of D.

Frame 4-1: Example of a quality requirement for the OPT

Quality requirement	ISO 9126 subcharacteristic
The total payment time should be as short as possible	time-behaviour

The product quality profile does not take the feasibility of the wishes into account, so this profile is indicated with 'wanted value'. Next to this 'wanted value' also a 'target value' is derived. For this quality profile the wishes were analysed together with the project manager on feasibility and the product quality profile adjusted accordingly. For the OPT almost every wish was accepted as valid and taken into account. The few rejected wishes had no impact on the resulting product quality profile.

4.1.2. Estimate product quality

Next to the quality needs identification also a product quality estimation is needed to make decisions about improving the software development process. Therefore it is necessary to find out the state of the current software development process, and how it contributes to product quality. The input material comes from two sources. First, the OPT baselining process executed in the SPIRITS project [van Veenendaal, 1997] identifies which actions are taken in the software development process. Second, a literature research executed in the PROFES project identifies what the impact of actions is on product quality [Soerjoesing, 1998b]. These relationships have been stored in the knowledge base of actions.
From the baselining report can be concluded that 34 actions are currently taken in the software development process. By applying the PPD models in the knowledge base for each of these actions an estimation can be done of its contribution to product quality. Gathering all actions taken per ISO 9126 quality (sub)characteristic enables estimation of product quality. This estimation represents the product quality that will probably be reached when using the current process, when no additional actions are taken. The result may be presented by means of a product quality profile.

4.1.3. Adapt software development process

In the third phase, the results from the first two phases are compared in order to make decisions about adapting the software development process. In figure 4-1 both product quality profiles indicating the quality needs and the estimated product quality, are presented together.

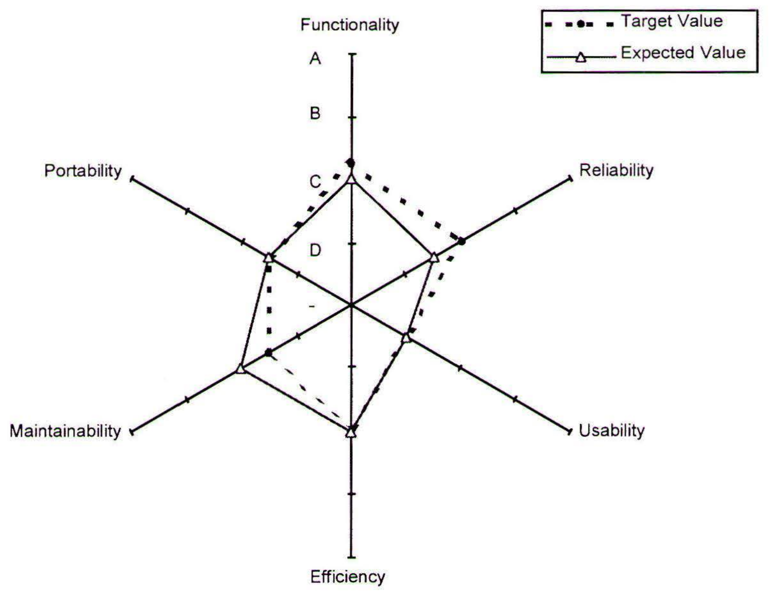


Figure 4-1: Quality needs vs. estimated quality for the OPT

The main conclusion to be drawn from figure 4-1 is that the project team appears to control the quality of the product quite well. For the ISO 9126 characteristics usability, efficiency and portability the current process is likely to reach needed quality. For maintainability the current process will probably do even better than needed. However, two areas for improvement are still identified:

1. It is expected that the functionality (especially the subcharacteristic suitability) of the OPT product is at risk, mainly because there are many country specific requirements which are currently not known, but to which the product has to comply.
2. It is expected that the reliability (especially the subcharacteristic maturity) of the OPT might be insufficient. In the software development process actions are taken that should improve reliability, but these actions are not executed completely.

Based on these results a meeting with the project team was convened in which the results of the investigation were presented. Also candidate improvement actions to address the improvement areas were proposed in this meeting. The following decisions were taken during the meeting.

- Action is taken by marketing to ensure the availability of the country specific requirements. Also involvement of the national Sales and Service Departments is needed for this requirements specification.
- The project team experienced the analysis with respect to reliability as strange, because the customers of the development team are quite happy with the reliability of the product. An investigation of the reliability of the product in the field is started by means of a measurement programme on product reliability after release.
- Also for the OPT reliability, there is a large dependence on SSD product testing. In order to trace the extend to which this is done, a system test checklist for the SSD is created. Using such a checklist the SSD can check whether full testing has been done sufficiently.

4.1.4. Evaluate effect

The final phase in the Product Oriented Process Improvement model is the evaluation of the effects of the process improvement actions and the evaluation of the product quality at the end of the project. As the project is not finished yet, this last evaluation could not be done.
The GQM measurement programme for OPT reliability after release, however, has been started and the first data have been analysed. The first feedback session is held recently. The main conclusion from this feedback session is that OPT reliability is quite good. The measurement programme should be continued, however, especially when the OPT is installed in the field. Also the SSDs should get involved in the measurement programme.

4.2. Omega project

This section deals with the Omega case study. In the Omega project the Omega fuel station management system is developed. More information on the Omega and the Omega project can be found in appendix D.
For the Omega project the first three phases of the Product Oriented Process Improvement model have been completed and the fourth phase has recently been started. As the steps taken resemble the steps of the OPT case study, this section will focus mainly on differences in the approach and the results. For a more detailed description of this case study I refer to [Soerjoesing & van Uijtregt, 1998], [PROFES consortium, 1998b] and [Soerjoesing, 1998a].

4.2.1. Identify product quality needs

For the first phase of the Product Oriented Process Improvement model the same steps were taken as in the OPT case study. After making sure there was support from the project team, a global survey for the quality needs was done, according to the embedded systems questionnaire [SPACE-UFO, 1998a]. After that, interviews were held with the Omega stakeholders to gather the quality requirements for the extensive survey. In this extensive survey it appeared that the score for reliability was too low in the global survey as was also identified in the OPT case study. Therefore the score was increased from D to C.

In order to validate whether the questions in the embedded software questionnaire were sufficiently objective to be asked to just one person, two persons were interviewed: the Omega project manager and a senior software engineer. During the interview the questions the two of them disagreed were recorded. It appeared that the two interviewees had no differences of opinion for the questions. Therefore it has been decided that it is acceptable to obtain the answers from only one person in the future. The project manager should be able to answer all the questions. For more detailed results of this additional investigation I refer to [SPACE-UFO, 1998c].

In contrary to the OPT project, where no difference existed between the wanted and the target value of the quality needs, the meeting with the project manager to analyse the quality requirements on their feasibility, did result in some changes. An example of a rejected quality requirement can be found in frame 4-2. The rejection of this requirement lead to omit co-existence. The score dropped from 'D' to '-'.

Frame 4-2: Example of a rejected quality requirement for the Omega system

Quality requirement	**ISO 9126 subcharacteristic**
It has to be possible to run other applications like a word editor on the Omega system	co-existence

4.2.2. Estimate product quality

The second phase of the Product Oriented Process Improvement model is done in the same way as was done in the OPT project. Input for this phase were the BOOTSTRAP assessment report [PROFES consortium, 1998a] and the information in the knowledge base of actions. From the BOOTSTRAP assessment report could be concluded that 41 actions are taken in the software development process. The results of the estimation can be found in figure 4-2

4.2.3. Adapt software development process

In the third phase, the results from the first two phases are compared in order to make decisions about adapting the software development process. In figure 4-2 both product quality profiles indicating the quality needs and the estimated product quality, are presented together.

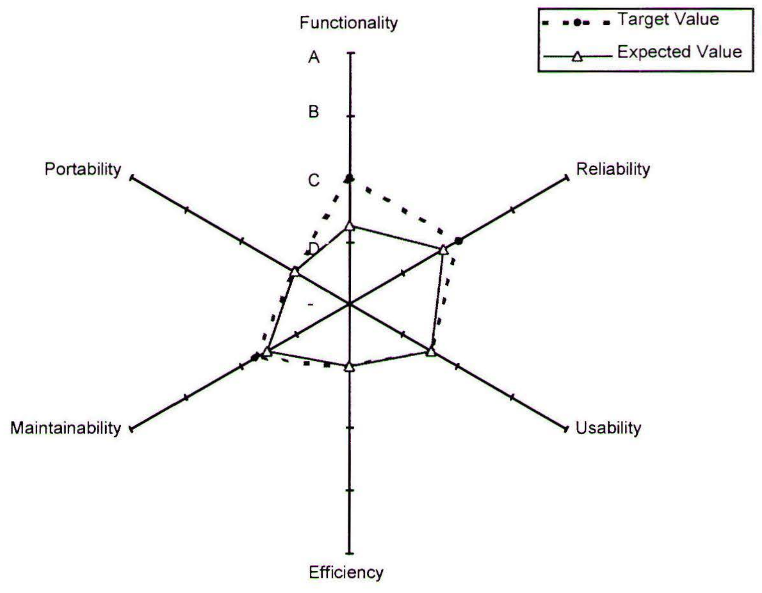


Figure 4-2: Quality needs vs. estimated quality for the Omega system

From figure 4-2 is concluded that the Omega project has more problems in reaching the desired quality then the OPT project. The problem areas identified in the investigation are presented in frame 4-3.

Frame 4-3: Problem areas for the Omega system

Functionality	*Maintainability*
Suitability	Changeability
Security	*Portability*
Reliability	Replaceability
Maturity	
Recoverability	

After deliberation between the quality manager and the project manager it was decided to focus on two problem areas for improvement: maturity and suitability. For these areas candidate improvement actions were suggested. In a meeting with the quality manager and the project manager some of these improvement actions were rejected, but still the decision was made to implement eleven actions for improvement. The actions taken concern configuration management, the testing process and project planning activities. Based on the information in the knowledge base for actions the expected impact of these actions on maturity and suitability was derived. In table 4-1 this impact can be found.

Table 4-1: Improvement actions taken for the Omega project

Improvement actions	Maturity	Suitability
Put the Omega sources, documents and test scripts under configuration management		++
Introduce detailed process for Configuration Management		++
Carry out module audits as soon as CM is established	++	
Full regression testing	+++	+++
Full system testing	+++	
Introduction of new test methods	+++	
Automate testing	+++	
Maintain project planning with a planning tool		
Set up hour registration including procedures and guidelines		
Deliver monthly project effort expenditure reports		
Include Requirements acquisition and Support activities in planning		++

4.2.4. Evaluate effect

The last phase in the Product Oriented Process Improvement model is the evaluation of the effects of the process improvement actions and the evaluation of the product quality at the end of the project. As the project is not finished yet, this last evaluation could not be done.
To measure the effect of actions taken with respect to the testing process a GQM programme has been started. Data is already being collected and the first feedback session is planned to be held soon.
Also tools have been developed for continuously tracking of defects and the registration of engineering hours, in order to make it more easier to manage the project. The first information that has been extracted from these tools is that the test team has started off well; many defects have been found.

4.3. Cost/Benefit analysis

In this section a cost/benefit analysis is done to evaluate the business impact of the application of the method in the case studies. Doing such an analysis is rather difficult, because it is easier to measure cost than to measure benefits. Cost can be deducted by measuring the effort in person hours, but it is very difficult to express benefits in financial terms, because they are often indirect and not objectively measurable.
Although it may be possible to estimate only the financial benefits by calculating expected profits, this will not be done in order to avoid the concept of multiple subjectivity, i.e. a number of subjective valuations are required to translate these benefits into a monetary value [Berghout, 1997].
Instead the cost/benefit analysis will be done as follows. The calculated cost (measured in man hours) will be valued against both financial and non-financial benefits. This may seem a strange way of doing a cost/benefit analysis. However, since the benefits can not be measured, an alternative for the complete cost/benefit analysis is to identify whether the effort expenditure (cost) has been worth the benefits [Casimir, 1998]. This level of evaluation is feasible and will provide insight for the decision whether such improvement activities should be continued or not.

4.3.1. Cost of the Product Oriented Process Improvement method

The cost is determined by measuring the effort in person hours. No translation will be made into a financial number, because this is a straight forward calculation. Furthermore it is not necessary, because the benefits will not be expressed in financial terms either.

In the tables 4-2 and 4-3 the results of the cost analysis for the OPT and the Omega projects are presented.

Table 4-2: Effort needed for the method application for the OPT case study (in person hours)

Phase	Quality Engineer	Quality Manager	Project Manager	Software Engineer	Total
Identify product quality needs	**50**	**4**	**8**	**10**	**72**
Gain commitment	8	4	2	2	16
Global survey quality needs	2		1		3
Multi-Party Chain chart	2		1		3
Extensive survey quality needs	36		4	8	48
Determine PQP quality needs	2				2
Estimate product quality	**4**				**4**
Product quality estimation	4				4
Adapt software development process	**30**	**1.5**	**1.5**	**3**	**36**
Comparison needs vs. estimation	2				2
Analyse candidate improvement actions	1				1
Write quality investigation report	24				24
Select process improvement actions in project meeting	3	1.5	1.5	3	9
Evaluate effect	**79**	**1.5**	**4**	**10.5**	**95**
Conduct GQM interviews	16		1	2	19
Develop GQM plan and measurement plan	24				24
Perform GQM data collection	24			4	28
GQM data analysis	8				8
GQM feedback session	7	1.5	3	4.5	16
Total	**163**	**7**	**13.5**	**23.5**	**207**

Table 4-3: Effort needed for the method application for the Omega case study (in person hours)

Phase	Quality Engineer	Quality Manager	Project Manager	Software Engineer	Total
Identify product quality needs	**74**	**8**	**8**	**15**	**105**
Gain commitment	8	4	2	2	16
Global survey quality needs	2		1	1	4
Multi-Party Chain chart	2		1		3
Extensive survey quality needs	60	4	4	12	80
Determine PQP quality needs	2				2
Estimate product quality	**4**				**4**
Product quality estimation	4				4
Adapt software development process	**70**	**11.5**	**3.5**	**15**	**100**
Comparison needs vs. estimation	2				2
Analyse candidate improvement actions	1				1
Write quality investigation report	24				24
Hold project meeting to present results	3	1.5	1.5	15	21
Develop Process Improvement Plan	40	10	2		52
Evaluate effect	**40**	**1**		**2**	**43**
Conduct GQM interviews	16	1		2	19
Develop GQM plan and measurement plan	24				24
Total	**188**	**20.5**	**11.5**	**32**	**252**

Note that there are some differences in the application of the method between the two projects, which influences the needed effort.

- In the extensive survey of the quality needs for the Omega, five interviews were held, for the OPT three interviews.
- For the OPT it was possible to select process improvement actions in the project meeting. For the Omega project this proved to be impossible. Instead, a complete investigation was conducted for the Omega, indicated with 'develop process improvement plan'.
- For the OPT the first feedback session has been held, already. For the Omega the first feedback session is planned to be held in the beginning of 1999.

4.3.2. Benefits of the Product Oriented Process Improvement method

It is difficult to indicate specific benefits to the application of the method that can be expressed in financial terms, like increased sales or reduced cost. This is especially difficult, because the application of the method has not been completed yet and the projects are still running. Even when the projects are finished this will still be difficult, because it is difficult to measure benefits objectively. For example, an increase in sales is rarely caused by increased product quality alone.

However, benefits can be assigned to the influence of the method application and can be determined in non-financial terms. These benefits can later be balanced against the cost. The benefits will be divided in direct and indirect benefits. The direct benefits are benefits that are directly related to the steps taken in the method and the benefits the method aimed for. The indirect benefits are the positive side effects of the application of the methods.

The direct benefits of the OPT and Omega case studies include:

- The quality needs for the OPT and Omega products are made explicit in measurable terms.
- Strong and weak sides of the OPT and Omega software development processes have been discovered.
- Based on the differences between the quality needs and the strengths and weaknesses of the software development process, problem areas have been identified for both projects, for which the development process is expected to be insufficient to meet the quality needs.
- The main national requirements for the OPT are now available, after effort was being invested as a result of the quality investigation.
- For the OPT a test checklist is created for the SSDs, which is expected to improve the SSD testing process and the OPT reliability.
- The status of OPT reliability is now much clearer as a result of the GQM measurement programme.
- A separate Omega test group has been started in Bladel and although the first feedback session of the GQM measurement programme has not been held yet, the first indications are that the testing process has improved and more bugs are being found, which will effect the reliability of the Omega.
- In the Omega project the requirements engineering phase is being taken into account in the planning better, so more effort is invested in this phase, which has to lead to more accurate requirements.
- The Omega software is put under configuration management, which will help in the version management of the software.

The indirect benefits of the OPT and Omega case studies include:
- The co-operation between quality assurance and the OPT software engineers has improved considerably.
- Direct attention and priority to software quality has increased quality awareness in the project teams.

4.3.3. Are the benefits worth the cost?

In the previous two subsections the cost and benefits of both projects are presented. The question that is now raised, is whether the benefits gained in the projects are worth the cost.

The first remark that has to be made about the benefits is that most of the results become visible when the project will be finished and when the products are being sold. For example, an increase in reliability will definitely have a positive influence on the service cost when the systems are installed in the field, but this effect is not known before. At this moment already some benefits are found. The evaluation of the benefits versus the cost is at this moment difficult to make, because the benefits and the cost seem to be close.

Because of the improved relationship between the quality assurance department and the OPT development team, the benefits seem to outweigh the cost, already considerably. The cost, 207 hours, are not that high and the increase in the attention towards quality and the co-operation with the quality assurance department even helps in justifying the budget of the quality assurance department in general. In this light, the effort has to be split into effort of the quality assurance department and effort of the project team. The cost for the project team consists only of 37 person hours, which is really not that much. The effort of the quality assurance department, 170 person hours, would otherwise be spent on other quality assurance activities like ISO 9001 audits and process assessments. The advantage of the new approach in relation to these activities is that the quality assurance department gets much closer to the project team. Also the increased attention towards requirements engineering, the creation of the SSD checklist and the gained insights in OPT reliability contribute to the justification.

For the Omega project the cost are substantially higher than for the OPT project. Even here, the total effort for the project team is no more than 43.5 hours. What has to be noted, however, is that no feedback session has been held yet, which would increase the cost compared to the OPT project. One of the major benefits for the project is that the renewed attention to software quality has increased the quality awareness and the attitude towards software quality. Also the Omega test team in Bladel has apparently good results, according to the information from the defect tracking tool, which will affect the reliability of the Omega and hopefully decrease the number of problems in the field. The configuration management system has been started which will help the software engineers in managing different software versions and extra attention is being given to the requirements definition phase, which has to lead to gaining more requirements. These benefits seem to make the balance for the Omega come out favourably for the new approach.

Further down the road, when the projects are finished and the results become available and based on these results new improvement actions are started, the benefits will undoubtedly increase further. It is therefore recommended to continue the method application in the future.

Summarising the following has to be noted. The cost of applying the method in real life projects is considerable, but not too high. Especially when the distinction is made between effort of the project team and effort of the quality assurance department. The benefits that are gained in the OPT case study already outweighs the cost considerably before a complete cycle has been finished. For the Omega case study the cost and benefits seem to be closer together, but the balance seems to be positive. The benefits are expected to become more clear and to increase after the project is finished. Therefore it is recommended to continue with the Product Oriented Process Improvement method.

5. Product quality profile scales

During the case studies with respect to the OPT and the Omega system it appeared that the scale used for the product quality profiles was inadequate. In this chapter the result of an analysis of this scale to indicate its weaknesses and what has to be done to improve the scale is presented. First a description on the different types of measurement scales that are used to assign a certain value to an object is given. In this case, a value for a level of quality. The chapter finishes with a proposal for a new scale for the product quality profiles.

5.1. Measurement scales

Measurement is the assignment of objects or events to a numerical system. Five levels of measurement scale types are commonly distinguished: nominal, ordinal, interval, ratio and absolute [Fenton & Pleeger, 1997]. The scale types listed are shown in increasing levels of richness, which means that the scales become more sophisticated and it is allowed to perform stronger statistical analysis methods.

5.1.1. Nominal scale

Nominal measurement consists of assigning items to groups or categories. No quantitative information is conveyed and no ordering of the items is implied. Nominal scales are therefore qualitative rather than quantitative. Frequency distributions are usually used to analyse data measured on a nominal scale. Variables measured on a nominal scale are often referred to as categorical or qualitative variables.

Some examples of measurement on a nominal scale are:
- classifying origin of defects into classes like design error, not tested fully, external reasons;
- classifying failure detection phases into classes like field, integration, and testing.

5.1.2. Ordinal scale

The ordinal scale has the same characteristics as a nominal scale, but has the additional characteristic that the categories can also be ranked in that one class is better, or higher than another class. The ranges between two classes have no meaning for the ordinal scale.

Some examples of measurement on an ordinal scale are:
- classifying failures into severity classes like Fatal, Major, and minor;
- classifying complexity into classes like high, medium, and low.

5.1.3. Interval scale

The interval scale is used when not only an order exists in more and less, but also is known how much more and less. Interval scales do not have a true zero point, however, and therefore it is not possible to make statements about how many times higher one score is than another.

A common example is the Celsius scale for indicating temperature, where a 10 degree difference has the same meaning anywhere along the scale. However, because the zero point is arbitrary, the freezing point for water, you can not say that a temperature of 30 degrees is twice as warm as one of 15 degrees.

5.1.4. Ratio scale

Ratio variables are very similar to interval variables; in addition to all the properties of interval variables, they feature an identifiable absolute zero point, thus they allow for statements such as x is two times more than y. A much used example to describe the distinction between interval and ratio scales is temperature. In contrast to the Celsius scale it is allowed to say that a temperature of 200 degrees is twice as high than one of 100 degrees with the Kelvin temperature scale, because the Kelvin scale has identified the true zero point for temperature, the temperature at which molecules stop moving.

5.1.5. Absolute scale

The measurement for an absolute scale is simply made by counting the number of elements, which you want to measure. The absolute scale resembles the ratio scale strongly. The only difference is that for the absolute scale the scale is unique (i.e. there is only one scale allowed for the measurement).
The absolute scale is the most restrictive scale. As such the scale can not be used often. A much used example to illustrate this, is the metric lines of code (LOC). LOC is an absolute scale measurement of the attribute 'number of lines of code', because there is only one way to measure this attribute. LOC is not an absolute scale measurement of the attribute 'size', however, because there are also other ways to measure this attribute, such as KLOC, number of characters and number of bytes.

5.2. ISO 14598-5 scale

The product quality profiles are used to indicate:

- the current quality, the quality level the previous version of the product has;
- the quality needs, the quality level the stakeholders want the product to be;
- the target quality, the level you want to reach at the end of the project;
- the expected quality, the quality level you will probably reach using the current process;
- final quality, the quality level finally reached at the end of the project.

The final quality then can be used again in the next project in which it will be used as current quality. By expressing these values all on the same scale you get the means to compare the values easily. Then you also can make statements about whether the quality level of a product is sufficient in a certain area or even if the project has met its goals.

For the case studies, presented in chapter 4, a modified version of the ISO 14598-5 standard is used as the scale for the various product quality profiles. The scale indicates risk, when quality is insufficient and not meeting requirements in a certain area, as indicated with ISO 9126.

The values of the scales are defined as follows:

-	Not relevant
D	Economic risk
C	High economic risk
B	Life risk for users
A	Life risk for other people than users

Because during the case studies it turned out that the steps between two successive values were too small and it seemed hard to assign the proper values to each quality subcharacteristics, it was decided to refine the scale using quartiles. The quartiles are indicated with plusses and minuses, so the steps between D and C are indicated as follows: D, D^+, D^{++}, C^-, C. Definitions have not been formulated for each quartile, so it is not exactly clear what is meant by a rate of for example D^+.

In terms of the analysis of section 6.1 the ISO 14598-5 scale is certainly on an ordinal scale. The risk progresses from - through D, C, B to A. The risk increases each step, but the amount in which it increases is not known and not the same for each step (the amount of increase in risk from D to C is not the same as the amount of increase in risk from C to B).

5.3. The drawbacks of the ISO 14598-5 scale

The modified ISO 14598-5 scale used for the product quality profiles has certain drawbacks.

The main drawback of the ISO 14598-5 scale is that the scale indicates risk if the requirements are not met and the quality is insufficient. Because the product quality profiles are used to express a quality level and you want to compare different levels with each other, the risk scale of ISO 14598-5 is not useful. Especially with respect to the final quality level, using risk is odd. The interpretation of the risk scale would be that the 'higher' the quality of the final product gets, the more risk you get for your company.
Of course, the risk only results from the mismatch between needed quality and final quality. If the final quality is much lower than the wanted quality, you will get risk for your company, in that sales decrease for example, or risk for users, because errors occur which cause life-threatening situations. If the levels are equal, however, you have neutralised the risk.

Furthermore the scale combines economic risk and life risk on one scale. For RPS, in which safety is no problem from software point of view, this means that half of the scale is useless. In the Omega and OPT case studies this can already be found in that the highest score reached is C^+.

Another drawback is that the lowest level '-' has the meaning 'normal'. Always some attention is given to each of the quality (sub)characteristics and no extra effort is needed for a quality characteristic indicated with '-'. For the quality needs this is not such a big problem, because always a certain need for quality exists, which could be indicated with the term 'normal'. Indicating poor quality on such a scale in an evaluation, becomes problematic, because there is no value below the 'normal' level.

Also the scale suggests that an equal score for two different characteristic means the same for the both of them. This is of course a wrong assumption, because by definition two different quality characteristics have different quality needs and have other means of coping with these needs.

The final drawback is somewhat technical. The ISO 14598-5 scale is an ordinal scale as was already determined in section 5.2, but the scores A, B, C, and D suggest that the scale is at least on an interval scale. The step from A to B appears to be same size as the step from B to C. From this perspective it would be better to use scores like poor, adequate and good, because then you do not suggest the scale to be richer than it is.

5.4. Improvements to the product quality profile scale

Based on the foregoing it is now possible to state the conditions the product quality profile scale has to comply with.

- The product quality profile scale should indicate the quality level for each ISO 9126 quality (sub)characteristic. It should indicate how good the quality is, or has to be. This indication should be company specific. The level will have to be drawn-up according to a RPS perspective. In that case the complete scale will be used.
- The term used for a value should already indicate what the value means.
- It should be defined what each level of quality means.
- Metrics should be provided with the scale, so it is possible to objectively determine the level.
- I guess it will be very difficult to come up with a scale that is richer than the ordinal scale, because it is hard to make quality quantifiable without losing information. If possible the scale should be richer than the ordinal scale, however.

5.5. Proposed product quality profile scale

Based on the demands stated in section 5.4 a new product quality profile scale will be proposed. It is decided to create a scale which is also on the ordinal level, because of the expected problems in trying to create a scale on a richer level and the time constraints of the research.
Because the scale will be on an ordinal level, this should be reflected in the terminology. Ratings like 1, 2, 3 ... and even A, B, C, ... suggest otherwise and should therefore be avoided. Furthermore it is decided to create a scale with five levels. This has the advantage that there is the possibility to get a score that is in the middle of the scale. One might say that this is a disadvantage, because people tend to answer defensively with the middle answer. Because of the way that the value is assigned I want to dismiss this argument. The product quality profiles are constructed in three different stages:

1. <u>The global survey of the quality needs.</u> Here the score will be obtained indirectly, using the relation matrix for the embedded software questionnaire.
2. <u>The extensive survey of the quality needs.</u> In this step the final value will be assigned by the quality engineer based on the information obtained from the interviewee.
3. <u>The product quality estimation.</u> In this step the value will again be assigned by the quality engineer based on the process assessment report.

The values of the scale are not allowed to be judgmental. A scale like satisfactory - good - excellent can therefore not be used, because by definition then all outcomes should get 'satisfactory'. That is, satisfactory to the user's quality needs.
A scale that does represent a certain kind of level, but is not judgmental is the high - medium - low scale. For some quality needs, surely the level 'low' can be used. This 'low' level then is satisfactory for a user. The advantage of this scale is also that the 'distance' between the values are intuitively equal. To indicate especially high demands, the value essential is used, and to indicate that a certain characteristic does not apply for a product,

the term 'not relevant' will be used. In frame 5-1 the definitions for each of the values of the scale can be found.

Frame 5-1: Definitions for the values of the proposed product quality profile scale

not relevant	no special attention is needed for this area
low	the quality level is or has to be of a low level
medium	the quality level is or has to be of a medium level
high	the quality level is or has to be of a high level
essential	the quality level is or has to be excellent to assure safety or assure approval

An alternative scale is the scale provided in frame 5-2. This scale simply makes a reference to the relation of the quality level to the 'normal' level of quality. The drawback of this scale is that it is not clear who decides on the question 'what is normal quality?'. The scale's simplicity makes it, however, also appealing.

Frame 5-2: Alternative product quality profile scale

not relevant	no special attention is needed for this area
below normal	the quality level is or has to be below the normal level
normal	the quality level is or has to be like the normal level
above normal	the quality level is or has to be above the normal level
essential	the quality level is or has to be excellent to assure safety or assure approval

In the case studies the stakeholders were able to assign priorities to the product quality requirements according to the low - medium - high - essential division. Therefore it has proven its strength and it is decided to use this scale.

The definitions presented in frame 5-1 have to be used for each of the ISO 9126 quality (sub)characteristics. I thought about formulating separate definitions for each of the quality (sub)characteristics, but dropped it after realising that probably no better definitions than terms like 'highly reliable' for reliability and so forth could be expected.
Therefore I recommend that for each ISO 9126 (sub)characteristic metrics are determined instead, which can help in indicating the meaning of a specific value for a specific subcharacteristic. The metrics will have to be company specific, to keep the scale company specific. ISO 9126-2 will be a good starting point to find these metrics. For some of the quality (sub)characteristics it may be hard to find good metrics, however, because many metrics only cover part of a quality (sub)characteristic.
For example, a metric that could be used for the ISO 9126 quality subcharacteristic maturity is the Mean Time Between Failures [ISO/IEC, 1996], the average operation time that passes between two successive failures. In frame 5-3 an example is given, how such a metric could be used.

Frame 5-3: Example of using metrics to make the scale explicit

ISO 9126 subcharacteristic		**Metric**
Reliability		Mean Time Between Fatal Failures
	not relevant	Not applicable
	low	24 hours
	medium	168 hours
	high	672 hours
	essential	8760 hours

In this research the metrics will not be defined for each of the characteristics, because in my opinion this would be a complete research on its own.

6. Product and process approaches

In chapter 3 the Product Oriented Process Improvement model is developed. In this chapter this model will be examined closer with respect to the two different views on quality improvement: product oriented quality improvement and process oriented quality improvement as introduced in chapter 2.

6.1. Product or Process

In this section the Product Oriented Process Improvement model will be analysed with respect to the product and process approaches already described in chapter 2. The model will be analysed phase by phase.

6.1.1. Identify product quality needs

The first phase of the Product Oriented Process Improvement model is the identification of the product quality needs. As the term itself already hints, this phase mainly consists of product related steps.
The first step of this phase, however, gain commitment, is a step that supports the execution of the complete method. Therefore it is difficult to relate it to process improvement or product improvement. The goal of the method is to improve the software process, based on the product quality needs. Therefore it is decided that gain commitment is both a process oriented step and a product oriented step and will be indicated as such.
The second step in this phase is the global survey of the quality needs. The embedded software questionnaire [SPACE-UFO consortium, 1998a] containing product related questions is used to gather information about the product quality needs. Also an initial product quality profile, indicating these product quality needs is determined. Therefore this

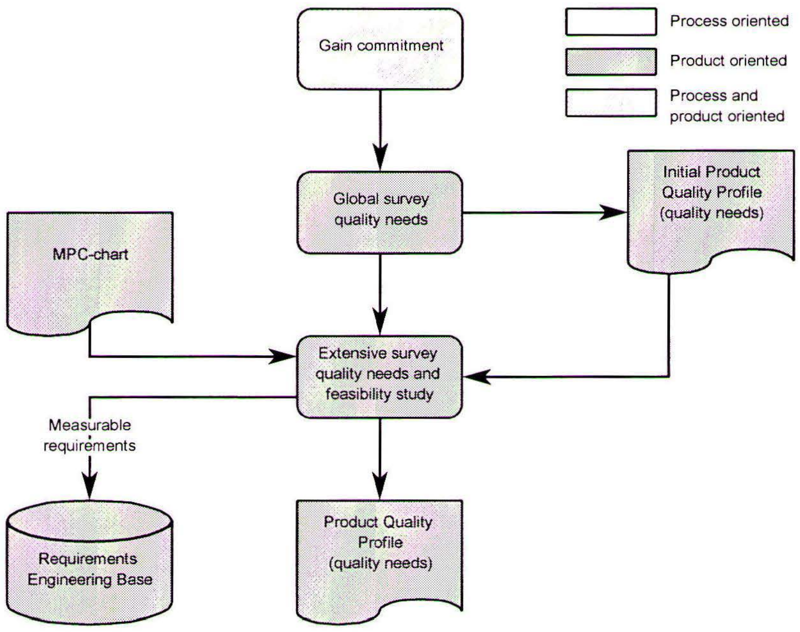

Process oriented
Product oriented
Process and product oriented
Gain commitment
Global survey quality needs
Initial Product Quality Profile (quality needs)
MPC-chart
Extensive survey quality needs and feasibility study
Measurable requirements
Requirements Engineering Base
Product Quality Profile (quality needs)

Figure 6-1: Identify product quality needs phase

step is definitely a product related step.
After the global survey, the extensive survey is started to get more detailed information about the product quality needs. For the extensive survey, the various stakeholders in specifying product (quality) requirements, as identified in the Multi-Party Chain chart, are interviewed in order to gather the product quality requirements. Based on the expressed quality requirements a final product quality profile for the quality needs is drawn up. This step is definitely related to the software product, but in the case studies, it turned out that some stakeholders, especially developers, express their wishes in terms of process terminology. For the Omega system a wish was that it is necessary to do full regression testing. This is clearly a wish in terms of the process. What is actually meant on a product level is that the system has to be reliable. One way of doing that, and to cope with changes in the software, is full regression testing. Whether this is the best solution is not certain and it is better to let the designers make that decision. Therefore it is necessary that the process related wishes are translated into product related wishes.

In figure 6-1 the first phase, identify product quality needs, is presented with the division in process and product related steps. Product related steps are indicated in grey, process related steps are indicated in white.

6.1.2. Estimate product quality

In the estimate product quality phase the status of the current process is used to make an estimation of product quality. Two sources are used as input for this step, the process assessment report and the knowledge base.
The process assessment report identifies what the current software development process is like and therefore it can be derived which actions are currently done in the development process. This input is therefore clearly process related.
The knowledge base contains information about the impact of process actions on product quality. As such, the knowledge base is used to translate process information into an expected product quality level. Therefore the knowledge base cannot be seen as a process or product related input, but must be seen as both process related as product related.
The estimation process itself has to be described as product related, because the step aims to come to an expected level of product quality. The expected level of product quality is expressed in a product quality profile, also product related.

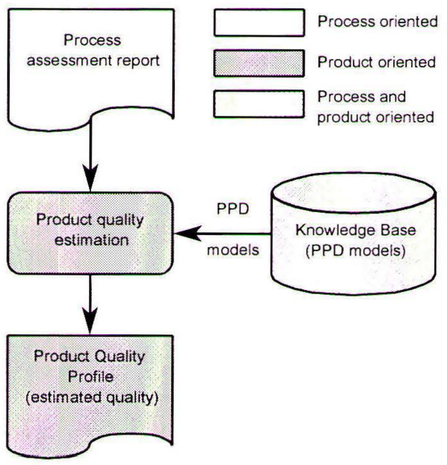

Process assessment report
Process oriented
Product oriented
Process and product oriented
Product quality estimation
PPD
models
Knowledge Base (PPD models)
Product Quality Profile (estimated quality)

Figure 6-2: Estimate product quality phase

In figure 6-2 the above described analysis is presented graphically. The knowledge base is cross-hatched to indicate that it is related to both the process as the product.

6.1.3. Adapt software development process

In the phase adapt software development process, decisions are taken to adapt the software development process. The decisions are based on the product quality analysis performed in the first two phases.
The phase starts with both product quality profiles, the one indicating quality needs and the estimated quality which is based on the current process. The two quality profiles are compared in order to identify problem areas; areas for which the levels of quality needs and estimated quality are different. Clearly this step is related to the software product quality approach.
After this first step for each of the problem areas, candidate process improvement actions are identified using the knowledge base. This knowledge base provides information on the impact of process improvement actions on the product quality. Because the emphasis in this step is on the identification of software process improvement actions, this step has to be classified as a process related step. As the software process improvement actions are only selected in order to accommodate certain problem areas for which the product quality is expected to become below target, this step has also a product related angle as well.
A selection of which process improvement actions actually will be implemented is made in co-operation with the project team. Finally a process improvement plan is written and the process improvement actions are implemented in the project. In these last steps, the emphasis is definitely on the process

In figure 6-3 the adapt software development process phase with its division into product and process related steps is presented.

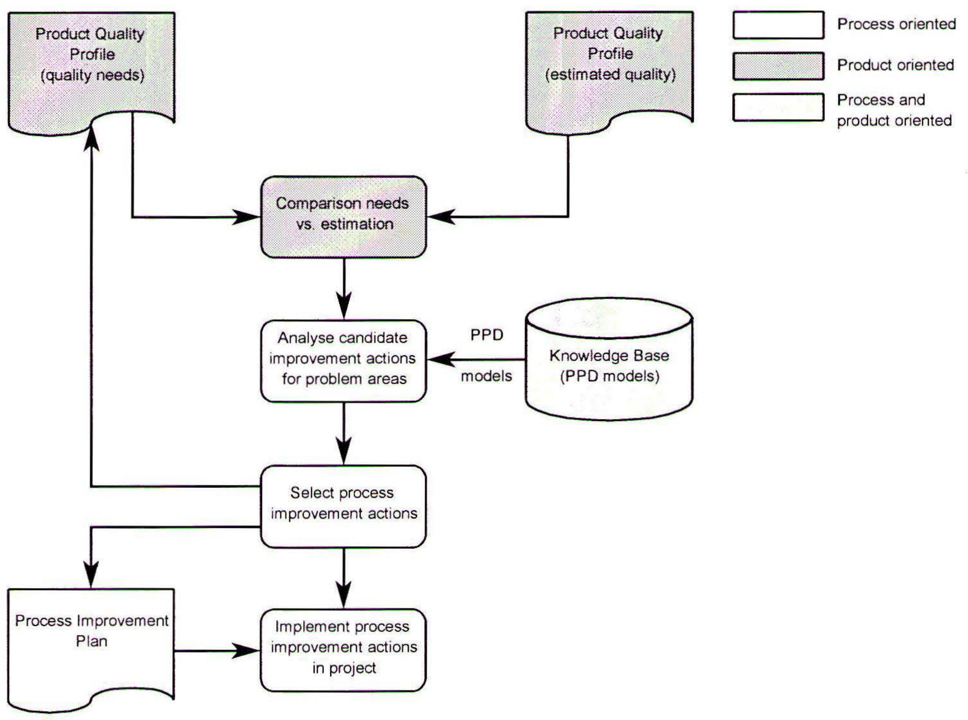

Product Quality Profile (quality needs)
Product Quality Profile (estimated quality)
Process oriented
Product oriented
Process and product oriented
Comparison needs vs. estimation
Analyse candidate improvement actions for problem areas
PPD models
Knowledge Base (PPD models)
Select process improvement actions
Process Improvement Plan
Implement process improvement actions in project

Figure 6-3: Adapt software development process phase

6.1.4. Evaluate effect

The last phase of the Product Oriented Process Improvement model is the evaluation of the effects of the project. This evaluation is done in two ways, evaluating the product itself and evaluating the effect of process improvement actions.
In the product quality evaluation the product quality requirement metrics are used to evaluate to what extent the project has met the needs with respect to product quality. Therefore it qualifies as a product oriented evaluation approach.
The evaluation of the effect of the process improvement actions is done by starting measurement programmes for these improvement actions. Therefore these steps can be classified as process related steps. The last step, update knowledge base, is considered to be as well process related as product related. In this step Product Process Dependency models are being derived based on the evaluation. It therefore contains the link between product and process. The PPD models are stored in the knowledge base.

In figure 6-4 the evaluation phase is shown according to the analysis presented above.

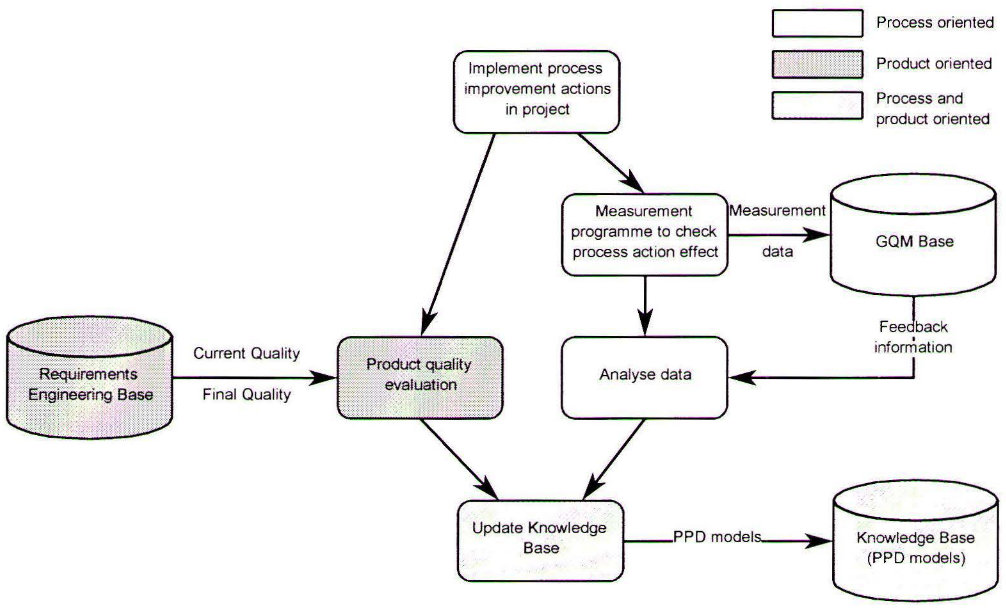

Process oriented
Product oriented
Process and product oriented
Implement process improvement actions in project
Measurement programme to check process action effect
Measurement data
GQM Base
Feedback information
Requirements Engineering Base
Current Quality
Final Quality
Product quality evaluation
Analyse data
Update Knowledge Base
PPD models
Knowledge Base (PPD models)

Figure 6-4: Evaluate effect phase

6.1.5. Product Oriented Process Improvement model

In chapter 2 is concluded that it is necessary to use as well process oriented approaches as product oriented approaches in order to come to better quality products; products that meet the user/customer requirements. From the analysis above can be concluded that it is indeed possible to use product and process oriented quality improvement methods together in order to come to better software products. Software Process Improvement approaches like a process assessment and the implementation of software process improvement actions in projects are combined with surveys of product quality needs, estimations of product quality, and product quality evaluations, using the ISO 9126 product approach.
The distinction in process and product related steps is however not as clear as it seems. Some product related steps need process related input or have process related sides and vice versa. While using the Product Oriented Process Improvement model this has to be taken into account, because it otherwise might cut back on the results. For example, in the

extensive survey for product quality needs, the interviewer has to take into account that the interviewee often tends to answer in terms of solutions (process related) instead of problems (product related). If the interviewer does not notice this, the true reason why a solution is brought up, may be lost. The interviewer must ask the interviewee why he wants this particular solution, and what problem he wants to solve with this solution. Then the quality problems that cause the interviewee to mention such a solution is known exactly and can be reckoned with. This does not mean, however, that the solutions the interviewee brings up, have to be ignored. In fact these solutions may be of great use in the step 'select process improvement actions. Therefore this indirect link from the extensive survey of the quality needs to the selection of process improvement actions will be included in the model.
In figure 6-5 the complete Product Oriented Process Improvement model with the division into process and product related steps can be found.

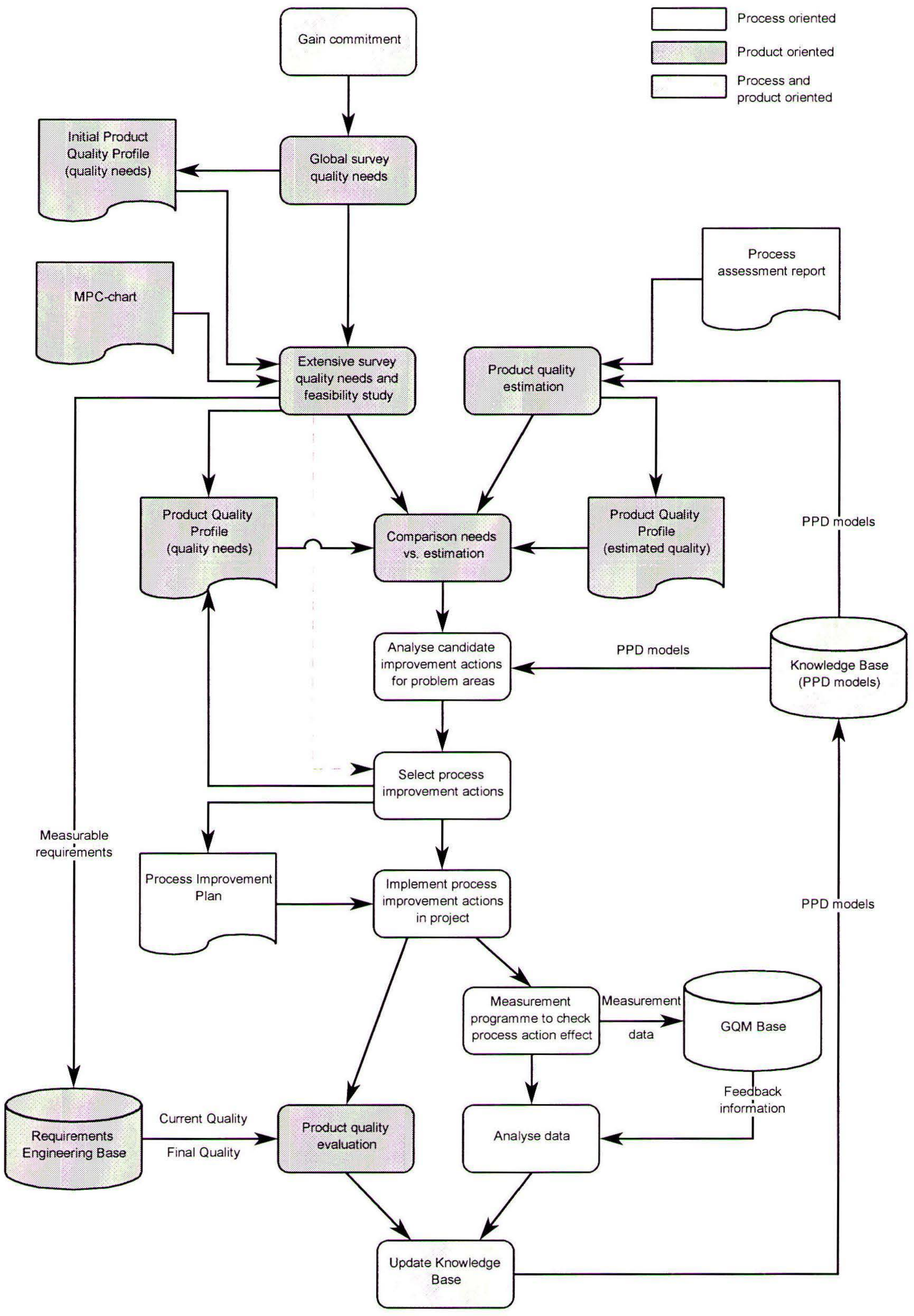

Process oriented
Product oriented
Process and product oriented
Gain commitment
Initial Product Quality Profile (quality needs)
Global survey quality needs
Process assessment report
MPC-chart
Extensive survey quality needs and feasibility study
Product quality estimation
Product Quality Profile (quality needs)
Comparison needs vs. estimation
Product Quality Profile (estimated quality)
PPD models
Analyse candidate improvement actions for problem areas
PPD models
Knowledge Base (PPD models)
Select process improvement actions
Measurable requirements
Process Improvement Plan
Implement process improvement actions in project
PPD models
Measurement programme to check process action effect
Measurement data
GQM Base
Feedback information
Requirements Engineering Base
Current Quality
Final Quality
Product quality evaluation
Analyse data
Update Knowledge Base

Figure 6-5: Product Oriented Process Improvement model

6.2. Product or process related metrics

In the evaluation phase of the Product Oriented Process Improvement model a product quality evaluation is done by collecting information by the definition of metrics. These metrics are specified for the product quality requirements, identified in the extensive survey of the product quality needs.
In section 2.3 software metrics are divided in three different entity classes: processes, products and resources. In this section this distinction is used to analyse the metrics from the OPT case study.

The product quality evaluation uses metrics which are specified for the product quality requirements. It is therefore expected that these metrics can be classified as product metrics.

Actually analysing the metrics for the OPT project reveals that a reasonable number of metrics have to be classified as process metrics. In figure 6-6 the distribution over product and process metrics for each of the parties can be found. From the graph can be concluded that there is a major difference between RPS internal parties and RPS external parties. The internal parties development, management, marketing and SSD all have process metrics next to the expected product metrics. The external parties bank, customer, end-user and government all the requirements can be classified as product metrics. None of the requirements can be classified as a resource metric.
Of course the difference found between internal and external parties can be explained quite easily. The internal parties know about the way in which the product is developed and therefore it is likely that matters concerning the process are brought up. The external parties are only concerned with the behaviour of the product itself and that is why for the external parties only product metrics are found.

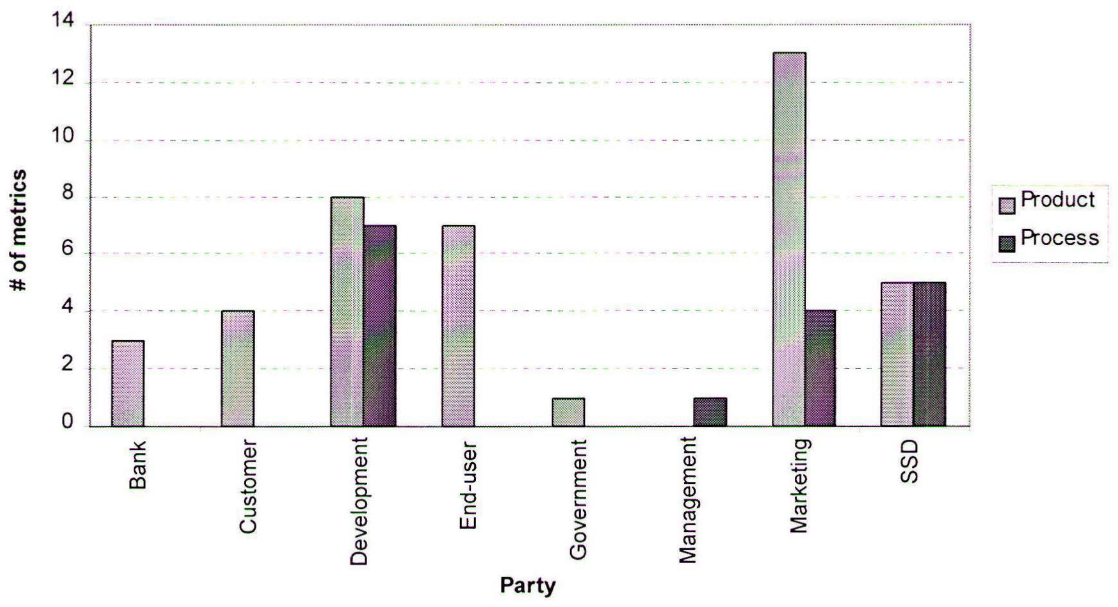


Figure 6-6: Classification of OPT metrics in product and process

Frame 6-1: Example of a internal process metric

Metric	**Current value**	**Wanted value**
Number of months needed to implement a new feature	2 to 4 months	2 months

In frame 6-1 an example of a metric that has to be classified as a process metric can be found. The metric 'number of months needed for the implementation of a new feature' says something about the process 'implementing a new feature'. Therefore it has to be classified as a process metric.
The metric is related to a requirement from management that it must be able to change the functionality of the OPT, as result of market requirements, quickly. The reason for this requirement can best be explained as follows. The retail petroleum systems market changes quickly. Competitors are continuously searching for new functionality, which makes their product stand out in the market, in order to get more sales. What management wants to accomplish with this requirement is that RPS' products are able to follow these trends in the market quickly and that no sales are lost because of these extra features. At the moment it takes from two to four months to update the product, this time should be shorter, two months maximum.
Basically this wish is based on needs for the OPT product, developing a product that is competitive in the market. The terminology of the requirement, however, must already be classified as related to the software process: 'changing OPT functionality quickly'. The metric 'number of months to implement a feature' seems to be completely related to the software process.

The above shows that it is very difficult to look at product and process separately. Starting with a desire for the product, keeping the product up to date with the competitors, the metric, lead time of implementing a new feature, ends up to be related to the software process. Looking at the requirement itself, I do not think it is less valid, because it is described in process terms, in spite of the fact that the goal of the interview was to find out which product quality requirements exist. I also think that a requirement like this can be used to identify product quality needs. The requirement sets demands for the changeability of the product, which are undeniably there. A problem occurs however, with the product quality evaluation. It is not possible to evaluate the product with such a metric. Therefore during the interviews the interviewer has to deal with this fact and has to try to avoid it.

7. Conclusions and recommendations

To conclude this report the conclusions of my research are summarised in this chapter. Besides these this thesis is completed with recommendations that originate from my experience at RPS.

7.1. Conclusions

The conclusions are divided over three different subsections. First, the conclusions with respect to product or process related quality improvement and the way in which SPIRITS, SPACE-UFO and PROFES deal with this are presented. After that the conclusions from the integration of the three methods as developed within these projects are drawn. Finally conclusions about the Product Oriented Process Improvement model are provided.

7.1.1. Product or process oriented quality improvement

As the importance of embedded products and embedded product quality is increasing, software developers are active in improving the quality of embedded software. There are two different approaches in improving this quality: the process oriented and the product oriented approach. The process oriented approach tries to improve product quality indirectly, by controlling and improving the software development process. Thereby the real effect of the process improvement activities on the product quality is neglected. The product oriented approach aims to make software product quality explicit, by the identification of product quality needs and the evaluation of the products to these needs. After identifying product quality needs the product still has to be developed. The software will be created using the software development process. In order to improve the software product quality, the software development process has to be improved. By integrating the product and process oriented approaches in an effective and efficient way, the quality improvement activities have to get the best results.
This is recognised in three quality improvement projects RPS participated in: SPACE-UFO, SPIRITS, and PROFES. For each of the projects the specific emphasis of integrating process and product oriented approaches is different. SPACE-UFO puts the emphasis on identifying quality needs and evaluating product quality, but recognised that it is also necessary to take specific process improvement actions to meet the product quality needs. In spite of the fact that the acronym of SPIRITS (Software Process ImpRovement in embedded IT environmentS) suggest that it is mainly about software process improvement, SPIRITS contains also methods for identifying product quality needs and evaluating product quality. Furthermore SPIRITS realises that the relation between product and process has to be established and therefore a specific tool has been constructed. PROFES goes even one step further and by means of literature research these product and process relations have been determined.

7.1.2. Integrating the methods

For RPS it is important that one method for quality improvement is created which takes the best parts of each of the projects. Therefore the methods of each of the projects are analysed on their strengths and weaknesses in chapter 3. Below some conclusions are drawn about the strong and weak points of the projects.

- PROFES defines a special step to gain commitment in the project team. What is awkward is that this step is done after identifying product quality needs. Of course also commitment from the interviewees is needed for the interviews.
- SPACE-UFO has created the questionnaire to be filled in by various users. The questions are objective to such an extent that it is only necessary to interview one person using the questionnaire. A second person would just provide the same answers. The project manager is the right person to interview, because he is most familiar with the product and is able to give all answers.
- Specifying product quality needs according to the SPIRITS Multi-Party Chain approach gives a lot of detailed information, but the interviews lack structure, which makes the results dependent on the skills of the interviewer. The SPACE-UFO way of specifying product quality needs is the use of the embedded software questionnaire. This questionnaire works very easily, structured, quickly and is quite accurate, but gives to little detailed information. The two techniques should be used subsequently. The results of the embedded software questionnaire should be used as input for the Multi-Party Chain interviews. The initial product quality profile can provide a structure for the interviews in that the interviewer is already familiar with aspects of the quality needs for such a product.
- SPACE-UFO aims at coming towards a method for the evaluation of software products. Because it is decided to leave the development of actual evaluation modules out of the SPACE-UFO project it is difficult to use the method for product evaluation.

7.1.3. The Product Oriented Process Improvement model

The end result of integrating the best parts of SPIRITS, SPACE-UFO and PROFES is the Product Oriented Process Improvement model, which incorporates the following general idea for software quality improvement. The first phase in making better quality products, is identifying the product quality needs, because it is important to know where you are going. Knowing where you are going is not enough, however, because it is also necessary to know where you are at the moment. Therefore the second phase contains a method for estimating what product quality you will likely get with the current process. The results of the first two phases are in analysed phase 3. For product quality areas where the current process is not likely to deliver the quality needs, actions will have to be implemented. In order to check whether the quality improvement actions have the required effect, the product quality is evaluated and measurement programmes are started. Feedback is tended to by means of a knowledge base.

The method has been applied in two RPS case studies, OPT and Omega. The results in these case studies are satisfactory. It appears that focused process improvement actions can be implemented in the projects according to the product quality investigations. Although the quality improvement programmes have not been finished yet, the benefits already outweigh the costs. When the improvement projects finish, the benefits are expected to become clearer and increase even more. From the case studies some additional conclusions can be drawn.

- The interviewer has to take into account that interviewees in product quality needs specification step tend to speak in terms of solutions. Especially interviewees representing internal parties bring up process related improvement actions. The goal of the interviews is to identify the product quality needs, however. Therefore it is important that the interviewer is prepared for this fact, so the product related needs that cause the interviewee to bring up such solutions are revealed as well.
- The extent to which certain process actions are implemented in the software development process are not taken into account in the PPD models. The effect of a full test on reliability is different from a test in which roughly the functionality is tested. The quality engineer has to take that into account during phase two in which an estimation of product quality is made, based on the software development process.
- The influence between two process actions is also not recognised in the PPD models. Certain process actions can strengthen each other, but it is also possible that certain process actions counteract each other. The quality engineer has to take such interactions into account as well.
- The estimation of product quality based actions taken in the development process, is not expected to be very accurate. In fact the estimation is not much more than an educated guess and it is highly dependent on the skills of the estimator. Nevertheless, the estimation can reveal product quality areas to which the current software development process is insufficiently suited to.
- The product quality profiles have to be seen as a graphical representation of the results of the investigations, not as the final outcome. The findings that lead to the special characteristics of a product quality profile always have to be conveyed together with the profile.
- The modified ISO 14598-5 scale that is used for the product quality profiles has certain drawbacks. The scale indicates risk if the requirements are not met and the quality is insufficient, instead of the quality level which is actually needed to compare different values. Also half of the scale is not applicable for RPS, since it involves risk to life. And the scale suggests being richer than the ordinal scale it is. A scale that addresses some of the drawbacks, is the not relevant – low – medium – high – essential scale, as represented in section 5.5.

7.2. Recommendations

Finally several recommendations are given to further enhance the RPS software development practice.

- The limitations of the PPD models have to be reckoned with. The extent to which process actions are implemented in the software development process and the influence between two process actions is not recognised in the method, or is at least not supported by the knowledge base. Although it is not feasible to research all influences between process actions, the method should be able to deal with the situation and the knowledge base should be modified to cope with it. Also a way should be developed to record partial implementation of process actions in the development process.

- In the previous section the conclusion is drawn that the ISO 14598-5 scale that is being used for the product quality profiles is not suitable. The scale that is proposed in section 5.5 addresses some of the drawbacks of this scale. For each of the ISO 9126 quality (sub)characteristics metrics still should be determined, which help in indicating the meaning of a specific value. Further research should be done to determine these metrics and find the values for the RPS context.
- The method for evaluating product quality is still not very good. The comparison of final values with the needed values of the quality requirement metrics is a good start for evaluation, but a better method has to be developed to improve this process.
- The OPT and Omega case studies have provided good results already. The main benefits, however, are expected to come when the method is applied continuously. The findings of the case studies, should be input for new improvement cycles in the future, so the Omega and OPT quality will improve even more. It is therefore recommended to continue the application of this method in the future.

Literature

Basili, V.R.; Caldiera, C.; Rombach, H.D., Goal Question Metric Paradigm, Encyclopedia of Software Engineering, volume 1, John Wiley & Sons, 1994.

Berghout, E., Evaluation of information system proposals: design of a decision support method, doctoral dissertation, Delft University of Technology, Delft, 1997.

BOOTSTRAP Institute, Bootstrap Assessor Training Course, BOOTSTRAP Institute, 1997.

Casimir, R., Kosten-batenanalyses worden bewust weggelaten, Automatiseringsgids, January 2, 1998 (in Dutch).

Deming, W. E., Out of the Crisis, MIT Center for Advanced Engineering Study, Cambridge, 1982.

Diaz, M.; Sligo, J., How Software Process Improvement helped Motorola, IEEE Software, September 1997.

Dion, R., Process Improvement and the Corporate Balance Sheet, IEEE Software, July 1993.

Downes, V.A.; Goldsack, S.J., Programming Embedded Systems with ADA, Prentice-Hall, Englewood Cliffs, 1982.

European Software Institute, European Software Institute WWW Site, European Software Institute, http://www.esi.es, 1998.

Fenton, N.E.; Pfleeger, S.L., Software Metrics - A Rigorous & Practical Approach, Internation Thomson Publishing, 1997.

Humphrey, W.S., Managing the Software Process, Addison Wesley Publishing Company, 1989.

Humphrey, W.S., Introduction to Software Process Improvement, Software Engineering Institute, Pittsburgh,1992.

Humphrey, W.S.; Snyder, T.R.; Willis, R.R., Software Process Improvement at Hughes Aircraft, IEEE Software, July 1991.

IEEE Standards Collection, Software Engineering, Institute of Electrical and Electronics Engineers, New York, 1994.

ISO/IEC, ISO 9126: Information Technology - Software quality characteristics and metrics, ISO/IEC, 1996.

Jonge, E.M.G. de, Requirements Engineering of Embedded Software Products, Graduation Report, RPS, Bladel, 1998.

Latum, F. van; Solingen, R. van; Oivo, M.; Hoisl, B.; Rombach, D.; Ruhe, G., Adopting GQM-Based Measurement in an Industrail Environment, IEEE Software, January 1998.

Looijen, M., Beheer van Informatiesystemen, Kluwer Bedrijfswetenschappen, Deventer, 1995.

Murez, J.P., RPS Development Process Model, Schlumberger RPS, Bladel, 1994.

PROFES consortium, Detailed specification of specific embedded systems characteristics, PROFES consortium, 1997a.

PROFES consortium, Specification of PROFES methodology, PROFES consortium, 1997b.

PROFES consortium, BOOTSTRAP Assessment Report, PROFES consortium, 1998a.

PROFES consortium, Schlumberger RPS Improvement Plan - Omega project, PROFES consortium, 1998b.

Rodenbach, E.; Solingen, R. van, Final Report of the SPIRITS project, Schlumberger RPS, Bladel, 1998.

Soerjoesing, S.P., Measurement Programme Documentation, Schlumberger RPS, Bladel, 1998a.

Soerjoesing, S.P., Product-Process Dependency Modelling, Research thesis, Schlumberger RPS, Bladel, 1998b.

Soerjoesing, S.P.; Uijtregt, A. van, Product Quality of the Omega systems, Schlumberger RPS, Bladel, 1998.

Solingen, R. van; Rodenbach, E., Embedded Software produkten voor benzinestations, Informatie (Dutch monthly), Kluwer, July/August 1996.

Solingen, R. van; Uijtregt, S. van, Partnership with Customers in Product Improvement - Testing Embedded Software Products in the Field, International Conference on Reliability, Quality and Safety of Software-Intensive Systems (ENCRESS '97), Athens, May 1997.

SPACE-UFO consortium, D2131: Specification report of instruments - addendum embedded software, SPACE-UFO consortium, 1998a.

SPACE-UFO consortium, D823P: Results of the SPACE-UFO project, SPACE-UFO consortium, 1998b.

SPACE-UFO consortium, Software Quality Specification for the Omega system, SPACE-UFO consortium, 1998c.

Uijtregt, A. van, GQM Measurement Programme Documentation, Schlumberger RPS, Bladel, 1998a.

Uijtregt, A. van, The Development of a Generic Software Measurement Information System, Schlumberger RPS, Bladel, 1998b.

Uijtregt, A. van; Solingen, R. van, Investigating product quality for the OPT system range, Schlumberger RPS, Bladel, 1998.

Veenendaal, E. van, Embedded Product Reliability & Process Improvement, Schlumberger RPS, Bladel, 1997.

Glossary of Terms

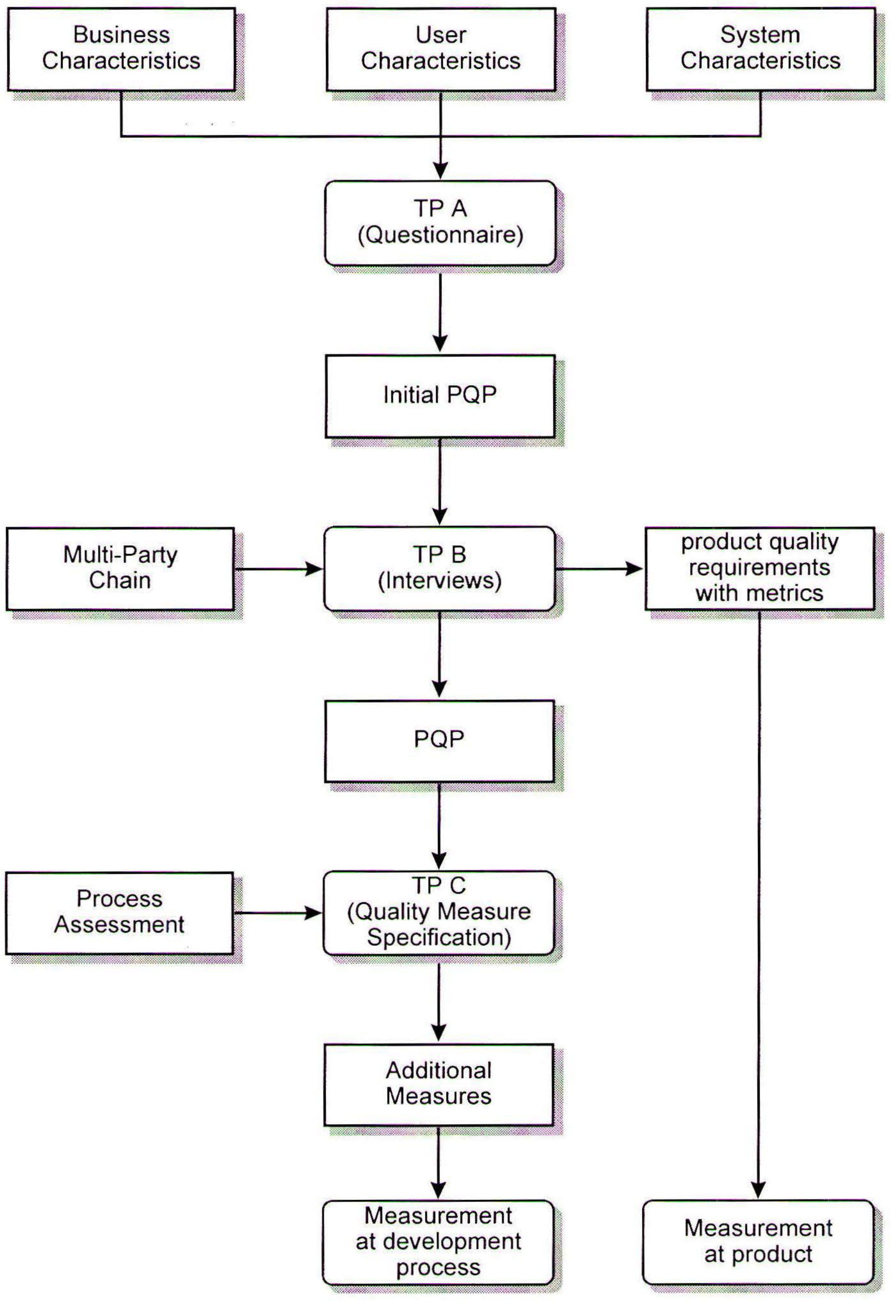
Embedded product	An autonomous physical unit, consisting of hardware (electronics) with software embedded in it, between which there is interaction by way of a specific interface [van Solingen & Rodenbach, 1996].
Measurement	The process by which number or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules.
Process action	A (decisive) step which is taken to achieve an expected or anticipated result [Soerjoesing, 1998b].
Process assessment	A general term for a review of the processes used in developing and maintaining software. Sometimes used more narrowly to mean the status of a project in the context of how closely it comes to its planned rate of progress. Also used more restrictively in context with other terms, such as 'stage assessment' or 'quality assessment'.
Product-Process Dependency	A verified relationship between a process action and a software product quality characteristic [Soerjoesing, 1998b].
Quality	The degree to which a system, component, or process meets customer or user needs or expectations [IEEE Standards Collection, 1994].
Quality assurance	The whole set of planned and systematic activities to provide a sufficient level of confidence that a product or a service is in conformance with the defined requirements [Looijen, 1995]
Quality requirement	A condition, capability or statement that must be met or possessed by a software product to satisfy a specific type or degree of quality as seen from the viewpoint of a party in a given role [de Jonge, 1998].
Software development process	is a term used to describe the people, methods, and tools used to produce software products [IEEE Standards Collection, 1994].
Software Process Improvement	The implementation of an appropriate culture and organisational procedure designed to support and enable the continual improvement of development processes, based on the results of measurement data and assessment results [European Software Institute, 1998].
Software product	The complete set of computer programs, procedures, and possibly associated documentation and data designated for delivery to a user [IEEE Standards Collection, 1994].

Appendices

Appendix A: ISO 9126 quality characteristics

Functionality:	The capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions.
Suitability:	The capability of the software to provide an appropriate set of functions for specified tasks and user objectives.
Accuracy:	The capability of the software to provide the right or agreed results or effects.
Interoperability:	The capability of the software to interact with one or more specified systems.
Security:	The capability of the software to prevent unintended access and resist deliberate attacks intended to gain unauthorised access to confidential information, or to make unauthorised modifications to information or to the program so as to provide the attacker with some advantage or so as to deny service to legitimate users.
Reliability:	The capability of the software to maintain the level of performance of the system when used under specified conditions
Maturity:	The capability of the software to avoid failure as a result of faults in the software.
Fault tolerance:	The capability of the software to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.
Recoverability:	The capability of the software to re-establish its level of performance and recover the data directly affected in the case of a failure.
Usability:	The capability of the software to be understood, learned, used and liked by the user, when used under specified conditions.
Understandability:	The capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use.
Learnability:	The capability of the software product to enable the user to learn its application.
Operability:	The capability of the software product to enable the user to operate and control it.
Attractiveness:	The capability of the software product to be liked by the user.
Efficiency:	The capability of the software to provide the required performance, relative to the amount of resources used, under stated conditions.
Time behaviour:	The capability of the software to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions.
Resource utilisation:	The capability of the software to use appropriate resources in an appropriate time when the software performs its function under stated conditions.
Maintainability:	The capability of the software to be modified.
Analysability:	The capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified.
Changeability:	The capability of the software product to enable a specified modification to be implemented.
Stability:	The capability of the software to minimise unexpected effects from modifications of the software.
Testability:	The capability of the software product to enable modified software to be validated.
Portability:	The capability of software to be transferred from one environment to another.
Adaptability:	The capability of the software to be modified for different specified environments without applying actions or means other than those provided for this purpose for the software considered.
Installability:	The capability of the software to be installed in a specified environment.
Co-existence:	The capability of the software to co-exist with other independent software in a common environment sharing common resources.
Replaceability:	The capability of the software to be used in place of other specified software in the environment of that software.

Appendix B: Initial Product Oriented Process Improvement model



Appendix C: Characterisation of the OPT project

This section contains the background information about the Outdoor Payment Terminal relevant for the OPT case study.

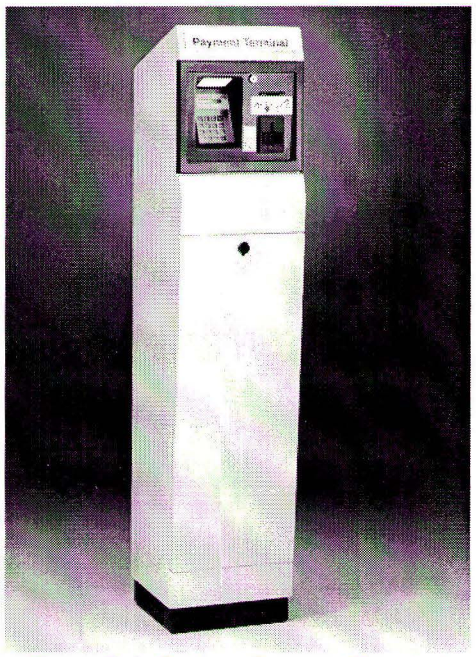


Figure C-1: Outdoor Payment Terminal

The family of Outdoor Payment Terminal Products (OPT) are products that provide the facility to purchase fuel without the necessary intervention of a station operator or cashier.
The fuel purchaser can initiate and complete a fuel purchase transaction with the use of an OPT. Therefore the OPT is equipped with several peripherals (functions):

• card reader	to pay with credit, debit, or private cards;
• cash acceptor	to pay for the fuel with cash
• user keyboard	to interact with the system
• user display	to interact with the system
• receipt printer	to provide a receipt of the transaction

To control the peripherals the OPT contains the following components as well:
- CPU board and I/O facilities
- power supply
- climate control
- housing
- interface to POS/Site controller

Apart from the functions described above that are in the OPT, we need the following functions provided by other parts of the fuelling system:

• card authorisation (CAM)	for off-line/on-line card authorisation
• EFT	to communicate transactions with a HOST
• transaction storage	for transaction record keeping (mainly for off-line
• transaction logging	either a journal printer or electronic journal
• forecourt controller (FCC)	dispenser control device
• dispensers	For fuel delivery

In figure C-2 the configuration of a typical fuelling system with two dispensers, a POS/Site controller, and an OPT is depicted.

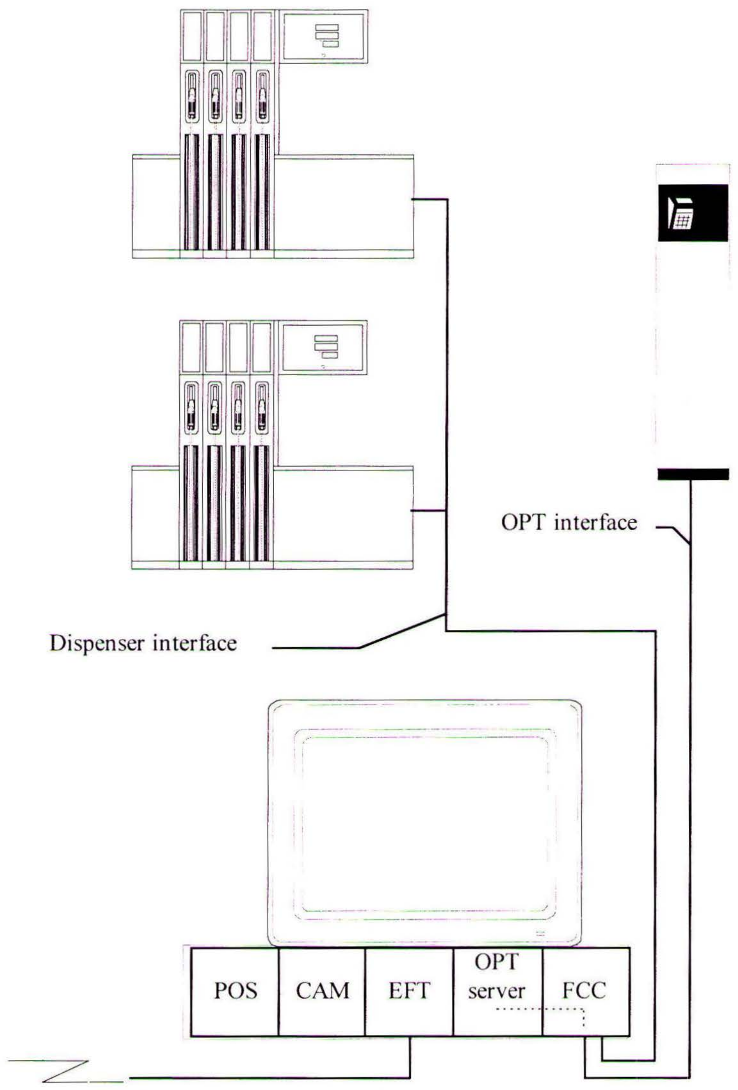


Figure C-2: Typical OPT configuration

Appendix D: Characterisation of the Omega project

This section contains the background information about the Omega project that is relevant for the Omega case study.

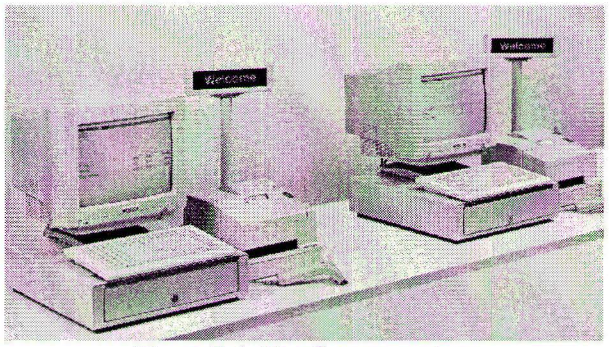


Figure D-1: The Omega system

The Omega Service Station Management System has as its main function the processing of petroleum retail transactions. This process consists of: releasing a filling point for a petrol filling, reading of the acquired amount of petrol, forwarding the price of the acquired petrol to the Point-of-Sale (POS), and finishing and storing the transaction after payment. Next to this, the system comprises a number of management functions. According to the requirements of the customer the system can be extended with more functions like, for example, card-handling, a payment-terminal or a telecommunication-link to the central computers of banks and oil companies.
The Omega system comprises a range, consisting of a low-end, mid-range and high-end system. These systems are embedded products. In the low-end system software is embedded in the hardware (memory chips), whereas in the mid-range and high-end system software is mainly stored at a hard disk. During start-up this software is partly read into the memory.

Appendix E: RPS Internal Documents

The following list of RPS internal documents are not publicly available for reasons of confidentiality.

OPT
- Quality Investigation
- GQM plan
- Feedback Session Presentation

Omega
- Quality Investigation
- Process Improvement Plan
- GQM plan

World Wide Calculator
- Quality Investigation

For more information about those documents, please contact:

Ing. E. Rodenbach
Quality Manager Special Projects - Systems Engineering Division
Retail Petroleum Systems
Industrieweg 5
5531 AD Bladel, The Netherlands
tel. +31 (0)497-389555
fax. +31 (0)497-381950