Umeå University
Department of Computing Science
25th October 2005

Master Thesis Project 20 credits
Supervisor: Annabella Loconsole
Spring 2005

A selection of requirements questions and metrics, to very small organizations

Author:
Jan Johansson

Abstract

To survive, every company needs to control their effectiveness and they should make continues improvements of their company. What is it that the customer wants, how should it work, what will it cost, how can they make the development cheaper, will they deliver what the customer ordered and so on and so forth. But to know *what* and *how* to improve, they need to measure their activities. They also need to compare the measures to previous measures and to predefined requirements. The process of analyzing, validating, defining and verifying all this are called *requirements management*(RM).

There exists a lot of different management principles that are more or less effective and more or less circumstantial. Some theories that has been proved to work are *Capability Maturity Model*(CMM) and *Goal Question Metric Paradigm*(GQM). These two theories attacks the management problems from different points of view, but they can be combined.

The problem is that those theories are developed for larger companies with several hundreds of employees, while not so much research is done for smaller companies with less than 50 employees.

The goal of this thesis is to design a set of requirements questions and metrics suitable for small organizations.

My contribution is a set of questions, and their related metrics, suitable for small organizations. The questions are divided into three groups that depend on the size of the organizations. In this way it is possible for a one man company to start measuring already from the beginning and to continue the improvement work while the company grows.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The requirements management process area is an important part of quality[1] [2] improvement. It is a way to control the continuous definition of software requirements as they change through the software life-cycle. Software measurement can help us in controlling the requirements process, in quantifying the amount of change to requirements and in predicting the cost of a change. In general software measurement can help to guide the activity of the management. Numerous software measures are found in the literature, but they need all to be tailored to the particular organization.

Due to the increasing complexity of the developed software the possibility for faults and defects also increases [SOL99]. Increased number of errors combined with optimistic time (and cost) estimations has given raise to the phrase "The software crisis" [SEI95] [ART85]. With increased number of software companies struggling to survive in the competition it is difficult for the customer to decide which company to hire. How will the customer be able to distinguish between a good company, who will deliver the right product at the right time to the right price, and a bad company, who will not be able to meet the same undertaking? A quality programme could be the difference to look for and especially one that works [HAK00].

But marketing is not the true reason why any software company should have some kind of quality programme. It is because this is the only way to achieve a good and solid software process that evolves with time to become better

[1] The word *quality* is derived from the Latin word "qualitas" that stands for nature, character, state, condition and quality, property, characteristic. [WWW1]

[2] Another interesting discussion about *quality* can be found at [WWW2]

and better.

> *If you don't measure, then you're left with only one reason to believe that you are still in control: hysterical optimism.*
> *– Tom DeMarco*

Therefore all businesses wants to improve their products, but how to do this if you do not know where you are at the moment? With improved software development process the company will be able to better estimate the time, and thus the cost, to develop a product. With better estimation it is possible streamlining the company to make better profit [KOT01] [AXC02].

But to make estimations you need to know *what* to measure and *how* to do it, and you need to do it correct over and over again so it is possible to compare the results, otherwise it will not be worth anything.

A lot of research has been done in this area and there exist many different theories on how to improve software quality.

One major drawback has been that Capability Maturity Model (CMM) were developed for use in large organizations and therefore require a lot of personnel with different roles. This mean that it will be very difficult for a small company to implement for instance the CMM and the Goal Question Metric (GQM). It is too extensive, there are many more roles than employees. It needs to be tailored to fit for the smaller organization. Previous work to tailor CMM for small organization is presented in [ORC00a] [ORC00b] [ORC00c] [LAR00].

1.1 Purpose

The purpose of this thesis is to tailor a set of requirements management questions and metrics (as described in [LOC01]) to different kind of organizations (small, medium, large enterprises) and to suggest improvement actions.

I will focus on organizations with a maximum of 50 employees because those are the ones that most likely need measures tailored with regard to the number of employees. This report is based on a previous effort to merge the two theories, CMM and GQM, into one extensive theory[LOC01]. Organizations with more than 50 employees are deemed to be able to use the original set

of measures proposed by Loconsole ([LOC01]).

1.2 Context

With increased number of software developers the customer can choose the developer who offers the lowest price. The question is how to decrease the development cost so your own company can offer the lower price? A well implemented quality system will help to lower the costs for error correction, improve time and cost estimation, reduce maintenance costs [TIT97].

There are many different reasons why you should make measurement of the development of software. If you measure you are able to characterize, evaluate, predict, and improve [PAR96]. Software measurement can help companies to implement high quality products.

With *characterization* you get understanding of products, processes, resources and environments and from this you will be able to establish a baseline that can be used for future assessment comparisons.

Evaluation lets you determine the status of the project in relation to the project plans. In this way you will know if you are drifting off course and will be able to get back on track. You are also able to assess quality, process, and product improvements.

To plan you have to *predict*. With the help of historical data you will be able to estimate risks, and design and cost tradeoffs. To do this you would have to understand the relationship between processes and products so that you can use the understanding to predict other attributes.

With the help of measures the organization can *improve* with respect to both process and product quality and performance. With good measurements the organization will be able to track the improvement efforts to validate success and failure. The measurement have to be quantitative so it is possible to compare with earlier and later results.

To get valid measures you would have to clearly identify *what* and *how* to measure and what rules to use for mapping the measure to a value that can be used as a metric [TIT97] [PAR96]. But all measures will not yield a specific metric, it will not be consistent. While the metric is a discrete point

(value) the measures is an evaluation of metrics.

Metrics can be both objective and subjective. The objective and subjective measurement address different questions and should be viewed as complements. Well defined rules on how the subjective measure will be gathered and evaluated, increase the consistency of these measurements and minimize errors and noise.

1.3 Related work

Few months ago when I first started this work I was struck by the immature management processes used in this area. In late 1960 and early 1970 Balanced Score Card (BSC) were developed in the area of economics, and the BSC are still in use. And they are proved useful as a management tool. Now, some 35-40 years later, the area of computer science have discovered the similarity between BSC and GQM[GOE03][AVE04].

Unfortunately, there has not been much research done based on small organizations in the real world with real applications. There are evaluations done of the application of CMM at large companies but the area of small companies are a quite black spot.

Laryd ([LAR00]) and Orci ([ORC00a], [ORC00b], [ORC00c]) have proposed tailored versions of CMM for small organizations.

Davis et al. ([DAV93]) address how to create software requirements specifications of high quality, namely without errors (knowledge or specification errors). This was done via a survey conducted by the authors. They suggest some metrics for requirement engineering but some of the metrics are very subjective and difficult to quantify.

Costello and Liu ([COS95]) focus on the requirements document and not at a specific phase. They claim that it will be possible to identify, isolate and resolve requirements problems before they can affect lower levels of the life-cycle phases.

Jiang et al. ([JIA03]) present their own concept for requirements engineering with a focus on the assessment of requirements engineering process. The authors propose a set of *concerns* related to the assessment, and those concerns

are based on theoretical research as well as a survey in the industry.

Li and Smidts ([LIM03]) aim to identify software engineering measures which can be considered good indicators of software reliability. They did this via expert opinion elicatitation and used no experimental data. They used 10 internationally experts to do the elicitation, but there is no more information about these experts, if they where scientists or from the industry.

Loconsole ([LOC01]) is so far the only one to have combined CMM and GQM to create a set or requirements management measures suitable for large organizations and her paper is the foundation to this thesis.

1.4 Method

I conducted a thorough literature study to be able to grasp the width of this area. Learning the essentials about CMM and GQM, searched and picked out the most relevant papers on the subject.

I have mainly read a lot of articles and reports but also some books and web-pages. After comparisons between all the used resources I have drawn my conclusions that will be presented further in this thesis.

1.5 Outline

In chapter 2 and 3 CMM and GQM are introduced to the reader and after that I summarize the application of the GQM to the Requirement Management KPA of the CMM performed by [LOC01].

In chapter 4 a classification of the different organizations sizes and the original roles for those organizations are presented.

The tailoring of the questions and metrics for the first goal of the Requirement Management KPA and the second goal are found in section 4.3 and 4.4respectively. The results are presented in section 4.5 and the conclusions begin at chapter 5.

The original questions and measures as proposed by Loconsole are presented in the appendix A. Abbreviations are also found in the appendix B.

Chapter 2

Capability Maturity Model (CMM)

The Capability Maturity Model is designed to provide a way for organizations to get control over their software development process and to increase their professionalism and maturity [SEI95]. It provides a framework with descriptions to evolutionary improve the maturity of an organization.

2.1 Software process improvement methods

There exists a lot of different models for managing quality, SPICE, Bootstrap, iso9000, tickIT, PSP etcetera. A short introduction to these theories are presented below.

The Software Process Improvement and Capability dEtermination (SPICE), is a project that try to harmonize the previous work of software process assessment. It tries to measure process capability by measuring the implementation and institutionalization of specific process but not in a direct way as the CMM's maturity levels.

Bootstrap was a project done as part of the European Strategic Program for Research in Information Technology. Its goal was to develop a method for software-process assessment, quantitative measurement, and improvement.

In executing that goal, Bootstrap enhanced and refined the Software Engineering Institutes's process-assessment method and adapted it to the needs of the European software industry-including non-defense sectors like banking, insurance, and administration. This adaptation provided a method that could be applied to a variety of software-producing units, small to medium software companies or departments that produce software within a large company. Although the Bootstrap project completed in 1993, its attribute-based method for assessing process maturity continues to evolve. Bootstrap is developed with regard to both CMM and ISO9000 in mind.

International Organisation for Standardization (ISO) develop a lot of different standardizations and among these there is the 9000-series. Those cope with overall quality management(9001) and in specific areas as in software(9000-3).

The Personal Software Process (PSP) is a system designed for the developer to use on his own to gather and analyze data about his own work.

These models are mostly derived as results of the idea called Total Quality Management, TQM.

> *Management practices designed to improve the performance of organizational processes in business and industry.*
> *Based on concepts developed by statistician and management theorist W. Edwards Deming, TQM includes techniques for achieving efficiency, solving problems, imposing standardization and statistical control, and regulating design, housekeeping, and other aspects of business or production processes.*
> *– Britannica Concise Encyclopedia. 2004.*

2.2 CMM

CMM is most common in USA but the use of CMM are spreading around the world. CMM was created to assess companies, now it is also used to improve the process and organization of companies. At SEI's homepage and at the CMM-site there are a lot of statistics presented that shows the success of implementing versions of CMM in different organizations. Companies report that they reduce cost, meet schedule, increase productivity, higher quality, higher customer satisfaction and so on. These reports and studies

are available at the SEI homepage [WWW3].

A lot of extensions have been developed to the original CMM; Capability Maturity Model for Software(SW-CMM), Systems engineering Capability Model(SECM) and the Integrated Product Development Capability Maturity Model(IPD-CMM). All those different extension have been integrated into one more extensive model called CMMI("I" is short for integration) that can be used throughout the entire organization.

2.3 Structure of CMM

CMM is composed of five maturity levels[1]. All levels except level 1 is composed of several *key process areas*, (KPA). Each KPA is in turn organized into five sections called common features. To accomplish the goals for the KPA the *key practices* specified for each common feature must be met. This structure is shown in figure 2.1.

The components of the CMM include:

Maturity level
The top-level structure of CMM is called maturity level. Those five levels are well-defined evolutionary steps towards achieving a mature software process.

Process capability
Process capability, for software, describes the range of expected results that can be achieved by following a software process. In this way it is possible to predict the most likely outcome to expect from the next software project undertaken of the organization.

Key process area
Each one of the maturity levels is composed of key process areas (KPA). Each KPA identifies a set of related activities considered important at that maturity level. The KPA's exists within just one single maturity level. For example, one of the KPA's for level 2 is Requirements management.

Goals
The goals summarize the key practices of a KPA and can thus be used to

[1] It is a ordinal scale with integer-values from 1 to 5 but there is no concept or distance associated with the scale.

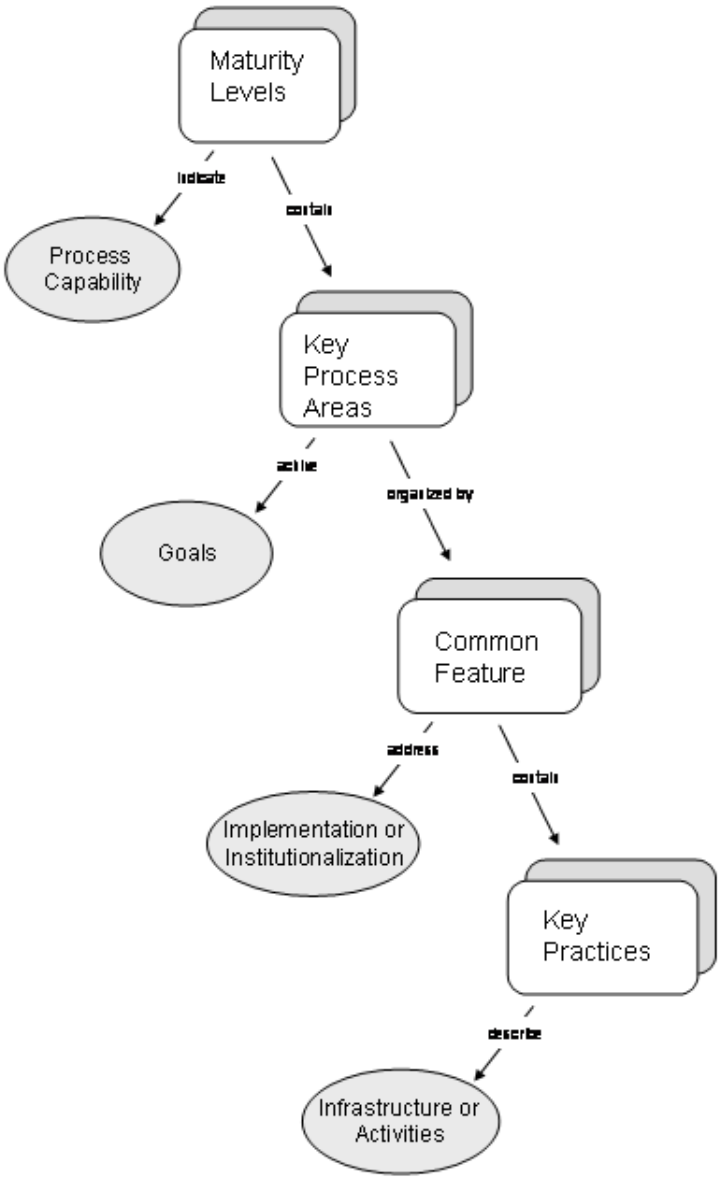

Figure 2.1: The structure of the Capability Maturity Level [SEI95]

measure if a project, or organization, has implemented the KPA. The goals signify the scope, boundaries and the intent of each KPA.

Common features

The key practices are divided among five Common Features sections: Commitment to Perform, Ability to Perform, Activities Performed, Measurement and Analysis, and Verifying Implementation. The common features are attributes that indicates whether if the implementation and instutionalisation of a KPA is effective, repeatable and lasting.

The Activities Performed common feature describes implementation activities. The other four common features describe the instutionalisation factors, which make a process a part of the organizations culture.

Key practices

Each KPA is described in terms of key practices that when implemented help to satisfy the goals of that KPA.

2.4 CMM Maturity Levels

By improving software processes for developing and maintaining software organizations will advance through the levels of maturity[SEI95]. The different levels are shown in figure 2.2.

Level 1 – Initial Level

Level 1 is often referred to as chaos instead of Initial Level. The organization has no general idea of how it is developing and maintaining software. The software process, if it exists, is inferior to what it should be to be able to deliver the software. Success depends often on one or two individuals and their heroic efforts in this kind of ad hoc process. Schedules, budgets, functionality and product quality are generally unpredictable.

Level 2 – Repeatable Level

Policies for managing software projects and procedures to implement those policies are established at the Repeatable Level. The organization plans new project based on knowledge from prior successful projects.

There are basic software management controls at Level 2. Managers track software costs, schedules and functionality; problems in meeting commit-

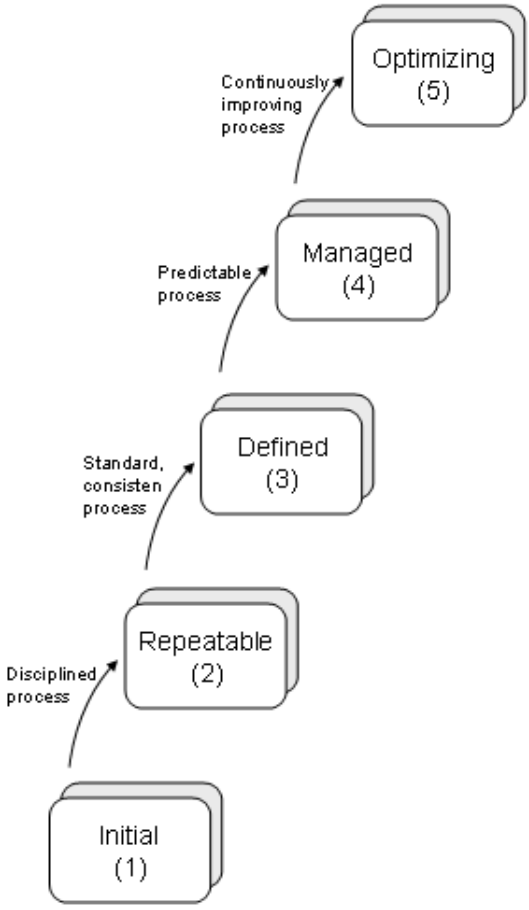

Figure 2.2: The five levels of the Capability Maturity Model [SEI95]

ments are identified when they arise. Software requirements are baselined and products developed are controlled. Organizations at Level 2 can be described as disciplined because planning and tracking of the software project is stable and earlier success can be repeated.

Level 3 – Defined Level

At the Defined Level, the standard process for developing and maintaining software across the organization is documented including both software engineering and management processes, and those are integrated with each other. This is referred to as the organizations standard software process.

Projects within a Level 3 organization tailor the standard software process to develop their own defined software process, that will fit their specific project. A defined software process contains a coherent, integrated set of well-defined software engineering and management processes. A well-defined process can be characterized as including readiness criteria, inputs, standards and procedures for performing the work, verification mechanisms. outputs and completion criteria. This way management has good insight into the technical progress on all projects.

Level 4 – Managed Level

At Managed level quantitative quality goals are in use for both software products and processes within the organization. All across the organization productivity and quality is measured. A software process database is used to collect and analyze the data. The measurement at level 4 is well-defined and consistent to help in the quantitative assessment of the projects software processes and products.

The process capability of Level 4 is predictable due to the measured process that operates within measurable limits. At this Level it is possible to predict trends in process and quality within the boundaries of these limits. When these limits are exceeded, action is taken to correct the situation.

Level 5 – Optimizing Level

At Optimizing Level all the organization is focused on continuous process improvement. The organization is no longer reactive, it has means and ability to be proactive. New technologies are proposed based on cost benefit analyzes, good software processes are identified and distributed throughout the organization.

A Level 5 organization is continuously improving thanks to the continued

effort to improve their process capability. They analyze defects to determine their cause and software processes are evaluated to prevent known types of defects from recurring. Improvement occurs both by incremental advancement in the existing process and by innovations using new technologies and methods.

2.5 Requirements Management Key Process Areas

Each one of the maturity levels, with the exception of Level 1, contains several key Process Areas as listed in figure 2.3.

> *Each key process area identifies a cluster of related activities that, when performed collectivitly, achieve a set of goals considered important for enhancing process capability. The key process areas have been defined to reside at a single maturity level. The key process areas are building blocks that indicate the areas an organization should focus on to improve its software process. key process areas identify the issues that must be addressed to achieve a maturity level.*
> *– [SEI95]*

Among the level 2 KPAs, the requirements management KPA will be deeply investigated in this thesis.

According to Paulk, the Requirements Management is about establishing a common understanding between the customer and the software project of the customers proposed requirements. "Customer" can be interpreted in any way between an external customer to another internal group. This understanding will be the basis for planning, estimating, performing, tracking and managing the software project throughout its life cycle.

It is not easy to satisfy the stakeholders (customer, developers etcetera) all at the same time can be difficult. But with the help of requirements management it may be possible. You need to analyze the requirements, validate that the requirements are what the customer(and the stakeholders) wants, define what the designers are to develop and at last verify that the developed product corresponds to the agreed requirements [ZAV97]. The

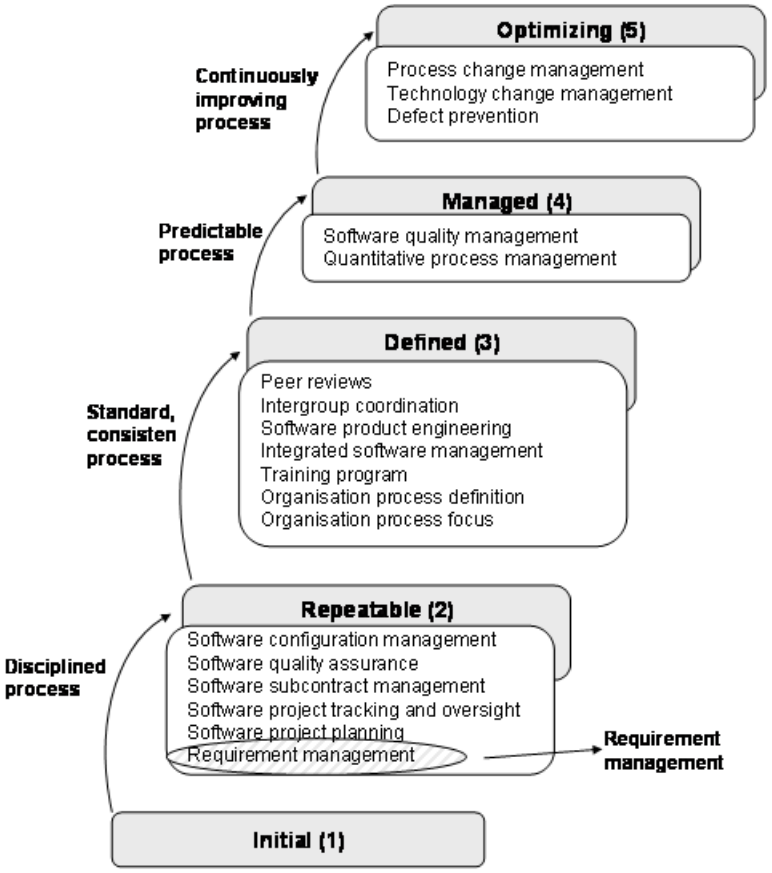

Figure 2.3: The Key Process Areas of the Capability Maturity Model [PAU93]

requirements management activities are also used to control the continuous definition of software requirements as they change through the software life-cycle.

If you measure you are able to characterize, evaluate, predict and improve [PAR96] your organization. With a well adapted requirement management you will be able to produce a "better´´ result. So requirement management are not just an important part of CMM, but an important part of every company that want to survive in the global competition. Requirements management it is a very important key process area of the CMM paradigm.

CMM defines two distinct goals for Requirements Management Key Process Area[PAU93]:

> *System requirements allocated to software are controlled to establish a baseline for software engineering and management use.*

and

> *Software plans, products and activities are kept consistent with the system requirements allocated to software.*

The first goal focuses on controlling the requirements on setting up a baseline . Without goals there is no way to know if the final product meet the requirements because lack of control.

The main focus of the second goal is consistency between the created products and their requirements.

Chapter 3

Goal Question Metric Paradigm

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

The Goal Question Metric Paradigm was developed in response to the need for a goal-oriented approach for measurement in software engineering and all data collected should be based on a rationale which is explicitly documented. The approach is top-down, defining the goal behind measuring software processes and products and using these goals to decide what to measure [DIF96] [BAS94].

To be effective a measurement must be [BAS94]:

1. Focused on specific goals;

2. Applied to all life-cycle products, processes and resources;

3. Interpreted based on characterization and understanding of the organizational context, environment and goals.

This supports the top-down model because stated measurement must be focused, based on goals and models. For an organization to have meaningful measures the organization must first establish its goals. This approach helps in the identification of useful and relevant metrics as well as in the analysis and interpretation of collected data. It enables an assessment of the validity of the conclusions drawn and avoids resistance against measurement programs.

The GQM-method contains four phases as shown in figure 3.1 [SOL99]:

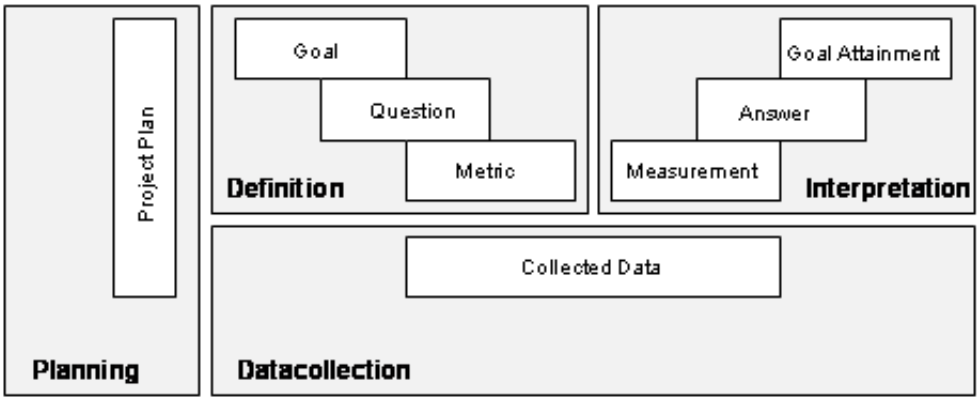

Figure 3.1: The four phases of the GQM-method [SOL99]

- The Planning phase, during which a project for measurement application is selected, defined, characterized and planned, resulting in a project plan.

- The Definition phase, during which the measurement programme is defined (goal, questions, metrics and hypotheses are defined) and documented.

- The Data Collection phase, during which actual data collection takes place, resulting in collected data.

- The Interpretation phase, during which collected data is processed with respect to the defined metrics into *measurement* results, that provide *answers* to the defined questions, after which *goal attainment* can be evaluated.

To make the GQM measurement a success the planning phase include training, management involvement and project planning to acquire all basic requirements. The GQM deliverables are constructed during the definition phase and are primarily based on interviews or other techniques. A goal, all questions, related metrics and expectations of the measurement are identified during the definition phase. The actual measurement can start when all definition activities are completed. During data collection phase all data are collected and stored in a database. Now the work with interpreting all data can begin with the interpretation phase. The measurements are used to answer the questions and these answers are used to see whether the stated goals have been attained .

3.1 GQM plan

The hierarchical structure called *GQM plan* starts with a goal (specifying purpose of measurement, object to be measured, issue to be measured and viewpoint from which the measure is taken). The goal is refined into questions that usually break down the issue into its major components. Each question is in turn refined into metrics. Some metrics can be used to give informations in order to answer more than one question. The GQM plan (or model) consists of three different levels as shown in figure 3.2:

Conceptual (Goal) A goal is defined for an *object*, for a variety of *purposes*, with respect to various *models of quality*, from various *points of view*, relative to a particular *environment*.

Operational (Question) A set of questions is used to define in a quantitative way the goal and to characterize the way the specific goal is going to be interpreted. The object of measurement is tried to characterize by the questions with respect to a selected quality issue and to determine its quality from the selected point of view.

Quantitative (Metric) A set of data is associated with every question in order to answer it in a quantitative way.

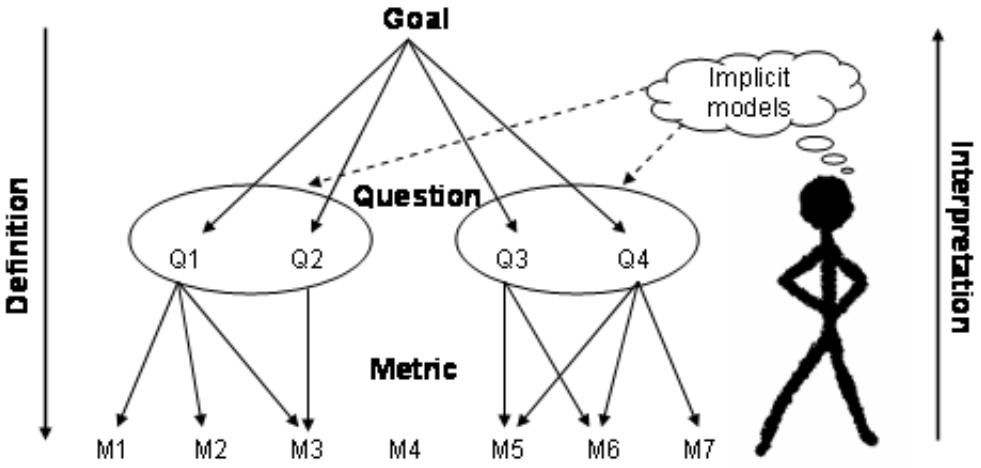

Figure 3.2: The Goal Question Metric Paradigm [SOL99]

3.2 GQM Goal Definition Template

The process of setting goals and refining them into quantifiable questions is complex and requires experience. For that reason templates exist to make the developing of GQM plans easier. A goal generation template contains five major aspects as follows:

Analyze	<**object of study**>
in order to	<**purpose**>
with respect to	<**quality focus**>
from perspective of	<**point of view**>
in the environment of	<**context**>

In some papers the aspects "object" and "purpose" are grouped together within "purpose of study", the aspects "quality focus" and "point of view" are grouped together within "perspective of study" and the final aspect "context" is labeled "environment of study".

Objects of Study

What is the primary target of the study.

- Processes

- Artifacts / Work products
- People
- Organization
- ...

Purpose of Measurement

The purpose of measurement: what is the learning objective. Often stated in one of the following generic study goals:

- Characterize: to record or measure features or attributes in the entity being studied.
  - Describe: Collect relevant information to document information about an object.
  - Monitor: Use measurement data to track progress or status.
  - Understand: Recognize patterns or propose hypotheses, theories or models.
- Evaluate:
  - Assess: Evaluate against a well-defined standard or baseline.
  - Compare: Evaluate two or more alternatives against each other.
  - Validate: Evaluate feasibility.
  - Appraise: Evaluate effectiveness or usefulness.

Attributes of Quality Focus

Specific attributed of the entity being studied.

- experience of programmers
- reliability of programs
- duration and cost of design phase
- management structure of an organization

Point of view

Who is interested in utilizing the information and clarifies rationale and usage of data.

- current project manager
- recruiting managers
- process development organization or process owner
- researcher

Context of study

What is the context and scope of the study, what is included and what is not.

- real-time software development
- multi-site, advanced development
- group-ware applications

Chapter 4

Tailoring

4.1 GQM application to the Requirements Management of CMM

Loconsole [LOC01] has shown that it is possible to combine CMM with GQM. The goals that CMM defined for each KPA as in figure 4.1 can be used for the first step in the GQM-method.

In this way the two goals in Requirements Management Key Process Area, see 2.5, can be used as goals in GQM and in turn refined into quantifiable questions, and from the questions refined again into metrics. 15 questions and 41 measures have been defined for the first goal, 7 questions and 12 measures for the second goal. Notice that some questions and metrics from the first goal is also used for the second goal. All original questions proposed by Loconsole for both goals in requirements management key process area can be found in appendix A. It is possible for organizations with more than 50 employees to use the proposed measures of Loconsole without doing any major tailoring of them with *regard to the number of available people.* Of course you still have to tailor the original set of measures in *regard to the specific project.*

In this chapter a tailored set of Loconsole's [LOC01] proposed metrics will be described for small organizations of different sizes; Small(S), Extra Small(XS) and Extra Extra Small(XXS) organizations.

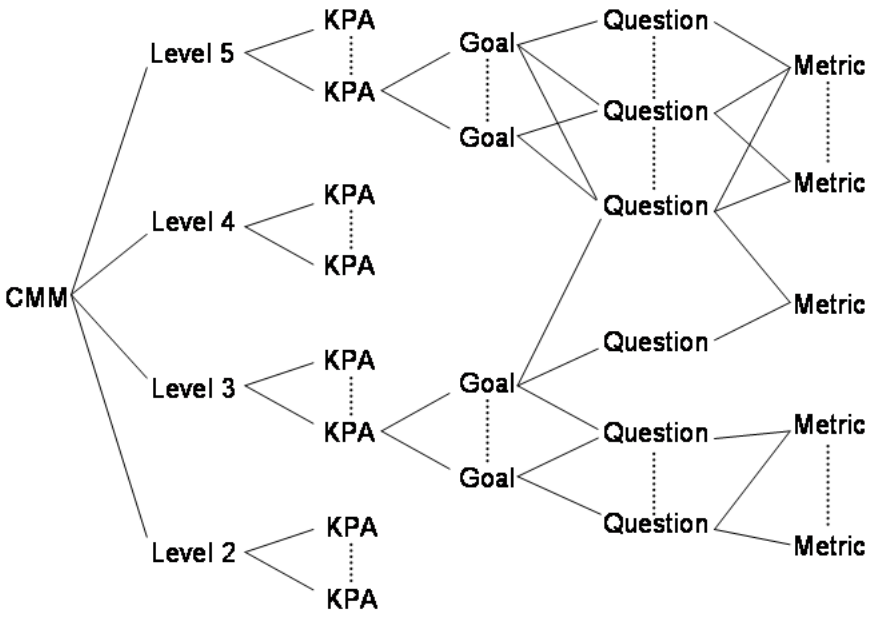

Figure 4.1: The relationship between CMM and GQM [LOC01]

4.2 S, XS, XXS companies

There exits a lot of different definitions of a small organization, in [PAU99] Paulk concluded that under a CMM workshop the participants could not agree on what "small" really meant. In this thesis the definitions that will be used for XXS, XS, and S organizations are those that Laryd ([LAR00]) proposed and they are presented in table 4.1

Table 4.1: Size of companies.

XXS	1—2 employees	one product
XS	3—15 employees	more than one product/version
S	16—50 employees	more than one product/version

However, the distinction between the different organizations are very buoyant and while an organization grows they can (and should!) continuously change the implementation of their quality management programme.

Already in the original definition of CMM it is stated that:

> *[...]a group could vary from a single individual assigned part-time, to several part-time individuals assigned from different departments, to several individuals dedicated full-time.*

In small organizations one individual will probably have several different roles, some roles can most likely be omitted, and some other roles shared. The important thing is to tailor the CMM to the specific project at hand.

Orci proposed in [ORC00a], [ORC00b] and in [ORC00c] the roles shown in table 4.2 for S, XS and XXS organizations.

Table 4.2: Proposed Roles in Small, eXtra Small and eXtra eXtra Small organizations.

Role	**S**	**XS**	**XXS**
Contract management group			
Customer SQA representative (CSQA)	CSQA	CSQA	CSQA/SQA
Documentation Support Group (DSG)	DSG		CSQA
Hardware engineering group			
Market and Sales (MS)	MS	MS	
Project Manager (PM)	PM	PM	SM/PM
Senior Manager (SM)	SM	SM	SM
Software Configuration Management (SCM)	SCM	SCM	SCM
Software Engineering group (SE)	SE	SE	SE
Software Estimating group			
SoftWare Manager (SWM)	SWM	SWM	SWM
Software Quality Assurance group (SQA)	SQA	SQA	
System engineering Group (SG)	SG	SG	SG
System Test Group (STG)	STG	STG	STG
Software Subcontract Manager (SSM)	SSM		
Software Configuration Control Board (SCCB)	SCCB		

These roles are not absolute and should only be seen as guidelines. Depending on the organizations goals and environment the roles may vary and likewise could the proposed questions and measures vary from project to project.

These category's, S, XS, XXS, all consist of a very wide range of different small organizations with very different conditions, regarding everything from different number of employees. Some may have subcontractors and some

don't, some may itself be a subcontractor; some may physically share one common office while others may be scattered around the world; some may develop an application ordered by a customer while some develop for the open market and so on. All these different prerequisites make it more or less impossible to propose measures that will fit everyone. You will have to tailor the proposed questions and measures to fit your specific project.

Each and every one of Loconsole's proposed questions will be discussed in the following paragraphs. The reasoning will be based on the previous work of Orci ([ORC00a], [ORC00b], [ORC00c]), Laryd and Orci ([LAR00]), Paulk ([PAU98], [PAU99]), and others.

The syntax for each question and metrics presented below will be the following:
Question
Measure
S|XS|XXS

4.3 First goal of the Requirements Management KPA

There are 41 measures connected to 15 questions for this first goal.

> *System requirements allocated to software are controlled to establish a baseline for software engineering and management use.*

4.3.1 What is the current status of each requirements?

Status of each requirement.
The different requirements can have various states during the software development process. The possible proposed states can be: new, analyzed, approved, documented, rejected, incorporated into the baseline, designed, implemented, tested etcetera. This is important to know for the management of every company so they can monitor the ongoing work throughout the whole project [JIA03] [COS95]. At least the PM should have total control of

the status of each requirement. The easy way is to use the documentation of the requirements and tick them as they are met.

S
In a company of this size it should be possible to use some kind of automated tool to help maintain control of all requirements status.

XS
These organizations might not need a tool support for checking the status of each requirement but it should be considered assuming the company might grow in the future.

XXS
Automated tool is overkill in a company this small but even in a one person company you need to know the progress of all requirements, especially if you develop for a customer.

4.3.2 What is the level of the stability of the requirements?

Number of initial requirement
Number of final requirements
Number of changes per requirement
Stability[1], and it's counterpart volatility[2], are important areas for a successful project. There are some reports claiming that as much as 11% of all software project failures are caused by volatility in the requirements[STA94]. Two major causes for project delay are requirements change and inadequate or complex requirements [ZOW05], the latter one also holds as one of the major causes for exceeding project cost. That time and cost are closely connected to each other are quite natural. Zowghi has also found that software from projects with a high level of requirements volatility more often will fail to pass customer acceptance test as well as having more usage complaints and failures/defects reported than software products from projects with a lower level of requirements volatility.

[1] the quality, state, or degree of being stable : as
the strength to stand or endure : FIRMNESS
the property of a body that causes it when disturbed from a condition of equilibrium or steady motion to develop forces or moments that restore the original condition
–Merriam-Webster Online Dictionary 2005-03-17

[2] Requirements Volatility: the ratio of requirements change (addition, deletion, and modification) to the total number of requirements for a given period of time.[ZOW05]

The measures defined for this question are all included in set of measures defined for the next question, 4.3.3. Therefore suggestions of measures for stability and volatility can be inferred from the discussion of the next question. This is done to ease the workload for these small companies.

4.3.3 Why are the requirements changed?

Number of initial requirements
Number of final requirements
Number of changes per requirement
Number of test cases per requirement
Type of change to requirement
Reason of change to requirements
Major source of request for a change to requirements
Phase where change was requested

Requirements change is unavoidable and changes must not mean poor requirements engineering [ZOW05]. The changes can be a result of a combination of many different factors. A PM should at least know how many requirements there were from the start of the project and how many they are when finished, including those that have been changed. It is also important to know, especially for future projects, the *reason* for change [JIA03] [LIM03]. If you know that you have many changes for the same reason, then you can put more focus in planning this in future projects reducing costs and time of change.

S

Small companies should try to cover all proposed measures because they are all deemed important at least for future projects.

XS

Some of the measures can probably be tailored depending on the specific application. For example the type of change might not be important for some organizations, but this is depending on the internal culture within the organization.

XXS

Number of initial requirements, Number of final requirements, and Number of changes per requirement may be sufficient but the reason of change is considered important and should be documented as well. The first three are

very easy to monitor, the reason demands a little more effort but I strongly advise to monitor this as well.

4.3.4 What is the cost of changing the requirements?

Cost of change to requirements
Size of a change to requirements
It is well known that a late change will generate a higher change cost than an early change. If the measure *Phase where change was requested* from the previous question was monitored it is possible to investigate all late changes and the reason why.

There is some discussion about the cost of changes to requirements, both positive and negative comments. The major objection is that it is a too complex ([JIA03]), and therefore too costly, question for companies of this size, and another objection is that the change is probably necessary to be able to deliver and keep the contract to the customer. On the contrary some deem it is so important that it is worth the time and effort.

I omit the suggestion of measures for small companies because the complexity of the question makes it probably too heavy for a small company. But if you consider them important for your project you should of course try to monitor them.

4.3.5 Is the number of changes to requirements manageable?

Total Number of requirements
Number of changes to requirements proposed
Number of changes to requirements open
Number of changes to requirements approved
Number of changes to requirements incorporated into baseline
Number of changes to requirements rejected
The computer software configuration item(s)(CSCI) affected by a change to requirements
Major source of request for a change to requirements
Requirement type for each change to requirements
Number of requirements affected by a change

Means that the amount of changes is increasing a lot in time and the company is not able to make the changes in time to deliver the software as planned. There are a lot of measurements related to this question, but the measures can be very easy to monitor if they are managed correctly from the beginning.

But the issue is if the question and its measures are important. Why are the changes needed? If it is because the customer wants the changes and needs the changes done, then you probably have to make the changes no matter what. Sure, it might be difficult and require a lot of work but you will, or should, get paid for it (see also the discussion from the previous question). Perhaps it is due to your own mistakes like created the wrong requirements because you did not understand what the customer wanted, then you still need to do the changes to be able to make the delivery.

The interesting thought is what happens if the answer you get after evaluating the answer from this question is *No*. If the answer is no, then there is a problem in the management of the requirements and you need to revise your software process and plans. The delivery to the customer is probably delayed.

S

If it is considered important for the specific project, all or some of the measures are suggested to get a better picture of the whole project and to help to analyze the process and the cost of change.

XS and XXS

Probably not applicable due to the amount of work and that the changes are probably necessary for the deliverance.

4.3.6 Does the number of changes to requirements decrease with time?

Number of changes to requirements per unit of time

There is not so much research done in this area and some of the papers show that this type of measure is controversial, some ([JIA03]) think this question and its measure do not contribute to the management of the project and hence do not use it while other ([JIA03]) consider it useful.

This measure is very dependent on the software development method (water-

fall, spiral etcetera) and what kind of product it is. For example if is a software that are developed in multiple stages/versions the number of changes to requirements might not be relevant.

If you consider the answer to this question important for your specific project you should of course monitor and use this.

4.3.7 How are affected groups and individuals informed about the changes?

Notification of Changes(NOC) shall be documented and distributed as a key communication document
Number of affected groups and individuals informed about NOC
Depending on the culture within the organization and how the work is done, it may not be necessary with a new document for every new change, use email, intranet or billboard (both real and virtual) instead.

However, it is important that all employees know about new changes and know *how* information about new changes will be distributed so they know where to look for updates. But it needs not be explicitly documented how it is done in companies of this size as long as people know it implicitly.

S
A company of this size should probably have documentation of the changes. Otherwise there is the risk that new employees will not get this information, it is easy to forget to distribute this information together with everything else.

XS
Depending on the organizations environment culture it is probably a good idea to keep track of the changes, but it does not have to be a formal document.

XXS
In a company as small as this it is highly unlikely that it is necessary to document changes. If there are two people they will probably just talk to each other about the current changes.

4.3.8 How many other requirements are affected by a requirement change?

Number of requirements affected by a change
This measure appear already in question 4.3.5. If the change will affect many other requirements, it will result in a delay in deliverance. If it is, then it must be negotiated with the customer so they understand that the delay is a result of their requested change of requirement.

S
Organizations at the upper level of **S** should consider if this question is applicable at the specific project.

XS and XXS
Probably not applicable due to the amount of work required.

4.3.9 In what way are the other requirements affected by a requirement change?

Type of change to requirements
Reason of change to requirements
Phase where change was requested
These measures have already been mentioned for questions 4.3.3. If no organized record of dependencies between requirements exist the workload to answer this question is not economically justifiable.

S
Organizations at the upper level of **S** should apply these 3 measures.

XS and XXS
Probably not applicable due to the amount of work required.

4.3.10 Is the size of the requirements manageable?

Size of requirements
What will the answer to this question yield? Will it contribute to anything in this specific project? Compare with the discussion for question *Is the*

number of changes to requirements manageable?(4.3.5).

This question is connected to the stability and volatility of requirements, larger requirements will increase the risk for volatility.

Outside negotiation with customer about the lead time for the application. The importance of this measure is questionable for organizations of this size. In general, small companies are suggested to not spend so much time in monitoring the size of requirements.

S
Some organization might consider this measure useful in an overall review for the project and might therefore measure this.

XS and XXS
Probably not applicable in organizations of this size.

4.3.11 How many incomplete, inconsistent and missing allocated requirements are identified?

Number of incomplete requirements
Number of inconsistent requirements
Number of missing requirements
The number of incomplete requirements can be obtained by the answers of the previous questions. The number of inconsistent requirements are deemed important and should be monitored [WIL97] [COS95] [JIA03] [ARM00]. Inconsistent requirements can give rise to ambiguity. The results of the ambiguity can be a software product that do not function as expected. This might very well be the difference between a happy customer and an angry one.

S
Organization of this size should be able to monitor all measures.

XS
Should at least manage inconsistent requirements, but all the measures are important and the organization should try to cover all of these measures.

XXS
Even a organization this small should at least manage inconsistent require-

ments due to its importance for the final product.

4.3.12 Does the number of "To Be Done"(TBD) decrease with time?

Number of TBD's in requirements specifications
Number of TBD's per unit of time
The importance of this question depends on what the organization is developing and for whom. If the organization have a customer who have order a specific program, then the organization have to meet all requirements and TBD's on delivery, thus the number of TBD will decrease with time. But if the organization do not develop in this way the TBD's might be irrelevant and hence can be omitted, but this is up to each project to decide. Compare with the discussion from the question *Does the number of changes to requirements decrease with time?* (4.3.6).

S and XS
If the question is considered important for the specific project, it can be monitored.

XXS
Not realistic in a company this small to take the time to collect data on TBD's but these measures can probably be obtained from other measures and documents used in the project.

4.3.13 How are the requirements defined and documented?

Kind of documentation
Depending on the organizations culture, measures can be collected [JIA03]. But it is not necessary that all requirements have their own documents and processes.

S
If it is considered important for the specific organization and their culture.

XS
If it is considered important for the specific organization and their culture, but it might not be worth the effort for a company this small. There is a

clear risk of documentation overload.

XXS
In a small company as this it is not likely that this question is necessary, they will probably just talk to each other.

4.3.14 Are the requirements scheduled for implementation into a particular release actually addressed as planned?

Number of requirements scheduled for each software build or release
With this question, a company wants to know if the number of requirements planned in the beginning is the same as the number of requirements delivered to the customer (for a particular release of the software). It is possible to judge each build or release as a separate project and hence monitor it through the previous stated measures like number of initial and final requirements for each build or release.

4.3.15 How many requirements are included in the baseline?

Number of baselined requirements
Phase when requirements are baselined
All requirement's should of course exist in some kind of documentation, which can be a formal baseline [3] or something else does not matter.

The baseline document is a requirements specification document which is considered definitive. Before the requirements are baselined, there can be many changes without doing any formal "request of change". When the requirements are baselined, to change the baseline document you need to apply for a change request. The baseline document is the one that should be distributed to the designers.

If the requirements are to be implemented into a particular release of the software the requirements must of course be in the requirement document or

[3] a line serving as a basis; especially : one of known measure or position used (as in surveying or navigation) to calculate or locate something
a set of critical observations or data used for comparison or a control
a starting point <the baseline of this discussion>
–Merriam-Webster Online Dictionary 2005-03-17

baseline document for that release.

It is possible that if the organization use two different documentations, one for requirements and one for the baseline, these can vary. And that would indicate that something is amiss. But that should already be obvious from the answer derived from question 4.3.1 and its measures. Furthermore, it is unlikely that XXS companies would use double documentations. And it is also unlikely that small companies like XXS and XS will have the strict theoretical application of a formal baseline, request of change etcetera.

In small companies were the salesperson are the same as the projectleader and are the same as the designer and are the same as the developer it is very unlikely that formal baselines will be used.

S and XS
Recommended measures for S and XS companies.

4.4 Second goal of the Requirements Management KPA

Software plans, products and activities are kept consistent with the system requirements allocated to software. There are 12 measures connected to the 7 questions for this second goal.

> *Software plans, products and activities are kept consistent with the system requirements allocated to software.*

4.4.1 Does the software product satisfy the requirements?

Functionality of the software
Number of initial requirements
Number of final requirements
Number of tests per requirement
Type of change to requirements
If the requirements are met the software should obviously satisfy the requirements, unless the requirements were wrong. The metric *Functionality of the software* is the one that is most important for this question, but at the same

time it should be obvious that the functionality of the software must satisfy the requirements.

In case the company is collecting data for the measures associated to the first goal the measures should already exist from question 4.3.3, with exception to the first one, and so for that reason it should be easy to do this evaluation if deemed important. But just that the product meets the requirements does not imply anything about it's quality or that the customer will be satisfied with it.

S and XS

The measures associated to this question are easy to monitor, especial if the company is collecting data for the measures associated to the first goal.

XXS

Maybe not so necessary to formal monitor the measures for this question in such a small company as this, but the metric "Functionality of the software" could be considered to monitor, if it is considered important for the specific project.

4.4.2 What is the impact of requirements changes on the software project?

Effort expended on Requirements Management activity
Time spent on upgrading
Number of documents affected by a change
Some questions for the first goal are connected to this one and therefore can be used to collect data to these measures if they are considered important for the specific project. All the questions connected to requirements changes can be useful to monitor for this question. If it is worth the time and effort to monitor this question and its measures is up to each company to evaluate.

S

All the measures are probably at hand and analyzing the measures can help the organization in its plans for future implementation work.

XS and XXS

Maybe companies of this size will not perform a formal measurement program, but they need to check the impact of a change in some way to see if

there might be delays in the delivery.

4.4.3 What is the status of the changes to software plans, work products, and activities?

Status of software plans, work products, and activities
The different documents as software plans, work products and activities, can also have various states. Some possible states are: identified, evaluated, assessed, documented, planned, communicated to effected groups and individuals, and tracked to completion. It should be quite easy to monitor these changes and if you think it is important you should of course monitor these. S and XS companies will most probably review their plans in weekly meetings. In these meetings they will (formally or informally) check the status of software plans, work products, and activities. In a XXS company being of 2 persons they will just talk to each other without a formal meeting.

4.4.4 Are the requirements scheduled for implementation in a particular release actually addressed as planned?

Number of requirements scheduled for each software build or release
Se the discussion from for question 4.3.14.

4.4.5 How are the requirements defined and documented?

Type of documentation
This can be managed depending on the organizations culture [JIA03]. But it is not necessary that all requirements have their own documents. However, it is necessary that all the requirements of the same kind are documented in the same way (with the same notation) so that the requirements are consistent in the requirements specification document.

S
If it is considered important for the organization and the specific project and especially if there has been problems of consistency and traceability. It should be questioned if this measure is necessary for the project at hand or

if it can be omitted. If just doing this measure on routine it will just add workload when it is not necessary.

XS
Probably not applicable, for the risk of documentation overload.

XXS
Not realistic in a company of this size were they easily can talk to each other.

4.4.6 Does the number of TBD's prevent satisfactory completion of the product?

Number of TBD's in requirements specifications
Intuitively the *number* of TBD's would increase with the size and complexity of the product and in turn also generate longer lead time. It is quite logically that the production time increase with the complexity.

If you deem this question interesting of course you can monitor this. Do not forget to set thresholds to compare with so that you can take actions based on the comparison.

4.4.7 Are all development work products consistent with the requirements?

Number of inconsistencies
This is important, if the requirements are inconsistent there is a imminent risk of volatility and the quality of the product are at risk [COS95] [WIL97].

The measure is recommended for S, XS, and XXS companies.

4.5 Summary

From the original set of 22 questions and 53 measures (38 different measures), I have selected a subset of 14 questions and 43 measures. For each question, the corresponding measures are shown but please note that the measures need to be tailored to the specific project at hand. The questions for XXS

companies are presented first and are followed by the questions for XS and S companies.

4.5.1 Questions for the first goal of the Requirements Management KPA

XXS

1—2 employees, one product

- What is the current status of each requirement?
  - Status of each requirement
- Why are the requirements changed?
  - Number of initial requirements
  - Number of final requirements
  - Number of changes per requirement
  - Number of test cases per requirement
  - Type of change to requirement
  - Reason of change to requirements
  - Major source of request for a change to requirements
  - Phase where change was requested
- How many incomplete, inconsistent and missing allocated requirements are identified?
  - Number of incomplete requirements
  - Number of inconsistent requirements
  - Number of missing requirements

XS

3—15 employees, more than one product/version

- Is the number of changes to requirements manageable?
  - Total Number of requirements

  - Number of changes to requirements proposed
  - Number of changes to requirements open
  - Number of changes to requirements approved
  - Number of changes to requirements incorporated into baseline
  - Number of changes to requirements rejected
  - The computer software configuration item(s)(CSCI) affected by a change to requirements
  - Major source of request for a change to requirements
  - Requirement type for each change to requirements
  - Number of requirements affected by a change
- How are affected groups and individuals informed about the changes?
  - Notification of Changes(NOC) shall be documented and distributed as a key communication document
  - Number of affected groups and individuals informed about NOC
- Does the number of "To Be Done"(TBD) decrease with time?
  - Number of TBD's in requirements specifications
  - Number of TBD's per unit of time
- How are the requirements defined and documented?
  - Kind of documentation
- How many requirements are included in the baseline?
  - Number of baselined requirements
  - Phase when requirements are baselined

S

16—50 employees, more than one product/version

- How many other requirements are affected by a requirement change?
  - Number of requirements affected by a change
- In what way are the other requirements affected by a requirement change?

  - Type of change to requirements
  - Reason of change to requirements
  - Phase where change was requested
- Is the size of the requirements manageable?
  - Size of requirements

4.5.2 Questions for the second goal of the Requirements Management KPA

XXS

1—2 employees, one product

- Are all development work products consistent with the requirements?
  - Number of inconsistencies
- Does the software product satisfy the requirements?
  - Functionality of the software
  - Number of initial requirements
  - Number of final requirements
  - Number of tests per requirement
  - Type of change to requirements

XS

3—15 employees, more than one product/version

- How are the requirements defined and documented?
  - Kind of documentation

S

16—50 employees, more than one product/version

- What is the impact of requirements changes on the software project?

- Kind of function point per requirements
- Effort expended on Requirements Management activity
- Time spent on upgrading
- Number of documents affected by a change

Chapter 5

Conclusions

While writing this thesis I have realized that it is more or less impossible to design a set of general questions and measures that will fit the wide variety of organizations that the software companies consists of. The set presented in this thesis is just an example.

5.1 Purpose

There is little research done in this area with regard on small organizations, most of the research conducted in large enterprises and some in medium sized (50 to 250 employees) enterprises. But almost none in small enterprises.

I will not propose any improvement actions for the software process. This, because it is impossible to say what to do without knowing the specific situation for each and every project, they are all different. All improvement actions are tightly correlated to each and every organization and their specific culture and structure.

I have only discussed organizations with a maximum of 50 employees because those are the ones that most likely need to tailor the measures with regard to the number of available employees.

To understand the results you need to know the different organization's sizes. The sizes used for the different organization were defined by Laryd in [LAR00] and they were presented in the table 4.1. As a organization

grows it will migrate to the next size.

5.1.1 Development

The theory used in this thesis is a combined version of Capability Maturity Model (CMM) and Goal Question Metrics model (GQM), presented by Loconsole ([LOC01]). She combines the Requirements Management Key Process Area in CMM level 2 with the theories of GQM and presents a set of questions and metrics. I have selected a subset of these questions and metrics to fit smaller organizations. The basic theories were developed for organizations with a large number of employees, there is at least 50 different roles in the original CMM.

Tailoring process

Due to the lack of major research in this area it has been real hard to tailor the questions and their measures. The tailoring has been quite subjective and should also be looked upon in this manner. All projects are unique and the internal culture within every company also contribute to complexity and the difficulty to produce general questions that will work for all companies everywhere. And truth to tell, that is impossible.

But there is some common characteristics and by focusing on them I have made the tailoring presented in this thesis.

5.2 Reflections

During my research I have realized that the golden solution does not exists. As Mark Paulk himself writes in his yearly CMM-newsletter in 1998 **Let Common Sense Prevail!** and this still holds true. All questions and measures are depending on the specific application and on the organization with their different individuals, different culture and different environment. No two projects are the other alike.

But as with common sense there exists question and measures that are more or less important as long as one remembers that it is still up to the specific

project to evaluate the measures.

There will not be any kind of mutual prioritizing of the questions due to the overhead of the specific application. The dependencies within the project group, such as cultural, educational and personal, are of greater importance in the process of choosing the right questions and measures than some kind of general tailoring. This is left for the developers and project management to discuss.

5.2.1 Limitation

Due to the shortage of research, comparable to each other, and investigations the results presented in this thesis should be used carefully. As stated above(in 5.1.1) most part of the tailoring is subjective and it can not be ruled out that a tailoring done by someone else would yield a somewhat different result. Faults, defects and digressions can easily occur with a basis like this.

One of the larger problems encountered is that it has been very difficult to compare results from other reports because it is not explicitly stated how the research was done and under which circumstances it was done. No one mention how the cultural bias within the company effect their results and how this can differ from another study.

Bibliography

[ANT01] Antón A., Carter R., Srikanth H., Sureka A., Williams L., Yang K., Yang L. *Tailored CMM for a Small e.Commerce Company, Level 2: Repeatable.* North Carolina State university, Department of Computer Science, Technical Report TR-2001-09, August 23, 2001.

[ARM00] Armour F., Miller G. *Advanced Use Case Modeling: Software Systems.* Addison-Wesley Professional, ISBN: 0201615924, 1st edition December 29, 2000.

[ART85] Arthur L.J. *Measuring Programmer Productivity and Software Quality.* John Wiley & Sons, 1985.

[AVE04] Aversano L., Bodhuin T., Canfora G., Tortorella M. *A Framework for Measuring Business Processes based on GQM.* IEEE, Proceedings of the 37th Hawaii International Conference on System Sciences, 2004.

[AXC02] Ax C., Johansson C., Kullvén H. *Den nya ekonomistyrningen.* Liber Ekonomi, 2002

[BAS88] Basili V., Rombach D. *The TAME Project: Towards Improvement-Orientated Software Environments.* IEEE Transactions Software Engineering, Vol. 14, No. 6, 1988.

[BAS94] Basili V., Caldiera G., Rombach D.. *The Goal Question Metric Approach.* Encyclopedia of Software Engineering, Volume 1, John Wiley & Sons, 1994.
http://wwwagse.informatik.uni-kl.de/publications/books/encyclo.gqm.pdf, 2004-11-01.
http://www.cs.umd.edu/users/mvz/handouts/gqm.pdf, 2004-10-06.

[BRA90] Brackett J.W. *Software Requirements, SEI Curriculum Module SEI-CM-19-1.2.* Carnegie Mellon University Software Engineering Institute, January 1990.

[COS95] Costello R., Liu D-B. *Metrics for Requirements Engineering.* Journal of Systems and Software, vol.29, no.1, April 1995.

[DAV93] Davis A., Overmyer S., Jordan K., Caruso J., Dandashi F., Dinh., Kincaid G., Ledeboer G., Reynolds P., Sitaram P., Ta A., Theofanos M. *Identifying and Measuring Quality in a Software Requirements Specification.* Proceedings of the First International Software Metrics Symposium, IEEE Computer Society Press, Los Alamitos, California 1993.

[DIF96] Differding C., Hoisl B. , Lott, C. *Technology Package for the Goal Question Metric Paradigm.* Internal Report Nr. 281/96, Department of Computer Science, University of Kaiserslautern, Germany, April 1996.

[GOE03] Goethert W., Fisher M. *Deriving Enterprise-Based Measures Using the Balanced Scorecard and Goal-Driven Measurement Techniques.* Technical Note CMU/SEI-2003-TN-024. Carnegie Mellon University Software Engineering Institute, October 2003.

[HAK00] Håkansson B. *Egendeklaration av programvaror.* http://www.teldok.org, TELDOK 39, 2000.

[JIA03] Jiang L., Eberlein A., Far BH. *Evaluating the Requirements Engineering Process Using Major Concerns.* Technical Report, RE Group, TR-2003-01, University of Calgary, August 2003.

[JOH97] Johnson D., Brodman J. *Tailoring the CMM for Small Businesses, Small Organizations, and Small Projects.* Software Process Newsletter, Committee on oftware Process, Technical Council on Software Engineering, IEEE Computer Society, No.8, Winter 1997.

[KOT01] Kotler P., Armstrong G., Saunders J. Wong V. *Principles of Marketing.* Prentice Hall Europe, 2001.

[KOT98] Kotonya, Sommerville G., Sommerville I. *Requirements Engineering: Processes and Techniques.* John Wiley & Sons, Ltd, 1998.

[LAR00] Laryd A., Orci T. *CMM för småföretag, Nivå 2.* Technical Report UMINF-00.10, Umeå University, Aug 2000.

[LIM03] Li M., Smidts C. *A Ranking of Software Engineering Measured Based on Expert Opinion.* IEEE Transactions on Software Engineering, vol.29, no.9, September 2003.

[LOC01] Loconsole A. *Measuring the requirements management key process area - Application of the Goal Question Metric to the Requirements Management Key Process Area of the Capability Maturity Model.* Proceedings of ESCOM - European Software Control and Metrics Conference, London, UK. April 2001.

[MIL88] Mills E. *Software Metrics, SEI Curriculum Module SEI-CM-12-1.1.* Carnegie Mellon University Software Engineering Institute, December 1998.

[NAT04] NATIONALENCYKLOPEDINS INTERNETTJÄNST. *Nationalencyklopedin.* NE.se.

[ORC00a] ORCI T. *Capability Maturity Model for Small Organizations, Level 2.* Technical Report UMINF 00.14, Department of Computing Science, Umeå University, September 2000.

[ORC00b] ORCI T. *Capability Maturity Model for Extra Small Organizations, Level 2.* Technical Report UMINF 00.13, Department of Computing Science, Umeå University, September 2000.

[ORC00c] ORCI T. *Capability Maturity Model for Extra Extra Small Organizations, Level 2.* Technical Report UMINF 00.12, Department of Computing Science, Umeå University, September 2000.

[PAR96] PARK R., GOETHERT W., FLORAC W. *Goal-Driven Software Measurement – A Guidebook.* Carnegie Mellon University Software Engineering Institute, August 1996.

[PAU93] PAULK M., WEBER C., GARCIA S., CHRISSIS M., BUSH M.. *Key Practices of the Capability Maturity Model, Version 1.1.* Technical Report CMU/SEI-93-TR-025, ESC-TR-93-178, Carnegie Mellon University Software Engineering Institute, February 1993.

[PAU96] PAULK M. *Effective CMM-Based Process Improvement.* Proceedings of the 6th International Conference on Software Quality, Ottawa, Canada, 28-31 October 1996, pp. 226-237.

[PAU98] PAULK M. *Using the Software CMM in Small Organizations.* http://wwwbroy.in.tum.de/lehre/vorlesungen/vse/WS2004/1998_cmm-small.pdf, TU München - Fakultät für Informatik, 2005-02-11.

[PAU99] PAULK M. *Using the Software CMM With Good Judgment.* ASQ Software Quality Professional, Vol.1, No.3, June 1999, pp. 19-29.

[SEI95] SOFTWARE ENGINEERING INSTITUTE, CARNEGIE MELLON UNIVERSITY . *The Capability Maturity Model: Guidelines for Improving the Software Process.* Addison-Wesley Professional, 1995.

[SOL99] 1 VAN SOLINGEN R., BERGHOUT E. *The Goal Question Metric Method.* McGraw-Hill Company, 1995

[STA94] THE STANDISH GROUP INTERNATIONAL INC. *The CHAOS report.* http://www1.standishgroup.com/sample_research/PDFpages/chaos1994.pdf, 2005-07-17.

[TIT97] SFK-DATA OCH SVERIGES VERKSTADSINDUSTRIER. *TickIT-handboken; handledning vid konstruktion och certifiering av kvalitetssystem för programvara med användning av SS-EN ISO 9001.* Industrilitteratur, April 1997.

[WIL97] WILSON W., ROSNEBERG L., HYATT L. *Automated Quality Analysis of Natural Language Requirement Specification.* Proceeding International Conferance on Software Engineering, NASA Software Assurance Technology Center, May 1997.

[ZAV97] ZAVE P. *Classification of Research Efforts in Requirements Engineering.* ACM Computing Surveys, 29(4), 1997.

[ZOW05] ZOWGHI D. *A longitudinal Study of Requirements Volatility in Software Development.* ASMA/SQA Meeting, University of Technology Sydney, May 2005.

[WWW1] NATIONALENCYKLOPEDIN.SE http://www.nationalencyklopedin.se, 2004-10-20.

[WWW2] STICKYMINDS.COM http://www.stickyminds.com/sitewide.asp?ObjectId=2409&ObjectType=COL&Function=edetail, 2004-10-23.

[WWW3] SOFTWARE ENGINEERING INSTITUTE http://www.sei.cmu.edu/, 2005-08-10

Appendix A

Questions and measures for the goals of the Requirements Management KPA

As suggested by Loconsole[LOC01]

A.1 Questions for the first goal of the Requirements Management KPA

Proposed questions for the first goal of the Requirements Management KPA

	Questions	**Measures**
1	What is the current status of each requirement?	Status of each requirement
2	What is the level of the stability of the requirements?	Number of initial requirement Number of final requirements Number of changes per requirement

Questions for the first goal of the Requirements Management KPA

	Questions	Measures
3	Why are the requirements changed?	Number of initial requirements Number of final requirements Number of changes per requirement Number of test cases per requirement Type of change to requirement Reason of change to requirements Major source of request for a change to requirements Phase where change was requested
4	What is the cost of changing the requirements?	Cost of change to requirements Size of a change to requirements
5	Is the number of changes to requirements manageable?	Total Number of requirements Number of changes to requirements proposed Number of changes to requirements open Number of changes to requirements approved Number of changes to requirements incorporated into baseline Number of changes to requirements rejected The computer software configuration item(s)(CSCI) affected by a change to requirements Major source of request for a change to requirements Requirement type for each change to requirements Number of requirements affected by a change
6	Does the number of changes to requirements decrease with time?	Number of changes to requirements per unit of time
7	How are affected groups and individuals informed about the changes?	Notification of Changes(NOC) shall be documented and distributed as a key communication document Number of affected groups and individuals informed about NOC
8	How many other requirements are affected by a requirement change?	Number of requirements affected by a change

APPENDIX A. QUESTIONS AND MEASURES FOR THE GOALS OF THE REQUIREMENTS MANAGEMENT KPA

	Questions	Measures
9	In what way are the other requirements affected by a requirement change?	Type of change to requirements Reason of change to requirements Phase where change was requested
10	Is the size of the requirements manageable?	Size of requirements
11	How many incomplete, inconsistent and missing allocated requirements are identified?	Number of incomplete requirements Number of inconsistent requirements Number of missing requirements
12	Does the number of "To Be Done"(TBD) decrease with time?	Number of TBD's in requirements specifications Number of TBD's per unit of time
13	How are the requirements defined and documented?	Kind of documentation
14	Are the requirements scheduled for implementation into a particular release actually addressed as planned?	Number of requirements scheduled for each software build or release
15	How many requirements are included in the baseline?	Number of baselined requirements Phase when requirements are baselined

A.2 Questions for the second goal of the Requirements Management KPA

Proposed questions for the second goal of the Requirements Management KPA

	Questions	Measures
1	Does the software product satisfy the requirements?	Functionality of the software Number of initial requirements Number of final requirements Number of tests per requirement Type of change to requirements
2	What is the impact of requirements changes on the software project?	Effort expended on Requirements Management activity Time spent on upgrading Number of documents affected by a change
3	What is the status of the changes to software plans, work products, and activities?	Status of software plans, work products, and activities?
4	Are the requirements scheduled for implementation in a particular release actually addressed as planned?	Number of requirements scheduled for each software build or release
5	How are the requirements defined and documented?	Type of documentation
6	Does the number of TBD's prevent satisfactory completion of the product?	Number of TBD's in requirements specifications
7	Are all development work products consistent with the requirements?	Number of inconsistencies

Appendix B

Abbreviations

BSC	Balanced Score Card
CMM	Capability Maturity Model
CSCI	Computer Software Configuration Item
GQM	Goal Question Metric
KPA	Key Process Area
NOC	Notification of Changes
PM	Project Manager
S, XS, XXS	Small, eXtra Small, eXtra eXtra Small companies
RM	Requirements Management
TBD	To Be Done