



**University of Applied Sciences**

## Quality Management

Module: Software Engineering 2

Students: Daniel Chiaradia (2109134)  
Kellermann Jennifer (2117248)  
Christoph Mülleneisen (2110454)  
Bodo Vossen (2116438)

Tutors: Herr Ferd van Odenhoven  
Frau Julia Stoll

Date: March 30, 2009

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Goals of Quality Management</b>	<b>2</b>
<b>3</b>	<b>Quality Assurance</b>	<b>3</b>
3.1	Standards . . . . .	3
3.2	Example . . . . .	4
<b>4</b>	<b>Quality Planning and Controlling</b>	<b>5</b>
4.1	Write quality plan right . . . . .	5
4.2	Control of Quality . . . . .	7
<b>5</b>	<b>Software measurements and metrics</b>	<b>8</b>
5.1	Meaning of software measurements . . . . .	9
5.2	Measurements . . . . .	10
5.3	Metrics . . . . .	10
<b>6</b>	<b>Conclusion</b>	<b>12</b>
<b>7</b>	<b>Appendix</b>	<b>13</b>
7.1	Discussion . . . . .	13
7.2	Table of authors . . . . .	14

## 1 Introduction

An overview is given about, why we need quality management and what is the goal of it. Furthermore you will find information how you can ensure quality, how to plan and control quality and some measurement aspects of quality.

The specification of a product is one of the larger problems within a software process. The usability is an example. It is very hard to understand what the customer means by usability. If the customer said that he needs a very good usability it isn't very clear what does he mean. The customer has its own vision of a good usability but he couldn't explain it. However, a specification never contains all requirements that the customer needs. And this is the main problem. The development team has to decide what the customer means with usability. Very often the customer gets software that doesn't fit his needs. So, the software has to be changed to meet the customer's needs. This costs a lot of extra money and working hours. Another important point is the visibility of the whole process. If you want to make your process clear and understandable you have some kind of common standards. This is very important for the customer and the company. If the process is visible than the company is able to make a good reflection of the process and can improve it. If the process is visible to the customer, he has a better idea what the product is like.

The communication and product quality can be improved by using quality management. If you use standards in your software process the communication will be more understandable because for example every document has the same structure. You gain a better visibility by using process standards because the whole process is more understandable by a common standard. In software engineering there is deep dependency between process and product quality. Many factors influence the process quality and implicit the product quality. Examples for such factors are process methods like if you are using reviews or not. this work is to share knowledge within the group. So everybody in our group knows exactly the same things about quality management.

## 2 Goals of Quality Management

This section defines the goals of quality management. All developers have to know why they have to use quality management within a software process, especially to get a good result/product.

### Definition

The topic of Quality is a very important topic for the industrial production, since the beginning of the 20th century. But at the beginning the understanding for quality was not the same like today. To ensure Quality at that time, this means to sort out defective products. There was no statistical

based management tool, to improve or ensure quality.

A beginning which goes more over the pure manufacturing control and the whole organisation with all areas includes, originated since the 70s and was fixed in 1987 in the first international quality management norms.

In the topical version of the norm ISO-9000 quality management is defined as:

“The concerted activities to supervise and steer an organization in terms of quality”

Under the topic of quality management we understand on the one hand a central management job settled at the leadership level and on the other hand the use of the concept "Organisation" shows that quality is not only concerns the production of goods, but also principle for every group of people and facilities who produce achievements for an aim audience, i.e. for her customers within her respective field of application is valid.

### **3 Quality Assurance**

For the software assurance there are many procedures and processes to look at the quality characteristics. The development team have to look, if these characteristics fit to the specifications, e.g. therefore we check, if they meet values of established standards.

Software quality assurance (SQA) help us to control, to take care and to ensure our product quality. SQA works with standards like the ISO 9000 an similar ones.

#### **3.1 Standards**

Standards are very important for software processes because you need something to guarantee your quality. Mostly standards are developed over a large period. Some methods within a project (not only software engineering) were very successful and provided a good framework to support the team. If something was successful there is a high chance that it will also successful in future projects. After a project has finished you analyze how useful the standard was by asking your team members, checking the time, that was spend on a process, and compare it with past projects. Another very important point where standards are important is the internal and external communication. For example in a large project with nearly 200 developers it can be very difficult to change something in a subsystem. If you want to do this you need a standardized way to do that otherwise you will lose the overview over your change requests.

You can find two types of standards:

1. Product standards: These standards affect the product. An example is the document structure or the source code style.
2. Process standards: These kind of standards affect the software process. They describe how the whole process is organized. For example how the specification has to be developed (Iterative process -> a short high level specification; waterfall model -> very detailed and huge specification)

The process quality influence the product quality, as already mention in section 3. Product standards influence the result of the project and the process standards ensure that product standards are followed. For example the waterfall model ensures a design phase will follow after an analysis phase. This guarantees that the design documents will refer to the analysis documents. So, that's the reason why they separate between these types of standards.

However, sometimes the developers don't understand the use of such standards and they ignore them. To prevent/solve this problem the developers should be involved in the decision process. It's very important to develop standards. When new technologies will be introduced you may have to rework your standards. For example for a new agile process you have to make your standard conform to this one because the process has changed and your process standards aren't conform to this agile process.

The ISO 9000 is a common standard for industrial processes. It will be explained in detail in the next section.

### **3.2 Example**

Over the years some of the used kinds of SQA proof themselves. These ones have become standards for the quality management. The most important and most used ones are ISO9000 Standards. ISO 900 is an international set of standards that can be used in the development of quality system in industries. In software development sector, we can use the ISO9001 document. If your company want to and also is able to, they can get certified. For it you have to follow several rules in administrating your team. The standard describes the approach within a company. E.g. how to decide which employees are qualified to do a special job, much about responsibilities, rights and others. The ISO Standard also advises to use a quality manual. In this document, which is available for everybody, it is described, which specifications are required from the customer, but also from the development team. It contains of the product's specifications and also the process' ones. In practice this can mean, that all documents have to follow a name konvention, every method has to be commented, tricky coding isn't allowed, all

code and documentation is to write in english and so on. The Iso Standards describes the 'how' but not the 'what' as you can see in the figure.



## 4 Quality Planning and Controlling

### 4.1 Write quality plan right

One important part for quality assurance is that the developer is able to write the quality plan. The quality plan is a very important document for the whole team. It describes exactly, what quality means for this project and it is divided in five main parts:

1. Product introduction In the product introduction you have to mention a description of the product, the quality expectation and for who you write this software.
2. Product plan The product plan could be a part of the project plan. In this section you have to enumerate the release dates and the responsibilities.
3. Process description This part describes your process . For example if you use Scrum or something similar.
4. Quality goals In the quality goals part you have to decide which quality attributes are the most important for you quality (look at list).
5. Risks and risk management Risks and risk management describes everything that could go wrong during your process, and might reduce your product/process quality.

During writing the plan you have to make some compromises, as I mentioned in the quality goals description. To show up what quality means there are a few quality attributes, and you have to decide which ones are the most important ones for your project. It is nearly impossible to optimise your project for all of them. These attributes can be divided in two groups. One group contains the attributes which are important for the customer, and the other one those which are important for the developer:

#### Customer's attributes

Attribute	Description
Safety	Describes if the product is safe. It is important that the program does not have any gaps, which could be misuse by criminals.
Reliability	Describes if the program is solid, or if it break down every day.
Resilience	Describes the program's attitude what happens if the program has a bug throughout.
Robustness	Describes the error robustness of the program.
Usability	This is perhaps the most difficult attribute. As mentioned in the introduction the customer can have a quite different opinion for usability is than the developer team. In general, usability describes if user can easily use this program.
Efficiency	Describes the performance of the program.

**Developer's attributes**

Attribute	Description
Security	Describes the safety from the developers point of view.
Understandability	Describes how easy or difficult it is to read up on the program.
Adaptability	Describes if the program can adapt to some other program or device.
Testability	Describes if the program is easy to test.
Modularity	Describes a bit of the design of the program. Normally/Always OO-programs consists of many classes.
Complexity	Describes how compley a program is. It is quite difficult to measure. Some aspects will be mentioned in the section measurements.
Portability	Describes if it is possible to run the program on another computer or device.
Reuseability	Describes if the developers can reuse some parts of the program. For example if you write your own List class, you can reuse it for your next project.
Learnability	Describes something common like understandability. How difficult it is to read up in the program.

One important thing is that you have to take care that the quality plan and the project plan are compatible to each other. If you change something in your project plan you propably have to change some aspects in your quality plan. There is a deep relationship between those two plans.

## 4.2 Control of Quality

The quality control is a part of the quality management that is directed upon the fulfillment of high-class requirements or quality golas. Besides, is distinguished in two measures:

1. Analytic measures (check)
2. Constructive measures (steering system)

By the analytic quality measures we understand the measures which check the quality of software and the values. This kind of measures plays a huge role for validation, verification and certification of software. Validation checks the correspondence of the product standards of the product at the end of the software development process. With verification the results of different product phases are

compared. With the certification the validation of the product must be proved by an independent testing organisation. Besides, it is checked whether the software has the “non functional” qualities which you have promised. Product parts or subsystems are tested to ensure the customer requirements have been observed. The resulting findings will be used to improve the development processes. With the constructive measure of quality control the mistake-preventing and mistake-avoiding processes will define or integrate test procedure and correction procedure into the processes. This can be:

- **Rules of the software Designs defines** Describe the architecture of the software or the entire system and individual components.
- Described in the IEEE – an example is the SDD – **Software Design Description** IEEE 1016
- **Code reviews (in a few points or constantly like pair programming)**. Two pairs of eyes are better than one - in some moments of development it can be better to implement or review software components with pair programming, to avoid or resolve problems.
- **Refactoring** This means to locate code fragments which are not implemented very well. To much instructions or missing documentations this parts must be reviewed and revised.
- **Defensive programming** Develop the software according to the method, that the software always be suspicious about every input from the users.

Resulting results can be used to improve the development process to circumvent an assigned problem in the future.

## 5 Software measurements and metrics

In the following part we will describe how software quality can be measured , and what are the problems by doing so. Years ago developers could count the lines of code to determine a quality measure, but in times of object oriented programming languages, this is not an adequate method. There are several reasons for that. For example you can write a “small” (program with just a few functions) program, but you write all you classes on your own (your write your own ArrayList, etc.). So you have many lines of code but just a few functionality. Otherwise you can write an even bigger project by using the Library (for example Java Library) and reuse the already implemented classes like ArrayList. The first project perhaps has more lines of code, but does it mean it has a higher quality? Probably not, because if you look at the quality attributes (see quality plan) the second project would have the better values. We should take a closer look at some attributes:

**Understandability:** New programmers would need more time to read up in the project, because he does not know how the List is implemented, and every programmer knows the ArrayList.

**Complexity:** Probably the program will become more complex. If you write your classes on your own you will have more code and more classes, instead of just one import line.

**Safety:** The last two were developer's attributes, so let's take a look to a customer attribute. If you write everything on your own, you could make mistakes, and those mistakes could be misused by criminals.

### 5.1 Meaning of software measurements

There are some kinds of quality assurance. In addition to the known reviews we have the possibility to proof our software by measurable values. This can be done by software, but also by developers. This method is called Software Measurement. The Software Measurements is less an alternative and more a tool for those time-consuming reviews to speed them up.

Only if there are comparable values (like times, numbers etc.), such a measurement will be possible. We use metrics to find such comparable values. Those metrics are usually mathematical functions, which derive indices from the characteristics of our program or the used processes.

When we have found indices, we are able to compare them to the indices of our established standards. If the compared index is near to the standards one, we can talk about a high software quality.

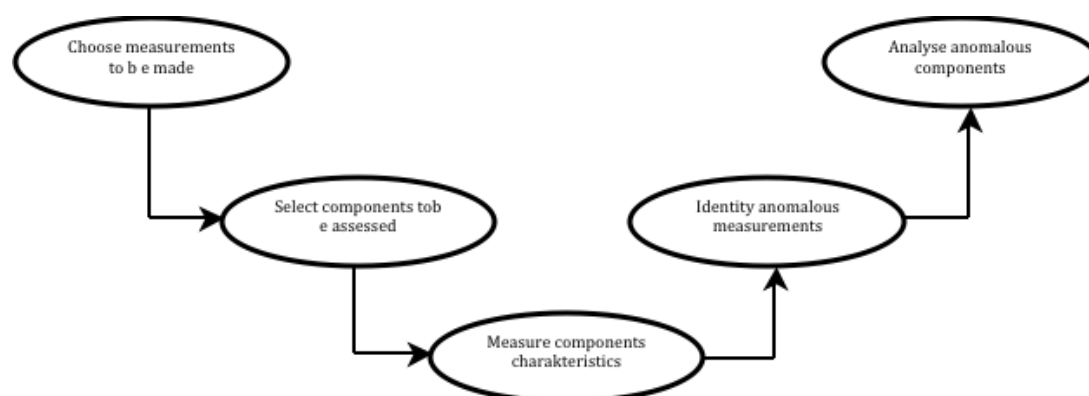
So if a standard needs 30 seconds to execute concrete sequence of actions and your implementation does take a minute, you definitely have to think about, if yours is working profitable or at least correctly.

## 5.2 Measurements

The process of software quality management is divided in five main parts. These five are:

1. **Choose measurements to be made:** Think about what you want to know,
2. **Select components to be assessed:** What components have to be assessed for the wanted result exactly,
3. **Measure components characteristics:** The measurement of the chosen components itself and the storing of the results,
4. **Identify anomalous measurements:** Comparison to calculated or earlier results and check the differences, and
5. **Analysis anomalous components:** If there are anomalous components, you have to inspect, if it is really an indicator for bad quality or are there other reasons for it.

The following figure shows the process:



If tested program parts are not used in the later process, the measurement results will be stored although. So, the stored data will become a database, which is big enough to compare with results from new measurements. Again, it will be possible to draw conclusions about our software quality.

## 5.3 Metrics

As already mentioned, we can help us to measure software quality with so-called metrics. In this case there are two different types of metrics. The first is the product metric, which evaluate the software and the second is the process metric, to evaluate the development process. For the assessment of these qualities we need measurable dimensions, which give us qualitative statements about the quality of the software. These dimensions are metrics and from it derived quality criteria, which we can find out by analysis and measuring of the software code. The quality criteria or even different metrics

form a clearly measurable basis. The identification of metrics, quality criteria and the associated analysis of the software code, are usually time-consuming tasks, so that a computer support through an appropriate application promises many advantages.

The most known product metrics and basis for more metrics is the software size, which can be measured in LOCs (Lines Of Code). We can use it to measure the productivity of the whole development team during a project. A often used measure method is also the so-called "Function Point" method. This method is based on the software design, the number of inputs, the output operations, file access, the external interfaces and the number of requests coming from outside of a module must be identified, added and combined with a correction factor. The results we receive can be saved in a database. So we have a result or a result set which we can compare with other results from the past or for the future to find possible anomalies. But the method of LOC counting or the Function Point method are not enough to measure software quality, because this methods are not exactly enough to measure and ensure software quality. These metrics do not allow any direct statement about the quality criteria. Only by many experimental analysis between static metrics and system complexity, understandability and maintainability of a program it can be possible to set relations.

To guarantee that our software quality can be measured, it is necessary to classify and summarize the quality requirements or to subdivide it in quality properties. This means that the quality of a program can be assessed by several quality characteristics. Important are however those who are relevant for the user and for the purpose.

Therefore, the ISO-9126 standard defines some software quality properties.

- Functionality
- Reliability
- Usability
- Efficiency
- Modifiability
- Portability

These properties are too abstract to take directly a statement about the quality. This must be displayed with concrete parameters. Concrete parameters for example are the numbers of moduls in a program, or the numbers of instructions in a modul. You can detect the numbers of errors inside a component with testing. That is a concrete dimension which can be reflected to the abstract property

“reliability”. Other properties - like usability - are difficult to identify with tests, because everybody interpret usability in a another way.

Other example is the property - portability - which operating systems can be used and is the performance on every other OS exactly the same (combined with the property “efficiency”).

Although the topic of software measurement in the field of software developers increasingly discussed and debated, is the systematic use of software measurement in the Industrial still very unusual.

## 6 Conclusion

Quality management is an integrated approach within a software process. It starts at the beginning of a project and ends with it. Quality Management ensures that the product fits to the customers needs. First quality management will define process and product standards (see section 3.1). These standards have to be adapted to the actual project because some approaches of a standard are counterproductive. Quality management can handicap the developer. You have look at the range/scale of your project. Sometimes it isn't necessary to write down every detail. E.g. if there are only two developers than you have to protocol only the essential points. The quality plan and control is very important during the whole process. You have to plan it, write a product plan and define what your quality goals are. Than you have to control your quality with assessments and observance of your process (see section 4.2). But to assess the quality, you have to find measurements (see section 5.2). This is a very critical point, because you can't find measurements for every property of your product rather process. For example for the maintainability of software it is hard to find measurements and in days of OO programming you cannot derive quality from the number of lines of code, because it attempts to avoid many lines of code by reusing components. Finally, you have to analyze your process to improve your standards that are used in this process. This is very important to improve your standards for the next projects (see section 3.1) Finally, quality management is mandatory in software process to ensure that your product fits to you specification.

## 7 Appendix

### 7.1 Discussion

It easy to meassure objectivly quality.

Thats not true, because Software has a lot of aspects which must be observed. For example the point of usability: How you can measure usability? People who are working a lot on different software, can fell the usability of your developed software as very comfortable, but whats about people who have not so much experience in using software. So to measure quality objectively you must create different metrics to receive measure data which must be saved and compared with measure data from the past to locate anomalies and to be able to say something about your software quality.

Quality management is a barrier for the developer.

Thats not true, because with quality management you can improve your development processes. With the regular tests and reviews in different steps of the development you can locate errors or mistakes. These mistakes or errors can be directly reflected to a period or process during the development. So you can ensure that these errors or mistakes are no longer happened for the future.

Quality manager can also be a part of the development team.

That is true. In my opinion I think it is the best way to ensure software quality, because he is sitting at the root of the processes. But a quality manager is also someone, who must observe every area in a organisation.

## 7.2 Table of authors

This table contains all authors of the text. The authors are responsible for content.

Authors name	Section
Introduction	Bodo Vossen & Daniel Chiaradia
Goals - Definition	Christoph Mülleneisen
Assurance - Standards	Daniel Chiaradia
Assurance - Examples	Jennifer Kellermann
Planning & Controlling - Plan	Bodo Vossen
Planning & Controlling - Control	Christoph Mülleneisen
Software Measurements & Metrics - Definitions	Jennifer Kellermann
Software Measurements & Metrics - Measurements	Jennifer Kellermann
Software Measurements & Metrics - Metrics	Christoph Mülleneisen
Conclusion	Daniel Chiaradia
Introduction - Planning & Controlling	Christoph Mülleneisen
Introduction - Measurements & Metrics	Bodo Vossen
Introduction - Goals	Daniel Chiaradia
Introduction - Assurance	Jennifer Kellermann
Appendix - Statements	Christoph Mülleneisen