LINKING SOFTWARE QUALITY TO SOFTWARE ENGINEERING ACTIVITIES, RESULTS FROM A CASE-STUDY

Jos J.M. Trienekens, Rob J. Kusters
University of Technology Eindhoven, Den Dolech 2 5600 MB Eindhoven, The Netherlands

Dennis C. Brussel
Centre for Autonation of Mission-critical Systems (CAMS), Royal Navy
De Ruyghweg 200, Den Helder, The Netherlands

Keywords: Software quality specification and implementation.

Abstract: Specification of software quality characteristics, such as reliability and usability, is an important aspect of software development. However, of equal importance is the implementation of quality during the design and construction of the software. This paper links software quality specification to software quality implementation using a multi-criteria decision analysis technique. The approach is validated in a case-study, at the Royal navy in The Netherlands.

1 INTRODUCTION

Command frigates (Dutch acronym LCF) are warships that are used in national and international task forces to support political decisions. An important aspect of the power of these frigates are the so-called guided missiles. The heart of the command function of such a warship is supported by the Combat Management System (CMS). This highly advanced software-intensive system integrates sensor and missile systems and is the central operating system of the cruiser. Up to twenty-two operators are working in the kernel of the command function on powerful working stations to identify air attacks and to determine defense actions. The information that is needed to identify foreign objects is provided by different types of sensors, e.g. from radar systems. The various signals are being collected by the sensors and are tracked and analysed by the kernel functionality of the CMS, called Multi Sensor Tracking (MST).

Recently the quality assurance team at CAMS, in close collaboration with selected user representatives and representatives of software engineering teams, have identified quality problems. Users, c.q. command function operators of the LCF, are not fully satisfied with the quality of the software, in particular the reliability and the time-behaviour of software applications. Also the software engineers have incrasing 'quality' problems, e.g. caused by the need to explicitly define distinct software quality attributes, and to carry out explicit engineering activities to implement quality into a software application.

One of the biggest problems is the high degree of subjectivity in the way quality is dealt with. Both domain experts, e.g. business context and information analysts, and software engineers have their own definitions and interpretations of quality which often leads to misunderstandings and confusion. Another problem is the increasing complexity of the advanced software applications, due to the extended functionality and the high level of integration of the different components. Software engineers often have to deal with conflicting quality requirements.

The management of CAMS decided to set up a project in that the concepts and terminology regarding software quality had to be clarified, and that had to result in an operational approach for the specification and implementation of software quality. The approach should limit subjectivity in the determination of the quality characteristics of a particular software application, and should link an operational specification of quality attributes to particular engineering activities Key issues in the project are:

J. M. Trienekens J., J. Kusters R. and C. Brussel D. (2007).
LINKING SOFTWARE QUALITY TO SOFTWARE ENGINEERING ACTIVITIES, RESULTS FROM A CASE-STUDY.
In *Proceedings of the Second International Conference on Software and Data Technologies - SE*, pages 117-122
DOI: 10.5220/0001325501170122

- the usage of complexity reduction techniques regarding the specification of quality requirements;
- the usage of techniques to support close collaboration between users and engineers regarding the definition and prioritisation of quality attributes.

The paper presents results regarding the decomposition of software quality characteristics, the determination of their relative importance by using AHP-techniques, and the determination of the contribution of particular engineering activities to the implementation of software quality.

2 BACKGROUND

The International Standard Organization (ISO), defines quality as 'the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs' (ISO/IEC 9126, 2001). ISO 9126 makes a dinstinction between internal and external quality, and quality in use. Internal quality attributes reflect e.g. the statical aspects of the software code of an application. External quality characteristics are determined by the dynamic aspects of a software application, and 'quality in use' is determined by the level to that a software application meets the expectations of the user. Although ISO9126 offers clear definitions of the quality characteristics and attributes, it doesn't offer guidance to the process of identification and prioritisation of quality characteristics, and it doesn't support software engineers in the process of implementing particular quality attributes (Trienekens and Kusters, 1999). As a consequence, currently a lot of subjectivity exist in the way that quality is dealt with. Both the specification and the implementation of quality into the software applications are characterised by vague and often ambiguous decisions of both domain experts, information analysts and software engineers.

3 LINKING QUALITY SPECIFICATION TO QUALITY IMPLEMENTATION: THE APPROACH

Specification and implementation of software quality cover complex decision processes because of issues such as the high degree of subjectivity, conflicting needs, weak structuredness of requirements, etc. Decision processes with these characteristics are being explored in a research area called Multi Criteria Decision Analysis (MCDA) (Roy, 1996). The objective of MCDA is to provide practitioners with a rationale for the ordering and rating (i.e. prioritisation) of different aspects of a process or product by using different types of criteria. At CAMS it has been concluded that MCDA could offer support in the specification and prioritisation of quality characteristics and attributes, and also regarding the selection of appropriate engineering activities to implement particular quality attributes into a software-intensive system.

Analytical Hierarchy Process (AHP) makes use of a set of techniques that offer support to reduce the complexity of a problem, and to determine the relative importances of particular aspects of a system (Saaty, 2001). This set of techniques offer domain experts and engineers the opportunity to clarify and motivate their decisions in complex engineering processes.

The first step in applying AHP in our case study is the specification of quality characteristics (i.e. the external quality) and quality attributes (i.e the internal quality). Based on the ISO9126 standard on software quality a hierarchical well-specified structure of quality characteristics and attributes has been developed

In the second step AHP is used to determine the weights to specify the relative importances of the various quality characteristics and attributes. AHP is a hierarchical method of pairwise comparisons of each quality (sub)-attribute against one another with respect to a higher level quality characteristic or attribute. Preferences among quality characteristics and attributes are converted into numerical weights, see for a clear report of an AHP application (Weil and Apostolakis, 2001).

In the third step the engineering activities are being determined that are needed to implement the particular quality attributes into the software application. For these engineering activities so-called contribution functions are being determined. A contribution function reflects the contribution of a particular engineering activity to the implementation of particular quality attributes.

In the fourth step, the performance index (PI) of each quality (sub)-attribute is calculated which expresses the relative contribution of a software quality (sub)-attribute to the overall quality of a software application. The following section presents the application of the stepwise approach in a case study.

4 VALIDATION OF THE APPROACH: A CASE STUDY

The case study has being executed on the Multi Sensor Tracking (MST) application which covers the kernel functionality of the Combat Management System of a frigate. To limit complexity in this paper a selection has been made from the kernel functionality, repectively the sub-function *localising signals* and the data objects *set of signals* and *location of signal data*.

Step 1: Development of the software quality hierarchy.
The hierarchical structure is in fact the result of the complexity reduction of a decision problem. It clarifies different types of decisions that have been made, such as the decision on the type of quality characteristics and attributes that should be recognised, and the decision in what way these characteristics and attributes are interrelated. Figure 1 shows that the total system quality is splitted up into three quality characteristics, respectively timeliness, reliability and user friendliness These quality characteristics are decomposed further into quality attributes and sub-attributes. For example reliability is decomposed into availability and operational reliability and the latter is decomposed further into failure rate and overload prevention. At CAMS for this task several domain experts have been selected, in conformance with (Karydas en Gifun, 2006). These experts have a broad experience regarding the type of functionality of the software to be developed. The 'time box' teams consist of both software developers and users representatives. In the team sessions various techniques are being used such as brainstorming and peer reviews. Different sources of information are used, respectively historical information, quality standards such as ISO/IEC 9126, and the current functional specification of the Combat Management System.

Step 2: Determination of the relative importance of quality characteristics and attributes
A weigth reflects the importance of a particular quality characteristic or attribute, to a particular characteristic or attribute at a higher level. Note, in Figure 1, that reliability has received the highest weigth, or the highest level of importance regarding the 'overall' quality of the system. Three sub-attributes of reliability, respectively overload prevention, redundancy and failure rate have received the highest weigths in the whole set of quality attributes and sub-attributes.

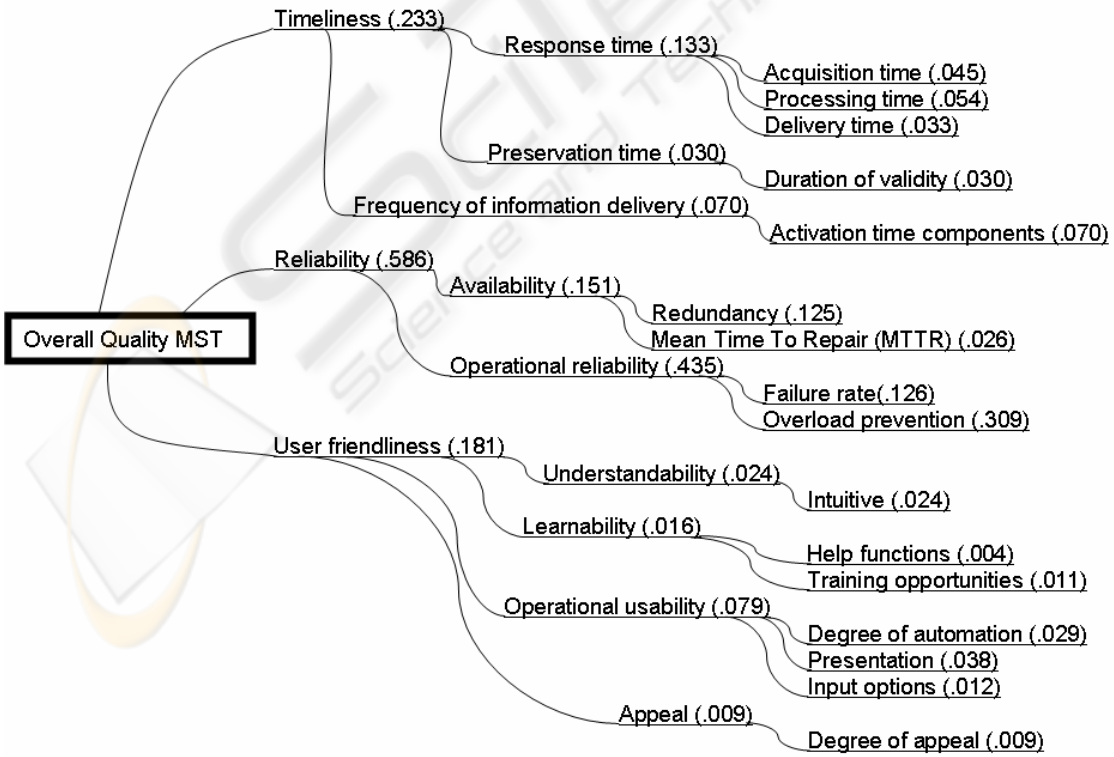

Figure 1: A software quality hierarchy.

Step 3: Determination of the quality contribution of engineering activities

To be able to implement quality (sub)-attributes into a software system appropriate engineering activities have to be determined. In order to reflect the distinct contributions of the particular engineering activities to the implementation of a quality attribute, again AHP is used. In the following we give an example of the way engineering activities are being specified and contribution factors are being determined.

In this example we take the quality (sub)-attribute ‘overload prevention’ which has the heighest weigth of the four attributes of the quality characteristic reliability. 'Overload prevention' of a particular software functionality is defined as the degree to that a particular software functionality prevents, that the maximum number of system tracks that a software functionality can handle, is reached. The objective of 'overload prevention' is to apply in a structured and well-defined way so-called *graceful degradation*. Regarding the particular engineering activities that are needed to implement 'overload prevention' two engineering aspects have to be recognised, respectively the 'detection of the overload threat' and the 'prevention of overload'. Different engineering activities have been specified by the engineering experts and it has been made clear what type of engineering activities are complementary or mutually exclusive. For example regarding an overload that cannot be detected it is also not possible to prevent it. Or, in case an operator has detected an overload threat, it is not possible to carry out an automated prevention algoritm. Subsequently the contribution factors have been determined for each of the engineering activities. These factors reflect the relative contribution of each of the distinct engineering activities to the implementation of the quality sub-attribute 'overload prevention'. Table 1 presents the result.

Table 1: Engineering activities and their contribution factors.

Engineering Activity	Description	Contribution factors
1	An overload threat will not be detected and will not be prevented. As a consequence not any track is being processed by the functionality.	0,07
2	An overload threat will be detected by an operator, but cannot be prevented. Subsequently all tracks are being processed by the functionality with quite some delay.	0,17
3	An overload threat will be detected by an operator but is not prevented in an automated way. The consequence is that alle low priority tracks are being deleted by the operator and all high priority tracks are processed by the functionality.	0,4
4	An overload threat will be detected in an automated way but will not be automatically prevented. Subsequently all low priority tracks have to be deleted by the operator and all high priority tracks can be processed by the functionality.	0,94
5	An overload threat will be detected automatically and prevented automatically so that all high priority tracks are being processed by the functionality.	1

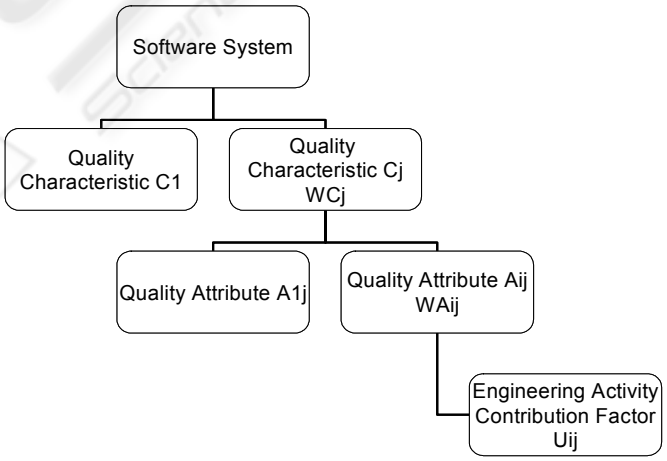

Figure 2: Hierarchical structure of quality characteristics and quality attributes.

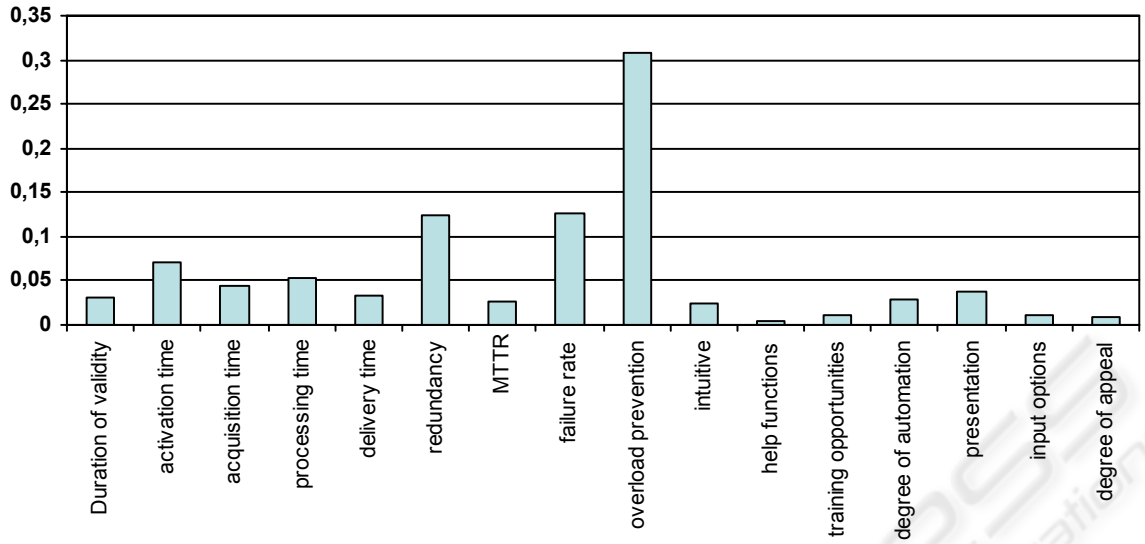

Figure 3: The relative contribution of attributes to 'overall' software quality.

Step 4: Determination of the performance index (PI) of the quality attributes

For each quality (sub)-attribute a so-called performance index (PI) is being calculated. The PI reflects the level to that particular quality attributes are being implemented, by carrying out particular engineering activities. The PI of a quality sub-attribute is the sum of the multiplications of the weights of the sub-attributes by the contribution factors of the particlar specific engineering activities.

$$PIC_j = \left(\sum_{i=1}^{n} (WA_{ij} * U_{ij})\right) / WC_j) * 100\%$$

n = number of attributes per quality characteristic.

PIC_j = Performance Index of quality characteristic Cj.

WC_j = Weight factor of quality Characteristic Cj

WA_{ij} = Weight factor of quality (sub)-Attribute Ai of quality characteristic Cj.

U_{ij} = contribution factor of engineering activity to quality attribute Aij of quality characteristic Cj.

In Figure 3 the relative contribution to the 'overall' quality of a particular software functionality is shown. The 'overall' quality is the result of particular engineering activities that have been carried out to implement the quality (sub)-attributes. Figure 3 clarifies the effectiveness of the engineering activities. The bar chart expresses the contribution of the different quality attributes to the total quality of the software system. Based on this information, engineers can predict the effects of particular engineering activities and it becomes easier for them to calculate the effort that is needed to reach a certain level of quality. Also via recalculations it becomes possible to predict the effects of an increase or decrease of particular engineering efforts regarding the implementation of particular quality attributes. For example it became clear that chosing engineering activities with 'average' contribution factors still resulted in a satisfactory level of particular quality (sub)-attributes, and a satisfactory 'overall' quality level of the software application.

5 LESSONS LEARNED

Although ISO9126 offered a useful quality basis, some of the quality attribute definitions had to be reinterpreted and/or translated to the specific technical context of the software-intensive system. During the project it was still necessary to redefine particular quality attributes as a result of brainstorm sessions and peer reviews. An interesting result of the case study was that some quality (sub)-attributes have been identified and defined that have not been discovered in previous projects at CAMS.

However, also some restrictions and shortcomings have been identified. First of all the time aspect. Both the collection of information in the beginning of the project, the brainstorm sessions, the peer reviews to develop the quality hierarchy, including the weigths took much time. This problem of applying MDCA and AHP in practice has also been recognised by (Weil en Apostolakis, 2001). In the case study only one piece of the software

functionality of the Multi Sensor Tracking (MST) has been investigated, and only with regard to a particular quality characteristic, i.e. 'overload prevention', which is one of the sub-attributes of the quality characteristics 'reliability'. Therefore it is suggested to determine first the critical parts of a complex software application and subsequently to apply then for only the critical parts of the software application the approach presented in this paper.

To solve the time problem it has been suggested to develop a database with information about the various type of software applications, their quality profiles (quality characteristics, attributes), and experiences from earlier work such as experiences regarding the interpretation and the redefinition of ISO9126 quality terminology. Also the contribution of the distinct engineering activities that are needed to implement particular quality (sub)-attributes should be defined and stored. This type of knowledge management should improve the efficiency of the engineering of software quality.

6 CONCLUSIONS

The CAMS department of the Dutch Department of Defense has investigated an approach to deal in a formal and systematic way with the quality of software. Software quality can be specified in a precise and formal way. The importance of the distinct quality attributes can be determined by applying AHP techniques.

A quality hierarchy forms a basis for software engineers to build in quality into a software application. Also regarding the determination of appropriate engineering activities, and their relative importances, AHP can be used. Contribution factors reflect the relative contribution of particular engineering activities to the 'overall' quality of a software application

A major drawback that has to be mentioned is the time aspect of applying MDCA and AHP. To solve this problem the reuse of previous work is stressed. A knowledge repository has been defined that captures quality profiles of software components and applications, best practices in quality specification and implementation, sets of weights and contribution factors, that can be reused in future projects.

REFERENCES

ISO/IEC 9126 *Software engineering - Software Product quality*, 2001. International Organization for Standardization,

Karydas D.M., J.F. Gifun, 2006. *A method for the effcient prioritization of infrastructure renewal projects,* Reliability Engineering and System Safety.

Roy B., 1996. *Multicriteria Methodology for Decision Aiding*, Kluwer Academic Publishers.

Saaty T.L., 2001. *Models, methodes concepts and applications of the Analytic Hierarchy Process*, Kluwer academic publishers.

Trienekens J.J.M. and R.J. Kusters, 1999. *Identifying embedded software quality, two approaches*, Quality and Reliability Journal, Addison Wesley.

Weil R., G.E. Apostolakis, 2001. *A methodology for the prioritization of operating experience in nuclear power plants,* Reliability Engineering and System Safety.