Module Qualities

logo
TBD

Gerrit Muller
Embedded Systems Institute
Den Dolech 2 (Laplace Building 0.10) P.O. Box 513, 5600 MB Eindhoven The Netherlands
gerrit.muller@embeddedsystems.nl

Abstract

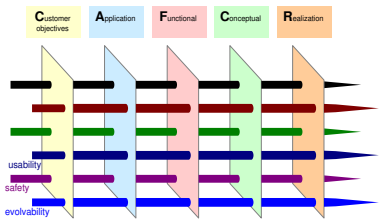
This module addresses the Qualities.

version: 0 status: draft 5th July 2004

Contents

Chapter 1

Qualities as Integrating Needles



1.1 Introduction

The 5 CAFCR views become more useful when the information in one view is used in relation with neighboring views. One of the starting points is the use of the stakeholder concerns. Many stakeholder concerns are abstracted in a large set of more generic qualities. These qualities are meaningful in every view in their own way. Figure 1.1 shows the qualities as cross cutting needles through the CAFCR views.

Section 1.2 shows an example of security as quality needle. In Section 1.3 a checklist of qualities is shown, with a definition of all qualities in the checklist.

1.2 Security as Example of a Quality Needle

As an example Figure 1.2 shows security issues for all the views. The green (upper) issues are the desired characteristics, specifications and mechanisms. The red issues are the threats to security. An excellent illustration of the security example can be found in [2].

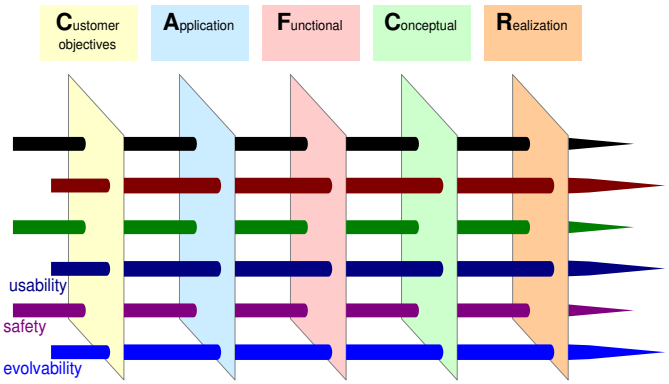


Figure 1.1: The quality needles are generic integrating concepts through the 5 CAFCR views

1.2.1 Customer Objectives View

A typical customer objective with respect to security is to keep sensitive information secure, in other words only a limited set of trusted people has access. The other people (non trusted) should not be able to see (or worse, to alter) this information.

1.2.2 Application View

The customer will perform many activities to obtain security: from selecting trustful people to appointing special guards and administrators who deploy a security policy. Such a policy will involve classifying people with respect to their need for information and their trustfulness, as well as classifying information according to the level of security. To recognize *trusted* people authentication is required by means of badges, passwords and in the future additional biometrics. Physical security by means of buildings, gates, locks, et cetera is also part of the security policy.

The security is threatened in many ways, from burglary to fraud, but also from simple issues like people forgetting their password and writing it on a yellow sticker. Social contacts of trusted people can unwillingly expose sensitive information, for instance when two managers are discussing business in a business lounge, while the competition is listening at the next table.

Unworkable procedures are a serious threat to security. For instance the forced change of passwords every month, resulting in many people writing down the password.

An interesting article is [1]. It shows how secret security procedures, in this case for passenger screening at airports, is vulnerable. It describes a method for terrorists how to reverse engineer the procedures empirically, which turns the effectiveness of the system from valuable to dangerous.

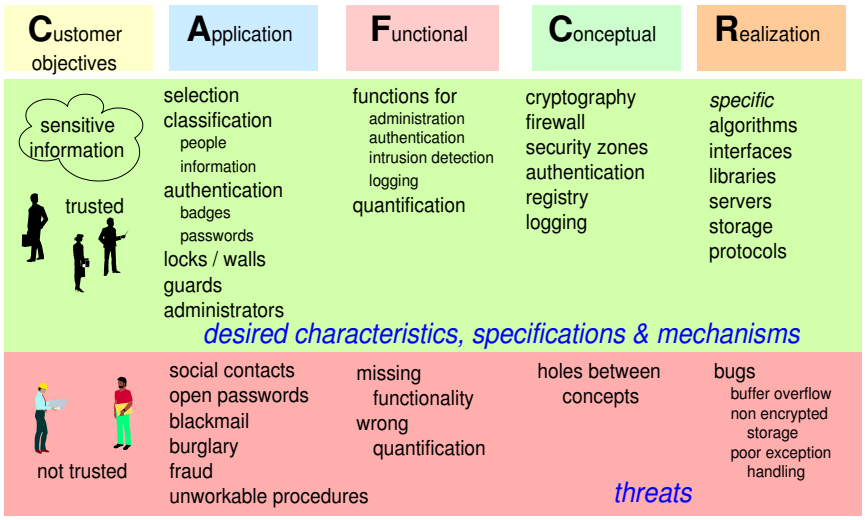


Figure 1.2: Example security through all views

1.2.3 Functional View

The system under consideration will have to fit in the customer's security. Functions for authentication and administration are required. The performance of the system needs to be expressed explicitly. For instance the required confidence level of encryption and the speed of authentication have to be specified.

Security threats are usually caused by missing functionality or wrong quantification. This threat will surface in the actual use, where the users will find work arounds that compromise the security.

1.2.4 Conceptual View

Many technological concepts have been invented to make systems secure, for example cryptography, firewalls, security zones, authentication, registry, and logging. Every concept covers a limited set of aspects of security. For instance cryptography makes stored or transmitted data non-interpretable for non-trusted people.

Problems in the conceptual view are usually due to the non-ideal combination of concepts. For instance cryptography requires keys. Authentication is used to access and validate keys. The interface between cryptography and authentication is a risky issue. Another risky issue is the transfer of keys. All interfaces between the concepts are suspicious areas, where poor design easily threatens the security.

1.2.5 Realization View

The concepts are realized in hardware and software with specific mechanisms, such as encryption algorithms and tamper free interfaces. These mechanisms can be implemented in libraries, running at a distributed computer infrastructure. Every specific hardware and software element involved in the security concepts in itself must be secure, in order to have a secure system.

A secure realization is far from trivial. Nearly all systems have bugs. The encryption algorithm may be applicable, but if the library implementation is poor then the overall security is still poor. Well known security related bugs are buffer overflow bugs, that are exploited by hackers to gain access. Another example is storage of very critical security data, such as passwords and encryption keys, in non encrypted form. In general exception handling is a source of security threats in security.

1.2.6 Conclusion

Security is a quality that is heavily determined by the customer's way of working (application view). To enable a security policy of the customer a well-designed and well-implemented system is required with security functionality fitting in this policy.

In practice the security policy of customers is a large source of problems. Heavy security features in the system will never solve such a shortcoming. Another common source of security problems is poor design and implementation, causing a fair policy to be corrupted by the non-secure system.

Note that a very much simplified description of security has been presented, with the main purpose of illustration. A real security description will be more extensive than described here.

1.3 Qualities Checklist

Figure 1.3 shows a large set of qualities that can be used as a checklist for architecting. This set is classified to ease the access to the list. The qualities are not independent nor orthogonal, so every classification is at its best a means not a goal.

The following sections describe the different qualities briefly, in the *functional view*. Note that every quality can in general be described in each of the views. For instance, if the system is a head end system for a cable operator, then the useability of the (head end) system describes in the functional view the useability of the system itself, while in the customer objectives view the useability deals with the cable operator services.

The descriptions below are not intended to be **the** definition. Rather the list is intended to be used as a checklist, i.e. as a means to get a more all round view on

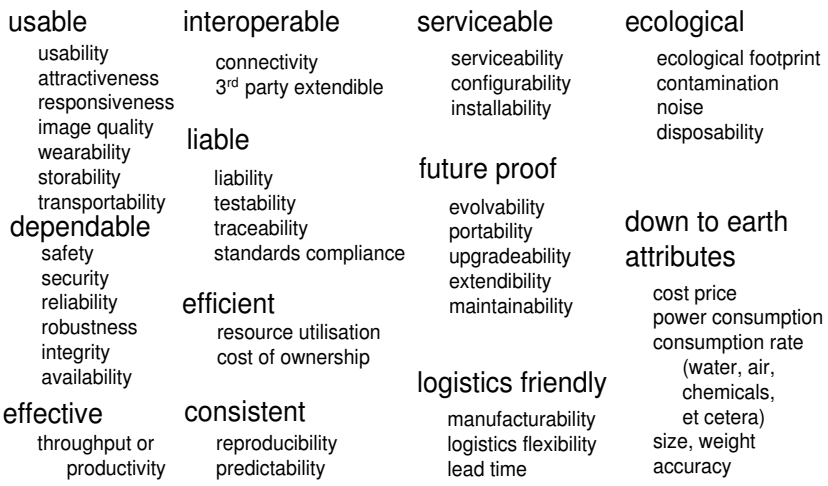


Figure 1.3: Checklist of qualities

the architecture.

1.3.1 Usable

useability The useability is a measure of usefulness and ease of use of a system.

attractiveness The appeal or attractiveness of the system.

responsiveness The speed of responding to inputs from outside.

image quality The quality of images (resolution, contrast, deformation, et cetera). This can be more generally used for output quality, so also sound quality for instance.

wearability The ease of wearing the system, or carrying the system around.

storability The ease of storing the system.

transportability The ease of transporting the system.

1.3.2 Dependable

safety The safety of the system. Note that this applies to all the stakeholders, for instance safety of the patient, operator, service employee, et cetera. Some people include the safety of the machine itself in this category. In my view this belongs to system reliability and robustness.

security The level of protection of the information in the system against unwanted access to the system.

reliability The probability that the systems operates reliable; the probability that the system is not broken and the software is not crashed. Here again the non-orthogonality of qualities is clear: an unreliable X-ray system is a safety risk when deployed for interventional surgery.

robustness The capability of the system to function in any (unforeseen) circumstances, including being foolproof for non-educated users.

integrity Does the system yield the *right* outputs.

availbility The availability of the system, often expressed in terms of (scheduled) uptime and the chance of unwanted downtime.

1.3.3 Effective

throughput or productivity The integral productivity level of the system. Often defined for a few use cases. Integral means here including aspects like start up shutdown, preventive maintenance, replacement of consumables et cetera. A bad attitude is to only specify the best case throughput, where all circumstances are ideal and even simple start up effects are ignored.

1.3.4 Interoperable

3^{rd} party extendable How open is the system for 3^{rd} party extensions? PCs are extremely open; many embedded systems are not extendable at all.

connectivity What other systems can be connected to the system and what applications are possible when connected?

1.3.5 Liable

liability The liability aspects with respect to the system; who is responsible for what, what are the legal liabilities, is the liability limited to an acceptable level?

testability The level of verifiability of the system, does the system perform as agreed upon?

traceability Is the operation of the system traceable? Traceability is required for determining liability aspects, but also for post mortem problem analysis.

standards compliance Large parts of the specification are defined in terms of compliance to standards.

1.3.6 Efficient

resource utilization The typical load of the system resources. Often specified for the same use cases as used for the productivity specification.

cost of ownership The cost of ownership is an integral estimate of all costs of owning and operating the system, including financing, personnel, maintenance, and consumables. Often only the sales price is taken as efficiency measure. This results in a suboptimal solution that minimize only the material cost.

1.3.7 Consistent

reproduceability Most systems are used highly repetitive. If the same operation is repeated over and over, the same result is expected all the time within the specified accuracy.

predictability The outcome of the system should be understandable for its users. Normally this means that the outcome should be predictable.

1.3.8 Serviceable

serviceability The ease of servicing the system: indication of consumable status, diagnostic capabilities in case of problems, accessibility of system internals, compatibility of replaceable units, et cetera.

configurability The ease of configuring (and maintaining, updating the configuration) the system

installability The ease of installing the system; for example the time, space and skills needed for installing.

1.3.9 Future Proof

evolvability The capability to change in (small) steps to adapt to new changing circumstances.

portability To be able to change the underlying platform, for instance from Windows NT to Linux, or from Windows 98SE to Windows XP.

upgradeability The capability of upgrading the entire or part of the system with improved features.

extendability The capability to add options or new features.

maintainability The capability of maintaining the well-being of the system, also under changing circumstances, such as end-of-life of parts or consumables, or new safety or security regulations.

1.3.10 Logistics Friendly

manufacturability The ease of manufacturing the system; for example time, space and skills needed for manufacturing.

logistics flexibility The capability to quickly adapt the logistics flow, for instance by fast ramp up (or down) supplier agreements, short lead times, low integration effort and second suppliers.

lead time The time between ordering the system and the actual delivery.

1.3.11 Ecological

ecological footprint The integral ecological load of the system, expressed in "original" ecological costs. This means that if electricity is used, the generation of electricity (and its inefficiency) is included in the footprint.

contamination The amount of contamination produced by the system

noise The (acoustical) noise produced by the system

disposability The way to get the system disposed, for instance the ability to decompose the system and to recycle the materials.

1.3.12 Down to Earth Attributes

These attributes (as the name indicates) are so trivial that no further description is given.

cost price

power consumption

consumption rate (water, air, chemicals, et cetera)

size, weight

accuracy

1.4 Summary

The qualities of a system can be generalized to the other CAFCR views. This generalization helps to understand the relationships between the views. Classification of the qualities is the basis for a checklist of qualities. This checklist is a tool for the architect: it helps the architect in determining the relevant qualities for the system to be created.

Bibliography

[1] Samidh Chakrabarti and Aaron Strauss. Carnival booth: An algorithm for defeating the computer-assisted passenger screening system. `http://swissnet.ai.mit.edu/6805/student-papers/spring02-papers/caps.htm`, 2002. Shows that security systems based on secret designs are more vulnerable and less secure.

[2] Charles C. Mann. Homeland insecurity. *The Atlantic Monthly*, pages 81–102, September 2002. Volume 290, No. 2T; Very nice interview with Bruce Schneier about security and the human factor.

[3] Gerrit Muller. The system architecture homepage. `http://www.extra.research.philips.com/natlab/sysarch/index.html`, 1999.

History

Version: 0, date: July 5, 2004 changed by: Gerrit Muller

- created module