Requirements Engineering: Quality Requirements / Classification of non-functional requirements

CECS 590

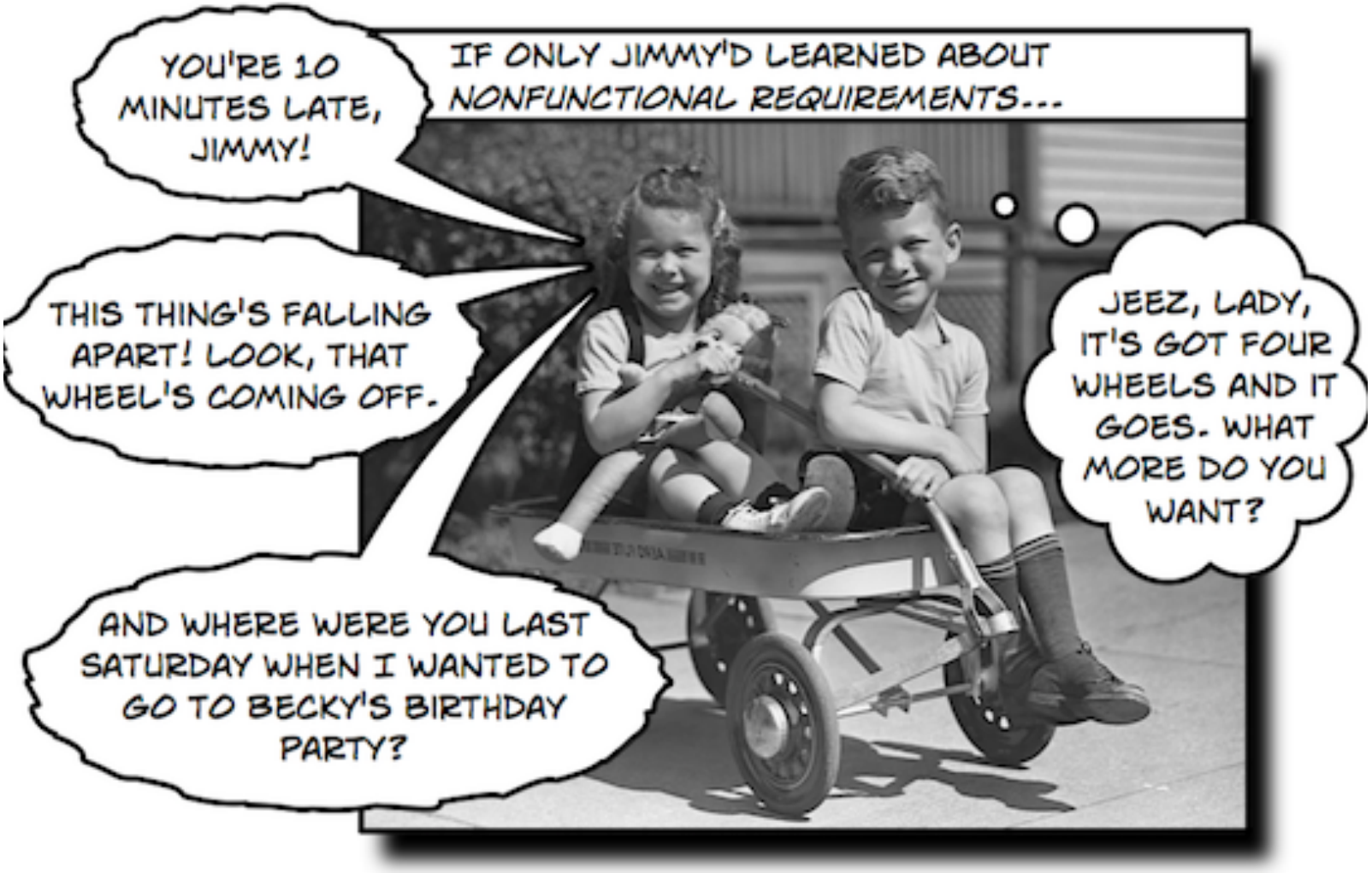
Recap time!

- Usage Model
  - What is it? How is it defined?
  - What is it for? Why do we need it?
  - What are the two major elements?
  - How are they defined? How to differentiate?

Requirements Engineering – Outline

- WHY do we need Requirements Engineering and what is it?
- Principles: Definitions, process, roles, problem/solution view, artifact orientation
- System Models: Decomposition and abstraction, system views
- Frameworks: What reference structures can I use for requirements?
- Business Case Analysis: Why are we building this system?
- Stakeholders: Who are the people to talk to about requirements?
- Goals and Constraints: What are the major objectives for the system?
- System Vision: What exactly do we want to achieve?
- Domain Models: What are the surrounding systems ours interacts with?
- Usage Models: How will the system interact with the user?
- Software quality models: How to determine the quality characteristics?
- **Quality requirements: How to specify which qualities need to be met?**
- Process requirements: How to specify constraints for development?
- Towards a system specification: How to hand over to design?
- Quality assurance: How to ensure that RE is done in a good way?
- Change management: How to evolve requirements?

Motivation


IF ONLY JIMMY'D LEARNED ABOUT NONFUNCTIONAL REQUIREMENTS...
YOU'RE 10 MINUTES LATE, JIMMY!
THIS THING'S FALLING APART! LOOK, THAT WHEEL'S COMING OFF.
AND WHERE WERE YOU LAST SATURDAY WHEN I WANTED TO GO TO BECKY'S BIRTHDAY PARTY?
JEEZ, LADY, IT'S GOT FOUR WHEELS AND IT GOES. WHAT MORE DO YOU WANT?

Classification of non-functional requirements

- Quality of a product
- Dimensions for the classification of requirements
- Examples
- Challenges with NFRs

K Rayker, stock.xchng

Which product is of higher quality?

„Quality is a complex and multifaceted concept.“

– *Garvin*

- Quality is often determined by not or hard to measure system characteristics
  („In the eye of the beholder.“)

→ **What are non-functional requirements?**

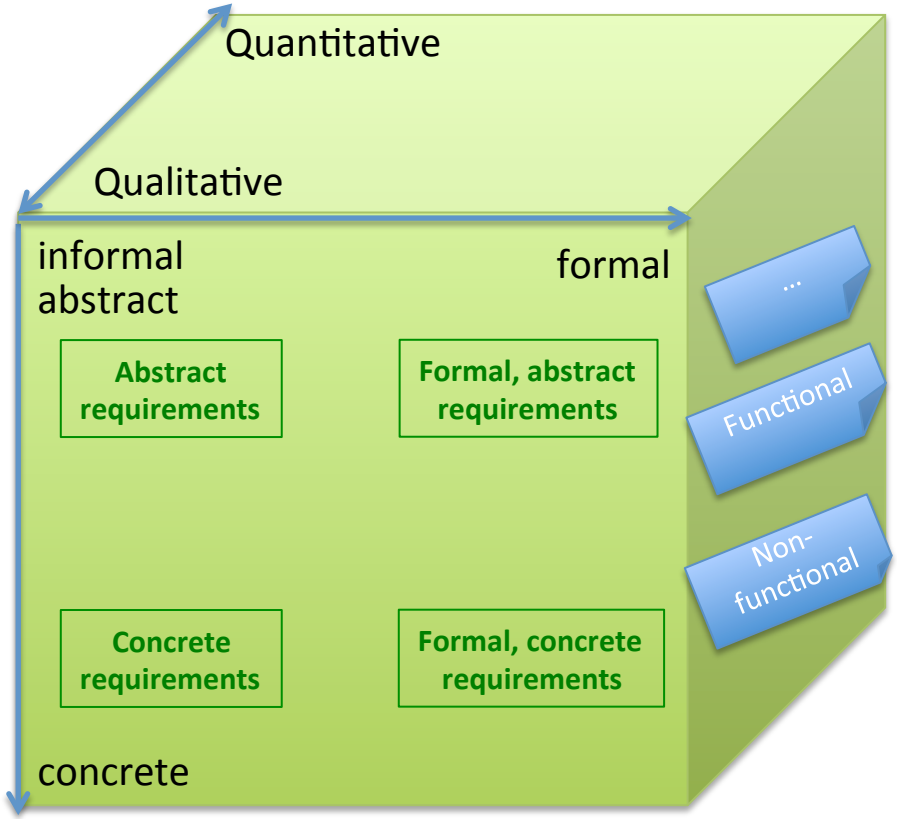
→ **Which interdependencies do non-functional requirements have with other classes of requirements?**

Classification of requirements: Dimensions

important

1. Formalisation
2. Degree of abstraction
3. Qualitative and quantitative
4. Functional, non-functional

→ Choice of degree of precision
- For making qualitative/quantitative statements
- For classifying requirements into functional/non-functional.



Example

- *The system is easy to use!*
  - Functional/qualitative implementation: *The system shall have a help function that provides the user with support at any time.*
  - Non-functional/quantitative: *The typical user understands the system after 10 minutes of training.*

A rough classification of requirements

- **Functional requirements:** All requirements refering to functionality (to functional behavior)
  - Qualitative: Which behavior is correct (compare Use Cases)
- **Non-functional requirements:** „Everything that is not functional“
  - Quantitative system characteristics: quality (realiability, performance, security, usability, adaptability, …)
  - Constraints for the implementation (programming language, operating system, off-the-shelf components, …)
  - Requirements w.r.t. the development process (process model, documentation, development standard, milestones, costs, …)
- **Furthermore:** Objectives with regard to marketing, rights, deployment and operating constraints, etc. …

Quality requirements (1st Classification)

Requirements to the quality characteristics of the system

- Quantitative characteristic w.r.t. the behavior and the functional usage („Quality in Use“)
  Examples:
  - Usability
  - Reliability
  - …
- Characteristics, that exceed the functional usage of the system (remaining „Product quality“)
  Examples:
  - Maintainability
  - Reusability
  - …

ATM
Examples?

Simple textual template

NFR	<description>
Rationale	<goal the requirement is derived from>
Satisfaction criterion	<metric that needs to be achieved>
Measurement	<how the metric will be measured/determined>
Risk	<in case this is requirement is not met>

Sarah's way at Intel: The landing zone

- "Our" NFR template
- Plus values for
  - Minimum
  - Target
  - Outstanding

NFR	<description>
Rationale	<goal the requirement is derived from>
Satisfaction criterion	<metric that needs to be achieved>
Measurement	<how the metric will be measured/determined>
Risk	<in case this is requirement is not met>

- Enables you to do tradeoffs in a matrix

Example: Vacation	Minimum / worst acceptb	Target	Outstanding
Length	3 days	10 days	1 month
Cost	$1000	$500	$250

Let‘s try that

NFR	<description>
Rationale	<goal>
Satisfaction criterion	<metric>
Measurement	<how metric is measured>
Risk	<in

Prepare example

Tradeoff

Criterion	Minimum / worst acceptb	T

Constraints (1st classification)

- (Project-specific) process requirements are requirements for the development process
  Examples:
  - Process model
  - Project plan (milestones, budget, deadlines)
  - Quality assurance (application of security norms)

- Implementation constraints and constraints and limitations for the implementation
  (Product Constraints)
  Examples:
  - Structure/Architecture,
  - Platforms and technologies
  - ...

Non-functional requirements in practice (Examples)

		Usability (NFR)	A system component shall save a users edits when ever possible.
)010PE01		Performance (NFR)	The perceived response time shall not be too high.

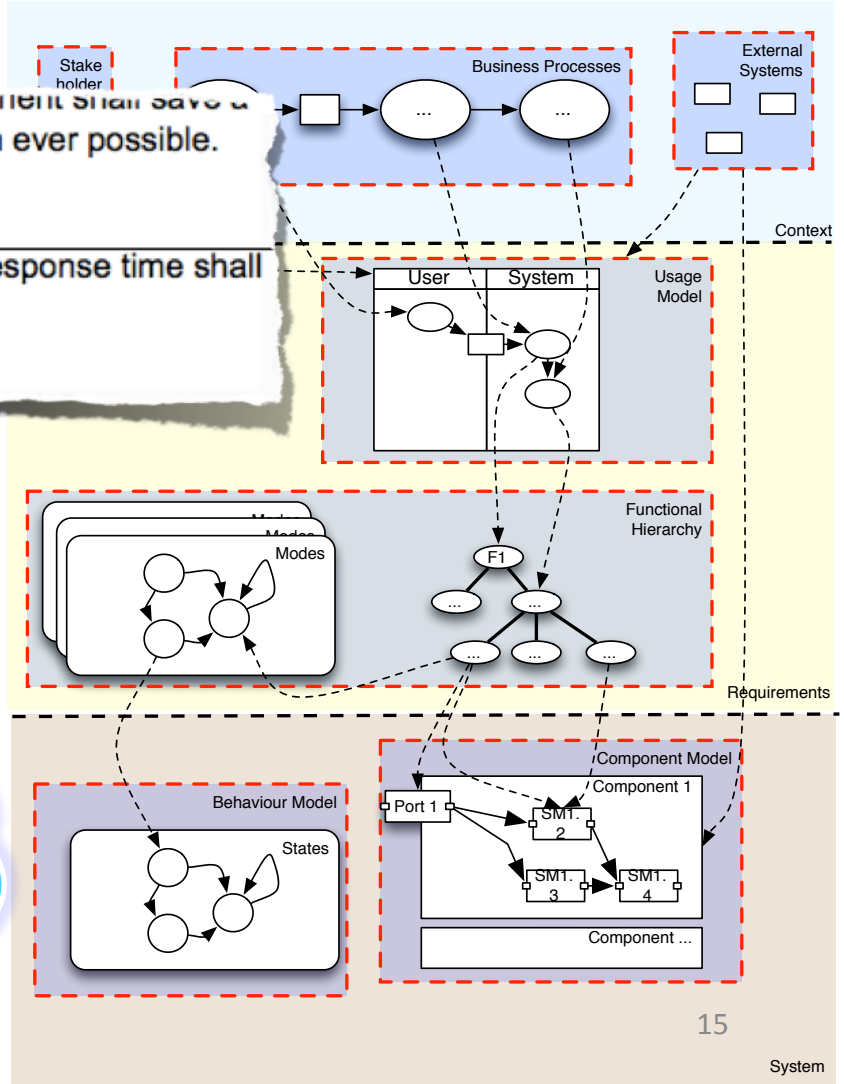
Which problems may arise with this requirement?

Non-functional requirements in practice (Examples)

		Usability (NFR)	A system component shall save a users edits when ever possible.
010PE01		Performance (NFR)	The perceived response time shall not be too high.

On which abstraction level does this requirement belong?

How could we specify this requirement in a measurable way?

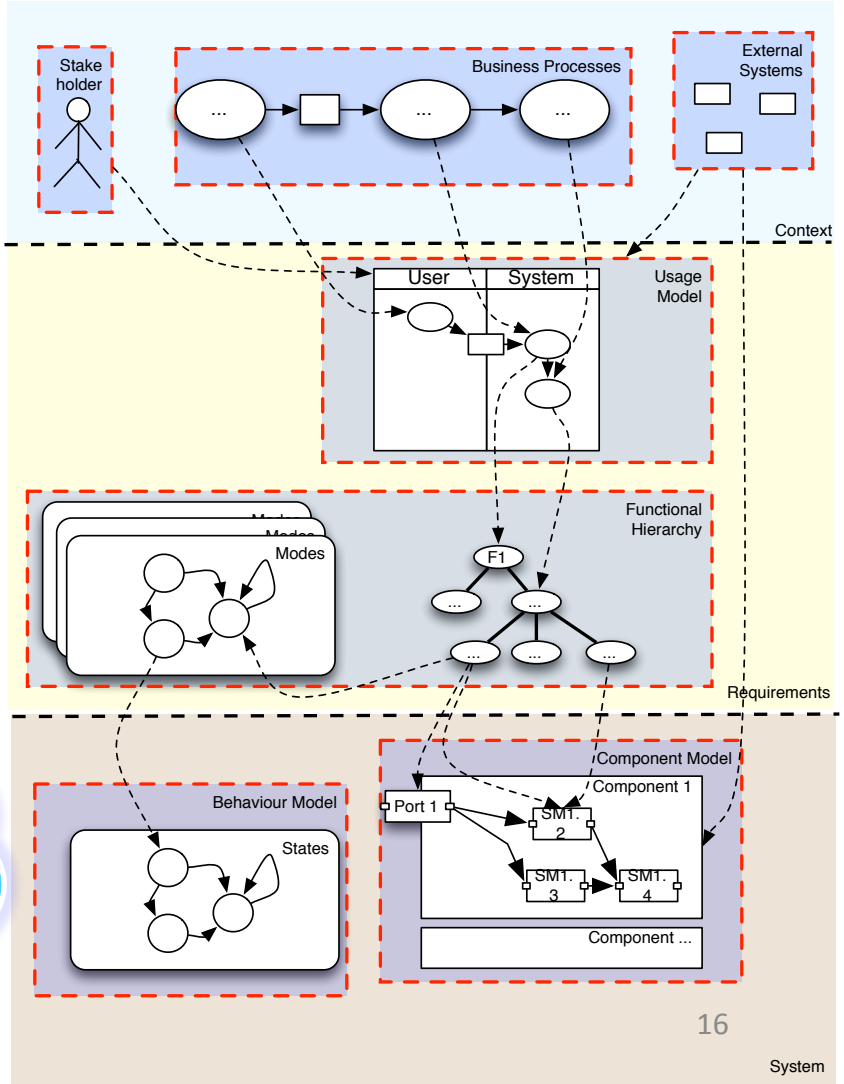


Non-functional requirements in practice (Examples)

"The system shall be maintainable."

On which abstraction level does this requirement belong?

How could we specify this requirement in a measurable way?



important

Summary: 3 challenges

1. Crosscutting Concerns: qualitative/quantitative statements
   - Partially w.r.t. functionality (e.g. performance)
   - Partially across functionality (e.g. maintainability)
2. Elicitation, assessment and evaluation
   - Often general wish w.r.t. characteristics, but not concrete
   - No statement to which extent the characteristic must be present
   - Requirements (and implementation) possible on different levels of abstraction with different concepts (and expressivity)
   - Abstraction levels influence modeling concepts and characteristics (as well as interdependencies), and therefore also the classification
3. Classification of non-functional requirements
   - Depends on the underlying quality model
   - Depends on the underlying system model and modeling concepts

Reading…

GUEST EDITOR'S INTRODUCTION

SOFTWARE QUALITY:

THE ELUSIVE TARGET

BARBARA KITCHENHAM, *National Computing Centre*
SHARI LAWRENCE PFLEEGER, *Systems/Software, Inc.*

If you are a software developer, manager, or maintainer, quality is often on your mind. But what do you really mean by software quality? Is your definition adequate? Is the software you produce better or worse than you would like it to be? In this special issue, we put software quality on trial, examining both the definition and evaluation of our software products and processes.

In the recent past, when bank statements contained errors or the telephone network broke down, the general public usually blamed "the computer," making no distinction between hardware and software. However, high-profile disasters and the ensuing debates in the press are alerting more people to the crucial nature of software quality in their everyday lives. Before long, we can expect increasing public concern about the pervasiveness of software, not only in public services but also in consumer products like automobiles, washing machines, telephones, and electric shavers. Consequently, we software professionals need to worry about the quality of all our products — from large, complex, stand-alone systems to small embedded ones.

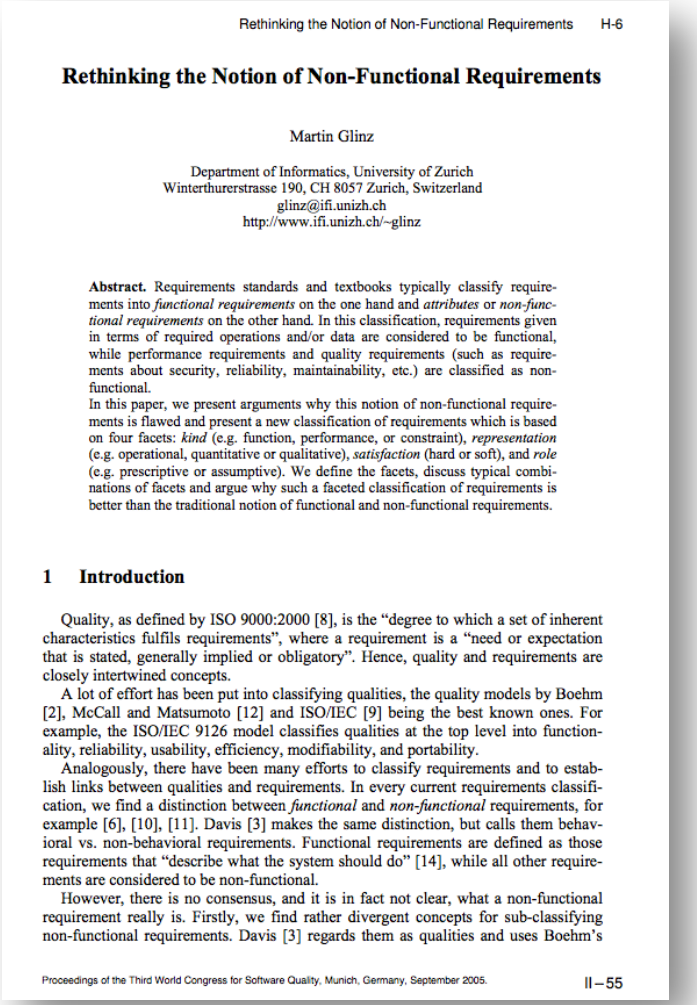
So how do we assess "adequate" quality in a software product? The context is important. Errors tolerated in word-processing software may not be acceptable in control software for a nuclear-power plant. Thus, we must reexamine the meanings of "safety-critical" and "mission-critical" in the context of software's contribution to the larger functionality and quality of products and businesses. At the same time, we must ask ourselves who is responsible for setting quality goals and making sure they are achieved.

WHAT DOES QUALITY REALLY MEAN?

Most of us are affected by the quality of the software we create because our organization's viability depends on it. And most software-related tools and methods — including those described in *IEEE Software* — claim to assess or im-

12 0740-7459/96/$05.00 © 1996 IEEE JANUARY 1996

Authorized licensed use limited to: T U MUENCHEN. Downloaded on July 10,2010 at 08:03:15 UTC from IEEE Xplore. Restrictions apply.



Rethinking the Notion of Non-Functional Requirements

Martin Glinz

Department of Informatics, University of Zurich
Winterthurerstrasse 190, CH 8057 Zurich, Switzerland
glinz@ifi.unizh.ch
http://www.ifi.unizh.ch/~glinz

Abstract. Requirements standards and textbooks typically classify requirements into *functional requirements* on the one hand and *attributes* or *non-functional requirements* on the other hand. In this classification, requirements given in terms of required operations and/or data are considered to be functional, while performance requirements and quality requirements (such as requirements about security, reliability, maintainability, etc.) are classified as non-functional.
In this paper, we present arguments why this notion of non-functional requirements is flawed and present a new classification of requirements which is based on four facets: *kind* (e.g. function, performance, or constraint), *representation* (e.g. operational, quantitative or qualitative), *satisfaction* (hard or soft), and *role* (e.g. prescriptive or assumptive). We define the facets, discuss typical combinations of facets and argue why such a faceted classification of requirements is better than the traditional notion of functional and non-functional requirements.

1 Introduction

Quality, as defined by ISO 9000:2000 [8], is the "degree to which a set of inherent characteristics fulfils requirements", where a requirement is a "need or expectation that is stated, generally implied or obligatory". Hence, quality and requirements are closely intertwined concepts.

A lot of effort has been put into classifying qualities, the quality models by Boehm [2], McCall and Matsumoto [12] and ISO/IEC [9] being the best known ones. For example, the ISO/IEC 9126 model classifies qualities at the top level into functionality, reliability, usability, efficiency, modifiability, and portability.

Analogously, there have been many efforts to classify requirements and to establish links between qualities and requirements. In every current requirements classification, we find a distinction between *functional* and *non-functional* requirements, for example [6], [10], [11]. Davis [3] makes the same distinction, but calls them behavioral vs. non-behavioral requirements. Functional requirements are defined as those requirements that "describe what the system should do" [14], while all other requirements are considered to be non-functional.

However, there is no consensus, and it is in fact not clear, what a non-functional requirement really is. Firstly, we find rather divergent concepts for sub-classifying non-functional requirements. Davis [3] regards them as qualities and uses Boehm's

Proceedings of the Third World Congress for Software Quality, Munich, Germany, September 2005. II – 55