Non-Functional Requirements

Gregor v. Bochmann, University of Ottawa

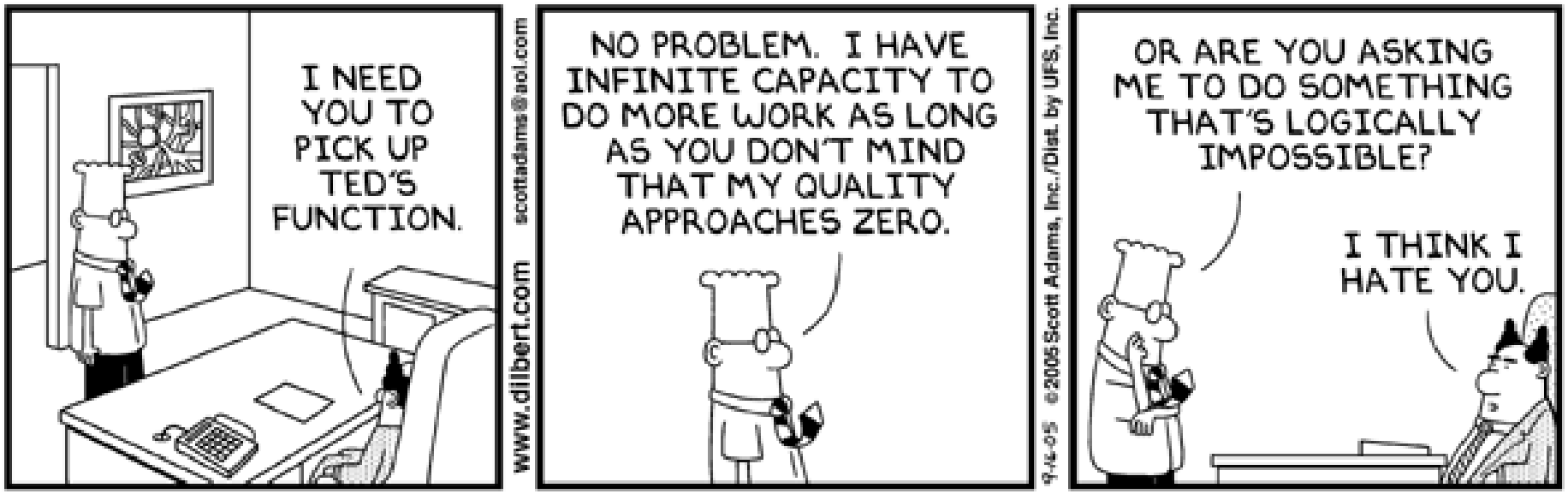
Based on Powerpoint slides by Gunter Mussbacher (2009)
with material from:
Jo Atlee, Dan Berry (both University of Waterloo); R. Pressman;
D. Damian; Amyot 2008, Somé 2008

Table of Contents

- Non-Functional Requirements and Software Quality Attributes
  - Software Quality
  - Classifications of Non-Functional Requirements
  - Quality Measures

- To measure is to know. If you can not measure it, you can not improve it.[1]

[1] Lord Kelvin (1824 - 1907)



© Scott Adams, Inc./Dist. by UFS, Inc.

Software Quality (1)

- Most definitions require compliance with requirements
- "Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software."[1]
- Implication:
  - We need to be able to explicitly quantify requirements and verify that any solution meets them
  - We need measures

[1] Pressman, 1997

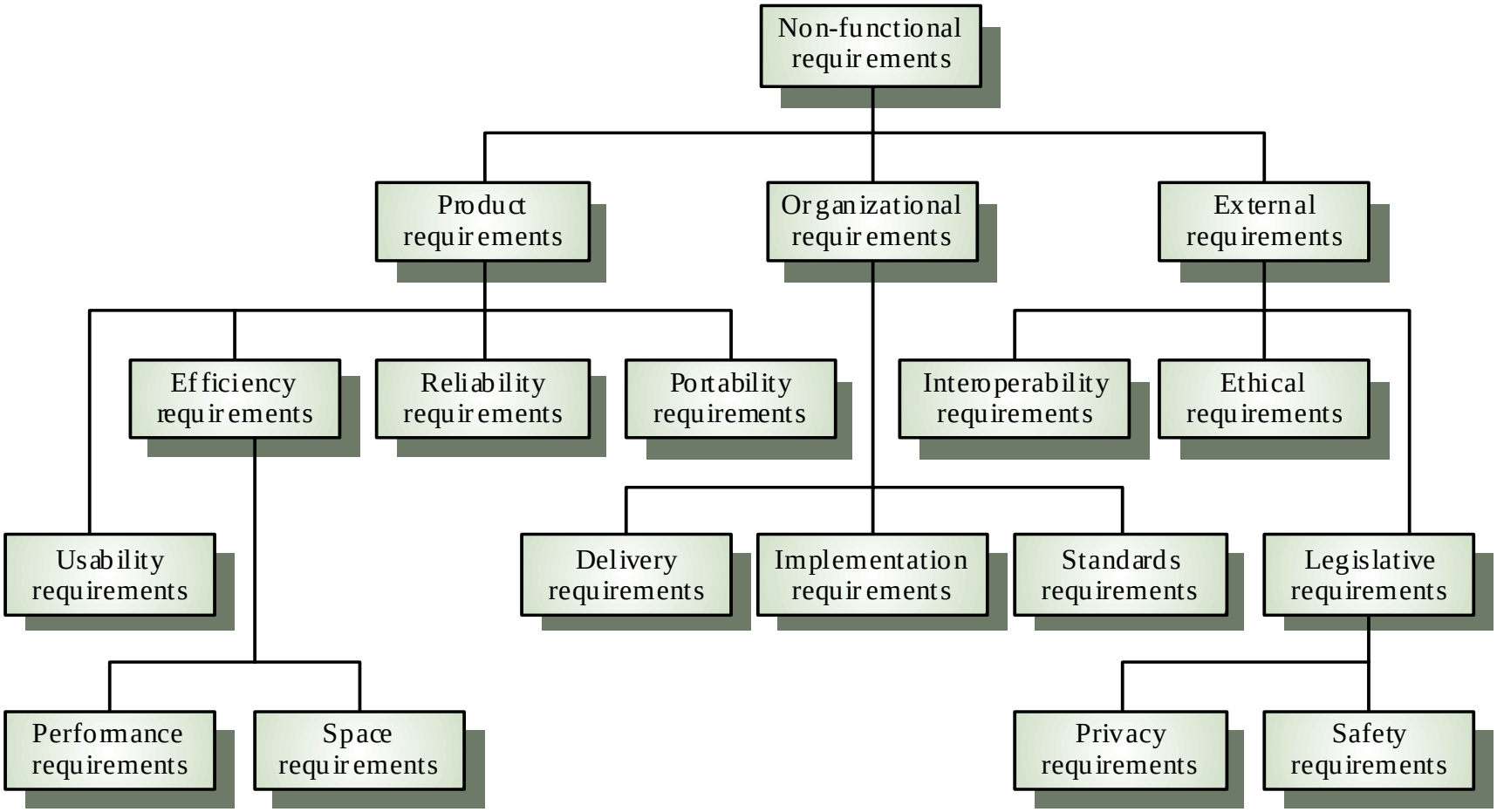
Software Quality (2)

- An interesting phenomenon:

Measurable objectives are usually achieved!

- Therefore, unless you have unrealistic values, requirements are usually met
- Important to know what measures exist!
- The chosen values, however, will have an impact on the amount of work during development as well as the number of alternatives and architectural designs from which developers may choose to meet the requirements

Types of Non-Functional Requirements (NFRs)



Source: Gerald Kotonya and Ian Sommerville, Requirements Engineering – Processes and Techniques, Wiley, 1998

Other Classification of NFRs (1)

- Product-oriented attributes
  - Performance : (a) response time, (b) throughput (number of operations performed per second)
  - Usability: effort required to learn, use, provide input and interpret results of a program
  - Efficiency: minimal use of resources (memory, processor, disk, network…)
  - Reliability: of computations, precision
  - Security
  - Robustness: in the presence of faults, stress, invalid inputs…
  - Adaptability: to other environments or problems
  - Scalability: for large number of users or quantities of data
  - Cost: total cost of ownership (TCO) for acquisition, installation, use, disposal

Other Classification of NFRs (2)

- Product family-oriented attributes
  - Portability: does it work for several platforms
  - Modifiability: addition of new functionalities
  - Reusability: of components, code, designs, and even requirements in other systems

  - They are frequently demanded by developers to:
    - Reduce development costs
    - Increase revenues by creating several versions derived from a product or by personalizing it

Other Classification of NFRs (3)

- Process-oriented attributes
  - Maintainability: changes to functionalities, repairs
  - Readability: of code, documents
  - Testability: ease of testing and error reporting
  - Understandability: of design, architecture, code
  - Integrability: ability to integrate components
  - Complexity: degree of dependency and interaction between components

Yet Another Classification of NFRs[1]

Acquisition	User concern	Quality attribute
Performance – how well does it function?	How well does it utilize a resource? How secure is it? What confidence can be placed in what it does? How well will it perform under adverse conditions? How easy is it to use it?	Efficiency Integrity Reliability Survivability Usability
Design – how valid is the design?	How well does it conform to requirements? How easy is it to repair? How easy is it to verify its performance?	Correctness Maintainability Verifiability
Adaptation – how adaptable is it?	How easy is it to expand or upgrade its capability or performance? How easy is it to change? How easy is it to interfere with another system? How easy is it to transport? How easy is it to convert for use with another application?	Expandability Flexibility Interoperability Portability Reusability

Note: It iws surprising that response time and throughput are not mentioned under Performance

[1] Damian, 2005

Quantification

- Non-functional requirements need to be measurable
  - Avoid subjective characterization: good, optimal, better...
- Values are not just randomly specified
  - Must have a rational
  - Stakeholder must understand trade-offs
  - Important to rank and prioritize the requirements
- Precise numbers are unlikely to be known at the beginning of the requirement process
  - Do not slow down your initial elicitation process
  - Ensure that quality attributes are identified
  - Negotiate precise values later during the process

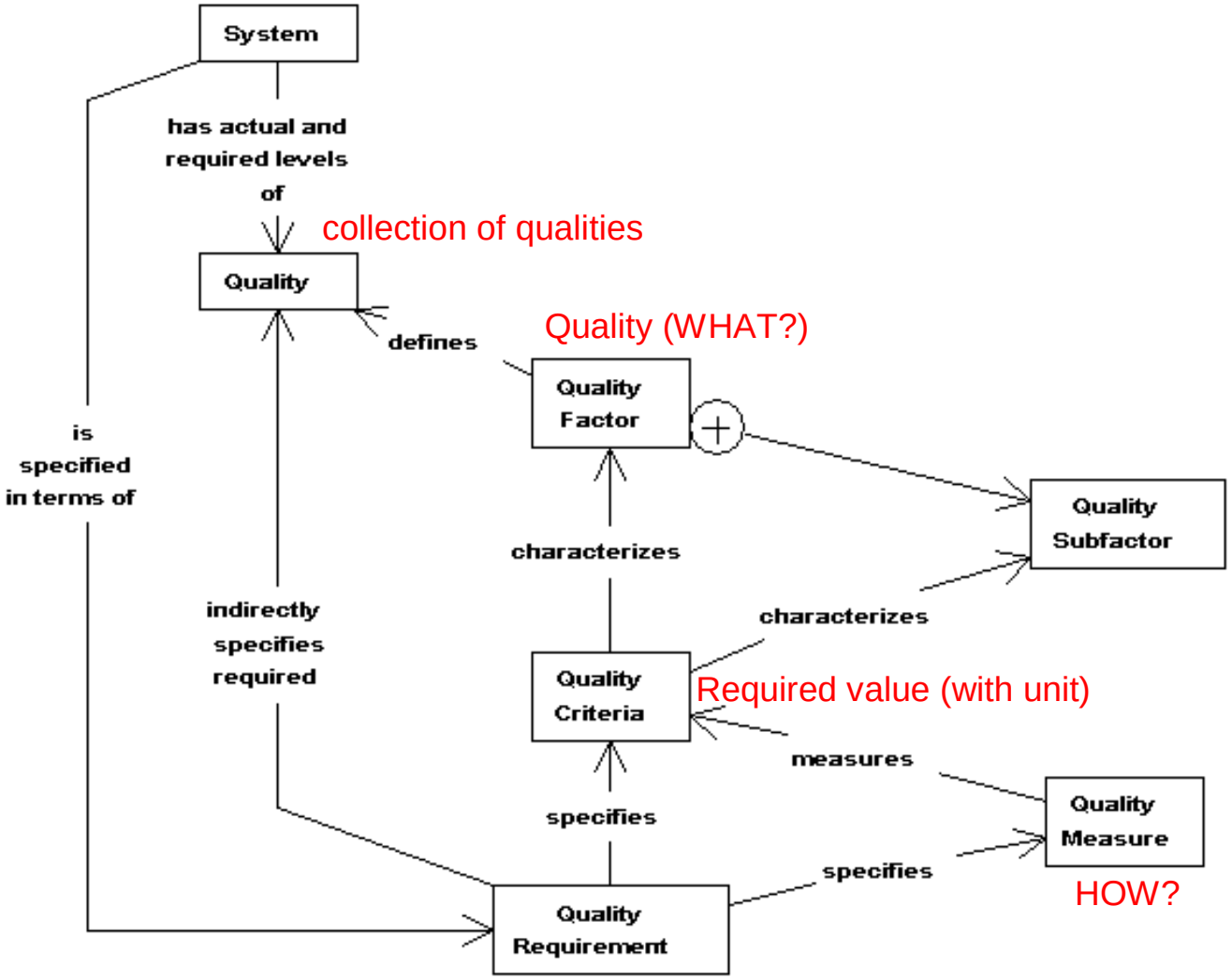
Measures vs. Metrics

- We use measures in a generic way but there is actually a distinction between measures and metrics

- For example, consider reliability
  - Metric: mean time between failures
  - Measure: number of failures in a period of time (an observation!)

- Reading the text on Wikipedia about software and performance metrics, I get the impression that metrics and measure mean the same thing. To define a measure, you have to define WHAT you measure (that is, the quality), the metric units used with the measurement values, and HOW you measure what is measured. For the example of Reliability above, the first line defines the quality (WHAT?), and the second lines defines a measurement method (HOW?). – G.v. B.

Some Relationships



Source: D. Firesmith, http://www.jot.fm/issues/issue_2003_09/column6/

Performance Measures (1)

- Lots of measures
  - Response time, number of events processed/denied in some interval of time, throughput, capacity, usage ratio, jitter, loss of information, latency...
  - Usually with probabilities, confidence interval
- Can be modeled and simulated (mainly at the architectural level) – performance prediction
  - Queuing model (LQN), process algebra, stochastic Petri nets
  - Arrival rates, distributions of service requests
  - Sensitivity analysis, scalability analysis

Performance Measures (2)

- Examples of performance requirements
  - The system shall be able to process 100 payment transactions per second in peak load.
  - In standard workload, the CPU usage shall be less than 50%, leaving 50% for background jobs.
  - Production of a simple report shall take less than 20 seconds for 95% of the cases.
  - Scrolling one page up or down in a 200 page document shall take at most 1 second.

Reliability Measures (1)

- Measure degree to which the system performs as required
  - Includes resistance to failure
  - Ability to perform a required function under stated conditions for a specified period of time
  - Very important for critical, continuous, or scientific systems
- Can be measured using
  - Probability that system will perform its required function for a specified interval under stated conditions
  - Mean-time to failure
  - Defect rate
  - Degree of precision for computations

Reliability Measures (2)

- Examples
  - The precision of calculations shall be at least $1/10^6$.
  - The system defect rate shall be less than 1 failure per 1000 hours of operation.
  - No more than 1 per 1000000 transactions shall result in a failure requiring a system restart.

Availability Measures (1)

- Definition: Percentage of time that the system is up and running correctly
- Can be calculated based on Mean-Time to Failure (MTBF) and Mean-Time to Repair (MTTR)
  - MTBF : Length of time between failures
  - MTTR : Length of time needed to resume operation after a failure
  - Availability = MTBF/(MTBF+MTTR)
- May lead to architectural requirements
  - Redundant components (lower MTBF)
  - Modifiability of components (lower MTTR)
  - Special types of components (e.g., self-diagnostic)
- Measurement: The mean time to failure and mean time to repair of critical components must be identified (typically measured) or estimated
- Modeling reliability and availability: e.g. Markov models

Availability Measures (2)

- Examples
  - The system shall meet or exceed 99.99% uptime.
  - The system shall not be unavailable more than 1 hour per 1000 hours of operation.
  - Less than 20 seconds shall be needed to restart the system after a failure 95% of the time. (This is a MTTR requirement)

Availability	Downtime
90%	36.5 days/year
99%	3.65 days/year
99.9%	8.76 hours/year
99.99%	52 minutes/year
99.999%	5 minutes/year
99.9999%	31 seconds/year

Security Measures (1)

There are at least two measures:

1. The ability to resist unauthorized attempts at usage
2. Continue providing service to legitimate users while under denial of service attack (resistance to DoS attacks)

- Measurement methods:
  - Success rate in authentication
  - Resistance to known attacks (to be enumerated)
  - Time/efforts/resources needed to find a key (probability of finding the key)
  - Probability/time/resources to detect an attack
  - Percentage of useful services still available during an attack
  - Percentage of successful attacks
  - Lifespan of a password, of a session
  - Encryption level

Security Measures (2)

- May lead to architectural requirements
  - Authentication, authorization, audit
  - Detection mechanisms
  - Firewall, encrypted communication channels
- Can also be modeled (logic ...)

- Examples of requirements
  - The application shall identify all of its client applications before allowing them to use its capabilities.
  - The application shall ensure that the name of the employee in the official human resource and payroll databases exactly matches the name printed on the employee's social security card.
  - At least 99% of intrusions shall be detected within 10 seconds.

Usability Measures (1)

In general, concerns ease of use and of training end users. The following more specific measures can be identified:

- Learnability
  - Proportion of functionalities or tasks mastered after a given training time
- Efficiency
  - Acceptable response time
  - Number of tasks performed or problems resolved in a given time
  - Number of mouse clicks needed to get to information or functionality
- Memorability
  - Number (or ratio) of learned tasks that can still be performed after not using the system for a given time period
- Error avoidance
  - Number of error per time period and user class
  - Number of calls to user support

Usability Measures (2)

- Error handling
  - Mean time to recover from an error and be able to continue the task
- User satisfaction
  - Satisfaction ratio per user class
  - Usage ratio
- Examples
  - Four out of five users shall be able to book a guest within 5 minutes after a 2-hour introduction to the system.
  - Novice users shall perform tasks X and Y in 15 minutes. Experienced users shall perform tasks X and Y in 2 minutes.
  - At least 80% of customers polled after a 3 months usage period shall rate their satisfaction with the system at 7 and more on a scale of 1 to 10.

Maintainability Measures (1)

- Measures ability to make changes quickly and cost effectively
  - Extension with new functionality
  - Deleting unwanted capabilities
  - Adaptation to new operating environments (portability)
  - Restructuring (rationalizing, modularizing, optimizing, creating reusable components)
- Can be measured in terms of
  - Coupling/cohesion metrics, number of anti-patterns, cyclomatic complexity
  - Mean time to fix a defect, mean time to add new functionality
  - Quality/quantity of documentation
- Measurement tools
  - code analysis tools such as IBM Structural Analysis for Java (http://www.alphaworks.ibm.com/tech/sa4j)

Maintainability Measures (2)

- Examples of requirements
  - Every program module must be assessed for maintainability according to procedure xx. 70% must obtain “highly maintainable” and none “poor”.
  - The cyclomatic complexity of code must not exceed 7.
    No method in any object may exceed 200 lines of code.
  - Installation of a new version shall leave all database contents and all personal settings unchanged.
  - The product shall provide facilities for tracing any database field to places where it is used.

Testability Measures

Measures the ability to detect, isolate, and fix defects

- Time to run tests
- Time to setup testing environment (development and execution)
- Probability of visible failure in presence of a defect
- Test coverage (requirements coverage, code coverage…)

- May lead to architectural requirements
  - Mechanisms for monitoring
  - Access points and additional control

- Examples
  - The delivered system shall include unit tests that ensure 100% branch coverage.
  - Development must use regression tests allowing for full retesting in 12 hours.

Portability Measures

Measure ability of the system to run under different computing environments

- Hardware, software, OS, languages, versions, combination of these

- Can be measured as
  - Number of targeted platforms (hardware, OS…)
  - Proportion of platform specific components or functionality
  - Mean time to port to a different platform

- Examples
  - No more than 5% of the system implementation shall be specific to the operating system.
  - The meantime needed to replace the current Relational Database System with another Relational Database System shall not exceed 2 hours. No data loss should ensue.

Integrability and Reusability Measures

Integrability

- Measures ability to make separated components work together
- Can be expressed as
  - Mean time to integrate with a new interfacing system

Reusability

- Measures ability that existing components can be reused in new applications
- Can be expressed as
  - Percentage of reused requirements, design elements, code, tests…
  - Coupling of components
  - Degree of use of frameworks

Robustness Measures

Measure ability to cope with the unexpected

- Percentage of failures on invalid inputs
- Degree of service degradation
  - Minimum performance under extreme loads
  - Active services in presence of faults
  - Length of time for which system is required to manage stress conditions

- Examples
  - The estimated loss of data in case of a disk crash shall be less than 0.01%.
  - The system shall be able to handle up to 10000 concurrent users when satisfying all their requirements and up to 25000 concurrent users with browsing capabilities.

Domain-specific Measures

The most appropriate quality measures may vary from one application domain to another, e.g.:

- Performance
  - Web-based system:
    Number of requests processed per second
  - Video games:
    Number of 3D images per second
- Accessibility
  - Web-based system:
    Compliance with standards for the blind
  - Video games:
    Compliance with age/content ratings systems (e.g., no violence)

Other Non-Functional Requirements

- What about NFRs such as “fun” or “cool” or “beautiful” or “exciting”?
- How can these be measured?

- The lists of existing quality attributes are interesting but they do not include all NFRs.
- It is sometimes better to let customers do their brainstorming before proposing the conventional NFR categories.

- In any case, we must also refine those goals into measurable requirements.