Non-Functional Requirements and Quality Measures

Daniel Amyot, University of Ottawa

Based on Powerpoint slides by Gunter Mussbacher (2009)
with material from:
J. Atlee, D. Berry, R. Pressman, D. Damian, S. Somé, A. Mavin, J. Eckhardt

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 cannot measure it, you cannot improve it.[1]

[1] Lord Kelvin (1824 - 1907)

Software Quality (1)

- Software quality goes beyond functional requirements
- 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 hence need *measures*

[1] Pressman, 1997

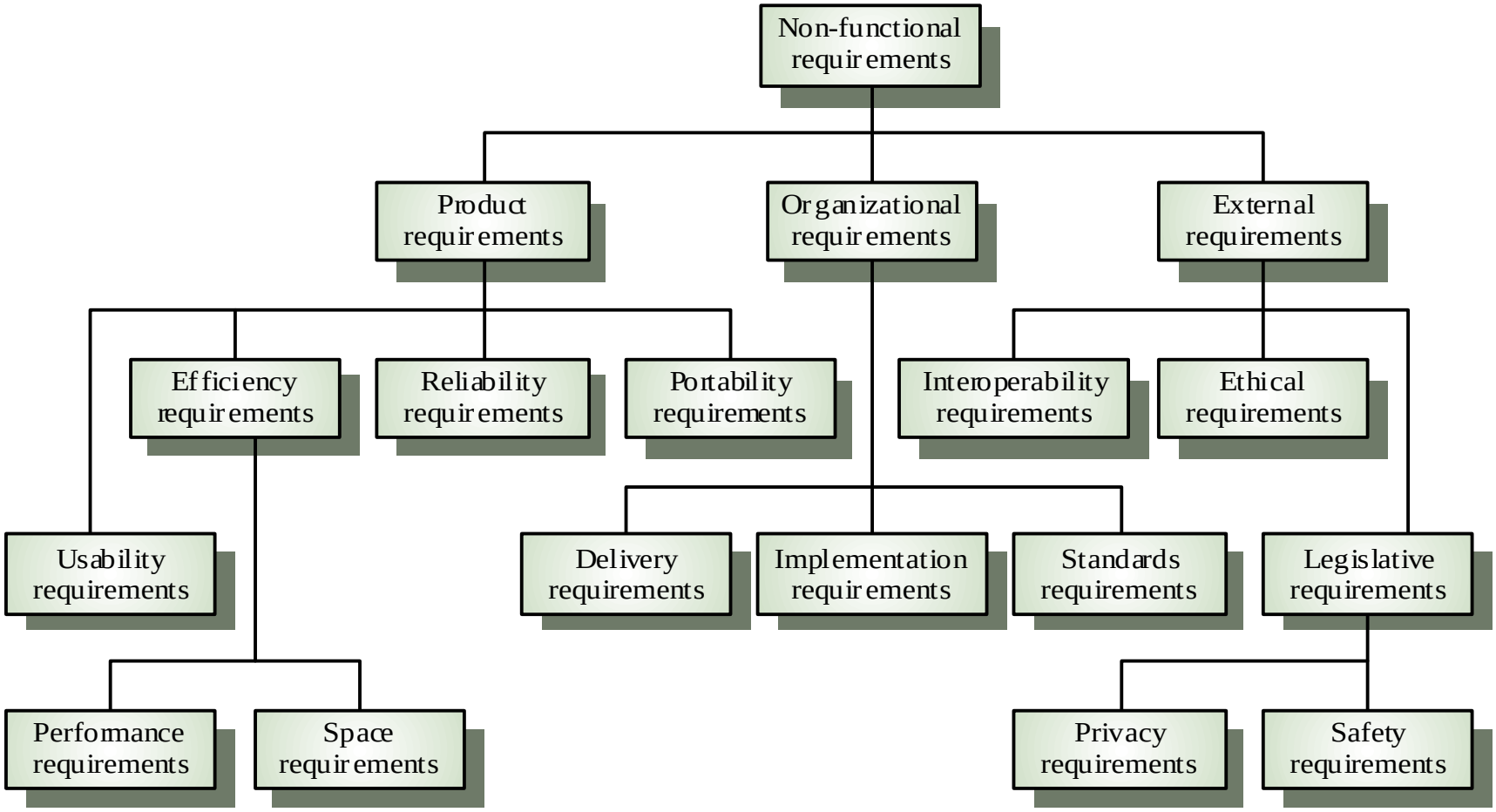
Software Quality (2)

- 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)

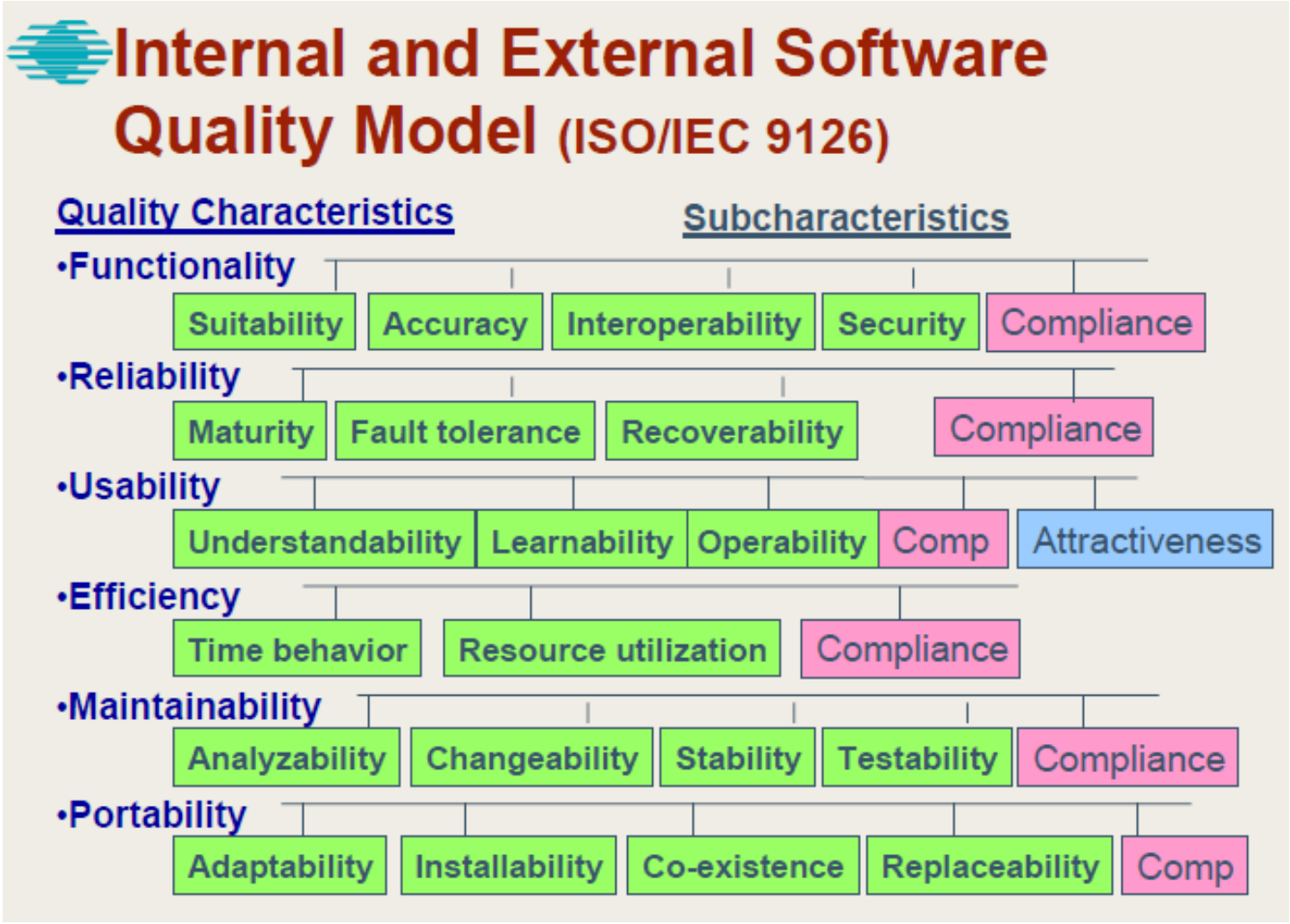


Source: Gerald Kotonya and Ian Sommerville, Requirements Engineering – Processes and Techniques, Wiley, 1998

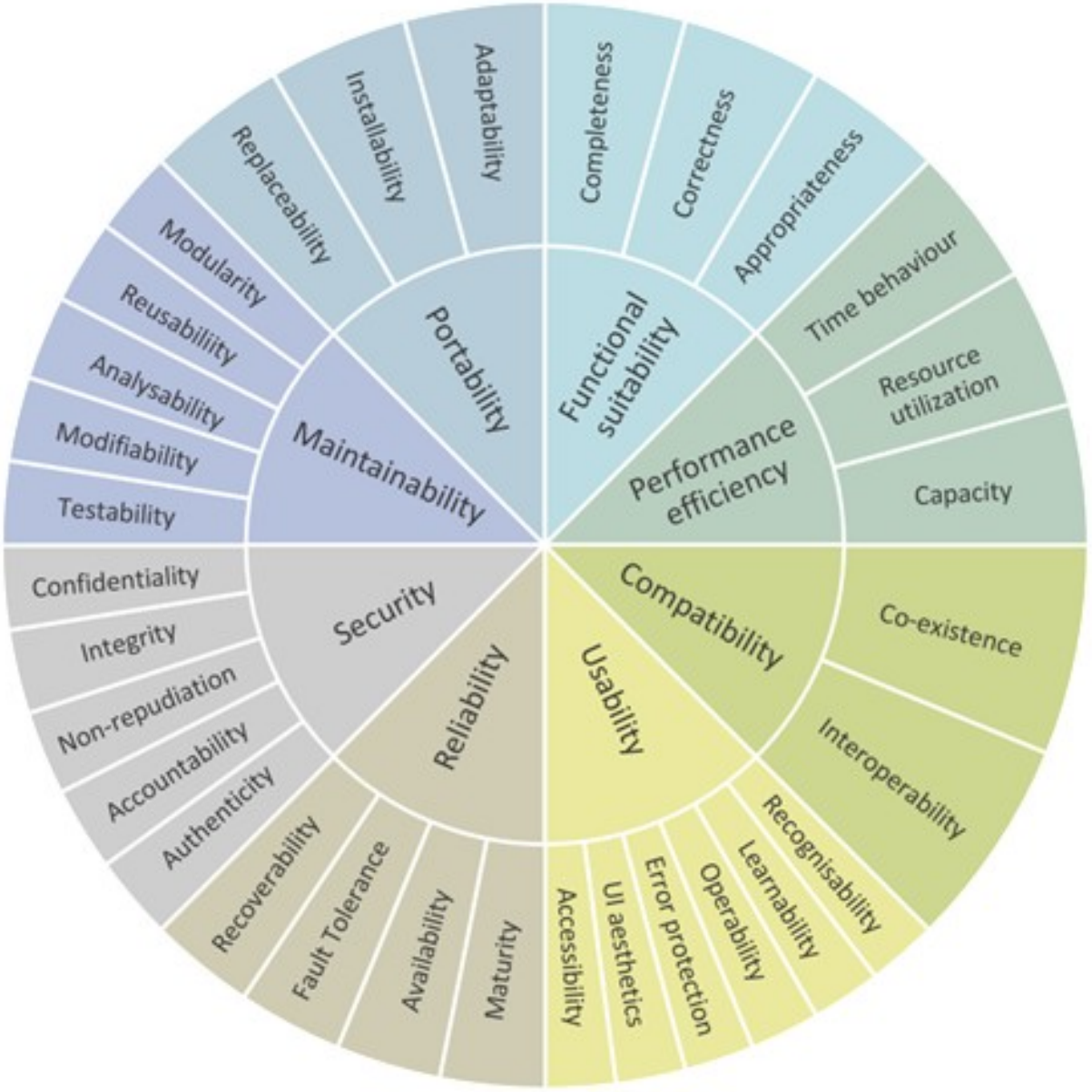
Interesting Resources on NFRs

- ISO/IEC 9126 Software engineering — Product quality
  - http://en.wikipedia.org/wiki/ISO/IEC_9126
  - Now superceeded
- ISO/IEC 25000:2014 — Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE)
  - https://www.iso.org/standard/64764.html
- "ilities"
  - http://en.wikipedia.org/wiki/Ilities
- *Software Quality*, and how to measure it
  - http://en.wikipedia.org/wiki/Software_quality

ISO/IEC 9126 Qualities



ISO/IEC 25000 Qualities



3). Non-Functional requirements

Quantification

- Non-functional requirements need to be measurable
  - Avoid subjective characterization: good, optimal, better...
- Values are not just randomly specified
  - Must have a rationale (e.g., why 25% less time instead of 20%, 30%?)
  - Stakeholder must understand goals and trade-offs
- 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
  - Measures are often used to talk about simple quantified observations
  - Metrics are often derived from simple measures

- For example, consider performance:
  - Measure: 500 transactions in one hour
  - Measure: 10 visitors in one hour
  - Metric: Number of transactions per visitor
    - 50 transactions/visitor in the above case

Confusion Matrix: True/False Positives/Negatives

		Condition (as determined by "Gold standard")	
	Total population	Condition positive	Condition negative
Test outcome	Test outcome positive	?	
	Test outcome negative		

[http://en.wikipedia.org/wiki/Sensitivity_and_specificity]

NFR Challenges

- The principal challenge with NFRs is to
  - Select relevant NFRs that can impact decisions during system development
  - **Select relevant metrics (with units) for these NFRs**
  - Determine realistic quantities
  - Assess and specify confidence levels in these quantities

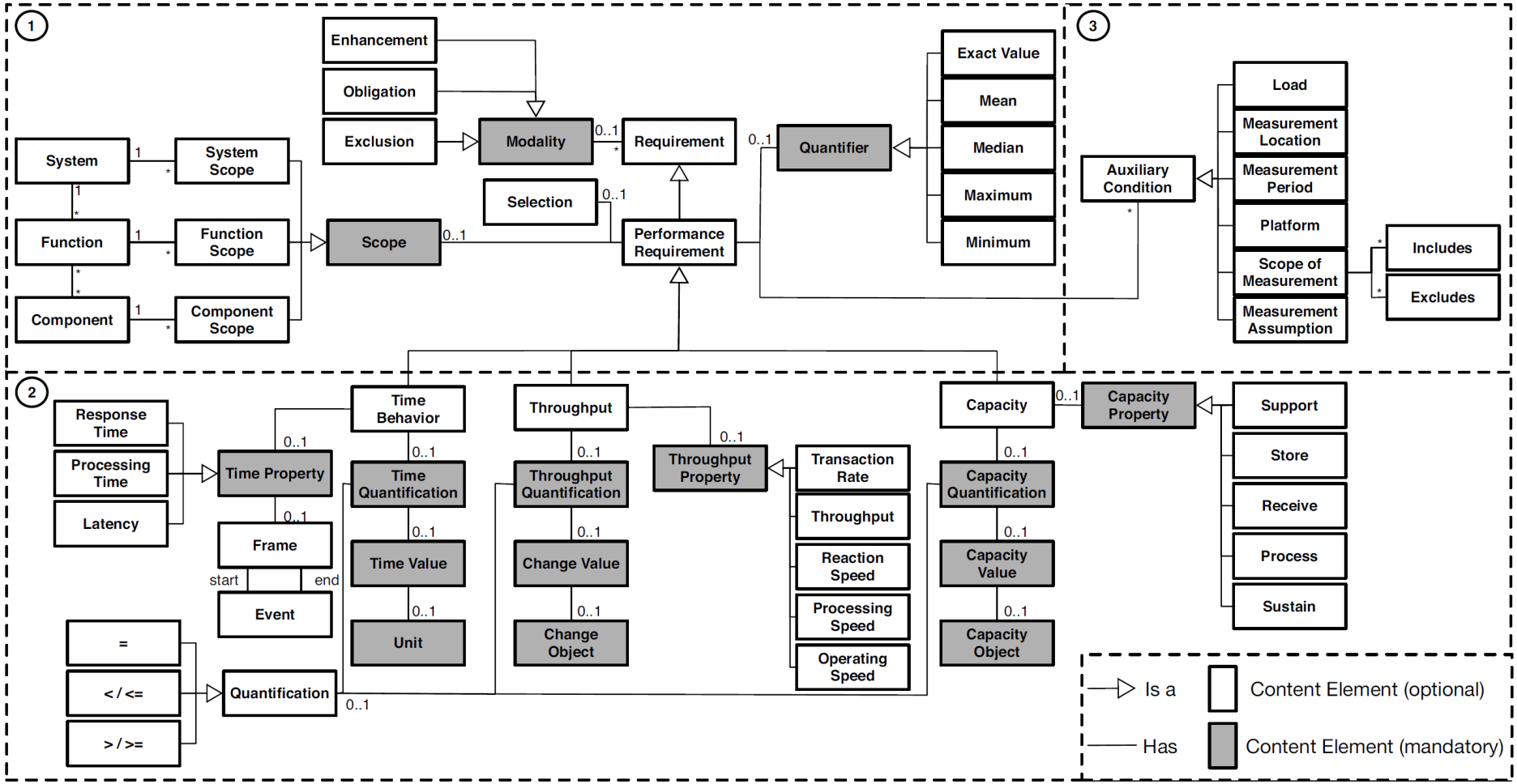
Performance Measures (1)

- 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
- Some can be modeled and simulated (mainly at the architectural level) – performance prediction
  - Queueing model (LQN), process algebra, stochastic Petri nets
  - Arrival rates, distributions of service requests
  - Sensitivity analysis, scalability analysis
  - Extensibility (what if we added this resource…)

Performance Measures (2) - Examples

- In case of peak load, the system shall have a minimal transaction rate of 100 payments per second.
- In standard workload, the CPU usage shall be less than 50%.
- Production of a simple report shall take less than 20 seconds for at least 95% of the cases.
- Scrolling one page in a 200 page document shall take at most 250 milliseconds.

Performance Measures (3) Metamodel



J. Eckhardt et al., Challenging Incompleteness of Performance Requirements by Sentence Patterns. *2016 IEEE 24th International Requirements Engineering Conference*, 46-55, IEEE CS, 2016

Performance Measures (4) Patterns

[In <x>% of all cases]	The system	must/shall	have	a		of	ms		
[In case of <x>]	The <operation>	could/may		a mean	response time	of <	s	[between event <A> and event <B>]	
	The <component>	must not		a median	processing time	of <=	min		<x>
				a maximal	latency	of >	h		
				a minimal		of >=			

(a) Sentence Patterns for Time Behavior Requirements

[In <x>% of all cases]	The system	must/shall	have	a	transaction rate	of	items		ms	
[In case of <x>]	The <operation>	could/may		a mean	throughput	of <	orders		s	
	The <component>	must not		a median	reaction speed	of <=	requests	per	min	<x>
				a maximal	processing speed	of >	<data>		h	
				a minimal	operating speed	of >=				

(b) Sentence Patterns for Throughput Requirements

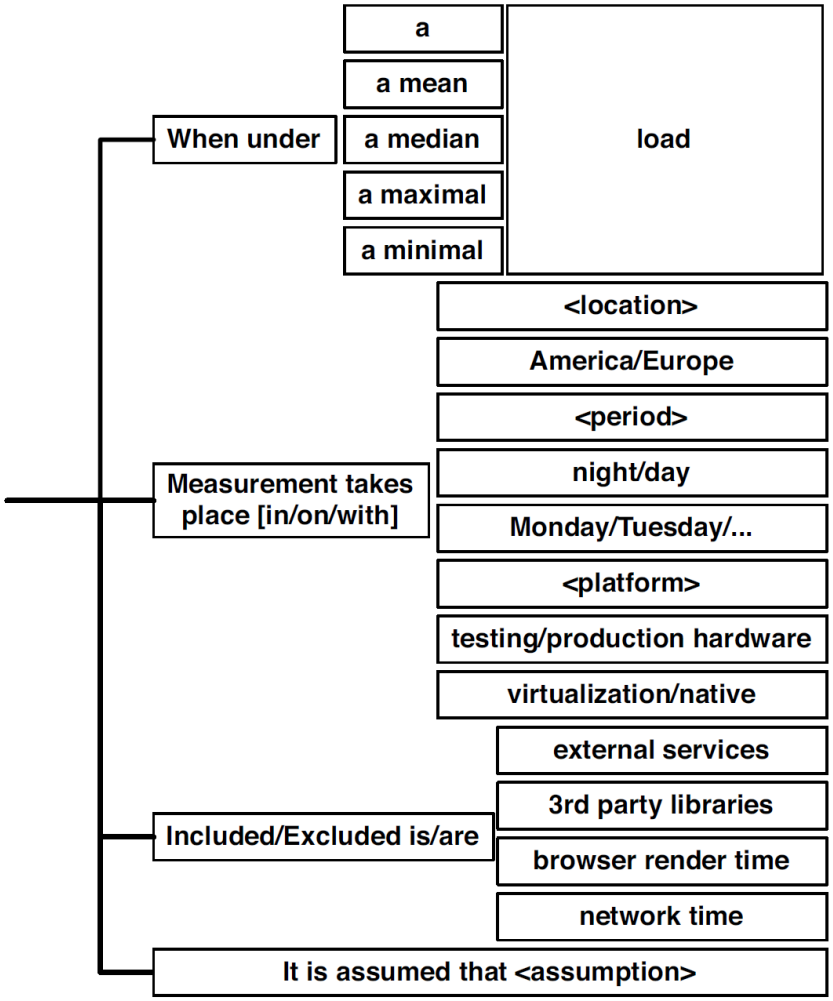
In case of peak load, the system shall have a minimal transaction rate of 100 payments per second.

Performance Measures (5) Patterns

[In <x>% of all cases]	The system	must/shall	be able to	support	a value	of	<x>	concurrent users			
[In case of <x>]	The <operation>	could/may		store	a mean	of <		users/objects/rows			
	The <component>	must not		receive	a median	of <=		<data>			
				process	a maximum	of >		items	per	ms	
				sustain	a minimum	of >=		orders		s	
								requests		min	
								<data>		h	

(c) Sentence Patterns for Capacity Requirements

Performance Measures (6) Patterns



(d) Sentence Patterns for Cross-cutting Performance Concepts

- "**Cross-cutting**" = Applicable to all types of performance requirements.
- **Load**, i.e., under which load should we measure the performance aspect? E.g., "When under a maximal load...".
- **Measurement location**, i.e., where should the measurement take place? E.g., "The measurement shall take place in Berlin, Germany".
- **Measurement period**, i.e., at which time of day, month, or year should the measurements be performed? E.g., "The measurement shall take place on weekdays between 9AM and 10AM".
- **Platform**, i.e., on which platform should we measure? E.g., "The measurement shall take place on an ARMv8 platform".
- **Scope of Measurement**, i.e., what is included and what is excluded in the measurement? E.g., "... included is the browser render time and excluded is the network time".
- **Measurement Assumption**, i.e., are there further assumptions that constrain the measurement? E.g., "... We assume that signal X is present".

Reliability Measures (1)

- Measure of the degree to which the system performs during a request
  - 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
  - Defect rate
  - Degree of precision for computations

Reliability Measures (2)

- Examples
  - Function X's 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.

Availability Measures (1)

- Definition: Proportion of time that the system is up and running correctly
  - Probability that the system is there when needed!
- Can be calculated based on Mean-Time to Failure (MTTF) and Mean-Time to Repair (MTTR)
  - MTTF : Average duration between operation start and failure
  - MTTR : Average duration needed to resume operation after a failure
  - Availability = MTTF/(MTTF+MTTR)
- May lead to architectural requirements
  - Redundant components (lower MTTF)
  - Modifiability/Substitutability of components (lower MTTR)
  - Special types of components (e.g., self-diagnostic)
- Modeling/Estimating 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. (This is a MTBF requirement)
  - The system shall be restartable in less than 20 seconds after a failure at least 95% of the time. (This is a MTTR requirement)

- Availability Downtime
  - 90% 36.5
  - 99% 3.65
  - 99.9% 8.76
  - 99.99% 52 m
  - 99.999% 5
  - 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 its user before allowing it to use its capabilities.
  - The system shall detect at least 99% of intrusion attempts within 10 seconds.

Security vs Safety

Privacy and Confidentiality

- **Privacy**
  - Privacy refers to an individual's right to **be free from intrusion or interference by others**. It is a fundamental right in a free and democratic society. Individuals have privacy interests in relation to their bodies, personal information, expressed thoughts and opinions, personal communications with others, and spaces they occupy.

- **Confidentiality**
  - The ethical duty of confidentiality refers to the obligation of an individual or organization to **safeguard entrusted information**. The ethical duty of confidentiality includes obligations to protect information from unauthorized access, use, disclosure, modification, loss or theft.

- http://www.pre.ethics.gc.ca/eng/policy-politique/initiatives/tcps2-eptc2/chapter5-chapitre5

Privacy Measures

- It is insufficient to cite appropriate laws… "The system shall comply with The *Personal Information Protection and Documents Act* (PIPEDA)" does not help much (too coarse-grained)!
- Consider the relevant parts of laws, as well as other sources and standards. For example: US **Fair Information Practice Principles** (FIPPs) guidelines (source: http://player.slideplayer.com/18/6144807/#)

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
  - The system should enable at least 80% of users to book a guest within 5 minutes after a 2-hour introduction to the system.
  - The system shall enable **novice** users to perform tasks X and Y in less than 15 minutes.
  - The system shall enable **expert** users to perform tasks X and Y in less than 2 minutes.
  - The satisfaction level of the system shall be “very good” or better for at least 80% of customers polled after a 3 months usage period .

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 https://codeclimate.com/
  - Check their metrics and use them in your requirements!

Maintainability Measures (2)

- Examples of requirements
  - Every program module of the system must be assessed for maintainability according to procedure XYZ.
    - At least 70% of the program modules assessed according to procedure XZY must obtain “highly maintainable”
    - 0% of the program modules assessed according to procedure XZY must obtain “poor”.
  - The cyclomatic complexity of the system code must not exceed 7.
  - No method in any class of the system may exceed 200 lines of code.
  - The software shall allow upgrades to leave personal settings unchanged.

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 a minimum of 80% branch coverage.
  - Development shall use regression tests allowing for full retesting in less than 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/Enumeration 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.

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

- Example
  - The estimated loss of data in case of a disk crash shall be less than 0.01%.

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 **hedonic NFRs** such as "fun" or "cool" or "beautiful" or "exciting"?
- How can these be measured?

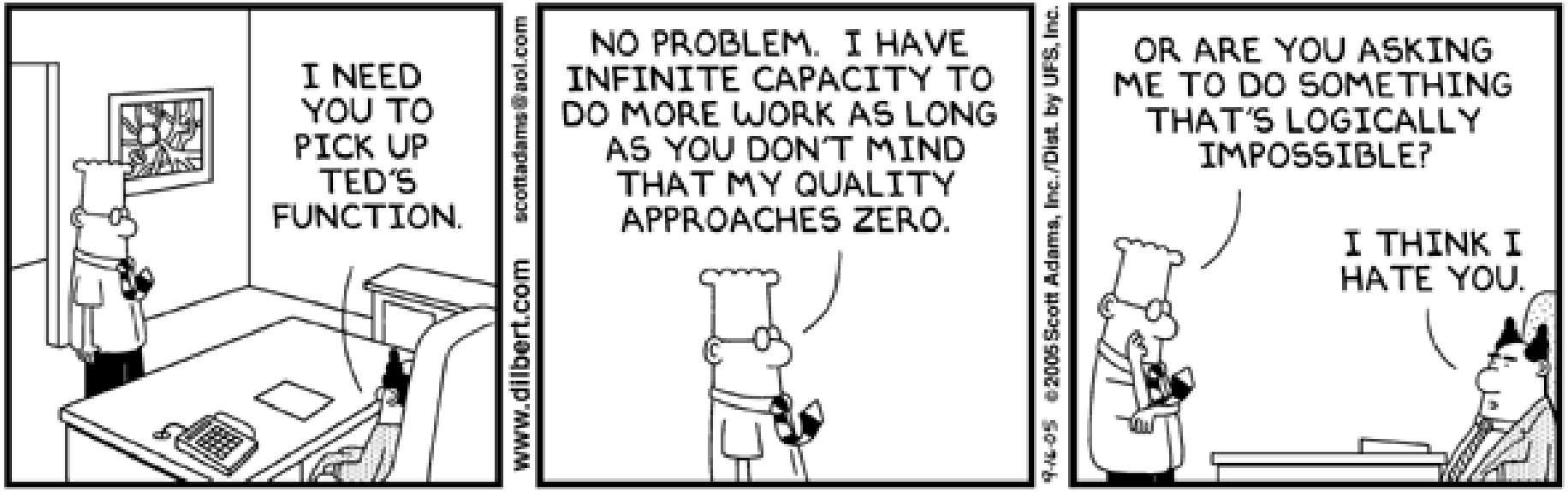
- The lists of existing quality attributes are interesting but they do not include all NFRs.
- It is sometimes better to let customers do their brainstorming before proposing the conventional NFR categories.

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

What About Moral and Ethical Requirements?

- Should a physician leave a patient suffer or die because of privacy issues? What does this mean in terms of information systems?
- If you are developing an autonomous car:
  - Who wants “driver override” as a feature?
  - Is this a safety feature? Is this augmenting or reducing uncertainty?
    - If a deer appears in front of the car, breaking may tilt the car forward
  - What if an accident is unavoidable and the choice is between killing X, Y, or one self?
- Risk management, legal aspects, and law enforcement… Is this a matter of what we can get away with?

Dilbert on Quality



Quiz!

1. Name three important challenges of quality requirements

2. Are there differences between requirements for
   1. Availability and Reliability?
   2. Safety and Security?
   3. Security, Anonymity, and Privacy?

3. What is the main NFR category of: “The system shall allow at least 90% of users to master over 80% of the system features in less than a week”. How would you test it?

4. How would you measure sustainability?