SE463
Software Requirements Specification & Analysis

Quality Requirements

Quality Requirements

- Functional requirements describe what the software is supposed to do
  - What services or tasks the software should provide
  - Black box input/output behaviour

- Quality requirements describe (extra) constraints on what constitutes an acceptable software solution

  e.g., How fast system should respond

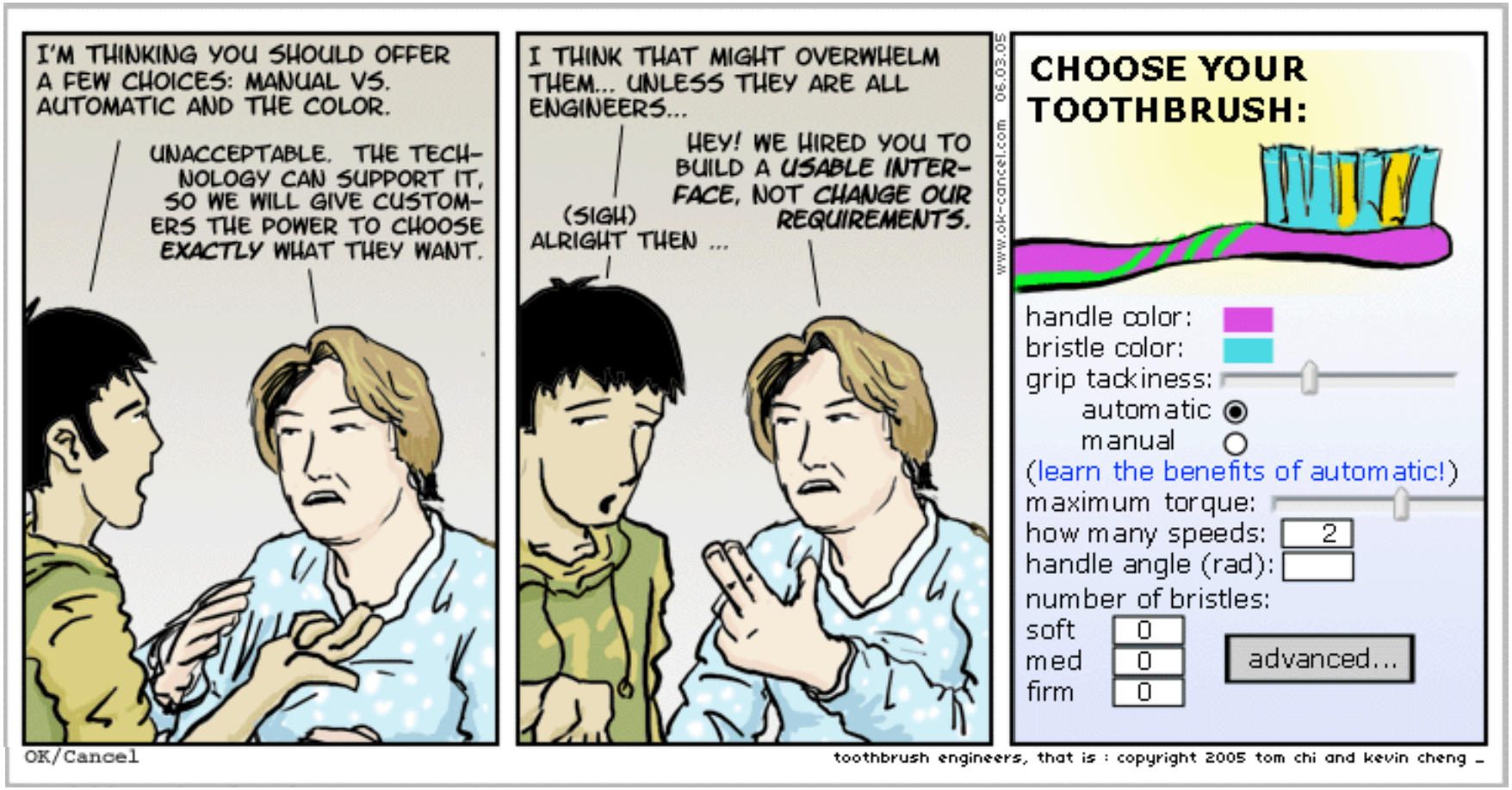
  Which deployment platforms should be supported

  How strong should security be

Quality Requirements

- Performance
  - execution speed
  - response time
  - throughput
    - e.g., “up to 30 simultaneous calls”
- Reliability
  - fault-tolerant
  - mean-time to failure
  - data backups
- Robustness
  - tolerates invalid input
  - fault-tolerant
  - fail-safe / -secure
  - degrades gracefully under stress
- Adaptability
  - ease of adding new functionality
  - reusable in other environments
  - self-optimizing
  - self-healing
- Security
  - controlled access to system, data
  - isolation of data, programs
  - protect against theft, vandalism
- Usability
  - how easy to learn / use
  - user productivity
- Scalability
  - workload
  - number of users
  - size of data sets
  - peak use
- Efficiency (capacity)
  - user productivity
  - utilization of resources
- Accuracy / precision
  - tolerance of computation errors
  - precision of computation results

Example: Useability



Other Nonfunctional Requirements

Design constraints

- interfaces to other systems
- COTS components
- programming language

Operating Constraints

- location
- size, power consumption
- temperature, humidity
- operating costs
- accessibility (for maintenance)

Product-family requirements

- modifiability
- portability
- reusability
- UI

Process Requirements

- Resources
  - personnel development
  - costs
  - development schedule
- Documentation
  - audience
  - conventions
  - readability
- Complexity (of code)
  - comments / KLOC
  - coupling / cohesion
  - cyclomatic complexity
  - use of multiple inherit. overloading, templates
- Standards compliance

“Motherhood” requirements

Expressions such as “reliable”, “user-friendly”, and “maintainable” are motherhood requirements.

- No one would explicitly ask their opposite

e.g., *slow, unreliable, user-hostile, unmaintainable*, …

(Virtually) every software system must have attributes such as “reliable”, “user-friendly”, and “maintainable”; what differs from product to product is:

- the degree to which each attribute is required, and
- the relative importance of one attribute over another.

Fit Criteria

"Anyone can build a bridge. It takes an engineer to build a bridge that barely stands."

[unknown source]

Fit criteria

A fit criterion quantifies the extent to which a quality requirement must be met.

Example:

- *75% of users shall judge the system to be as usable as the existing system*
- *After training, 90% of users shall be able to process a new account within 4 minutes*
- *A module will encapsulate the data representation of at most one data type*
- *Computation errors shall be fixed within 3 weeks of being reported*

Richer Fit Criteria

Requirement	Outstanding	Target	Minimum
Response time	0.1s	0.5s	1s
CPU utilization	20%	25%	30%
Usability	40 tasks/hr	30 tasks/hr	20 tasks/hr

Measurements

What gets measured gets done!

- Therefore, unless a quantified requirement is unrealistic, it will probably be met.
- There is a danger of focusing on what is measurable, and not on the true requirement
  - e.g., industry benchmarks (sometimes)

When you can't test before delivery

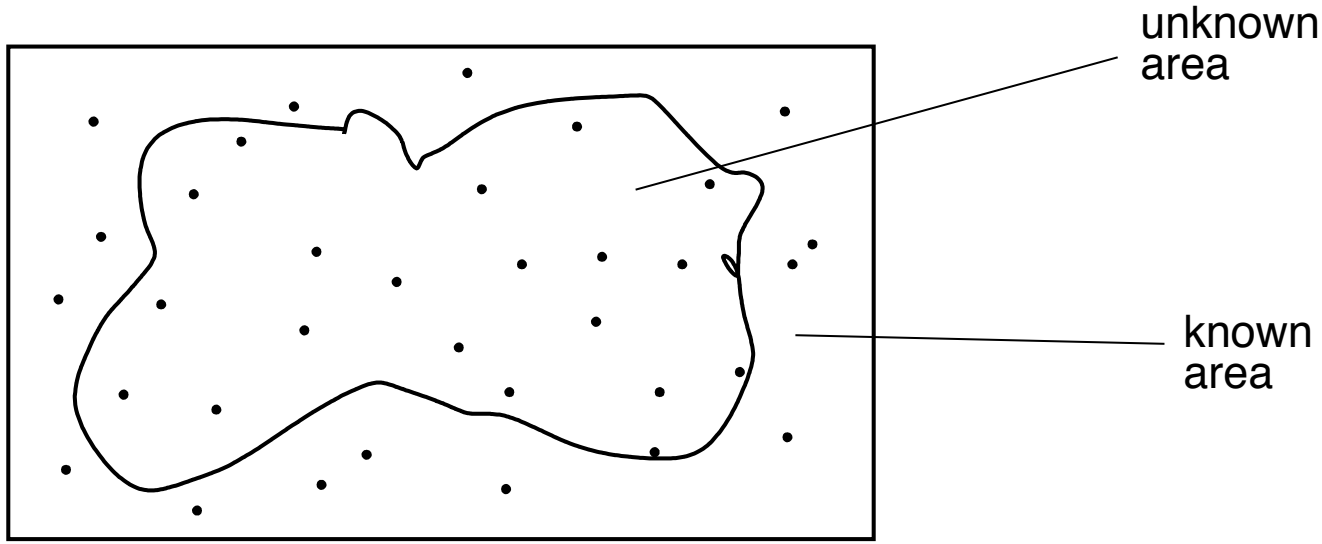
- Fit criteria that cannot be evaluated before the final product is delivered are harder to assess. For example:
  - *The system shall not be unavailable for more than a total of 3 minutes each year*
  - *The mean-time-to-failure shall be no less than 1 year*

- Possible approaches:
  - Measure the attributes of a prototype.
  - Measure secondary indicators

    e.g., number of user errors to assess usability
  - Estimate a system's quality attributes
  - Deliver system and pay penalty if requirements are not met

Monte Carlo techniques

Monte Carlo techniques: estimate an unknown quantity using a known quantity.

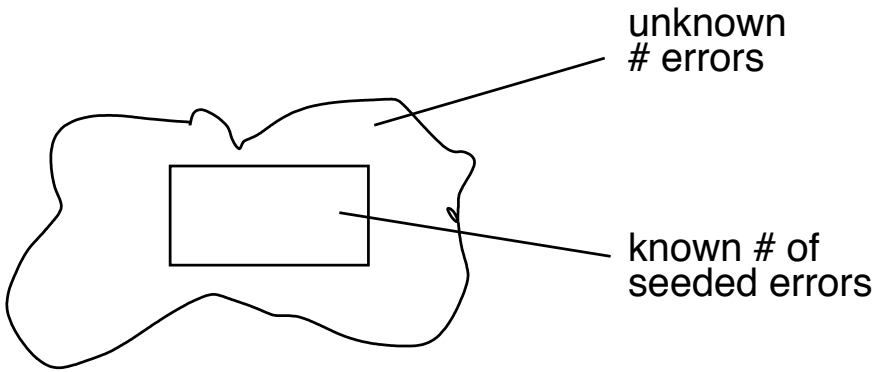


$$\frac{\text{Number of points in shape}}{\text{Total number of points}} \approx \frac{\text{Area of Shape}}{\text{Known Area of Rectangle}}$$

Monte Carlo techniques

We can use Monte Carlo techniques to estimate number of bugs remaining in a program (reliability).

- Plant a known number of errors into the program, which the testing team does not know about.
- Then compare the number of seeded errors the team detects with the total number of errors it detects, to arrive at an estimate of the total number of bugs in the program.



$$\frac{\text{\# detected seeded errors}}{\text{\# seeded errors}} \approx \frac{\text{\# detected errors}}{\text{\# errors in the program}}$$

Paper Airplane Exercise

Prioritizing Quality Requirements

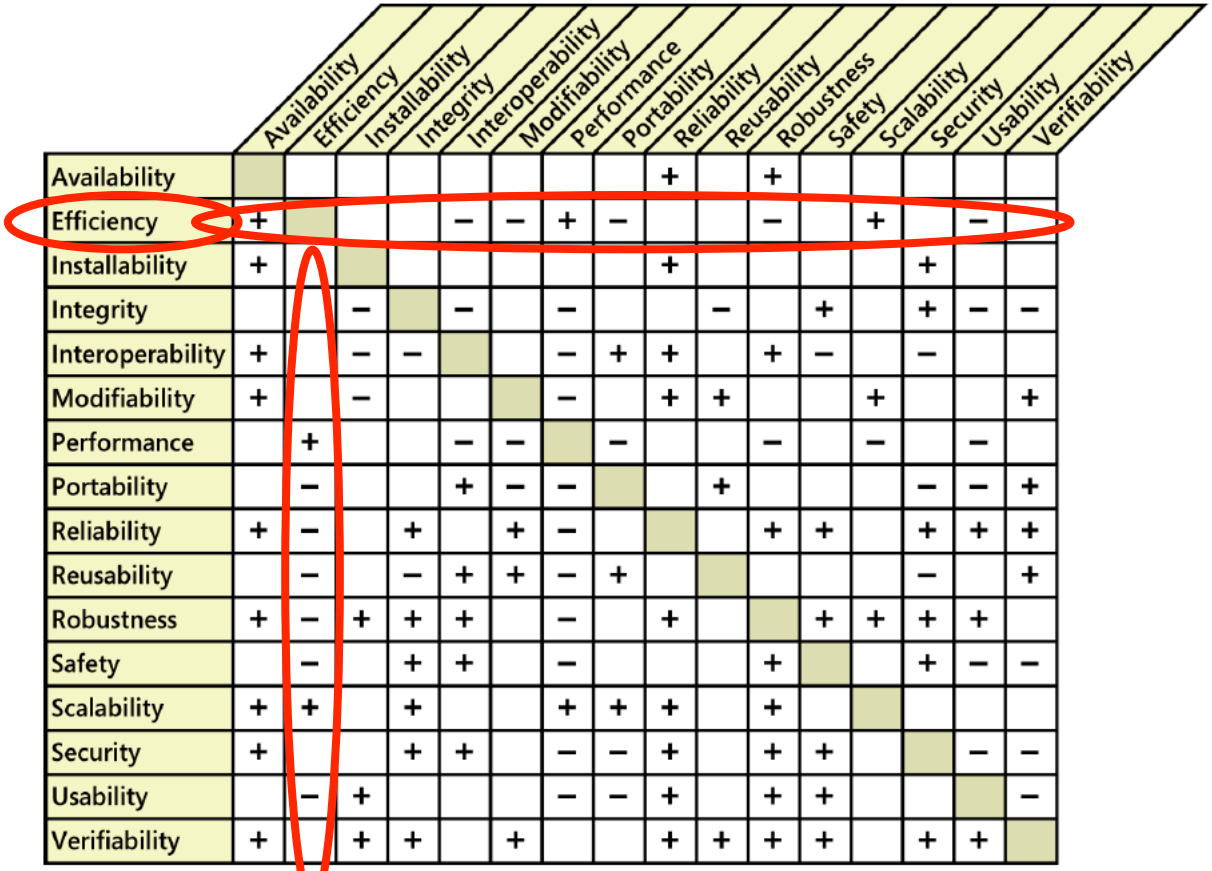
Many quality requirements conflict with one another:

- maintainability vs. robustness
  - simple design vs. design that monitors run-time / error recovery
- performance vs. security
- performance vs. reuse
- performance vs. portability
- robustness vs. testability

Also, some quality requirements conflicts with functional requirements

- performance vs. particular features (e.g., unlimited undo)

Typical Conflicts



	Availability	Efficiency	Installability	Integrity	Interoperability	Modifiability	Performance	Portability	Reliability	Reusability	Robustness	Safety	Scalability	Security	Usability	Verifiability
Availability									+		+					
Efficiency	+				−	−	+	−			−		+		−	
Installability	+								+					+		
Integrity			−		−		−			−		+		+	−	−
Interoperability	+		−	−			−	+	+		+	−		−		
Modifiability	+		−				−		+	+			+			+
Performance		+			−	−		−			−		−		−	
Portability		−			+	−	−			+				−	−	+
Reliability	+	−		+		+	−				+	+		+	+	+
Reusability		−		−	+	+	−	+						−		+
Robustness	+	−	+	+	+		−		+			+	+	+	+	
Safety		−		+	+		−				+			+	−	−
Scalability	+	+		+			+	+	+		+					
Security	+			+	+		−	−	+		+	+			−	−
Usability		−	+				−	−	+		+	+				−
Verifiability	+		+	+		+			+	+	+	+		+	+	

FIGURE 14-2 Positive and negative relationships among selected quality attributes.

Karl Wiegers, *Software Requirements*, (MS Press)

Summary

Quality requirements affect how a system accomplishes its functional goals

- Quality requirements are highly important to user experience
- Multiple types to consider
- Need to elicit and specify preferences
- Need to elicit and specify fit criteria

Deliverable #8

- Identify 6 quality requirements that your team believes are important to your project
  - State what the quality requirement means in the context of your project
  - Can have multiple requirements related to the same quality attribute, but must involve at least 4 quality attributes
- Use 100-dollar technique to prioritize quality requirements
  - Involve at least 2 concrete stakeholders, 1 not from your team
- Elicit fit criteria for top 3 quality requirements
  - Involve at least 2 concrete stakeholders, 1 not from your team