Lecture 8:
Quality requirements: **Lau**: 6
The QUality PERformance model [QUPER]

Björn Regnell
http://www.cs.lth.se/krav/
• Sprid ut er i salen per projekt gärna över två bänkrader
Functional reqs:

- What the system shall do
- Often intended to be implemented as a whole or else not implemented at all
- Often regards input/output data and functions that process the input data to produce the output

Non-functional reqs (NFR), Quality Requirements, (extra-functional reqs):

- How good the system shall do it
- Often measured on a scale
- Often put constraints on the system (or the development process)
- Often cross-cutting; may impact many functions

Performance
Reliability
Usability
Safety, Security
Interoperability
Maintainability

...
FR & QR are often tightly coupled

In practice it is often difficult to separate functional and quality requirements as quality requirements often are manifested into extra functionality.

Example: **Quality** requirement on security requires a log-in **function**.
Difficult trade-offs among QR

Quality requirements often counteract each other.

Common examples:

- Higher performance
  -> lower maintainability
- Higher security
  -> lower usability

Requires carefully considered trade-offs!
Quality requirements often determine choice of architecture

Cost?
Value?
Long-term vs short-term?
What quality features of a word processor do you appreciate?
Fig 6.1 Quality factors

**McCall**
US Airforce 1980

**Operation:**
- Integrity
- Correctness !!
- Reliability
- Usability
- Efficiency

**Revision:**
- Maintainability
- Testability
- Flexibility

**Transition:**
- Portability
- Interoperability
- Reusability !!

**ISO 9126**

**Functionality**
- Accuracy
- Security
- Interoperability
- Suitability !!
- Compliance !!

**Reliability**
- Maturity
- Fault tolerance !!
- Recoverability !!

**Usability**
- Efficiency

**Maintainability**
- Testability
- Changeability
- Analysability !!
- Stability !!

**Portability**
- Adaptability
- Installability !!
- Conformance !!
- Replaceability !!

Use as check lists
### Quality factors for Hotel system

<table>
<thead>
<tr>
<th></th>
<th>Critical</th>
<th>Important</th>
<th>As usual</th>
<th>Unimportant</th>
<th>Ignore</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integrity/security</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correctness</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reliability/availab.</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficiency</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Revision</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testability</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flexibility</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Transition</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portability</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interoperability</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reusability</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Installability</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Concerns:
1. Hard to run the hotel if system is down. Checking in guests is impossible since room status is not visible.
2. We aim at small hotels too. They have less qualified staff.
3. Customers have many kinds of account systems. They prioritize smooth integration with what they have.
4. Integration with spreadsheet etc. unimportant. Built-in statistics suffice.
5. Must be much easier than present system. Staff in small hotels should ideally do it themselves.
<table>
<thead>
<tr>
<th>Requirement (R)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1:</td>
<td>Product shall detect speed violation and take photo within 0.5 seconds.</td>
</tr>
<tr>
<td>R2:</td>
<td>Product shall compute a room occupation forecast within 2 minutes.</td>
</tr>
<tr>
<td>R3:</td>
<td>Product shall compute a room occupation forecast within 4 minutes.</td>
</tr>
<tr>
<td>R4:</td>
<td>Product shall compute a room occupation forecast within ___ minutes.</td>
</tr>
<tr>
<td>R5:</td>
<td>Product shall compute a room occupation forecast within ___ minutes. (Customer expects one minute.)</td>
</tr>
<tr>
<td>R6:</td>
<td>Forecast shall be computed with exponential trend smoothing and seasonal adjustments.</td>
</tr>
<tr>
<td>R7:</td>
<td>The supplier shall specify the forecast accuracy for hotels similar to ours.</td>
</tr>
</tbody>
</table>
**Fig 6.3C** Cost/benefit of response time

$ or ratio

Response time

From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002
Fig 6.4  Capacity and accuracy requirements

Capacity requirements:
R1: The product shall use < 16 MB of memory even if more is available.
R2: Number of simultaneous users < 2000
R3: Database volume:
    #guests < 10,000 growing 20% per year
    #rooms < 1,000
R4: Guest screen shall be able to show at least 200 rooms booked/occupied per day, e.g. for a company event with a single “customer”.

Accuracy requirements:
R5: The name field shall have 150 chars.
R6: Bookings shall be possible at least two years ahead.
R7: Sensor data shall be stored with 14 bit accuracy, expanding to 18 bits in two years.
R8: The product shall correctly recognize spoken letters and digits with factory background noise ___ % of the time. Tape B contains a sample recorded in the factory.

From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002
Performance requirements:

R1: Product shall be able to process 100 payment transactions per second in peak load.

R2: Product shall be able to process one alarm in 1 second, 1000 alarms in 5 seconds.

R3: In standard work load, CPU usage shall be less than 50% leaving 50% for background jobs.

R4: Scrolling one page up or down in a 200 page document shall take at most 1 s. Searching for a specific keyword shall take at most 5 s.

R5: When moving to the next field, typing must be possible within 0.2 s. When switching to the next screen, typing must be possible within 1.3 s. Showing simple report screens, less than 20 s. (Valid for 95% of the cases in standard load)

R6: A simple report shall take less than 20 s for 95% of the cases. None shall take above 80s. (UNREALISTIC)
Example:
Service time: Time to process one request
Average service time: 8 s (exp. distr.)
Average interarrival time: 10 s (exp. distr.)
System load: 8/10 = 0.8

Average response time:
5 × service time = 40 s
90% responses within:
12 × service time = 96 s
**Defect correction**

<table>
<thead>
<tr>
<th>Program errors</th>
<th>Usability problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected</td>
<td>Surprising?</td>
</tr>
<tr>
<td>Inspection OK</td>
<td>Inspection low hit-rate</td>
</tr>
<tr>
<td>Detect in test stage</td>
<td>Detect in design stage</td>
</tr>
<tr>
<td>Mostly simple</td>
<td>Often redesign</td>
</tr>
<tr>
<td>Test equipment OK</td>
<td>Subjects hard to find</td>
</tr>
</tbody>
</table>

**Usability**

*Fit for use* = tasks covered +

*Ease of use* =
- Ease of learning
- Task efficiency
- Ease of remembering
- Subjective satisfaction
- Understandability

From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002
Usability requirements?

R1: System shall be easy to use??

R2: 4 out of 5 new users can book a guest in 5 minutes, check in in 10 minutes, . . . New user means . . . Training . . .

Achieving usability

• Prototypes (mockups) before programming.
• Usability test the prototype.
• Redesign or revise the prototype.

Easier programming. High customer satisfaction.

Defect types

Program error: Not as intended by the programmer.
Missing functionality: Unsupported task or variant.
Usability problem: User cannot figure out . . .
Fig 6.6B Usability problems

Examples of usability problems

P1: User takes long time to start search. Doesn’t notice “Use F10”. Tries many other ways first.

P2: Believes task completed and result saved. Should have used Update before closing.

P3: Cannot figure out which discount code to give customer. Knows which field to use.

P4: Crazy to go through 6 screens to fill 10 fields.

Problem classification

Task failure: Task not completed - or believes it is completed.

Critical problem: Task failure or complaints that it is cumbersome.

Medium problem: Finds out solution after lengthy attempts.

Minor problem: Finds out solution after short attempts
**Usability test**
Realistic introduction
Realistic tasks

Note problems
- Observe only or
- Think aloud & ask

**Heuristic evaluation**
Expert’s predicted problems
≡ Inspection/Review

**Usability test:**
Cover all tasks?
Mockups find same problems as test with final system?

From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002
### Usability requirements

<table>
<thead>
<tr>
<th>Problem counts</th>
<th>Risk</th>
<th>Cust.</th>
<th>Suppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: At most 1 of 5 novices shall encounter critical problems during tasks Q and R. At most 5 medium problems on list.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Task time

<table>
<thead>
<tr>
<th>Task time</th>
<th>Risk</th>
<th>Cust.</th>
<th>Suppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2: Novice users shall perform tasks Q and R in 15 minutes. Experienced users tasks Q, R, S in 2 minutes.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Keystroke counts

<table>
<thead>
<tr>
<th>Keystroke counts</th>
<th>Risk</th>
<th>Cust.</th>
<th>Suppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3: Recording breakfast shall be possible with 5 keystrokes per guest. No mouse.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Opinion poll

<table>
<thead>
<tr>
<th>Opinion poll</th>
<th>Risk</th>
<th>Cust.</th>
<th>Suppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>R4: 80% of users shall find system easy to learn. 60% shall recommend system to others.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Score for understanding

<table>
<thead>
<tr>
<th>Score for understanding</th>
<th>Risk</th>
<th>Cust.</th>
<th>Suppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>R5: Show 5 users 10 common error messages, e.g. <em>Amount too large</em>. Ask for the cause. 80% of the answers shall be correct.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Usability requirements

<table>
<thead>
<tr>
<th><strong>Design-level reqs</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>R6: System shall use screen pictures in app. xx, buttons work as app. yy.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Product-level reqs</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>R7: For all code fields, user shall be able to select value from drop-down list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Guideline adherence</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>R8: System shall follow style guide zz. Menus shall have at most three levels.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Development process reqs</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>R9: Three prototype versions shall be made and usability tested during design.</td>
</tr>
</tbody>
</table>

From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002
### Fig 6.8A  Threats

<table>
<thead>
<tr>
<th>Threats</th>
<th>Violate</th>
<th>Preventions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input, e.g.</strong></td>
<td><strong>Examples</strong></td>
<td></td>
</tr>
<tr>
<td>Mistake</td>
<td>Integrity</td>
<td>Logical checks</td>
</tr>
<tr>
<td>Illegal access</td>
<td>Authenticity</td>
<td>Signature</td>
</tr>
<tr>
<td>Wire tapping</td>
<td>Confidentiality</td>
<td>Encryption</td>
</tr>
<tr>
<td><strong>Storing, e.g.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disk crash</td>
<td>Availability</td>
<td>RAID disks</td>
</tr>
<tr>
<td>Program error</td>
<td>Integrity</td>
<td>Test techniques</td>
</tr>
<tr>
<td>Virus deletes data</td>
<td>Availability</td>
<td>Firewall</td>
</tr>
<tr>
<td><strong>Output, e.g.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmission</td>
<td>Availability</td>
<td>Multiple lines</td>
</tr>
<tr>
<td>Fraud</td>
<td>Confidentiality</td>
<td>Auditing</td>
</tr>
<tr>
<td>Virus sends data</td>
<td>Authenticity</td>
<td>Encryption</td>
</tr>
</tbody>
</table>

From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002
Fig 6.9  Security requirements

R1: Safeguard against loss of database. Estimated losses to be < 1 per 50 years.
R2: Safeguard against disk crashes. Estimated losses to be < 1 per 100 years.
R3: Product shall use duplicated disks (RAID disks).
R4: Product shall safeguard against viruses that delete files. Remaining risk to be < _____.
R5: Product shall include firewalls for virus detection.
R6: Product shall follow good accounting practices. Supplier shall obtain certification.
R7: Product shall prevent users deleting invoices before transfer to the account system.
R8: The supplier shall as an option offer features for checking and reserving deposits made by credit cards.
R9: The supplier must enclose a risk assessment and suggest optional safeguards.

From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002
Examples of QR in reqT:
Capacity and Performance <=> Usability

Model(
    Quality("dbCapacity") has
        Spec("#guests < 10,000 growing 20% per year, #rooms < 1,000"),
    Quality("calendarAccuracy") has
        Spec("Bookings shall be possible at least two years ahead."),
    Quality("forecastPerformance") has
        Spec("Product shall compute a room occupation forecast within ___ minutes. (Customer expects one minute.)"),
    Quality("taskTimeUsability") has
        Spec("Novice users shall perform tasks Q and R in 15 minutes. Experienced users tasks Q, R, S in 2 minutes."),
    Quality("taskTimeUsability") requires (Task("Q"), Task("R"), Task("S")),
    Quality("peakLoadPerformance") has
        Spec("Product shall be able to process 100 payment transactions per second in peak load."))

[Examples modified from Lauesen: "Software Requirements - Styles and Techniques"]
Fig 6.10  Maintainance

Corrective maintenance

Preventive maintenance

Perfective maintenance

Product

New release

Maintenance cycle:

Report: Record and acknowledge.

Analyze: Error, change, usability, mistake?
          Cost/benefit?

Decide: Repair? reject? work-around?
        next release? train users?

Reply: Report decision to source.

Test: Test solution. Related defects?

Carry out: Install, transfer user data, inform.
### Maintainability requirements

#### Maintenance performance
- **R1:** Supplier’s hotline shall analyze 95% of reports within 2 work hours. Urgent defects (no work around) shall be repaired within 30 work hours in 95% of the cases.
- **R2:** When repairing a defect, related non-repaired defects shall be less than 0.5 in average.
- **R3:** For a period of two years, supplier shall enhance the product at a cost of ____ per Function Point.

#### Support features
- **R4:** Installation of a new version shall leave all database contents and personal settings unchanged.
- **R5:** Supplier shall station a qualified developer at the customer’s site.
- **R6:** Supplier shall deposit code and full documentation of every release and correction at ____________.

---

From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002
## Development process requirements

R7: Every program module must be assessed for maintainability according to procedure xx. 70% must obtain “highly maintainable” and none “poor”.

R8: Development must use regression test allowing full re-testing in 12 hours.

## Program complexity requirements

R9: The cyclomatic complexity of code may not exceed 7. No method in any object may exceed 200 lines of code.

## Product feature requirements

R10: Product shall log all actions and provide remote diagnostic functions.

R11: Product shall provide facilities for tracing any database field to places where it is used.
Forecast speed [Tag]: How quickly the system completes a forecast report [Gist]

Scale: average number of seconds from pushing button, to report appearing.

Meter: Measured 10 times by a stopwatch during busy hours in hotel reception.

Must: 8 minutes, because the competitive system does it this fast.

Plan: ____ (supplier, please specify).

Wish: 2 minutes.

Past: Done as batch job taking about an hour.
Paper [QUPER] in compendium

Supporting Roadmapping of Quality Requirements

Quality Requirements challenge in market-driven RE

Systematic prioritization of FEATURES is state-of-art in roadmapping and platform/product scoping

…but...

Prioritisation of QUALITIES is often handled ad hoc with no specific support for roadmapping

One FR imply many different qualities. How to scope both FR and QR together?
Improving Quality Requirements

Problem:
Quality requirements such as performance are often given without explanation
- Would just a little less still be almost as valuable?
- Would just a little less be very much cheaper?

One proposed solution:
Estimate cost-benefit breakpoints and barriers with QUPER = Quality Performance reference model
QUPER model views: Benefit, Cost, Roadmap

**Benefit view**
- Excessive
- Competitive advantage
- Useful
- Useless

**Saturation breakpoint**
**Differentiation breakpoint**
**Utility breakpoint**

**Quality level**

**Cost view**

**Utility breakpoint**

**Barrier**

**Quality level**

**Roadmap view**
- Competitor A
- Competitor B
- Target release $n_1$
- Target release $n_2$
- Bad Target
- Current

**Quality level**

(Feature X, Segment Y)
QUPER example steps

- **Step 1 - Description**
  - *Quality indicator:* Time to play music [seconds]
  - *Quality type:* Performance
  - *Definition:* Measured from player invoke button pressed until music is played using 2 GB memory stick type X with 100 tracks with average duration of 3 min

- **Step 2 - Current reference products**
  - *Competitor Product X:* 4 seconds
  - *Competitor Product Y:* 2 seconds
  - *Own Product Z (Qref):* 3 seconds

- **Step 3 – Current market expectations**
  - *Utility breakpoint:* 5 seconds
  - *Differentiation breakpoint:* 1.5 seconds
  - *Saturation breakpoint:* 0.2 seconds

- **Step 4 – Estimate the closest cost barrier (CB1)**
  - *Q1:* 2 seconds
  - *C1:* 4 weeks

- **Step 5 – Estimate the second cost barrier (CB2)**
  - *Q2:* 1 second
  - *C2:* 24 weeks

- **Step 6 – Candidate targets**
  - *Min target:* 2 seconds – This target is possible without a new architecture, but needs some software optimization.
  - *Max target:* 1 second – If we create a new architecture, this target (which is better than differentiation) will be easy to reach. Users might require this level of quality within 2 years.
val m = Model(
    Quality("mtts") has (
        Gist("Mean time to startup"),
        Spec("Measured in milliseconds using Test startup"),
        Breakpoint("utility") has Value(4000),
        Breakpoint("differentiation") has Value(1500),
        Breakpoint("saturation") has Value(200),
        Target("basic") has (
            Value(2000), Comment("Probably possible with existing architecture."))),
    Target("stretch") has (
        Value(1100), Comment("Probably needs new architecture."))),
    Barrier("first") has (Min(1900), Max(2100)),
    Barrier("second") has Value(1000),
    Product("competitorX") has Value(2000),
    Product("competitorY") has Value(3000),
    Test("startup") verifies Quality("mtts"),
    Test("startup") has (
        Spec("Calculate average time in milliseconds of the startup time over 10 executions from start button is pressed to logon screen is shown."),
        Target("stretch"))
)

Vad kommer på tentan?

• En del av uppgifterna blir påstående-anledning-frågor enligt exempel från tidigare år här: http://cs.lth.se/krav/exam-problems/

• En del av uppgifterna omfattar praktiskt arbete med några centrala tekniker ur litteraturen, t.ex. givet en domänbeskrivning skapa ett kontextdiagram eller givet ett kvalitetsområde skapa en roadmap-vy enligt Quper.

• Essäfrågor ur artikel el. bokkapitel: beskriv t.ex. ett område, en teknik, slutsatser från en undersökning. Givet nyckelord. Begränsad plats. (Språkbehandling är inte fokus men det ska vara läsbart och begripligt.)
Vad händer på lab 2?

- Totalt 90 minuter
- Ca 30-40 min: Kvalitetskrav
- Ca 50-60 min: Releaseplannering
- Obligatoriska förberedelser här:
  - http://cs.lth.se/krav/labs/lab2/
- Anmälning via SAM på samma sätt som lab1