Computer Science
UNIVERSITY OF TORONTO

lecture 15
software quality

csc302s

winter 2014

recap from last time

- static analysis
- a lot is done "live" by the IDE while you are coding.
- beyond what is done in your IDE:
  - attempts to find null dereference or null assignment
  - array index out of bounds etc.
  - other runtime errors not caught by compiler
  - duplicate code
- many false positives & negatives
- your mileage may vary

Software Quality

Understanding Quality

Importance of Process Quality

- tools for improving process quality

Software Quality Attributes

Tools for improving process quality

Software Quality Attributes

Challenge Problem

Context

- You built some software
- You tested it
- You shipped it

But:

- Is it any good?
- How would you know?
- Can you do a better job next time?

Q1: What is Quality?

“Quality is value to some person”

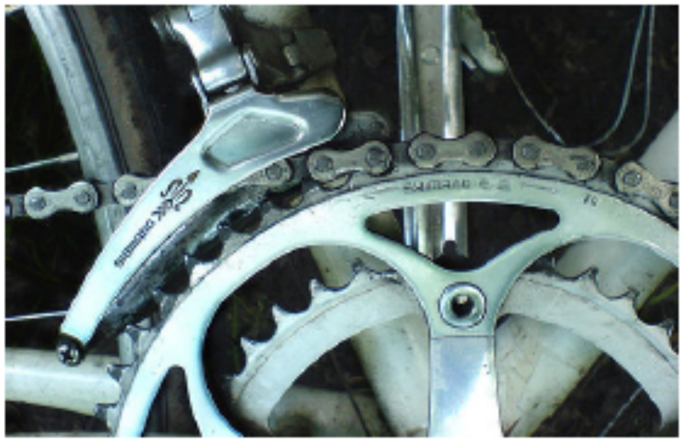
“Quality is fitness to purpose”

“Quality is exceeding the customer’s expectations”

4 Views of Quality

Quality in Use
(What's the end-user's experience?)

External Quality Attributes
(Does it pass all the tests?)



Internal Quality Attributes
(Is it well-designed?)

Process Quality
(Is it assembled correctly?)

Quality Assurance

V&V focuses on the quality of the product(s)

- requirements, models, specifications, designs, code,…

QA focuses on the quality of the processes

- How well are the processes documented?
- How well do people follow these processes?
- Does the organisation measure key quality indicators?
- Does the organisation learn from its mistakes?

Examples:

- ISO9001
- TickIt
- Capability Maturity Model (CMM)
- Total Quality Management (TQM)

Managing Quality (history)

Source: Adapted from Blum, 1992, p473-479. See also van Vliet, 1999, sections 6.3 and 6.6

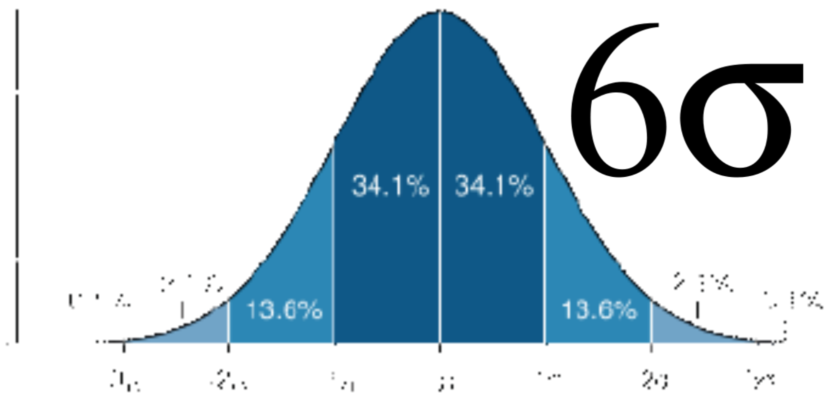
Industrial Engineering

- Product Inspection (1920s)
  - examine intermediate and final products and discard defective items
- Process Control (1960s)
  - monitor defect rates to identify defective process elements & control the process
- Design Improvement (1980s)
  - engineering the process and the product to minimize the potential for defects

Deming: Total Quality Management

- Use statistical methods to analyze industrial production processes
- Identify causes of defects and eliminate them
- Basic principles are counter-intuitive:
  - in the event of a defect (sample product out of bounds)…
    …don't adjust the controller or you'll make things worse.
  - Instead, analyze the process and improve it

Six Sigma



Key ideas:

Use statistics to measure defects

Design the process to reduce defects

Origin of the term

99.9999% of all items are with ±6σ of the mean on a normal curve

So a target of 6σ mean no more than 1 defective part per million

In practice, must allow for ±1.5σ drift in the mean over the long term

So we really only get ±4.5σ = 3.4 defective parts per million

For complex devices

100 parts: probability of a defective device is 0.0013

10,000 parts: probability of a defective device is 0.04 (I.e. 96% are okay….)

⇒ Design things to have fewer components

⇒ Control the manufacturing variability of the components

Applying This to Software

Quality Management for Software

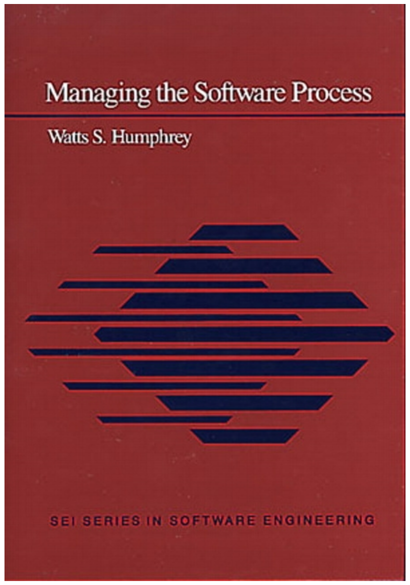
- No variability among individual product instances
- All defects are design errors (no manufacturing errors)
- Process improvement principles still apply (to the design process!)

Defect removal

- Two ways to remove defects:
  - fix the defects in each product (i.e patch the product)
  - fix the process that leads to defects (i.e. prevent them occurring)
- The latter is cost effective as it affects all subsequent projects

Defect prevention (from Humphrey)

- programmers must evaluate their own errors
- feedback is essential for defect prevention
- there is no single cure-all for defects
  - must eliminate causes one by one
- process improvement must be an integral part of the process
- process improvement takes time to learn


Managing the Software Process
Watts S. Humphrey
SEI SERIES IN SOFTWARE ENGINEERING

Process Modeling & improvement

Process Description

understand and describe current practices

Process Definition

Prescribe a process that reflects the organization's goals

Process customization

adapt the prescribed process model for each individual project

Process enactment

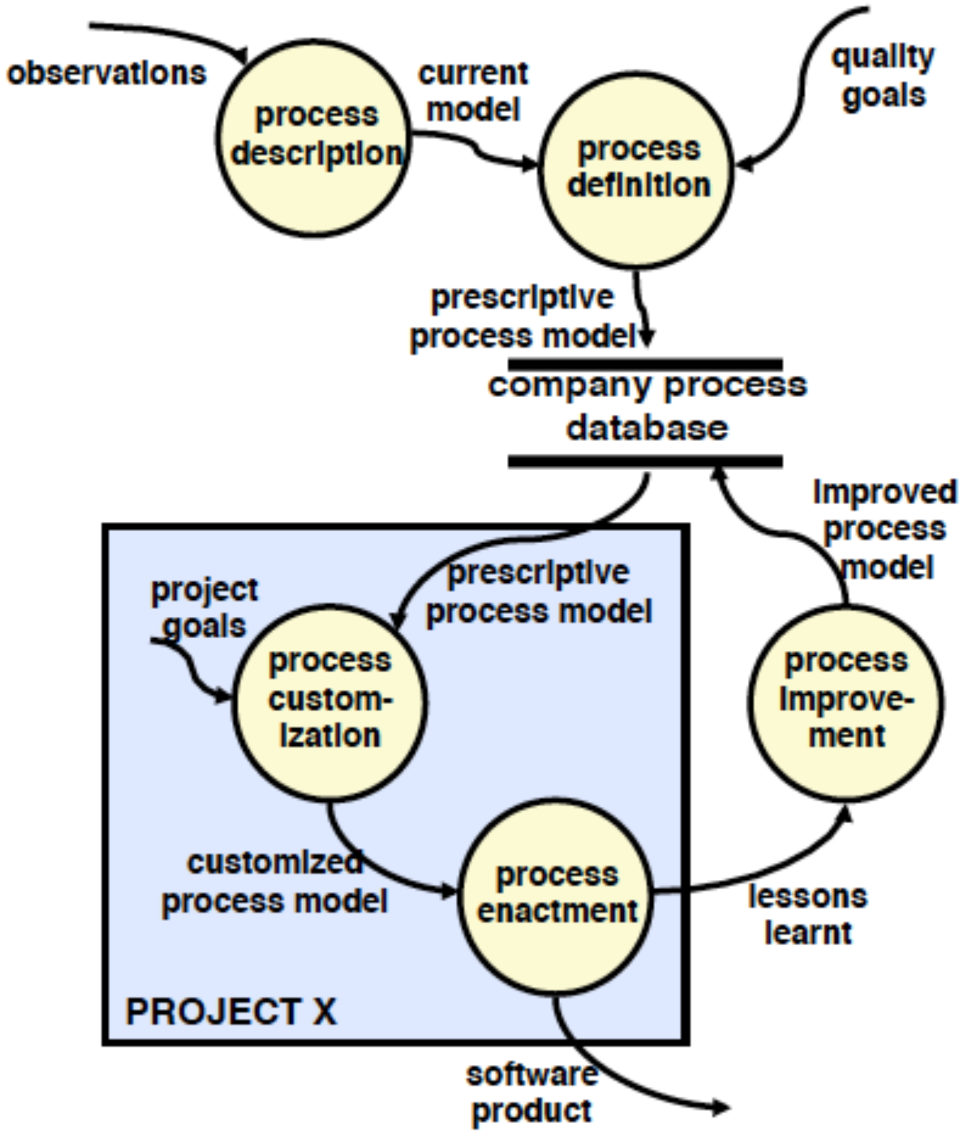
Carry out the process

- (develop the software!)
- collect process data

Process improvement

use lessons learnt from each project to improve the prescriptive model

- analyze defects to eliminate causes



e.g. Capability Maturity Model (CMM)

Source: Adapted from Humphrey, 1989, chapter 1. See also van Vliet, 1999, section 6.6.

Level	Characteristic	Key Challenges
5. Optimizing	Improvement fed back into process	Identify process indicators "Empower" individuals
4. Managed	(Quantitative) measured process	Automatic collection of process data Use process data to analyze and modify the process
3. Defined	(Qualitative) process defined and institutionalized	Process measurement Process analysis Quantitative Quality Plans
2. Repeatable	(Intuitive) process dependent on individuals	Establish a process group Identify a process architecture Introduce SE methods and tools
1. Initial	Ad hoc / Chaotic No cost estimation, planning, management.	Project Management Project Planning Configuration Mgmnt, Change Control Software Quality Assurance

Counterpoint: 6 Sigma for Software?

Software processes are fuzzy

- Depend on human behaviour, not predictable

Software Characteristics are not ordinal

- Cannot measure degree of conformance for software
- Mapping between software faults and failures is many-to-many
- Not all software anomalies are faults
- Not all failures result from the software itself
- Cannot accurately measure the number of faults in software

Typical defect rates

- NASA Space shuttle: 0.1 failures/KLOC (but it cost $1000 per line)
- Best military systems: 5 faults/KLOC
- Worst military systems: 55 faults/KLOC
- Six Sigma would demand 0.0034 faults/KLOC (?)

Arguments against QA

Costs may outweigh the benefits

- Costs: Increased documentation; more meetings; …
- Benefits: Improved quality of the process outputs (better software?)

Reduced "agility"

- Documenting the processes makes them less flexible

Reduced "thinking"

- Following the defined process gets in the way of thinking about the best way to do the job

Barrier to Innovation

- New ideas have to be incorporated into the Quality Plan and get signed off

Demotivation

- Extra bureaucracy makes people frustrated

How to assess software quality?

Source: Budgen, 1994, pp65-7

Reliability

- designer must be able to predict how the system will behave:
  - completeness - does it do everything it is supposed to do? (e.g. handle all possible inputs)
  - consistency - does it always behave as expected? (e.g. repeatability)
  - robustness - does it behave well under abnormal conditions? (e.g. resource failure)

Efficiency

- Use of resources such as processor time, memory, network bandwidth
  - This is less important than reliability in most cases

Maintainability

- How easy will it be to modify in the future?
  - perfective, adaptive, corrective

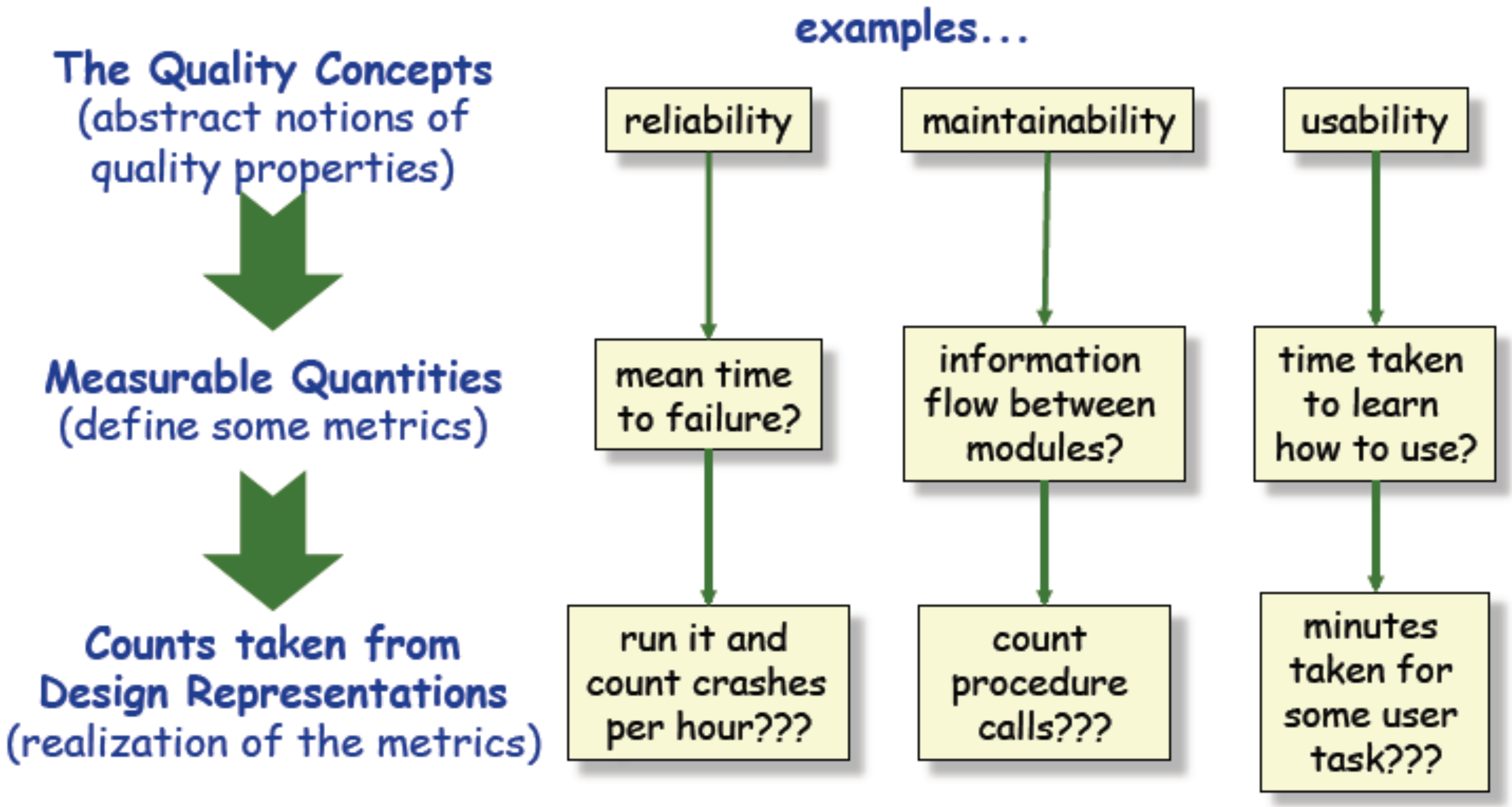
Usability

- How easy is it to use?

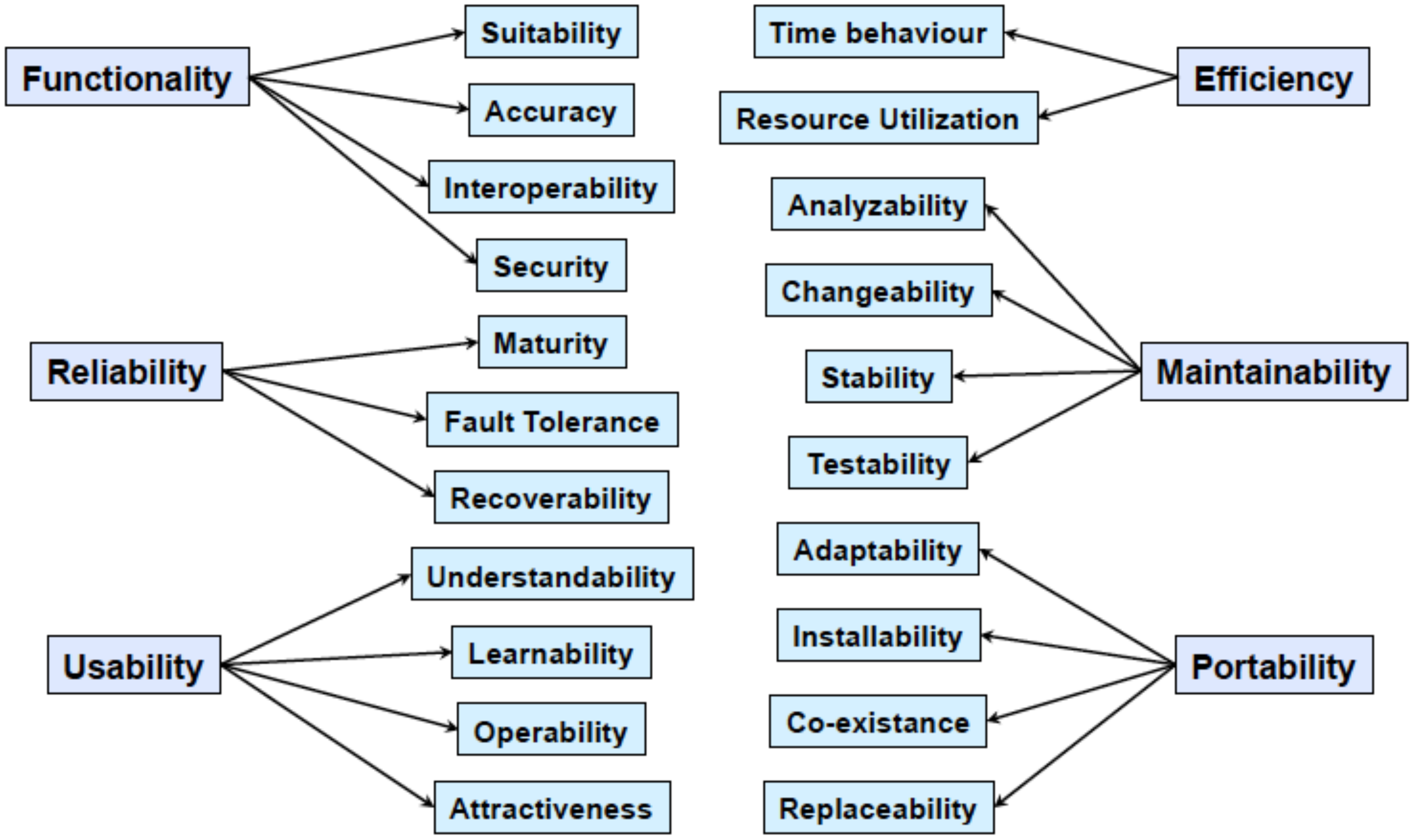
Measuring Quality

Source: Budgen, 1994, pp60-1

We have to turn our vague ideas about quality into measurables


examples...
The Quality Concepts (abstract notions of quality properties)
Measurable Quantities (define some metrics)
Counts taken from Design Representations (realization of the metrics)
reliability
maintainability
usability
mean time to failure?
information flow between modules?
time taken to learn how to use?
run it and count crashes per hour???
count procedure calls???
minutes taken for some user task???

ISO/IEC 25010:2011



Computer Science
UNIVERSITY OF TORONTO

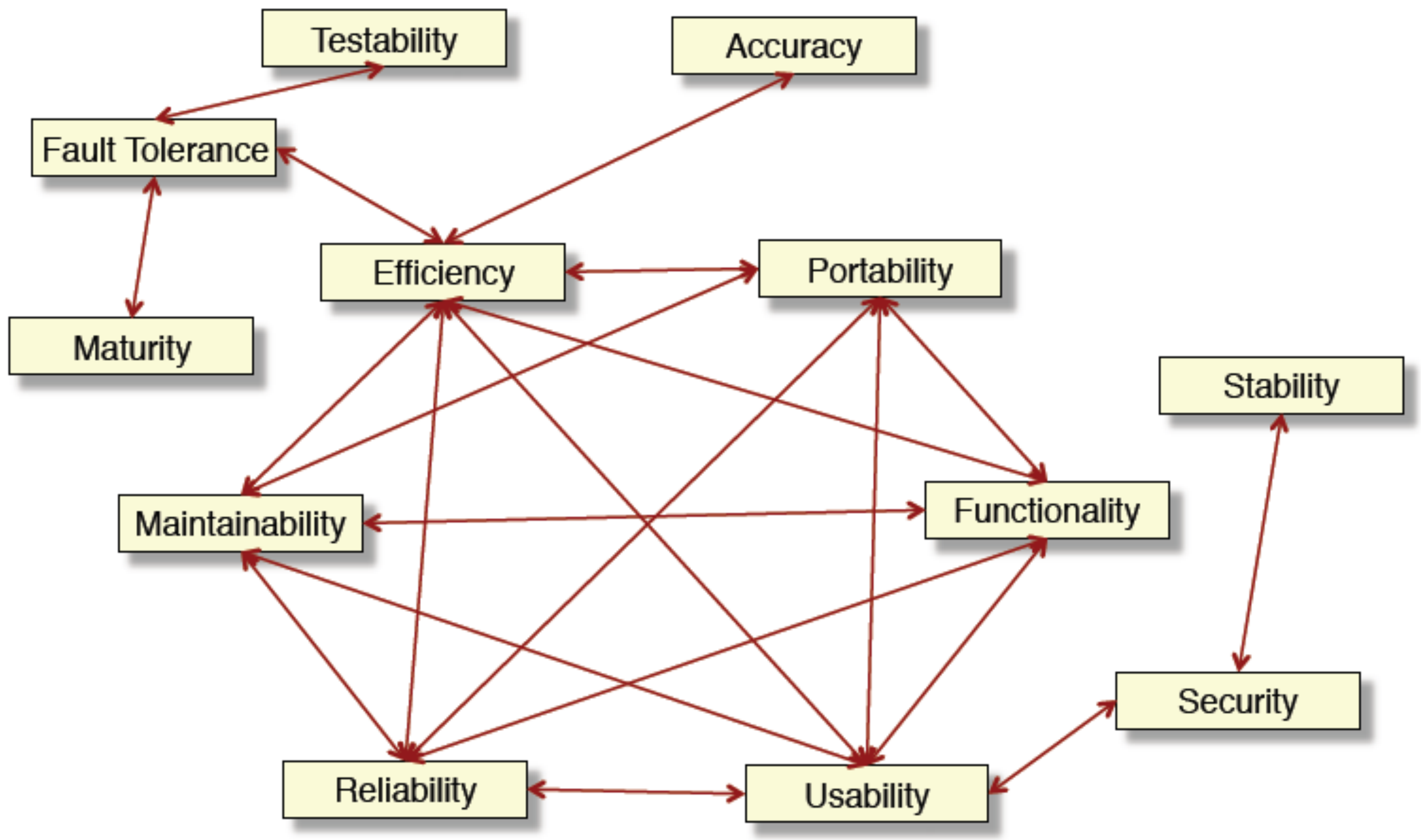
ISO/IEC 25010:201

The fundamental objective of the ISO/IEC 9126 standard is to address some of the well known human biases that can adversely affect the delivery and perception of a software development project. These biases include changing priorities after the start of a project or not having any clear definitions of "success." By clarifying, then agreeing on the project priorities and subsequently converting abstract priorities (compliance) to measurable values (output data can be validated against schema X with zero intervention), ISO/IEC 9126 tries to develop a common understanding of the project's objectives and goals.

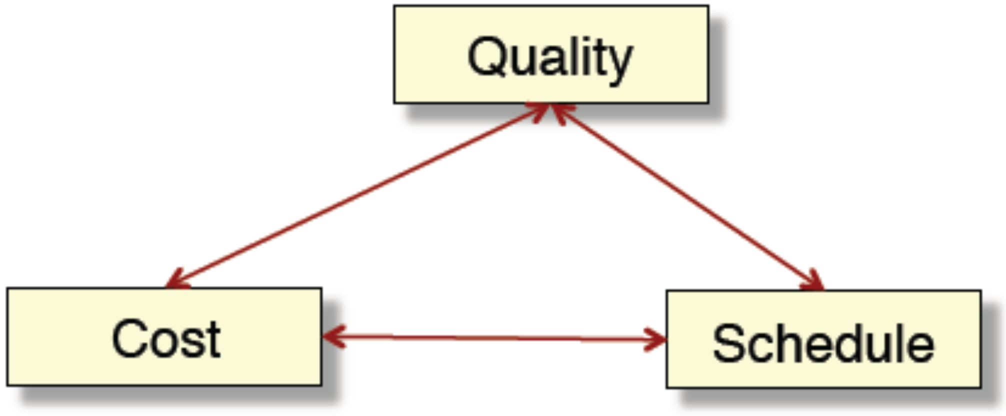
The standard is divided into four parts:

- quality model
- external metrics
- internal metrics
- quality in use metrics.

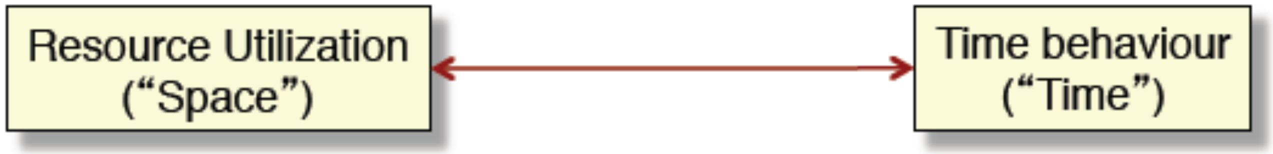
Conflicts between Quality factors



More abstractly…



"Better, Faster, Cheaper - pick any two"



Measurable Predictors of Quality

Source: Budgen, 1994, pp68-74

Simplicity

- the design meets its objectives and has no extra embellishments
- can be measured by looking for its converse, complexity:
  - control flow complexity (number of paths through the program)
  - information flow complexity (number of data items shared)
  - name space complexity (number of different identifiers and operators)

Modularity

- different concerns within the design have been separated
- can be measured by looking at:
  - cohesion (how well components of a module go together)
  - coupling (how much different modules have to communicate)

slide from lecture 2 – architecture

"Il semble que la perfection soit atteinte non quand il n'y a plus rien à ajouter, mais quand il n'y a plus rien à retrancher." – Antoine de Saint Exupéry, Terre des Hommes, 1939

(my) translation: *"perfection is finally attained not when there is no longer anything to add, but when there is no longer anything to take away"*