Lecture 22: Software Quality

Understanding Quality

Importance of Process Quality

- 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?

"Quality is value to some person"

"Quality is fitness to purpose"

"Quality is exceeding the customer's expectations"

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?

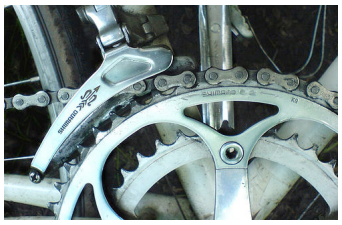
Examples:

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

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?)

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

Process Quality - key ideas

"You cannot test-in software quality"

- testing or inspection cannot improve the quality of a software product
  - (by that stage it is too late)

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 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 to detect defects
- 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

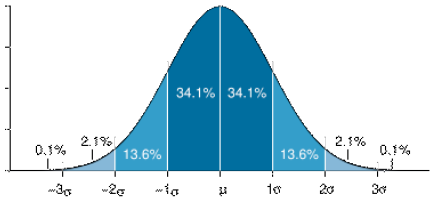
Deming and TQM

- 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

Adapted to Software

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

Six Sigma



Key ideas:

- Use statistics to measure defects
- Design the process to reduce defects

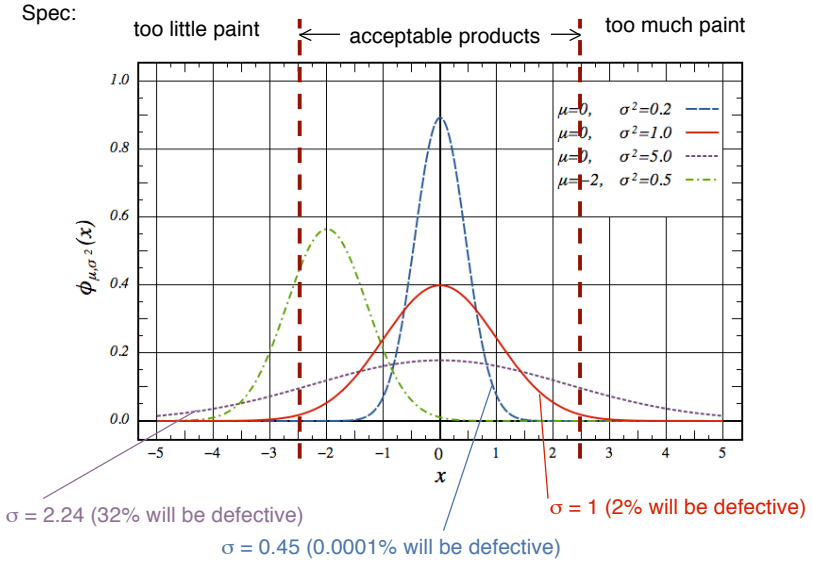
Origin of the term

- 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

Explaining 6-sigma



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