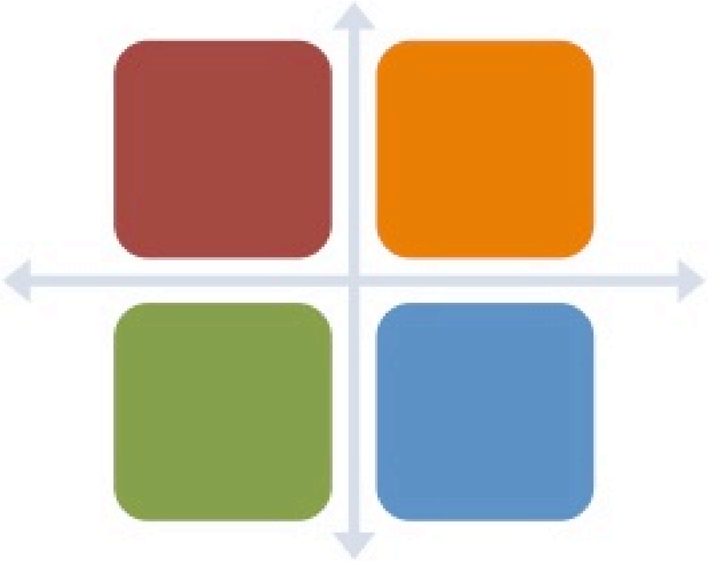


Why is software so bad? (Is it?)

Philippe Kruchten

August 2012

Philippe Kruchten, Ph.D., P.Eng., CSDP

Professor of Software Engineering

NSERC Chair in Design Engineering

Department of Electrical and Computer Engineering

University of British Columbia

Vancouver, BC Canada

pbk@ece.ubc.ca

Founder and president

Kruchten Engineering Services Ltd

Vancouver, BC Canada

philippe@kruchten.com

Why is software so bad?

Why is software so bad?

A survey

Why is software so bad?

“with a little help from my friends”

Bad for society?

or

Bad quality?

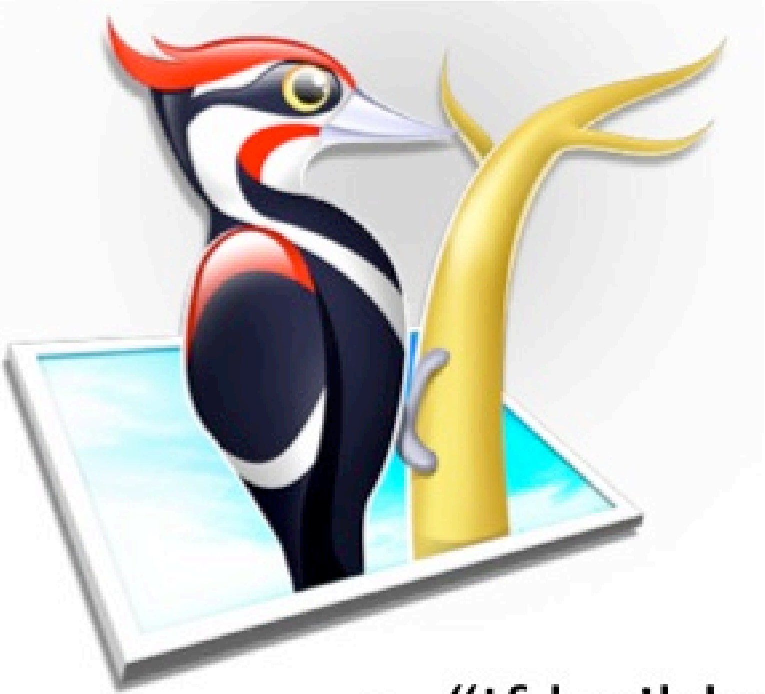
Is gravity bad?

Joe Marasco

Why is the quality of software so low?

Why is the quality of software so low?

Is it?

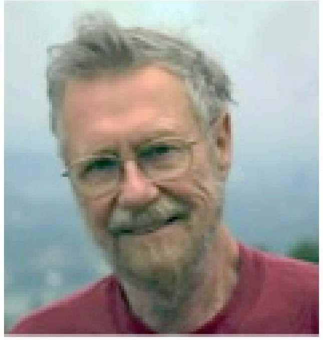


- "If builders built buildings the way programmers wrote programs, the first woodpecker to come along would destroy civilization."

Gerald Weinberg

“The Software Crisis”

- Bauer, NATO Conference, 1968….
- …Dijkstra, 1972



- …Gibbs, *Scientific American*, 1994
- …
- A.G. Yu, 2009

CHAOS reports

- Success: 34% (was 16% in 1994)
  - On time, on budget, with expected functionality
- Failure: 16% (was 31% in 1994)
  - Cancelled somewhere along the lifecycle, etc…
- “Challenged” projects: 51%
  - cost and schedule overrun, reduced functionality,
  - “restarts” is often the cause

Source: Standish Group

Chaos: Not the answer

- They are not looking at *quality* of the result

- They have been discredited
  - Opaque methodology
  - Sampling issues, “shoddy data”
  - Lumping wide range of data into a single pot

Eveleens & Verhoef 2010
Magne Jørgensen
Bob Glass

Neumann's Risk Forum

- Peter G. Neumann: Risks to the public in computers and related systems. *Software Engineering news (SEN).*
  ACM SigSoft

- Since 1985, 6 times a year…
- http://catless.ncl.ac.uk/Risks/

Really actively harmful stuff:

- Therac-25, 1987
- Ariane 5 launcher, 1996
- Denver airport luggage-handling system, 1994
- Airbus A330 crash, 1994
- NASA Mars Climate Orbiter, 1999
- Northeast power blackout, 2003
- …

- … actually not that much.

Aspects of Quality

- If *'accepted'* is the criterion, software is not bad.
- If *'admired'*, then it's as ugly as sin.
- If *'correct'*, then it is merely painful.
- If *'utilitarian'*, it's actually quite good.
- If *'good value for creation cost’,* it's beyond terrible.
- If *'good value for purchase cost'* it's better than diamonds.

P. F. Conroy

Quotes

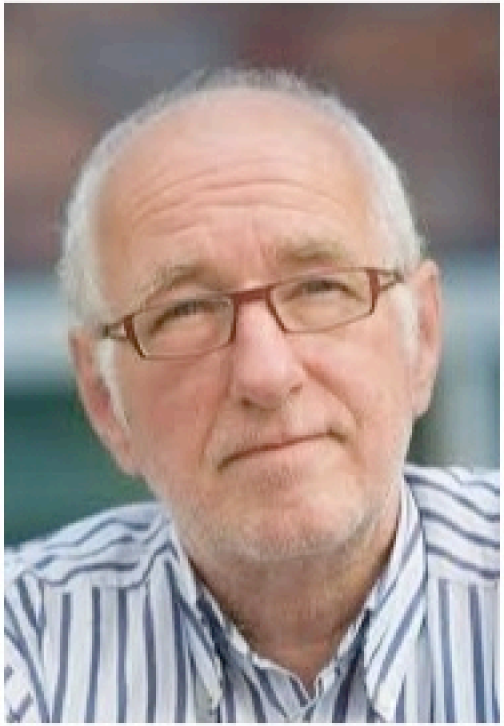
- At the leading edge, there are no subjects, no objects, only the track of Quality ahead, and if you have no formal way of evaluating, no way of acknowledging this Quality, then the train has no way of knowing where to go. Pirsig, 1977

- And what is good, Phaedrus,
  And what is not good—
  Need we ask anyone to tell us these things?

  Plato 370 BC

Aspects of Quality

- Transcendent quality: you like it, or not
- User-based quality: fitness for use
- Product-based quality: relates to attributes of the software
- Manufacturing-based quality: conformance to specs
- Value-based quality: profit and cost



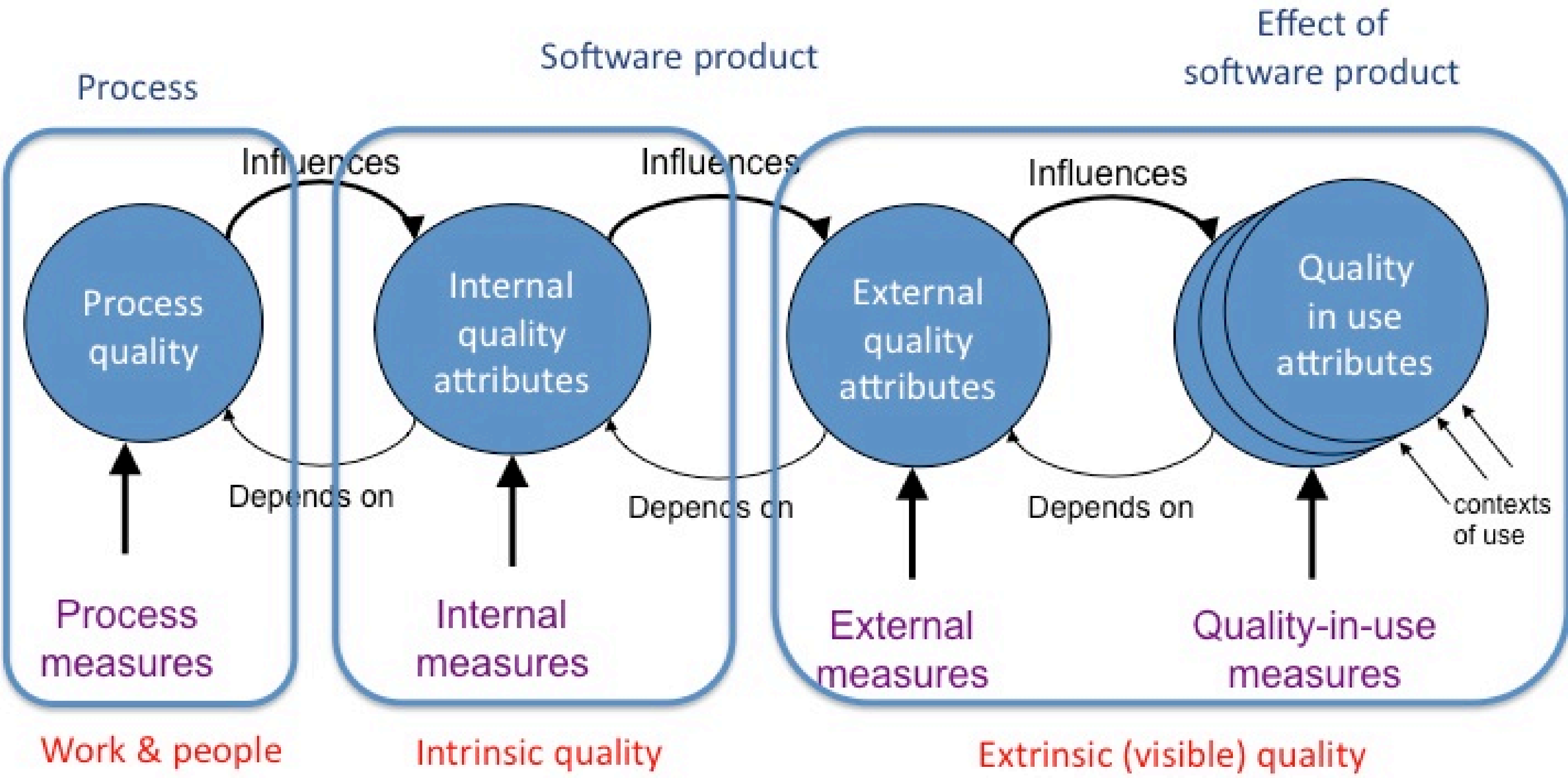
Hans van Vliet, VU Amsterdam

Aspects of Quality…

- **User-based quality: fitness for use**
  - Bad analysis of requirements
  - Misunderstanding of whom the user really is
  - Changing needs
- **Value-based quality: profit and cost**
  - Bad procurement or bad project management
  - Low bids, bad risk management, deployment or conversion costs

Hans van Vliet, VU Amsterdam

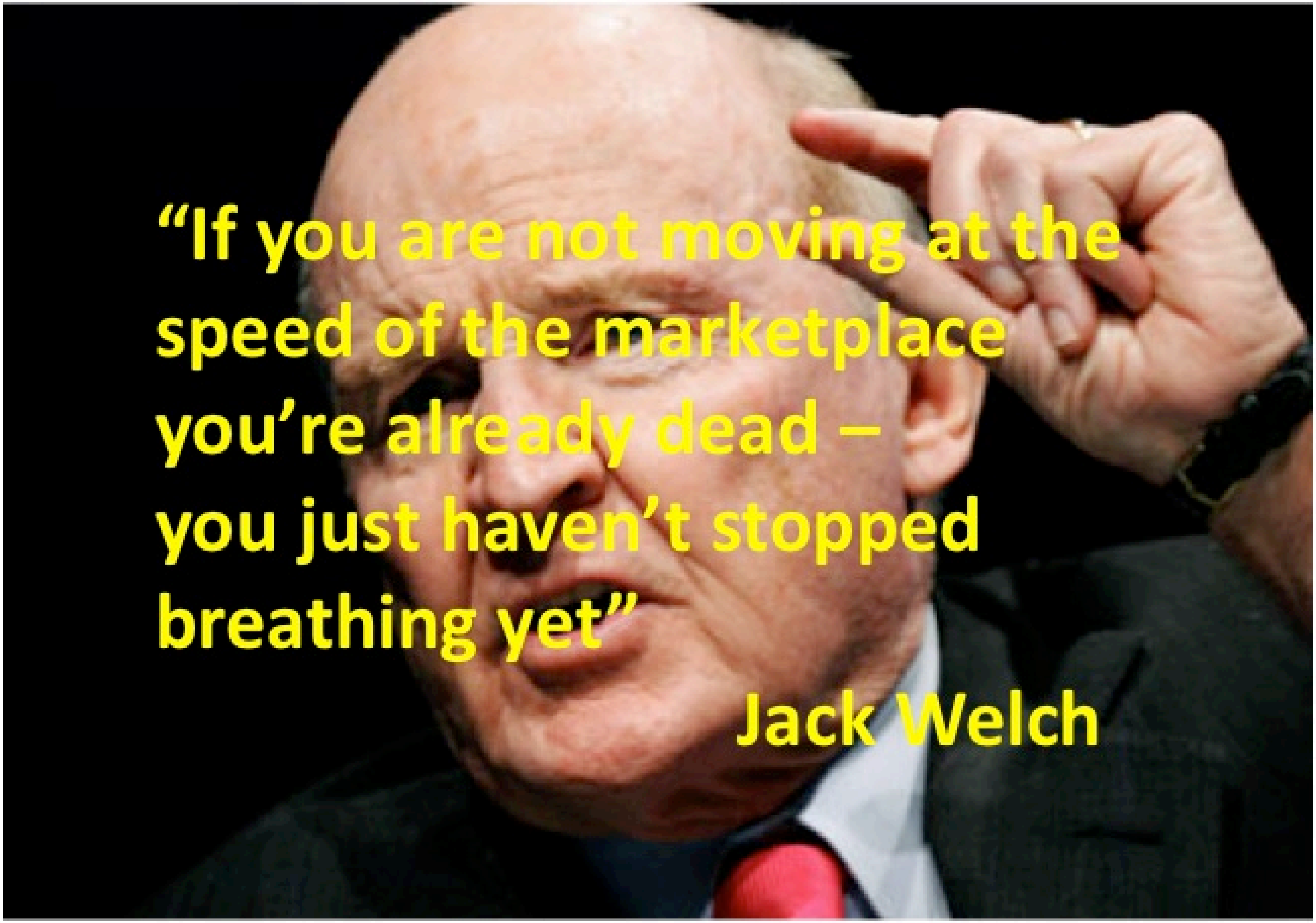
The ISO 9126 Quality Model



Root Causes of Low Software Quality?

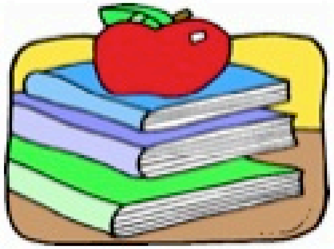
Many hypotheses:

- Time and budget pressures
- Software engineering education
- Lack of domain knowledge
- No systematic assessment of quality
- No systematic analysis of failures
- The lure of complexity
- General public education



“If you are not moving at the speed of the marketplace you’re already dead – you just haven’t stopped breathing yet”

Jack Welch

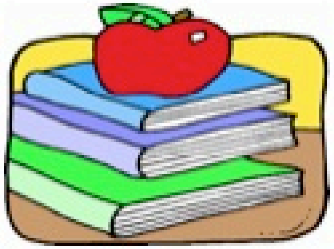


Education

- Computer science too focused on itself
  - The code, computer, the technologies
  - Not enough software **Engineering …**

"Computer science graduates are often expected to have an understanding of many issues surrounding the interaction between humans and computers when they are in the workplace. However, most computer science graduates are ill equipped to deal with such issues, and could benefit if they were given more consideration in the university curriculum."

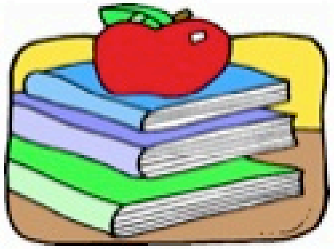
Paul Parsons, U. Western Ont.



Education

- Quality…
  - ISO9126, ISO 250xx series, IEEE 730,
- Complete lifecycle
  - Design, coding, testing,….

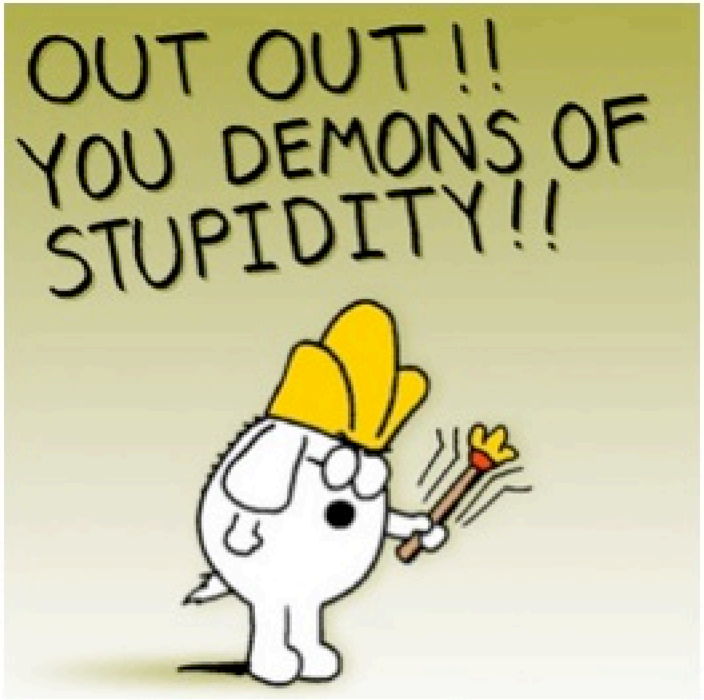
  But what about:
  - Deployment, update, training, conversion, maintenance, interoperation…



Application domain

- Software systems, yes, but applied to what?
- Interdisciplinary teams
- Cognitive biases
  - Anchoring bias
  - Representativeness bias
  - “GroupThink”

Getting some help?



Getting some help?

Government

or

Market

or

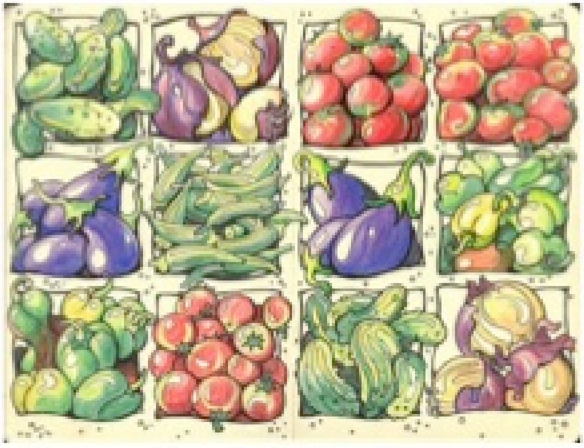
Profession

Government

- Quality by law?

- Only in very narrow domains
  - Safety critical systems, or “software that kills”

Market

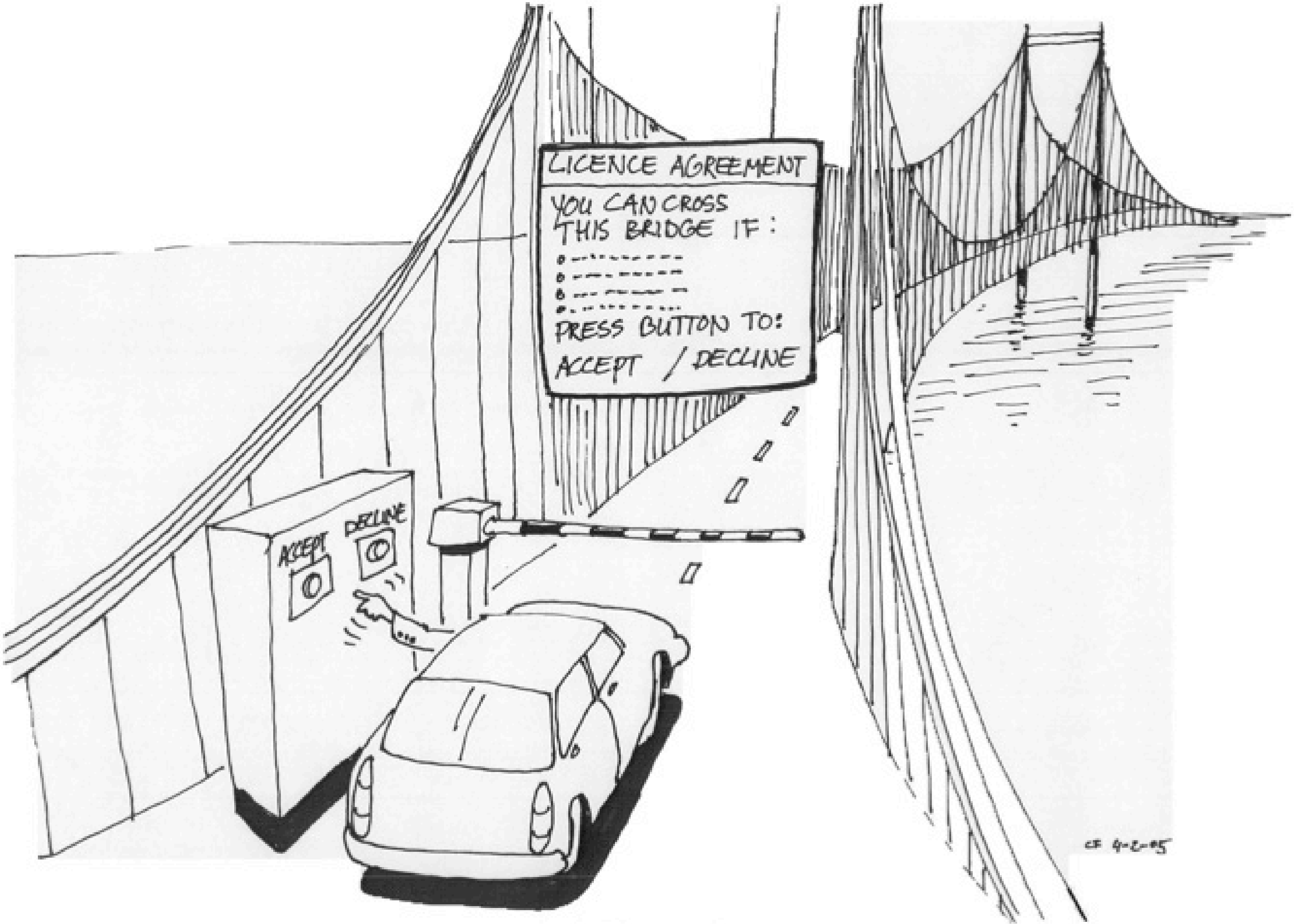


- Let the market decide
- When users have choices,
  they vote with their purchasing power.

- Globalization, outsourcing impact??

- Government: insure free market
  - Monopolies, trade barriers, abuse of patents...

- Vendor initiative: ex. AppStore


LICENCE AGREEMENT
YOU CAN CROSS THIS BRIDGE IF:
PRESS BUTTON TO:
ACCEPT / DECLINE
ACCEPT
DECLINE
CF 4-2-05

Profession

- Get organized

- Certifications
- Accreditation of education programs

- Licensing: professional software engineers
  - More than certification: license to practice in some areas
- Ethical behaviour and practices

We can raise the bar.

IEEE
computer
society

IEEE
Computer Society
Certified
Software
Development
Professional
csdp

I.S.P.

PMP
PMI.
Project
Management
Professional

A P
E G
Professional Engineers
and Geoscientists of BC

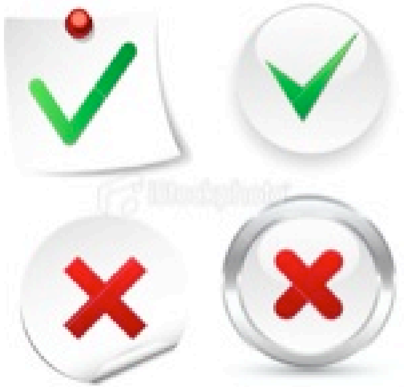
Serious Introspection Needed

- Low level of accountability ☹
- Running away…. ☹
- Better understanding of software quality

- Objective metrics
- Willingness to analyse and publish failure analysis

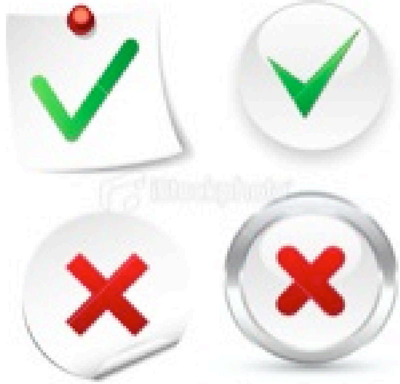
More quotes

- I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of meager and unsatisfactory kind. (Lord Kelvin, 1900)
- The difficulty in defining quality is to translate future needs into measurable characteristics, so that a product can be designed and turned out to give satisfaction at a price the user will pay. (Shewhart).



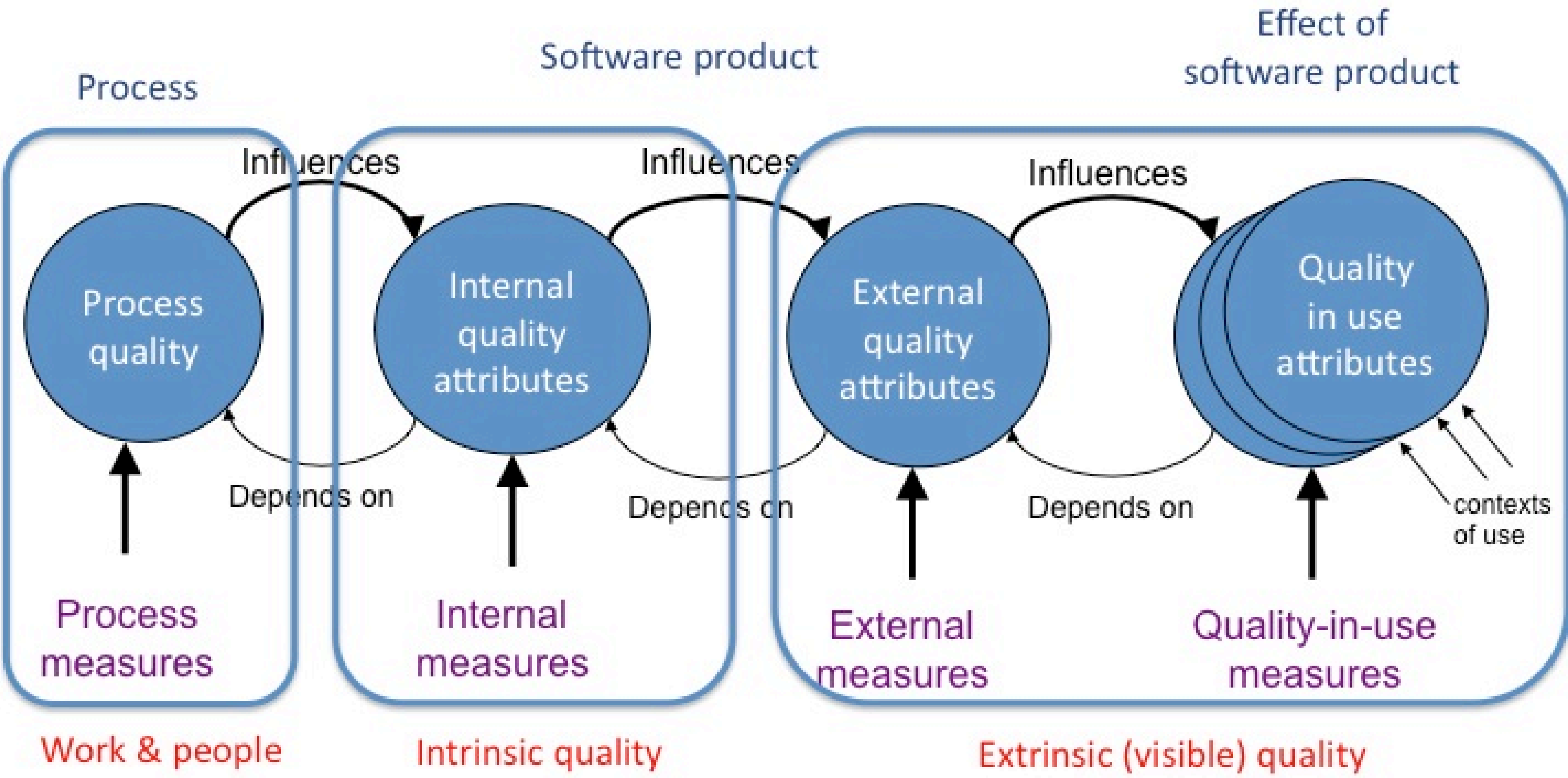
Quality > Verification > Testing

SW Verification

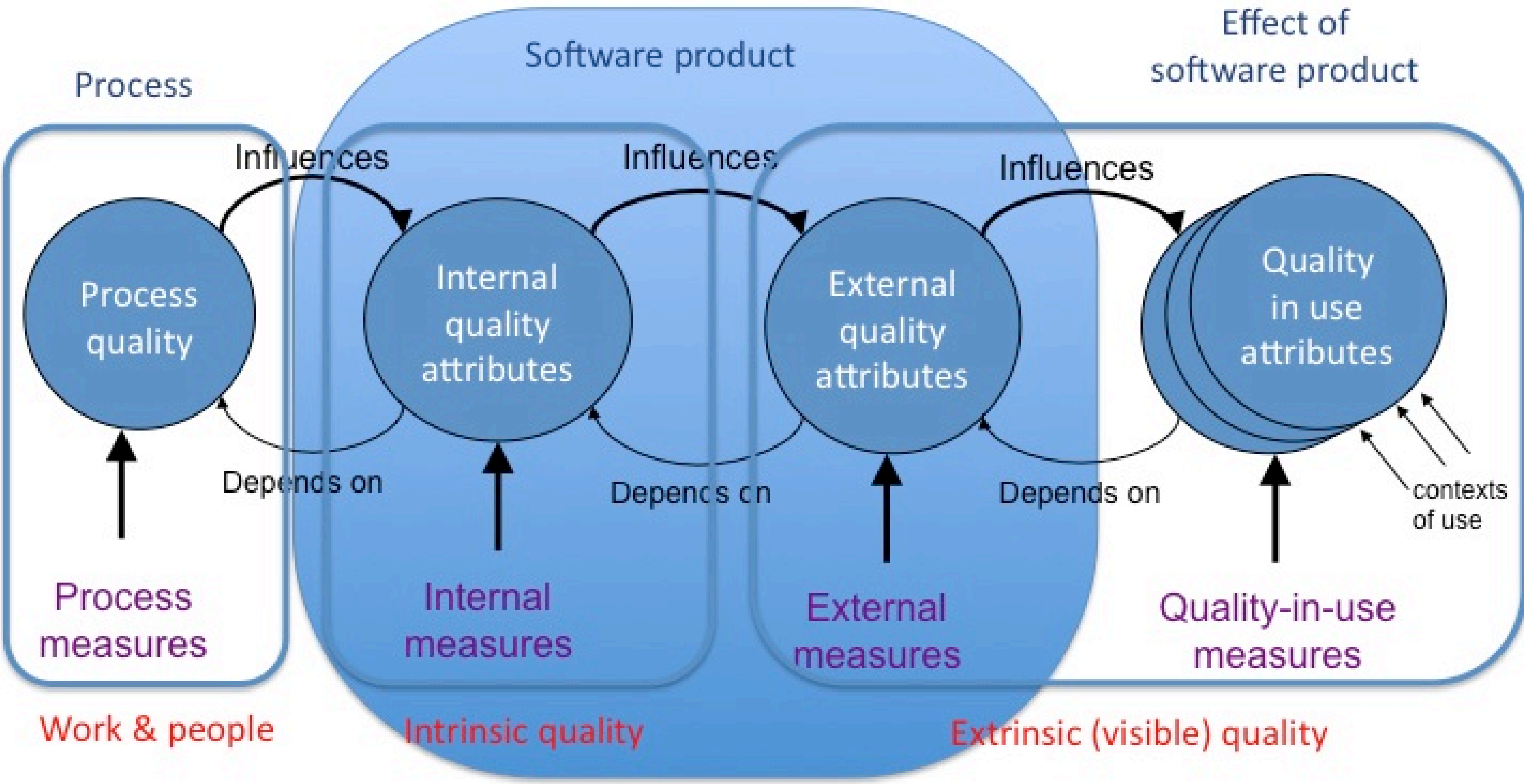


- Static verification
  - Coding convention
  - Bad practices, “code smells”
  - Non testable quality attributes
  - Software metrics: e.g., complexity
  - Formal proofs
- Dynamic verification
  - Test
    - Small granularity: unit test
    - Large granularity: system test, acceptance test
    - Functional / other (performance, load)
  - Other forms of experimentation
    - User observation, surveys

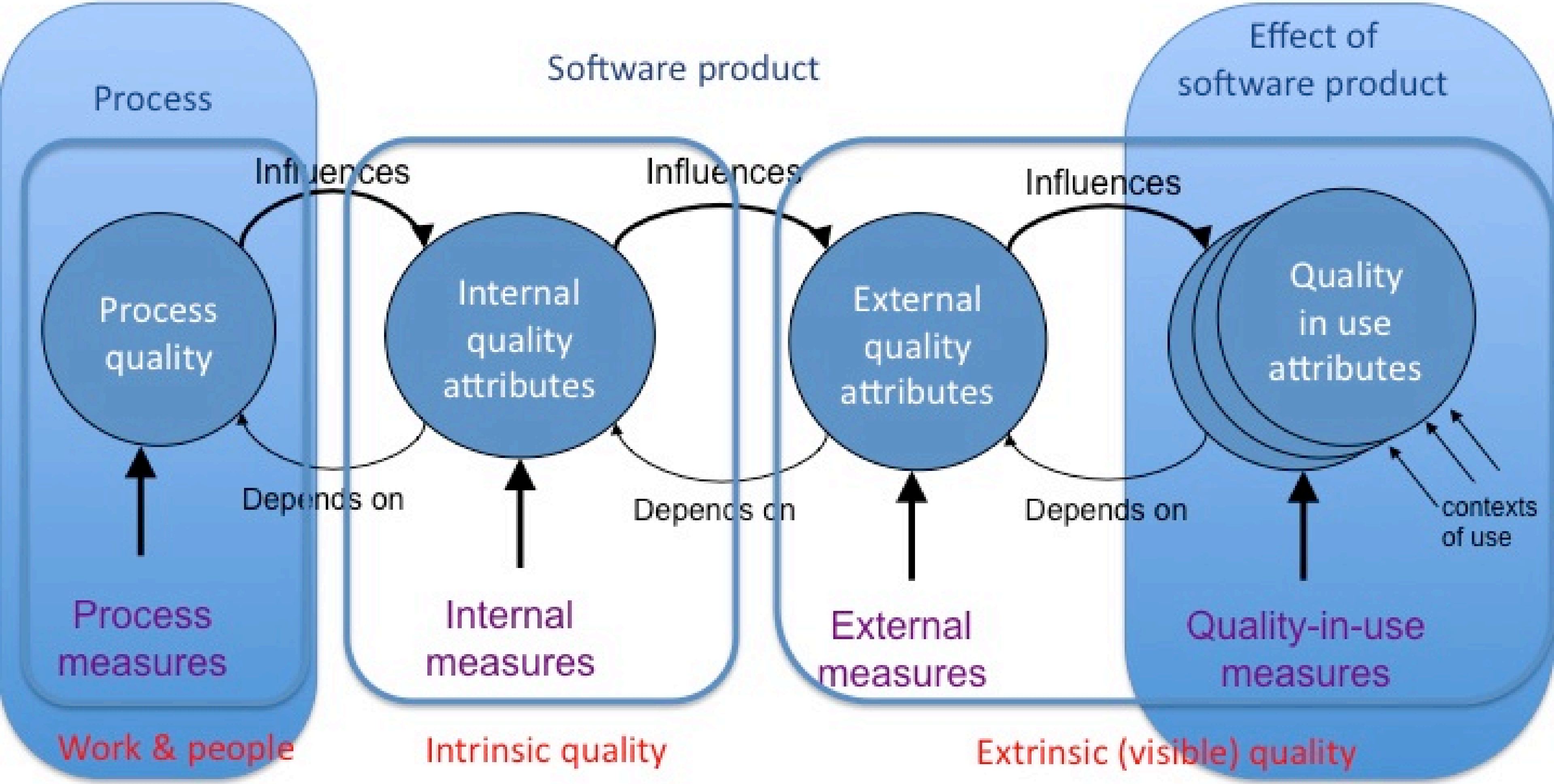
The ISO 9126 Quality Model



The ISO 9126 Quality Model

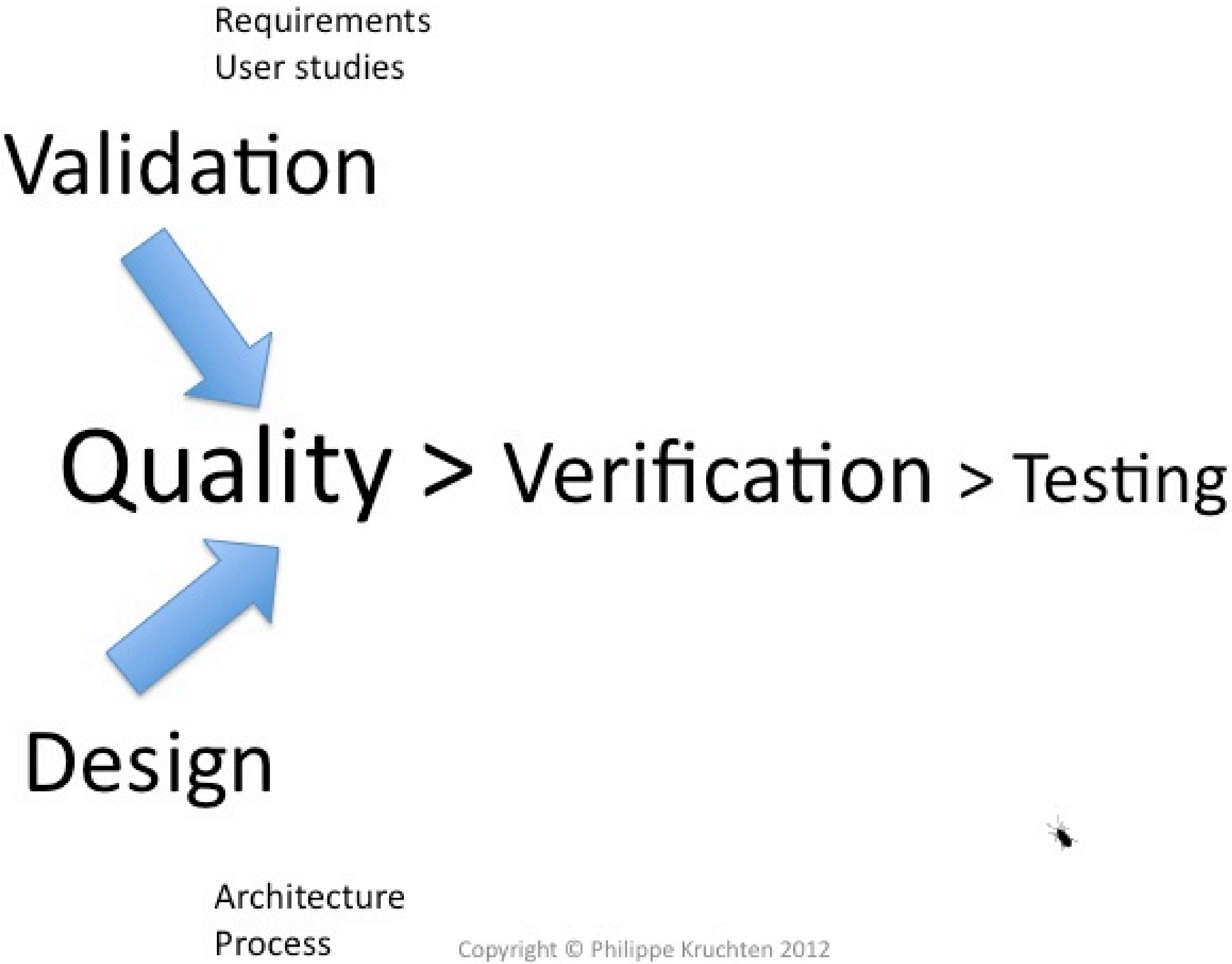


Shifting our focus



Shifting our focus on the people

- Development process
  - Decisions making
  - Communication and collaboration
- Usage
  - Understand "quality in use"
    /= defect-free
- Cognitive aspects
  - Knowledge, understanding, memory, mental models, biases…
- Test-Driven Design?



Requirements
User studies

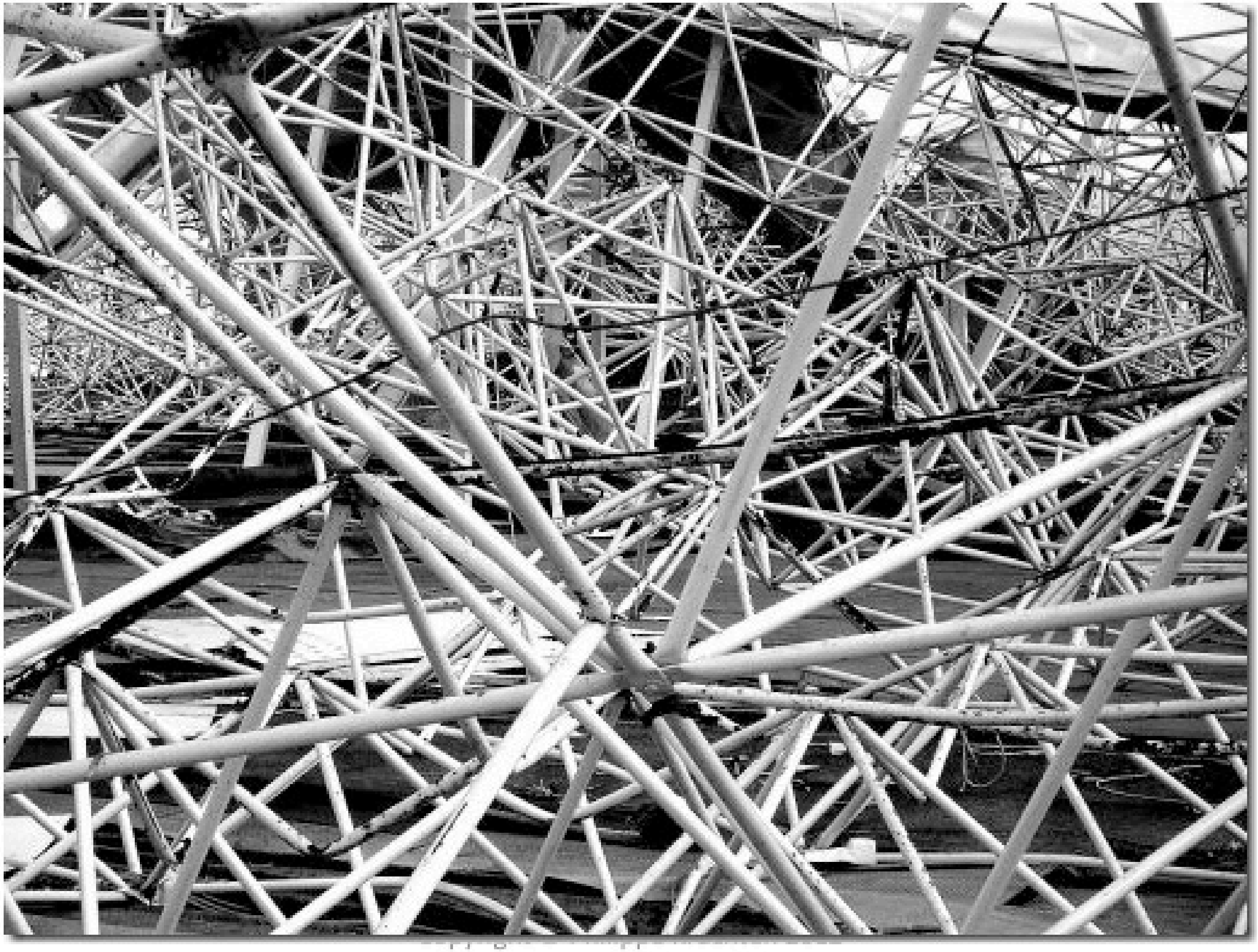
Validation

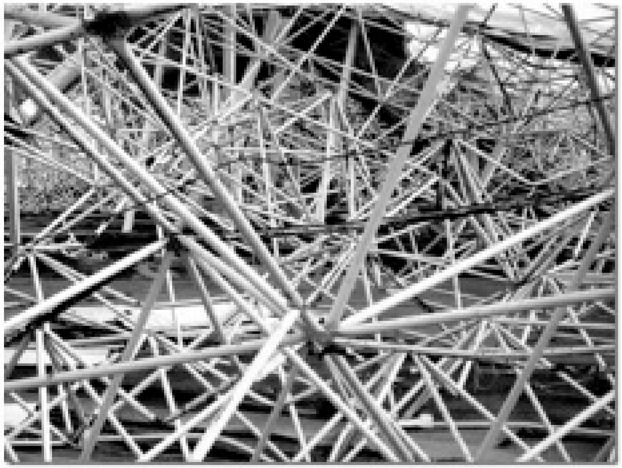
Quality > Verification > Testing

Design

Architecture
Process

Complexity





Scale

Many *things*

Diversity

many different kinds of *things*

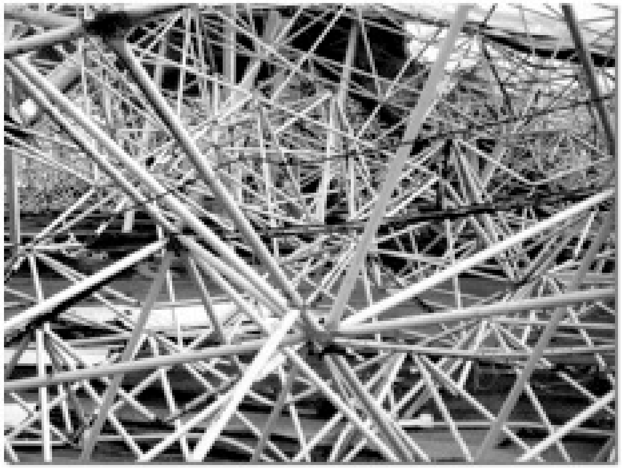
Interdependencies

between all these many *things*

Kinds of Things = features, services, LoC, classes, stakeholders, developers, sites, technologies, managers, ….

Intrinsic complexity

essential complexity

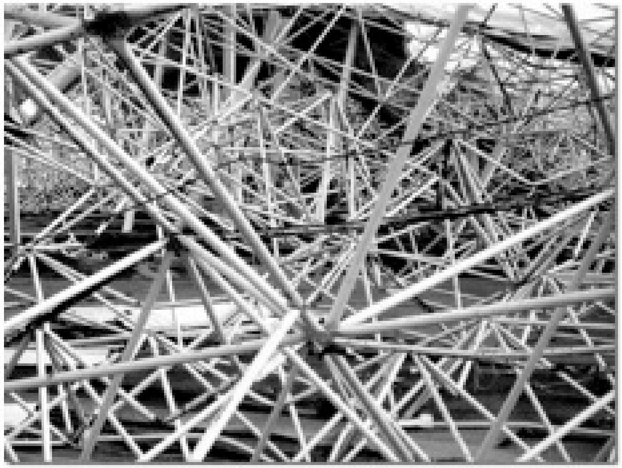


Extrinsic complexity

accidental complexity

Perceived complexity

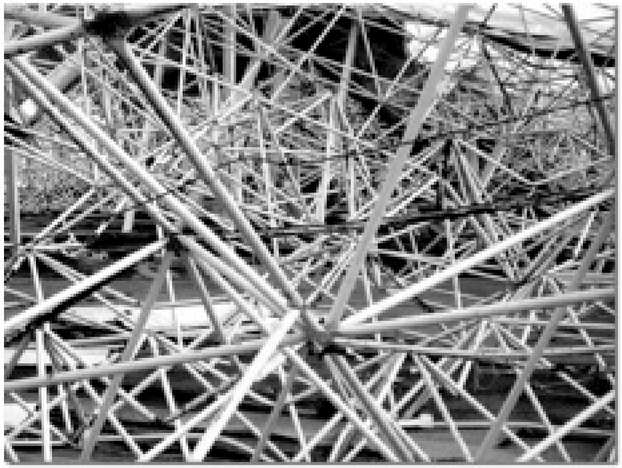
culture, familiarity, education



Determinism

Visibility

Complex /= confusing

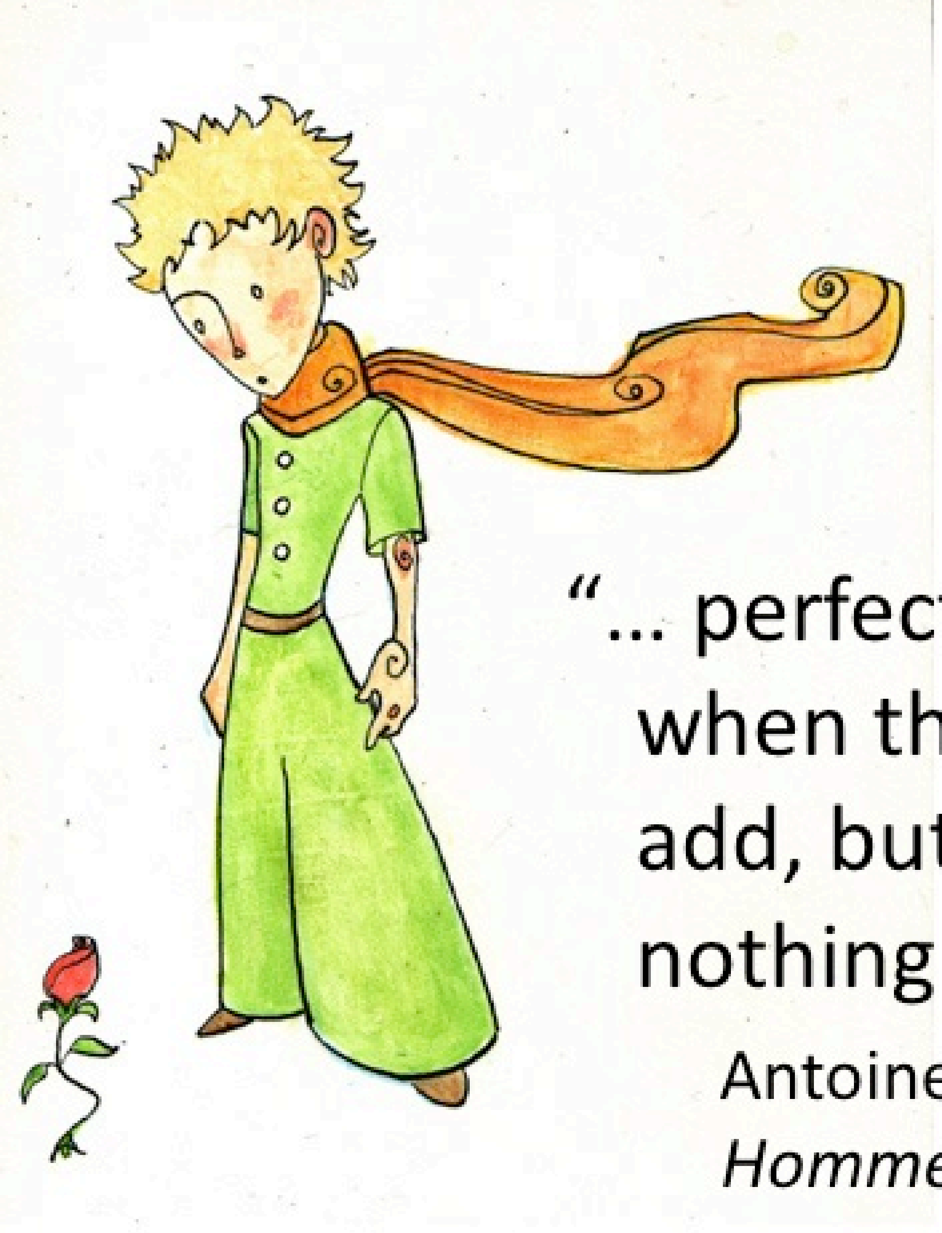


Complicated

Complex

Chaotic

Software bloat !



"… perfection is achieved not when there is nothing left to add, but when there is nothing left to take away."

Antoine de St. Exupéry, *Terre des Hommes,* 1939, chap.3

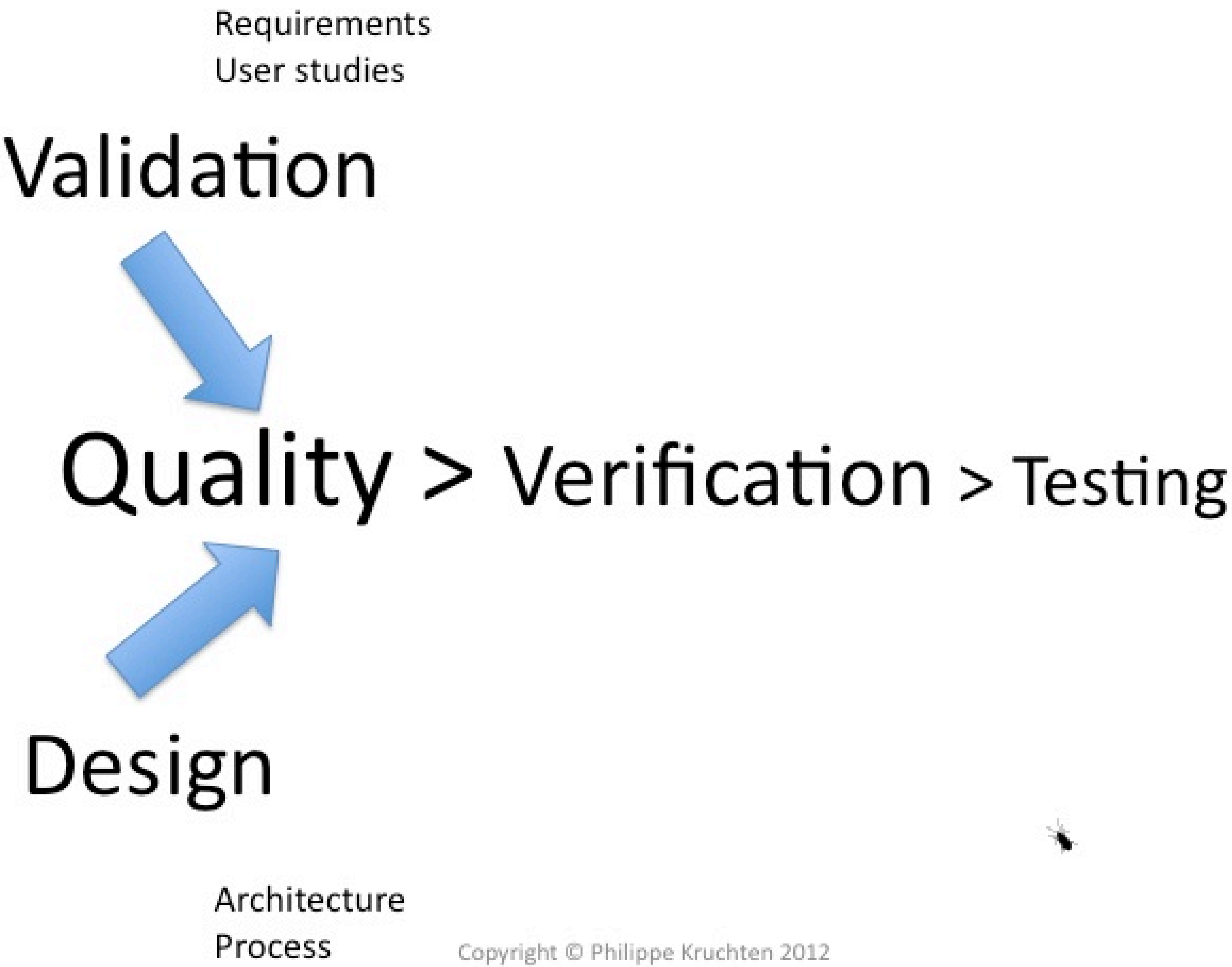
General public education

If the public at large understood more about software (or the systems that run it):

- They would be more discriminate in their choices
- Some perception of low quality would be reduced
- They would raise the bar

Summary

- Most software is good
  - Good for society, and good enough quality
- Some software is bad
  - market dynamics, education, quasi-monopolies
- Culprits?
  - No necessarily the developers
- Improvements?
  - Education mostly, both public and developers
  - Better understanding of quality (based on use and value)
  - More quality /= more testing



Requirements
User studies

Validation

Quality > Verification > Testing

Design

Architecture
Process

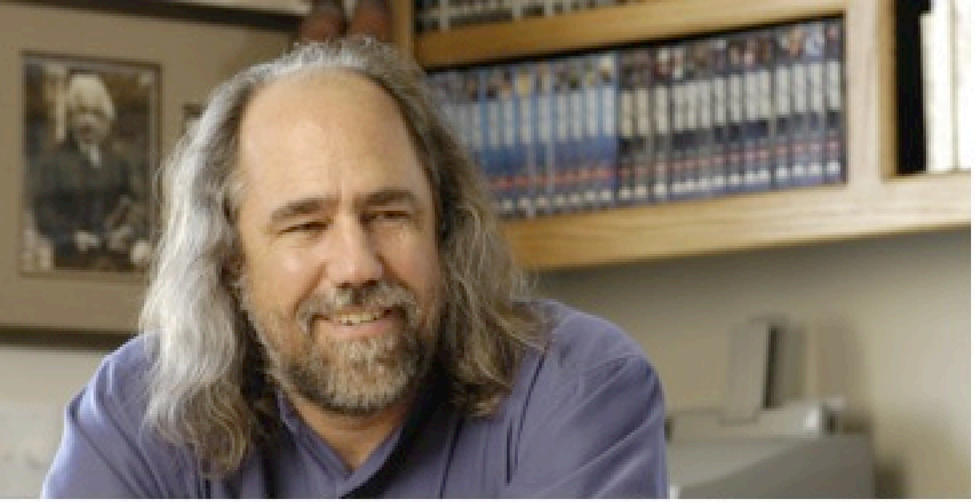
Introspection

"The continuing self-flagellation about poor quality is a kind of modernist indulgence that attempts to purges the soul by admission of guilt. We seem to crave purity, even when it's not truthful. That's a mistake. It's Occam's razor burn. When we don't understand things, we're better to admit it than to pretend we do. But, like our medical colleagues, we can't just wait for perfect knowledge to begin applying it. Moreover, because our subject is supporting all nature of human pursuit, so the map becomes our territory."

Robert Biddle, U. Ottawa

"My short answer is that "software" is better than most people think. We take for granted a lot of very sophisticated magic that goes on behind the scenes to make it all work. As with any product or service that is used in high volume, there are occasional failures as well as things that don't work very well much of the time. In particular, we are very hard on new software. I have come to judge the quality of a software product only after it has been in the field for twenty years; before that, it is still not mature. [...] Systems with solid architecture that can be incrementally improved with advances in underlying technology are from my point of view the modern-day version of cathedrals. [...] we need to judge our construction abilities on our cathedrals and not on a sampling of doghouses."

Joe Marasco



- While I may declare that software is “bad” I must temper my lament by observing that I say that largely because I know we can do much better. … However one measures good or bad, I celebrate the reality that software has changed humanity in profound and irreversible ways and that in this self-created cosmos of ones and zeros, there is a fierce and elegant beauty within the mind and matter of the software-intensive systems we have created.

Grady Booch, IBM Research

Thanks, guys

With a little help
from my friends…