# What is software quality?

UA/DETI/TQS
Ilídio Oliveira (ico@ua.pt)
v2019-02-12.

universidade de aveiro
departamento de eletrónica, telecomunicações e informática
deti

# “Classic” software-related failures

THERAC-25

- software errors caused radiation overexposures (3 deaths, 3 severe injuries) (1985-87)

ESA ARIANE-5

- code for previous models failed when reused in new operating conditions (1996)

London “112”

- emergency dispatch entered a chain of decline (1992)

# Multibancos ficaram com 2,7 milhões de euros em levantamentos feitos em 2012 (a

11 F

Fun
perf
os n
dos
retic

Conf
reco
entid
devo
debit
afect
pass

rimir

## Auditoria detectou 21 falhas de segurança no sistema informát

MARIANA OLIVEIRA

## Tribunal de Contas detecta falhas no sistema informático da Direcção-Geral de Alfândegas

LUSA 27/11/2006 - 13:55

## Consulados portugueses "intermitentes" por falhas no sistema informático

NATÁLIA FARIA 21/11/2012 - 15:00

Em duas horas, auditores conseguiram descobrir milhares de passwords FOTO: FERNANDO VELUDO/NFACTOS

TÓPICOS (/TOPICOS)

Justiça (/justica)

Mini
adia

Uma
dete

## Sistema de contagem electrónico falha no Parlamento e deputados votam só de pé

ROMANA BORJA-SANTOS 26/03/2009 - 17:19

por várias aplicações. Do conjunto, seis deficiências foram classificadas com risco

## BE diz que falha informática permite acesso a dados pessoais nas listas de professores

LUSA e PUBLICO.PT 18/06/2004 - 20:00

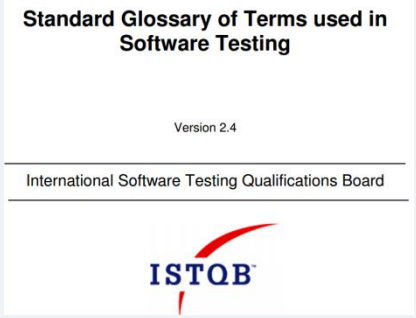
# Software drives the world...
# or: living among bugs


http://catless.ncl.ac.uk/Risks/

We can recognize the lack of quality...

Can we be systematic about the elements of software quality?

# Software quality defined



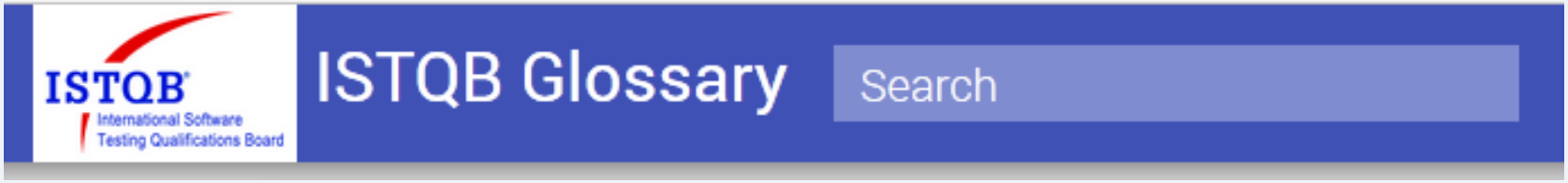

## IEEE

- Quality:  The degree to which a system, component or process meets customer or user needs or expectations (IEEE-610.12-1990, Standard Glossary)
- Quality metric: a function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute.

## ISTQB

- Quality:  The degree to which a component, system or process  meets specified requirements and/or user/customer needs and expectations. [After IEEE 610]
- Software quality: the totality of functionality and features of a software product that bear on its ability to satisfy stated or implied needs. [After ISO 9126]

# Essential vocabulary in software quality




https://www.istqb.org/downloads/glossary.html

# Software quality working definition

The degree to which a software product

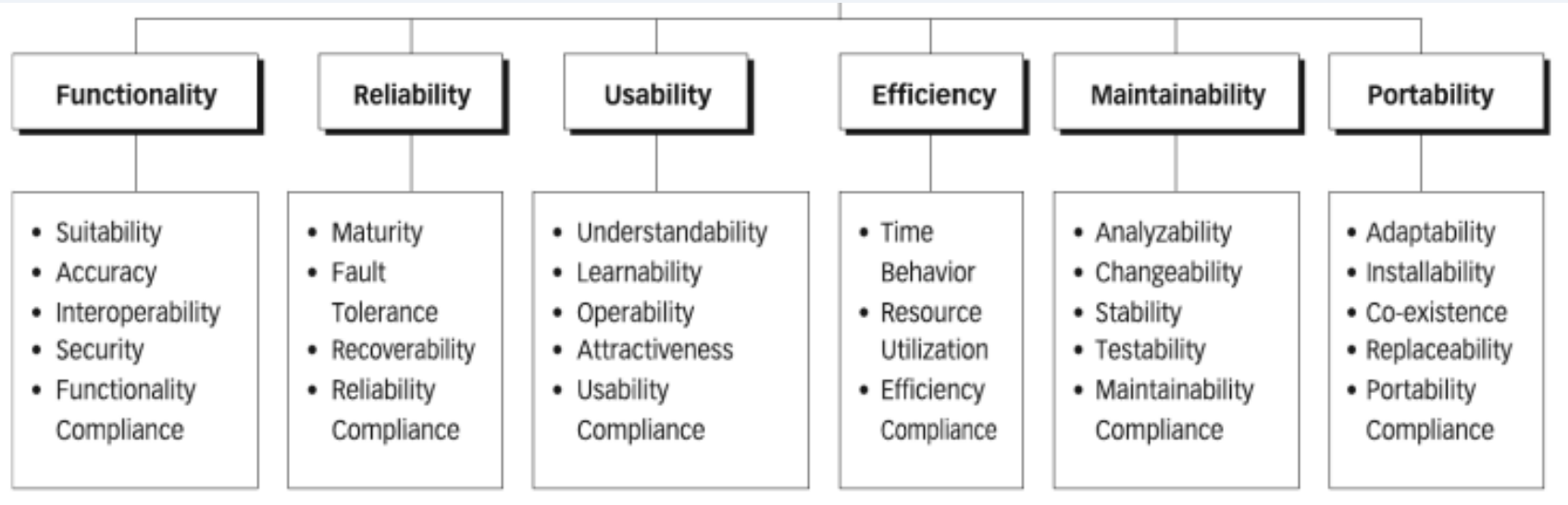
- fulfills the defined functional requirements
- meets the expectations of the customer/users w.r.t. to the system attributes
- meets the best practices of the industry.

# “Popular” views on quality factors

| | |
|---|---|
| *ISO 9126* | Functionality, reliability, usability, efficiency, maintainability, portability |
| *McCall and Matsumoto* | Integrity, correctness, reliability, usability, efficiency, maintainability, testability, flexibility, portability, interoperability, reusability |
| *IEEE 830* | Performance, reliability, availability, security, maintainability, portability |
| *ESA PSS-05* | Performance, documentation, quality, safety, reliability, maintainability |

ISO/IEC 25010:2011 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models

# ISO 9126-1 Quality Model




See: Table of Characteristics and sub-characteristics for the ISO 9126-1 Quality Model

# ISO 9126 Quality model: quality factors

## Functionality

- Suitability: the software is suitable for the intended tasks
- Accuracy: results/outputs are correct and precise
- Interoperability: ability of a software component to interact with other components or systems
- Security: resist to unintended access or modifications

## Non-Functional

- Reliability: amount of time that the software is available for use
- Usability: degree to which the software is easy to use
- Efficiency: software makes optimal use of system resources
- Maintainability: ease of repair/correct
- Portability: ease with which the software can be transposed from one environment to another

# Internal and external quality factors (McConnell, Code Complete)

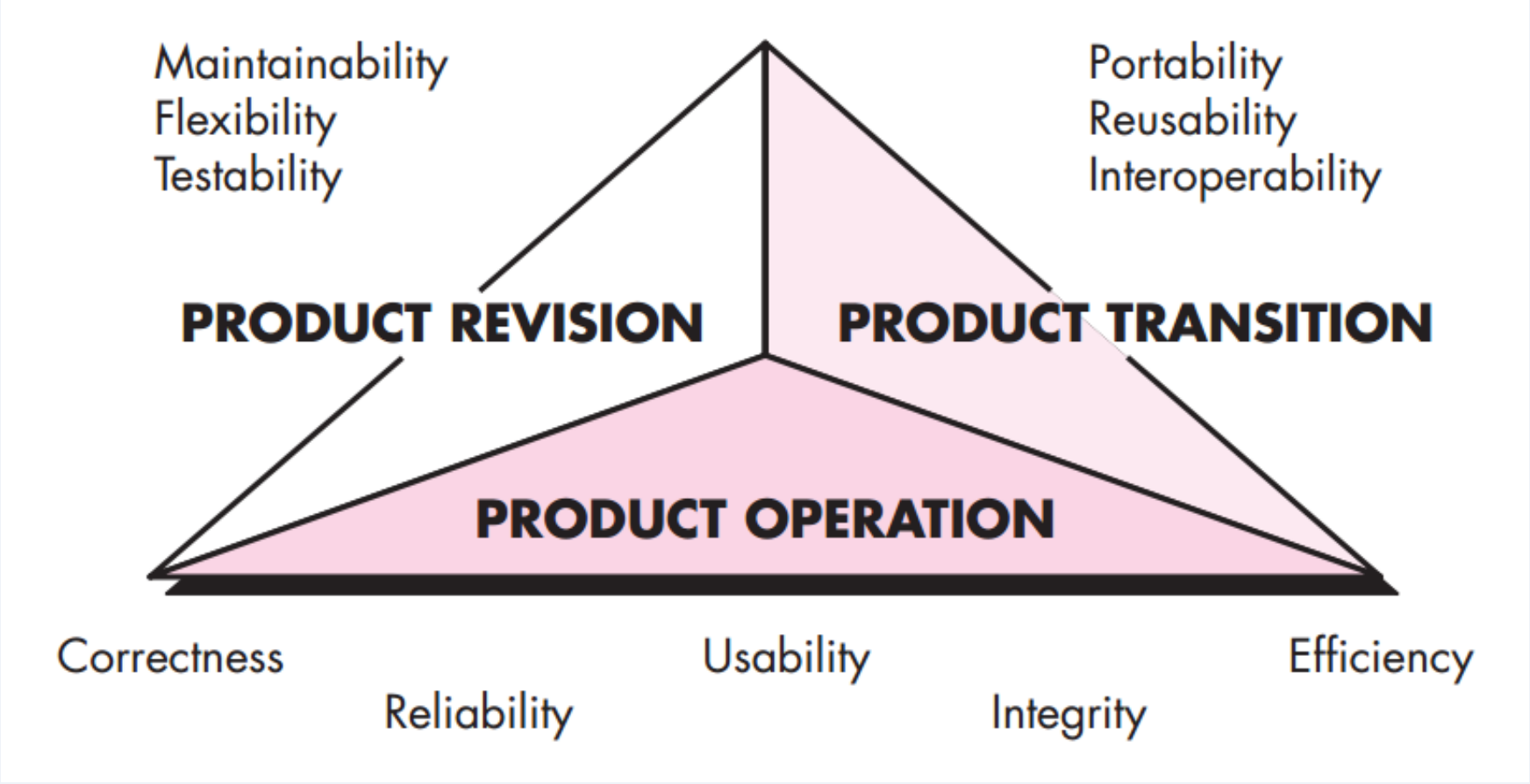
## External: those that the users/customers are aware of

- Correctness (constructed free from faults)
- Usability
- Efficiency (use of resources)
- Reliability (long times between failures)
- Integrity (state is always consistent)
- Adaptability (usable in new contexts)
- Robustness

## Internal: those that the programmer is aware of

- Maintainability.
- Flexibility + portability (fits new uses or environments)
- Reusability (parts can be reused)
- Readability + understandability
- Testability

# McCall's software quality factors [McC77]



# Being systematic about software quality

## SOFTWARE QUALITY ASSURANCE

set of activities (methodology) to control and monitor the software development process to attain the project goals with a certain level of confidence in quality terms.

## SOFTWARE QUALITY CONTROL

evaluates if software products are within the defined quality standards resorting to inspections and different kinds of tests

## SQA != SQC

SQC is aims at detecting and fixing defects. SQA aims at preventing  them.

# Practices of SQA

Testing

Software configuration management

- Versions management

Code improvement

- Reviews, shared practices, static analysis,…

Issue tracking and task management

Continuous integration

Formal methods (out of scope for our course)

VERIFICATION: ARE WE DOING THE SYSTEM IN THE RIGHT WAY?

Check work products against their specifications

Check modules consistency

Check against industry best practices

...

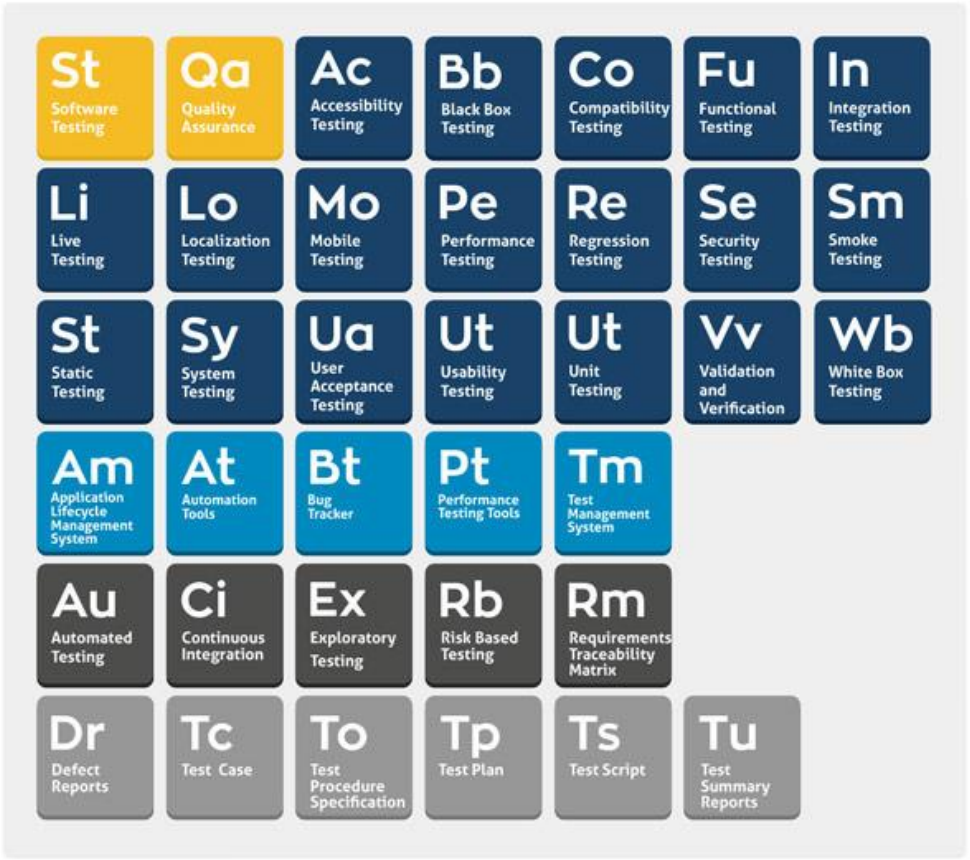
VALIDATION: ARE WE DOING THE RIGHT SYSTEM?

Check work-products against the user needs and expectations

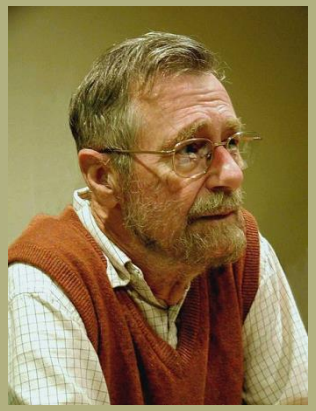
# Elements of Software Testing

Software testing is an essential service for any business implementing a new system or updating an old one. Software development technologies are constantly changing, but the essential elements for successful software testing remain the same

| General Definitions | Software Testing Types | Testing Tools | Testing Strategies | Testing Documentation |
|---|---|---|---|---|




https://www.qualitestgroup.com/blog/testing-as-a-lifestyle/elements-of-software-testing/



Program testing can be used to show the presence of bugs, but never to show their absence!  (1970)

Edsger W. Dijkstra,

# Practices of SQA

Testing

Software configuration management

- Versions management

Code improvement

- Reviews, shared practices, static analysis,…

Issue tracking and task management

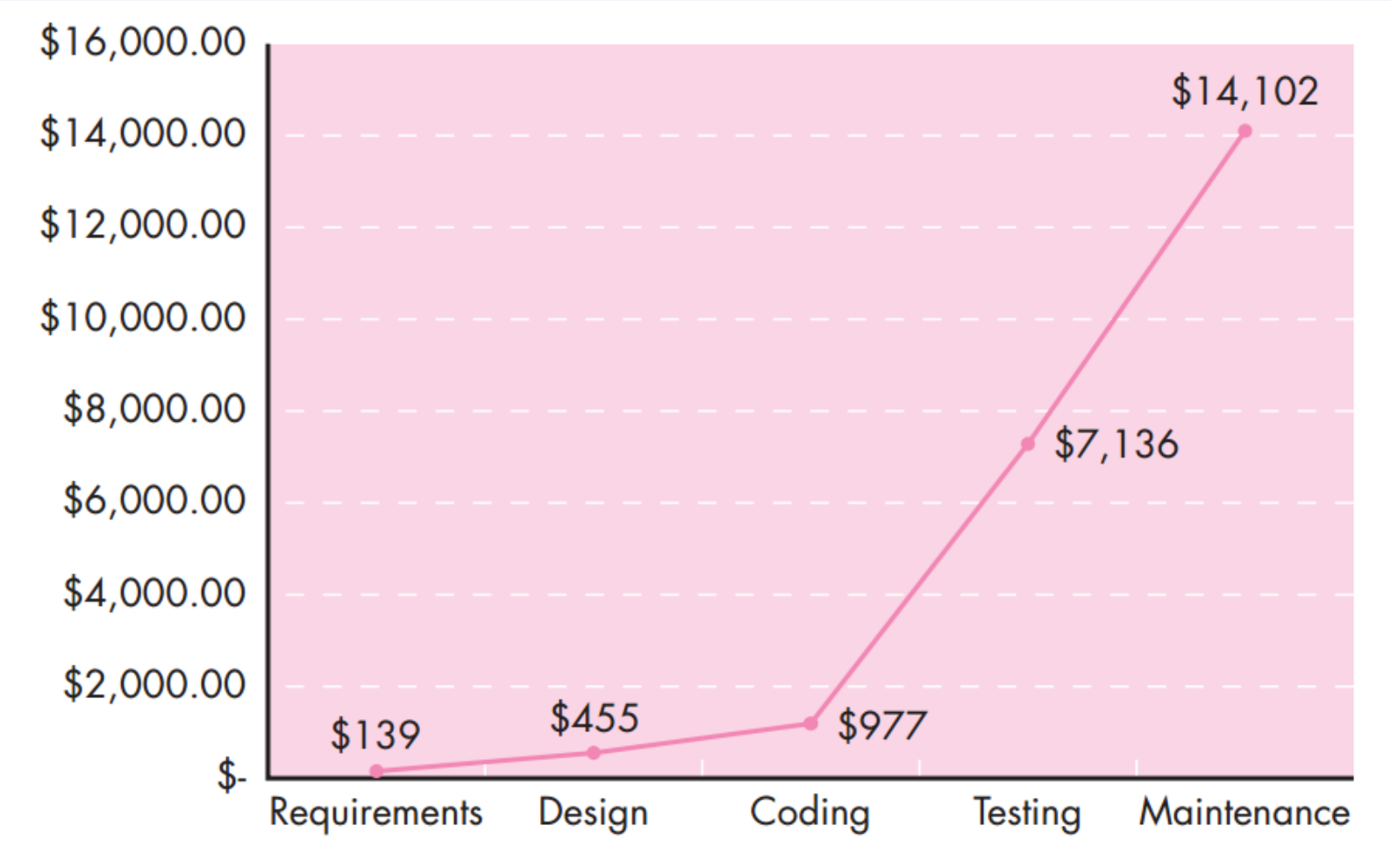
Continuous integration

Formal methods (out of scope for our course)

| Goal | Attribute | Metric |
|---|---|---|
| **Requirement quality** | Ambiguity | Number of ambiguous modifiers (e.g., many, large, human-friendly) |
| | Completeness | Number of TBA, TBD |
| | Understandability | Number of sections/subsections |
| | Volatility | Number of changes per requirement |
| | | Time (by activity) when change is requested |
| | Traceability | Number of requirements not traceable to design/code |
| | Model clarity | Number of UML models |
| | | Number of descriptive pages per model |
| | | Number of UML errors |
| **Design quality** | Architectural integrity | Existence of architectural model |
| | Component completeness | Number of components that trace to architectural model |
| | | Complexity of procedural design |
| | Interface complexity | Average number of pick to get to a typical function or content |
| | | Layout appropriateness |
| | Patterns | Number of patterns used |
| **Code quality** | Complexity | Cyclomatic complexity |
| | Maintainability | Design factors (Chapter 8) |
| | Understandability | Percent internal comments |
| | | Variable naming conventions |
| | Reusability | Percent reused components |
| | Documentation | Readability index |
| **QC effectiveness** | Resource allocation | Staff hour percentage per activity |
| | Completion rate | Actual vs. budgeted completion time |
| | Review effectiveness | See review metrics (Chapter 14) |
| | Testing effectiveness | Number of errors found and criticality |
| | | Effort required to correct an error |
| | | Origin of error |

Adapted from: Hyatt, L., and L. Rosenberg, “A Software Quality Model and Metrics for Identifying Project Risks and Assessing Software Quality,” NASA SATC, 1996, available at http://satc.gsfc.nasa.gov/support/STC_APR96/qualtiy/stc_qual.html

# The cost of correcting an error raises exponentially along the sw lifecycle

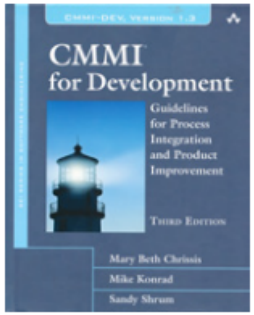



Boehm, B., and V. Basili, "Software Defect Reduction Top 10 List," IEEE Computer, vol. 34, no. 1, January 2001, pp. 135–137. http://doi.ieeecomputersociety.org/10.1109/2.962984

# Focus on the process quality

## HOW IT CAN HELP

The CMMI-DEV model provides guidance for improving your organization's capability to develop quality products and services that meet the needs of customers and end users. These best practices will help your organization improve efficiency, speed, and product quality fueled by a lower number of defects.




## KEY PROCESS AREAS

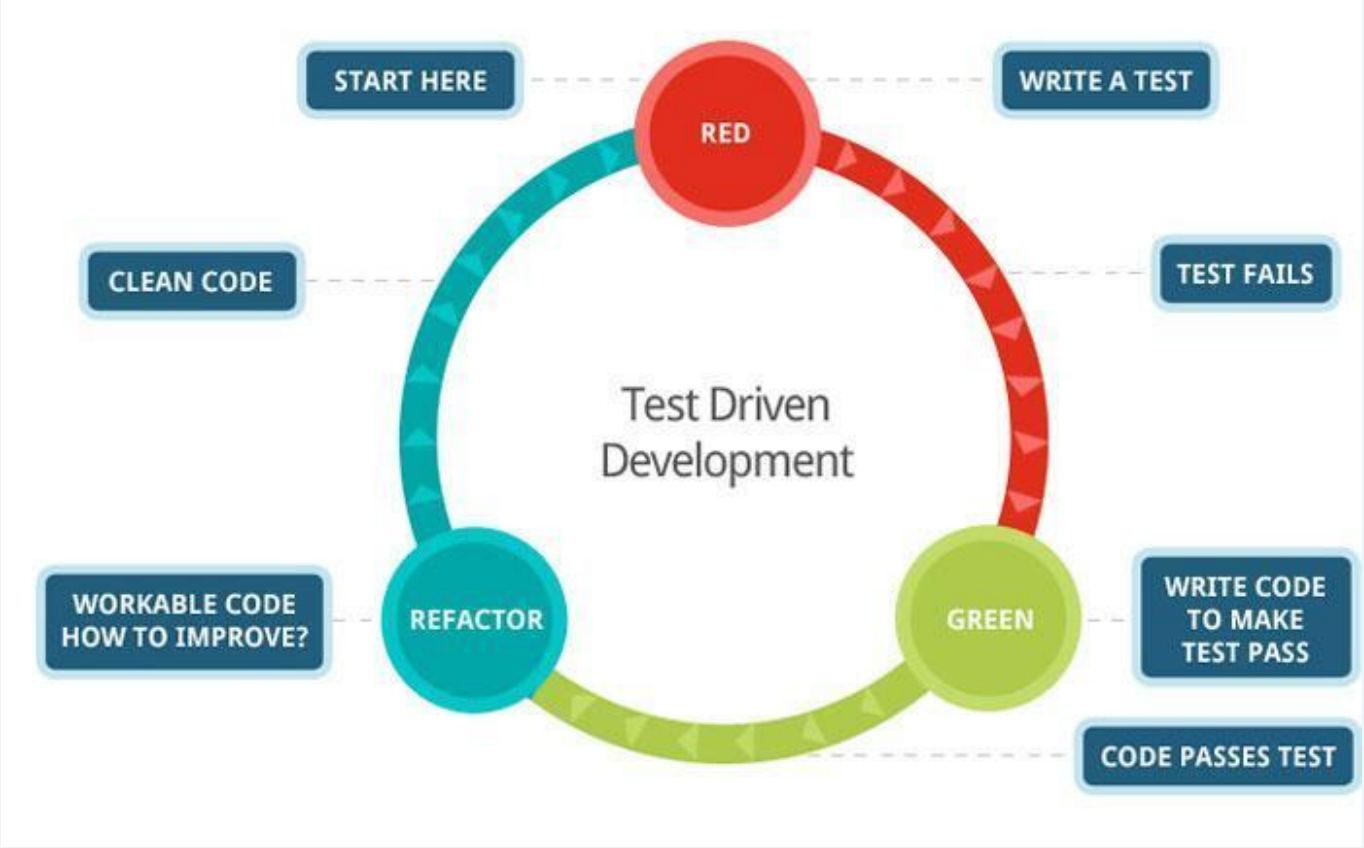
- Product Integration
- Requirements Development
- Technical Solution
- Validation
- Verification

**"We identified CMMI for Development as one of the most complete and widely recognized sets of industry best practices, allowing process improvements in a structured and systematic way. We were convinced that its adoption was essential to our success."**

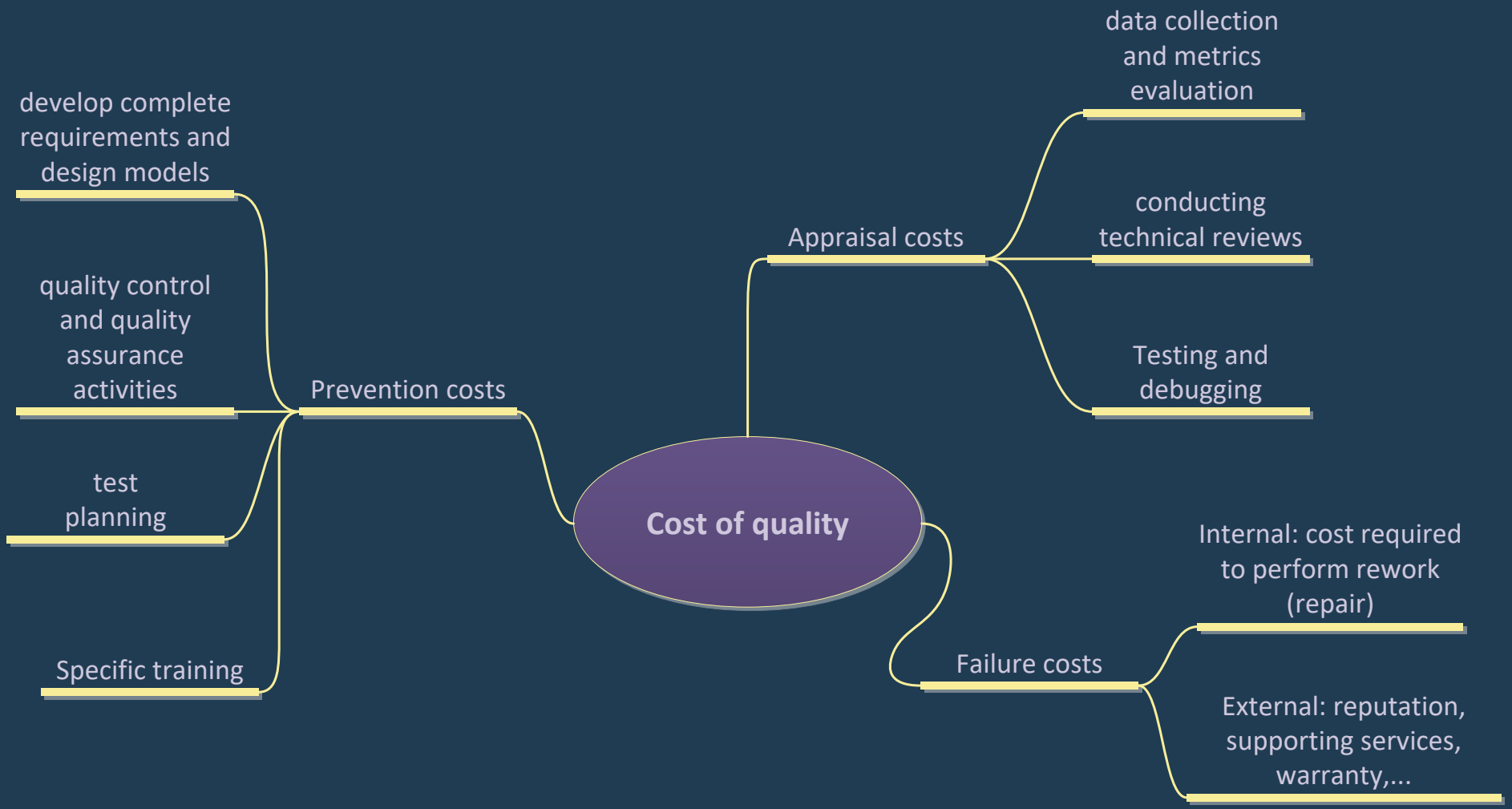
**– LUC CHIASSON,** Group Leader of the Quality Assurance and Continuous Improvement, CAE

http://cmmiinstitute.com/cmmi-models

# TDD: Test Driven Development




# Costs *vs* investments in software quality



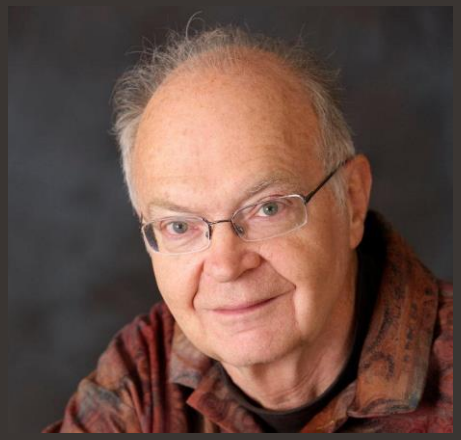
# Quality dilema: when good is good enough?

If you produce a software system that has terrible quality, you lose because no one will want to buy it.

If on the other hand you spend infinite time, extremely large effort, and huge sums of money to build the absolutely perfect piece of software, then it's going to take so long to complete and it will be so expensive to produce that you'll be out of business anyway. Either you missed the market window, or you simply exhausted all your resources.


http://www.artima.com/intv/serious2.html

# TeX testing strategy



The reward for coding errors found in Knuth's TeX and Metafont programs (as distinguished from errors in Knuth's books) followed an audacious scheme inspired by the Wheat and Chessboard Problem.[9] It started at $1.28,[7] and doubled every year until it reached $327.68.[5] Recipients of this "sweepstakes" reward include Chris Thompson (Cambridge) and Boguslaw Jackowski (Gdansk),[10] and also Peter Breitenlohner on 20 March 1995.[11]

Wikipedia, 2014-02