

TVVS

Software Testing, Verification and Validation

Introduction

Ana Paiva

apaiva@fe.up.pt

[//web.fe.up.pt/~apaiva](http://web.fe.up.pt/~apaiva)



FEUP

Universidade do Porto
Faculdade de Engenharia

Why do we need Software Testing?

▣ Software is everywhere

- Enterprise applications & e-commerce
- Public services (healthcare, justice, ...)
- Computer controlled systems (transport., I4.0, IoT,...)
- Media and entertainment



Why do we need Software Testing?

❑ Software **bugs** are everywhere

Some examples of Public Services in Portugal



P Entrar

BASE DE DADOS FOI A MESMA

Nova empresa resolveu colocação de professores em seis dias

SOFIA RODRIGUES

30 de Setembro de 2004, 9:35



negocios 🔍 👤 ASSINAR

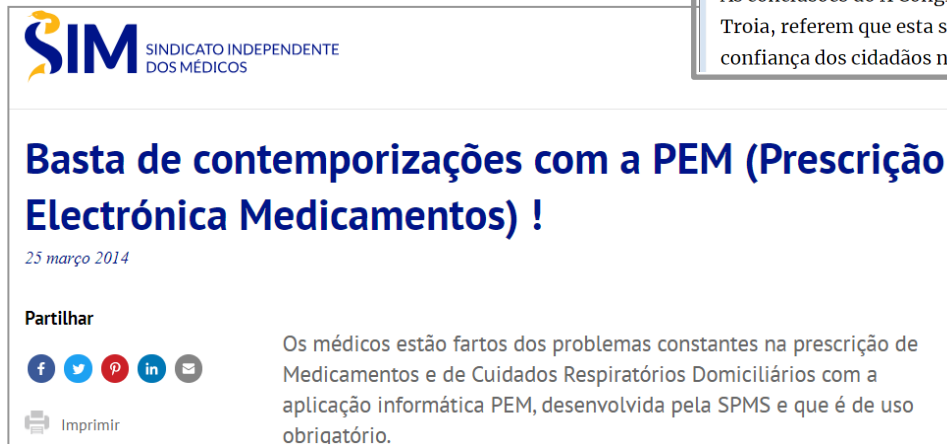
JUSTIÇA

Juízes: Problemas no Citius e nos tribunais estão a causar "enorme desgaste"

Lusa

04 de outubro de 2014, às 15:18

As conclusões do X Congresso dos Juízes, aprovadas por unanimidade em Troia, referem que esta situação pode causar "danos irreparáveis na confiança dos cidadãos na justiça".



SIM SINDICATO INDEPENDENTE DOS MÉDICOS

Basta de contemporizações com a PEM (Prescrição Electrónica Medicamentos) !

25 março 2014

Partilhar

Os médicos estão fartos dos problemas constantes na prescrição de Medicamentos e de Cuidados Respiratórios Domiciliários com a aplicação informática PEM, desenvolvida pela SPMS e que é de uso obrigatório.

Imprimir



OBSERVADOR 🔍 🔔 👤

PORTAL DAS FINANÇAS

Finanças demoraram seis meses a responder a alerta de segurança informática

23/8/2018, 8:09 → 165 6

Why do we need Software Testing?

▣ Software bugs cost millions

Facebook

In October 2021 Facebook experienced one of its largest power outages in a decade.

It resulted in a 5% drop in share price, losing billions from its market cap value. According to the company, it was caused by a bug which was supposed to identify and prevent commands (commands that could take systems offline accidentally) from being issued.

Lost in space

One of the subcontractors NASA used when building its Mars climate orbiter had used English units instead of the intended metric system, which caused the orbiter's thrusters to work incorrectly. Due to this bug, the orbiter crashed almost immediately when it arrived at Mars in 1999. The cost of the project was \$327 million, not to mention the lost time (it took almost a year for the orbiter to reach Mars).

Rocket launch errors

In 1996, a European Ariane 5 rocket was set to deliver a payload of satellites into Earth orbit, but problems with the software caused the launch rocket to veer off its path a mere 37 seconds after launch. As it started disintegrating, it self-destructed (a security measure). The problem was the result of code reuse from the launch system's predecessor, Ariane 4, which had very different flight conditions from Ariane 5. More than \$370 million were lost due to this error.



Why do we need Software Testing?

▣ Software bugs “kill” people

Flight crashes

In 1994 in Scotland, a Chinook helicopter crashed and killed all 29 passengers. While initially the pilot was blamed for the crash, that decision was later overturned since there was evidence that a systems error had been the actual cause. Another example of a software-induced flight crash happened in 1993, when an error in the flight-control software for the Swedish JAS 39 Gripen fighter aircraft was behind a widely publicized crash in Sweden.

Deadly radiation therapy

The Therac-25 medical radiation therapy device was involved in several cases where massive overdoses of radiation were administered to patients in 1985-87, a side effect of the buggy software powering the device. A number of patients received up to 100 times the intended dose, and at least three of them died as a direct result of the radiation overdose.

Another radiation dosage error happened in Panama City in 2000, where therapy planning software from US company Multidata delivered different doses depending on the order in which data was entered. This resulted in massive overdoses for some patients, and at least five died. The number of deaths could potentially be much higher, but it is difficult to know how many of the 21 who died in the following years did so as a result of their cancer or ill effects from the radiation treatment.



Why do we need Software Testing?

- ❑ Software bugs **expose** people's data

LinkedIn AutoFill plugin bug left user data exposed

By [Keumars Afifi-Sabet](#) published April 20, 2018

The flaw, now patched, could have allowed attackers to steal personal data undetected

A Google bug exposed the information of up to 500,000 users

PUBLISHED MON, OCT 8 2018•2:12 PM EDT | UPDATED MON, OCT 8 2018•4:33 PM EDT

Why do we need Software Testing?

How many bugs are injected per line of code? (asked to ChatGPT 😊)

- ❑ **Industry Average:** Historically, the software industry has often used a rule of thumb that there are about 1 to 5 defects per 1000 lines of code (KLOC). This is a very rough estimate and should not be taken as a strict rule.
- ❑ **Critical Systems:** In critical systems like aerospace, medical devices, or automotive software, defect density is expected to be significantly lower, often in the range of 0.1 to 0.5 defects per KLOC.
- ❑ **High-Quality Projects:** In well-managed, high-quality software projects with rigorous testing and code review processes, defect density can be even lower, potentially approaching 0.01 defects per KLOC or lower.

It's essential to note that these are rough estimates, and the actual defect density in a specific project can deviate significantly from these averages.

Ultimately, defect density is a useful metric for assessing software quality, but it should be interpreted with caution and in the context of the specific project and its requirements.

**SO, WHY DO NOT WE JUST TEST
MORE AND MORE?**



SOFTWARE TESTING IS CHALLENGING!

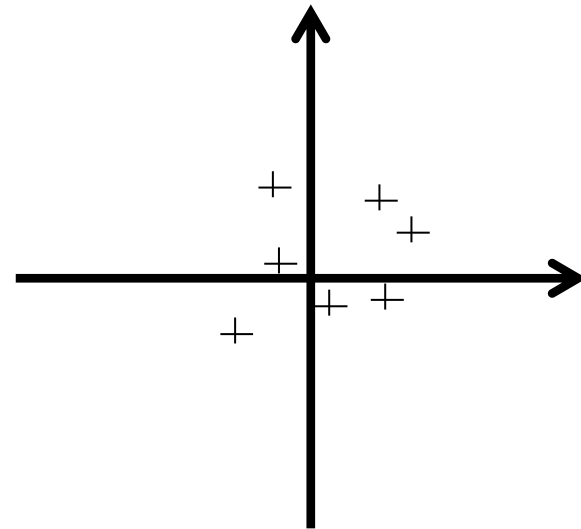
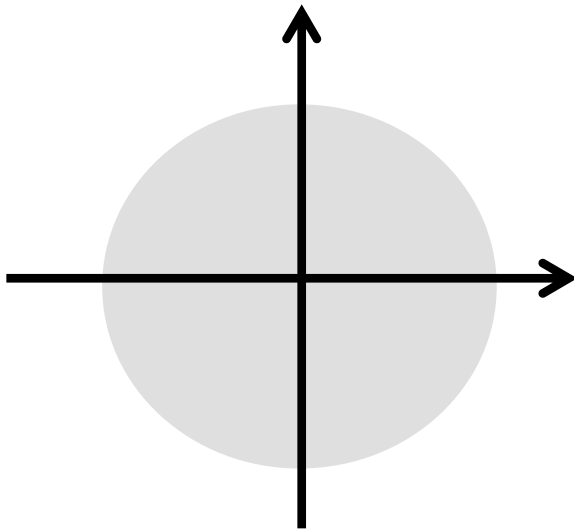


Some challenges...

- ❑ Bugs are not uniformly distributed
- ❑ How many tests are enough?
- ❑ When should we stop testing?
- ❑ Test automation vs. Manual tests?
- ❑ How to choose the best tool?
- ❑ Deal with time constraints
- ❑ Workers' skills
- ❑ Within the budget
- ❑ ...

Exhaustive testing is impossible

- Since **exhaustive testing is impossible**, how can we generate a set of manageable test cases with a high probability of detecting most defects?



- Dijkstra used to say: “Program testing can be used to show the presence of bugs, but never to show their absence”

Let's start testing...

- ▣ I just implemented a calculator capable of dividing two integers:

$$a/b$$

- ▣ Give me test cases (values for a and for b) to "ensure" the calculator is well-implemented



Two dimensions of quality

Verification and Validation (V&V) are both assessing the quality of a system, but they are different.

Verification: The primary goal is to ensure that the software is built according to the specified requirements and design, focusing on correctness and adherence to standards.

- Verification: "Are we building the software right?" (Focuses on correctness and adherence to specifications)

Validation: The primary goal is to ensure that the software satisfies user needs and delivers value in its intended environment, focusing on usability, functionality, and user satisfaction.

- Validation: "Are we building the right software?" (Focuses on fitness for purpose and user satisfaction)

Two dimensions of quality

When does it occur in the development lifecycle?

- **Verification:** Typically occurs during the development phase. It involves activities such as code reviews, static analysis, and unit testing to ensure that each component or module meets its design specifications.
- **Validation:** Usually occurs after the development phase. It involves activities such as user acceptance testing (UAT) and usability testing to assess the overall product's suitability for its intended use.

Note that Verification and Validation (V&V) are fundamental for ensuring the delivery of high-quality software systems.

**... AND HOW MUCH IT COSTS TO
INVEST IN QUALITY?**



Quality costs

▣ Costs of **conformance**

- All costs associated with planning and running tests (and revisions) just one time

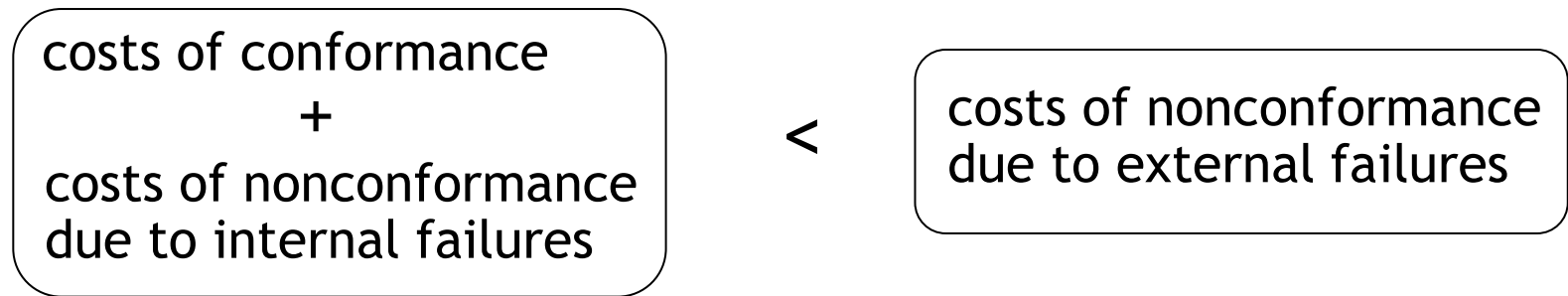
▣ Costs of **nonconformance**

- Costs due to internal failures (before release)
 - Cost of isolating, reporting and regression testing bugs (found before the product is released) to ensure that they're fixed
- Costs due to external failures (after release)
 - If bugs are missed and make it through to the customers, the result will be costly product support calls, possibly fixing, retesting, and releasing the software, and - in the worst-case-scenario - a product recall or lawsuits

[source: "Software Testing", Ron Patton]

Quality costs

- ▣ In his book "Quality is Free: The Art of Making Quality Certain", Philip Crosby argues that the costs of conformance plus the costs of nonconformance due to internal failures is (usually) less than the costs of nonconformance due to external failures



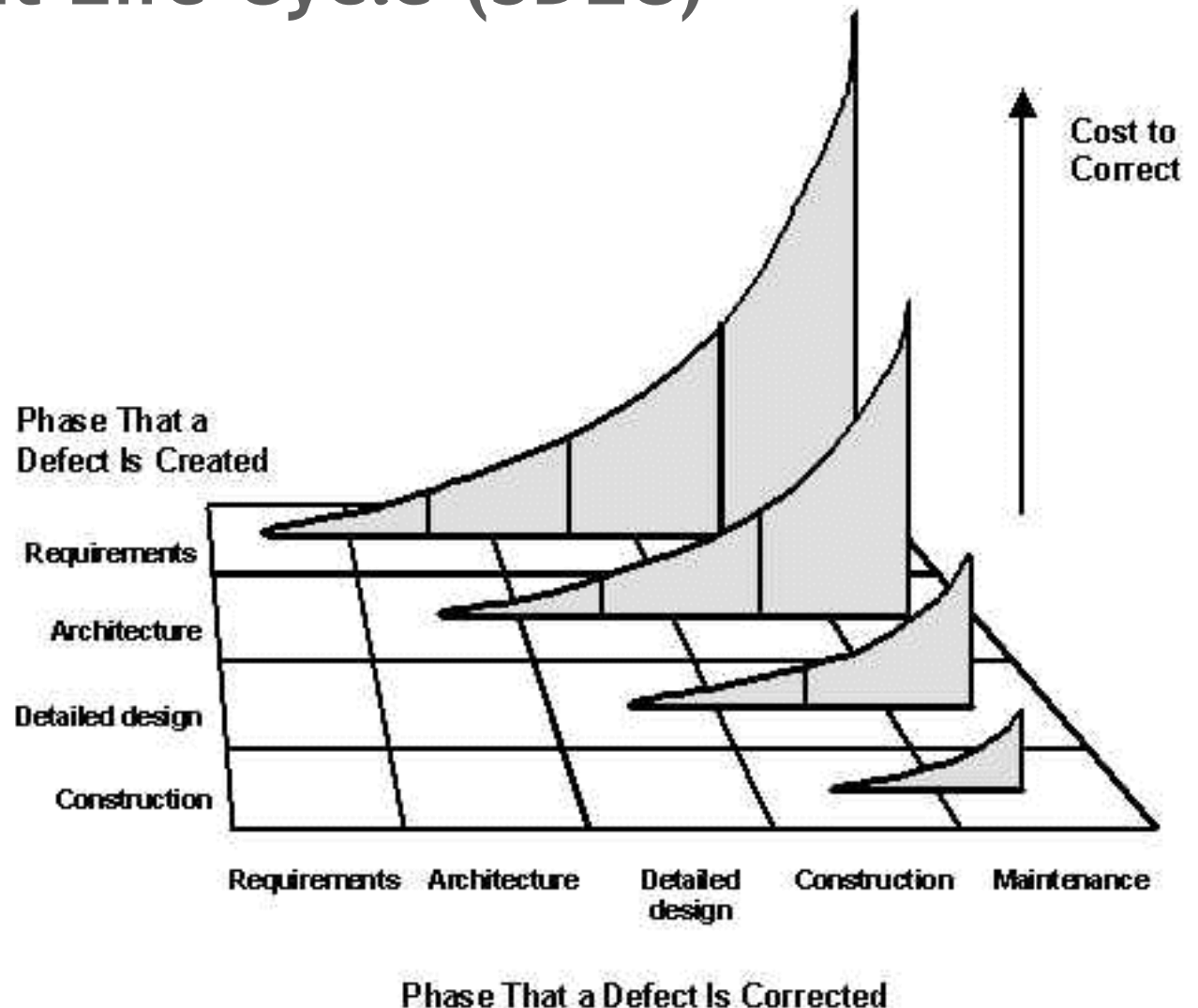
[source: Ron Patton, "Software testing"]

Quality costs along the Software Development Life Cycle (SDLC)

If a defect is created in the early stages of the development life cycle and corrected in the later stages, correcting it will be very expensive.

If a defect is created and removed in the same phase, it won't be too expensive.

The longer the bug remains, the more expensive it will be to fix.



Bugs along the SDLC (examples)

▣ Requirements Phase:

- **Unclear or Incomplete Requirements:** Ambiguous, incomplete, or changing requirements can lead to misunderstandings and result in defects.
- **Miscommunication:** Miscommunication between stakeholders, including developers and business analysts, can lead to requirements defects.

▣ Design Phase:

- **Poor Architecture:** Design decisions that don't consider scalability, performance, or maintainability can lead to design defects.
- **Incomplete or Inaccurate Designs:** Insufficiently detailed or incorrect design documentation can result in defects during implementation.

Bugs along the SDLC (examples)

□ Coding Phase:

- **Syntax and Logic Errors:** Coding errors such as syntax mistakes and logic flaws are frequently injected during this phase.
- **Inadequate Testing:** Failure to conduct unit testing or inadequate test coverage can leave defects undiscovered.
- **Inconsistent Coding Standards:** Not adhering to coding standards can lead to code defects.

□ Integration Phase:

- **Integration Errors:** Integration issues can arise due to incompatible interfaces or data mismatches when combining components or modules.
- **Interoperability Issues:** Incompatibility between different software parts can result in defects.

Bugs along the SDLC (examples)

□ Testing Phase:

- **Incomplete Test Coverage:** Some defects may remain undetected if not all possible scenarios are tested.
- **Test Data Issues:** Defects can occur when test data doesn't accurately represent real-world usage.
- **Environment Differences:** Test environments that differ from the production environment can lead to defects not manifesting until deployment.

□ Deployment Phase:

- **Configuration Errors:** Mistakes in configuring the software in the production environment can introduce defects.
- **Installation Problems:** Issues during the installation process can result in defects.

Bugs along the SDLC (examples)

□ Maintenance Phase:

- **Regression Defects:** Changes made during maintenance can inadvertently introduce new defects or reactivate old ones.
- **Documentation Updates:** Failure to update documentation to reflect changes can lead to misunderstandings and defects.

So,...

- ▣ We must perform Verification and Validation (V&V) to find the maximum number of errors, mistakes, bugs, faults, defects, and failures ☹ in the software system under test (SUT).

- ▣ Homework

Go to the ISTQB glossary at

glossary.istqb.org

and find out the meaning of

error, mistake, bug, fault, defect, and failure