UPskill

Metodologias de teste de software

P.PORTO

isep Instituto Superior de Engenharia do Porto

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
Instituto Superior de Engenharia do Porto

Teste de software

Testing

As Paul Ehrlich (scientist, Nobel laureate) puts it —

"To err is human, but to really foul things up you need a computer."

Teste de software

- Processo usado para identificar a correção, completude e qualidade do software desenvolvido
- Conjunto de atividades realizadas com o objetivo de encontrar erros no software
  - Para que este possa ser corrigido antes de ser entregue ao cliente
  - Para que o produto final seja forte, fiável e estável
  - Para procurar garantir que o software é isento de defeitos

OS 7 PRINCÍPIOS

Os 7 princípios

- 1 Testar mostra a presença de defeitos
- 2 Testar exaustivamente é impossível
- 3 Agrupar defeitos *(defect clustering)*
- 4 Paradoxo do Pesticida *(pesticide paradox)*
- 5 Ausência de erros é uma falácia
- 6 Testar precocemente
- 7 Dependente do contexto

Testar mostra a presença de defeitos

- Os erros de software podem causar custos elevados
  - O desenvolvedor tem custos associados à mão de obra necessária para a correção dos erros
  - O cliente tem custos associados aos erros de software
    - Os erros de software podem (potencialmente) causar custos monetários ou até perdas humanas
  - O cliente pode imputar custos ao desenvolvedor
- A única forma de poder detetar erros é testar o software

Testar exaustivamente é impossível

- Normalmente não é possível conhecer todos os pontos de falha de um software
- Normalmente os casos a testar são muitos
- Normalmente o custo e tempo associado aos testes depende do número de testes
  - Se testar um caso custa X então (grosso modo) testar N casos custa N*X
- O custo e o tempo de teste cresce de forma exponencial com o número de combinações de teste

Agrupar defeitos *(defect clustering)*

- Testar exaustivamente é impossível
  - Aquilo de que necessitamos é de um conjunto ótimo de testes baseado numa análise de risco
- Como determinar o risco?
  - Analisar um conjunto de operações para as quais a nossa sensibilidade nos diga que a probabilidade de poderem eventualmente provocar falhas é maior
  - Agrupar defeitos permite-nos avaliar quais são os módulos que devem ter um maior esforço de teste *(defect clustering)*

Agrupar defeitos *(defect clustering)*

- Uma pequena quantidade de módulos contém a maior parte dos defeitos detetados
  - A experiência permite-nos identificar esses módulos mais problemáticos
  - Ao testar exaustivamente esses módulos estaremos a garantir a correção de uma maior quantidade de defeitos

Pesticide Paradox

- Se executamos sempre os mesmos testes, então eventualmente deixaremos de encontrar defeitos, portanto…
  - Os testes têm de ser periodicamente revistos e reescritos, adicionando novos casos de teste
  - Testar reduz a probabilidade de defeitos no software, mas, mesmo que não encontremos defeitos ao testar isso não comprova a completa correção do software
  - O software nunca é 100% *"bug free"*

Ausência de erros - falácia

- O software nunca é 100% *"bug free"*
- O software tem de cumprir as necessidades e requisitos do cliente
  - Encontrar e resolver defeitos de um sistema que não cumpre as necessidades do cliente ou especificações não é uma tarefa produtiva
  - Portanto, os testes devem começar assim que possível durante o ciclo de vida do processo de desenvolvimento de software

Testar precocemente

- Os testes devem começar o mais cedo quanto possível para que qualquer defeito também seja precocemente capturado
- Os testes devem ser orientados ao tipo de sistema a desenvolver, ou seja, dependentes do contexto

Dependente do contexto

- A forma como se testa um determinado sistema A é diferente da forma como se testa o sistema B
  - A forma como se testa um website de comércio eletrónico é diferente da forma como se testa um ERP

O CICLO DE VIDA DO DESENVOLVIMENTO DE SOFTWARE

SDLC - Software Developement Life Cycle

- Requirements stage
  - Obter o máximo de informação possível sobre os detalhes e especificações sobre o software pretendido por parte do cliente
- Design stage
  - Planear as linguagens de programação adequadas para o desenvolvimento do projeto
- Build stage
  - Codificar o software
- Test stage
  - Testar o software para verificar que está desenvolvido e cumpre as especificações
- Maintenance stage
  - Alterações ao código para prever melhoramentos

Custos associados à correção de erros

- A análise a um grande número de projetos permite concluir que
  - Aproximadamente 50% dos defeitos são introduzidos nas fases de *Requirements* e *Design*
  - O custo para corrigir um defeito incrementa ao longo do ciclo de vida do desenvolvimento do software
  - Quanto mais cedo se detetar um erro, mais barato se torna corrigi-lo

Waterfall Model

- Requirements
  - E se cometermos um erro nesta fase?
- Design
  - E se cometermos um erro nesta fase?
- Developement
  - E se cometermos um erro nesta fase?
- Test
  - Podemos não encontrar erros…
- Maintenance
  - Mas temos um produto que o cliente não quer

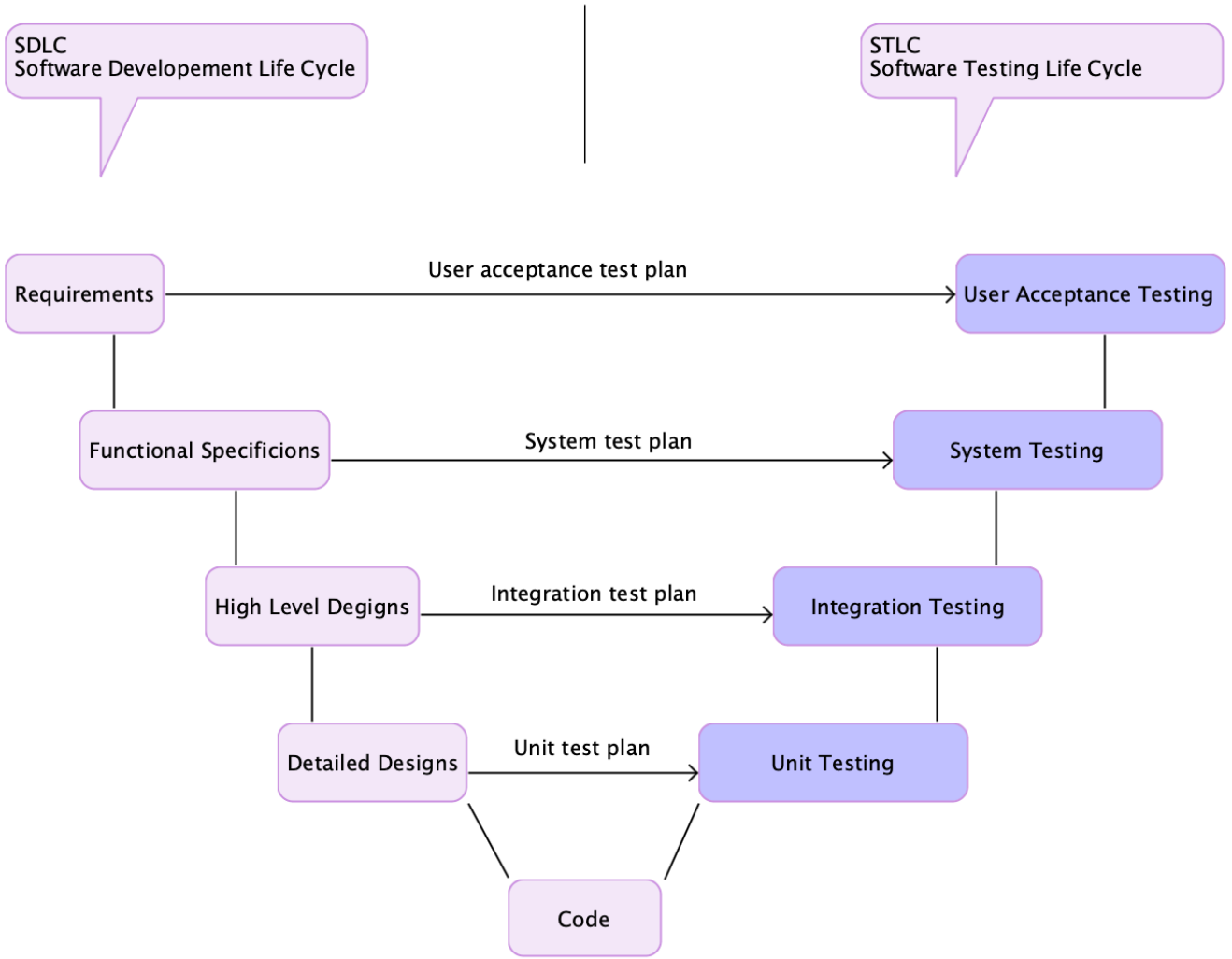
Iterative Model

- Phase 1
  - Requirements
  - Design
  - Developement
  - Test
- ...
- Phase N
  - Requirements
  - Design
  - Developement
  - Test

Iterative Model

- Modelos iterativos
  - RAD (Rapid Application Developement)
  - AGILE
  - …

V-Model of Testing



Resumo

- Existem vários modelos de desenvolvimento de software
- O modelo adotado para um projeto depende dos objetivos desse projeto
- Testar é uma atividade stand-alone e tem de ser adaptada ao modelo de desenvolvimento escolhido
- Seja qual for o modelo escolhido, os testes devem ser executados em todos os níveis (desde a fase de levantamento de requisitos até à fase de manutenção)