Uma imagem com ClipArt

Descrição gerada automaticamente

Ana Catarina Henriques(8170064), Rebeca Teixeira(8170274), Vânia Lemos(8170310)

Docentes: Cristóvão Sousa e Fábio Silva

Licenciatura em Engenharia Informática

19/11/2018

Índice

[Âmbito e referências 3](#_Toc530430116)

[*Dentro do âmbito* 3](#_Toc530430117)

[*Fora do âmbito* 3](#_Toc530430118)

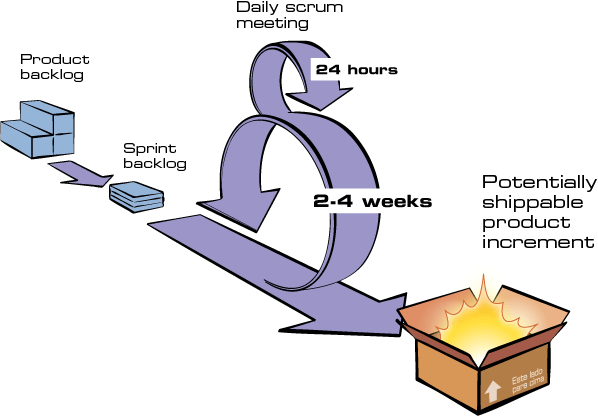
[*Referências* 4](#_Toc530430119)

[Definições 4](#_Toc530430120)

# Âmbito e referências

## Dentro do âmbito

Foi-nos solicitado pelos docentes da unidade curricular de Engenharia de Software II desenvolver um motor de pesquisa que, através de uma frase, procure e apresente o ficheiro com o maior grau de similaridade de acordo com o que o utilizador introduziu.

O planeamento de projetos de software é extremamente importante o uso de metodologias ágeis, de entre as quais a *Scrum.* Nesta os projetos são divididos em ciclos iterativos *(sprints*) que representam um intervalo de tempo para que um conjunto de atividades sejam executadas. No início de cada sprint, faz-se um planeamento (Sprint Planning Meeting) onde o Product Owner prioriza os itens do Product Backlog (lista das funcionalidades a serem implementadas) e a equipa seleciona as atividades que se implementarão durante esse Sprint. No final deste, a equipa tem de apresentar software funcional e iniciar um novo planeamento do sprint seguinte, reiniciando-se assim o ciclo.

Para que seja possível a entrega de software funcional é necessário que este tenha qualidade, o que implica a aplicação de:

* **Testes:**

Testing é um grupo de procedimentos realizados para avaliar algum aspeto ou parte de software, é usado para revelar defeitos no software e estabelecer o grau de qualidade do mesmo (avaliar atributos de qualidade de software tais como segurança, usabilidade e precisão).

Os testes de software são uma forma dinâmica de verificar se o programa, através de um conjunto de testes finito, age de acordo com o esperado, para isso, o tester (pessoa que está a testar) tem de conhecer os outputs em função dos inputs e das condições de execução. Os testes podem ser realizados a 3 níveis: testes unitários, testes de integração e testes de software, estes cobrem as atividades de validação (foco no processo) e verificação (foco no produto).

Os testes de sistema são uma parte crucial do ciclo de vida do desenvolvimento de um programa. Para muitos a melhor abordagem é garantir a prevenção de erros, logo os testes de software são uma abordagem preventiva e focada na qualidade final, sendo estes um bom indicador e uma boa abordagem de desenvolvimento de software.

Podemos assim concluir que é necessária uma escolha justificada e adequada dos testes a efetuar, de modo a obtermos o menor custo possível. Posto isto é necessário um planeamento de casos de teste obedecendo aos critérios de seleção e adequação de software, para que estes sejam eficazes e objetivos. Com base no nosso software vamos efetuar testes unitários, sendo estes testes de classe de equivalência e testes de valor limite (BVA).

* **Software Configuration Management:**

Software Configuration Management é uma área da engenharia de software responsável por fornecer o apoio para o desenvolvimento de software cujas principais funções são o controlo de versões, mudanças e auditoria das configurações.

## Fora do âmbito

Neste capítulo, pretendemos dar a conhecer os vários tipos de testes que existem, para além daqueles que iremos realizar.

Os testes são guiados para um objetivo específico, que pode ser mais ou menos específico e com vários graus de precisão. Os testes podem ser desenhados para verificar se as funcionalidades estão bem implementadas - testes de conformidade, testes de correção ou testes funcionais.

No entanto, as propriedades não funcionais também podem ser testadas - performance, confiabilidade e usabilidade. É preciso ter em conta que alguns testes são mais apropriados para software “custom-made” como por exemplo testes de instalação e outros para produtos de consumidor como os testes beta.

Para além dos testes que vamos realizar existem mais tipos de testes: **testes de aceitação/conformidade**- determinam quando o sistema cumpre os critérios de satisfação, geralmente por verificar o comportamento desejado em prol de requisitos de cliente; t**estes de instalação-** após o término do sistema e dos testes de aceitação, o software é verificado no ambiente em que será instalado; **testes alfa e beta**- depois de o software ser lançado, é por vezes dado a um pequeno e selecionado grupo de potenciais utilizadores; ***reliability achievement and evaluation***- melhora a confiabilidade ao identificar e corrigir os erros; ***regression testing***- o objetivo é mostrar que o comportamento do software não mudou ao lhe adicionar alterações exceto na medida em que deveria; ***performance testing***- este verifica se o software corresponde aos requisitos específicos e avalia características de performance; **security testing**- foca-se na verificação de que o software está protegido de ataques externos; **stress testing**- leva o sistema ao limite para testar os mecanismos de defesa do software; **back to back testing**- teste em que duas ou mais variantes do programa são executadas com o mesmo input e os outputs são comparados e erros analisados em caso de discrepância; **recovery testing**- verifica as capacidades do software de reiniciar depois de o sistema falhar ou de outro ”desastre”  
Existem várias técnicas para realizar testes de caixa branca, testes de caixa preta e de testes baseados na intuição tais como:

* **Testes baseados na intuição*: ad-hoc*-** testes que são feitos consoante as qualidades, intuição e experiência do engenheiro com programas semelhantes; ***Exploratory testing***- os testes não são definidos num plano de testes, no entanto são desenhados, executados e modificados de forma dinâmica.
* **Testes de caixa preta**: classe de equivalência; ***Parwise testing***- testes feitos ao combinar valores interessantes para cada conjunto de inputs em vez de os considerar todos; Boundary-values analysis(BVA); ***Random testing***- testes gerados aleotoriamente
* **Testes de caixa branca**: ***Control Flow-based criteria*** – destinado a abranger todos os *statements*, blocos de *statements* ou combinações especificas dos mesmos. ***Data flow based criteria***- o grafo de controlo de fluxo contem informação de como as variáveis são definidas, usadas e depois descartadas; ***Reference models for code-Based testing*-** A estrutura de controlo pode ser graficamente representada num fluxograma

## Referências

* IEEE Standard for Software Unit Testing
* SWEBOK

# Definições

|  |  |
| --- | --- |
| Conceito | Definição |
| Testes de Software | Os testes de software são uma forma dinâmica de verificar se um programa age como o esperado. |
| Testing | É um grupo de procedimentos realizados para avaliar algum aspeto ou parte do software. |
| Tester | Pessoa que realiza os testes de software. |
| Testes unitários | São testes que se realizam a programas mais pequenos ou então a partes específicas do código. Normalmente são realizados pelos programadores, visto que requerem a análise do código. |
| Teste de integração | Servem para testar a integração de diferentes componentes do software. |
| Teste de sistema | São testes que se efetuam ao sistema num todo. Geralmente são usados para testar se os requisitos não funcionais do mesmo estão a ser cumpridos. |
| Erro | Um erro é um engano na interpretação por parte da entidade que está a desenvolver o software. |
| Defeito | Um defeito é o resultado de um erro, ou seja, é uma anomalia que faz com que o software se comporte incorretamente (“bugs”). |
| Casos de teste | Conjunto de condições usadas para testar o software. |
| Plano de testes | O plano de testes é um documento extenso onse se definem as abordagens, os recursos, o âmbito e o planeamento das atividades de teste. |
| Classe de equivalência | É uma técnica para realizar testes que consiste em lançar um erro quando se insere inputs inválidos. Normalmente, para cada grupo de testes é efetuado um número específico de classes de equivalência. |
| Critério de adequação de software | Um critério de adequação de dados de teste é uma regra de paragem, ou seja, ajuda o tester a selecionar o conjunto de dados de teste para um programa bem como as propriedades do mesmo em que se deve focar. |
| Critérios de seleção | Os critérios de seleção ajudam o tester a estabelecer que os testes que deve efetuar para ter uma noção do que testar e como. |
| Testabilidade | O termo “testabilidade do software” em engenharia de software pode ter dois significados distintos, sendo os dois muito importantes. Tendo em conta os critérios definidos pode significar que o software é fácil de testar, ou ainda pode significar a possibilidade de medir a probabilidade que um caso de teste tem de ser bem sucedido ou de falhar. |
| Teste de caixa preta | São testes funcionais que se baseiam apenas nos inputs e outputs. |
| Teste de caixa branca | São testes estruturais que se focam em como a informação do software foi codificada. |
| Validação | É um processo, focado no produto, que determina se um sistema ou um componente de software satisfaz os requisitos especificados. |
| Verificação | É um processo, focado no processo, que avalia um sistema ou componente de software para determinar se as condições impostas no começo são satisfeitas numa determinada fase do desenvolvimento. |

## Submissão de código

Através do link do GIT é possivel ver o código anteriormente apresentado e excertos do mesmo: [https://github.com/rebeccateix20/es2.git](https://github.com/rebeccateix20/es2.git?fbclid=IwAR02prr7l8SQigIAV4wtXXAf1vvXnUdYQN4dNOc-pefLsXoWAcf9-3Y2mKs)

## Verificação de metodologias e Considerações finais

Executamos os testes no *idea intellij* usando o *JUint 5* e com auxílio da ferramenta *gradle*. Para conseguirmos fazer o controlo de versões e desenvolvimento paralelo dos testes, entre os vários elementos do grupo, utilizamos o *GitHub* como repositório online.

Com este projeto podemos ter uma pequena noção do que uma equipa de testes tem que fazer.

<https://github.com/vanialemos/ESII>

C:\Users\Proprietário\Documents\GitHub\ESII