<Project Name>

Configuration Management Plan

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introdução 4

1.1 Propósito 4

1.2 Âmbito 4

1.3 Definições, acrónimos e abreviaturas 5

1.4 Referências 6

1.5 Overview 6

2. Software Configuration Management 7

2.1 Organization, Responsibilities, and Interfaces 7

2.2 Tools, Environment, and Infrastructure 7

3. The Configuration Management Program 7

3.1 Configuration Identification 7

3.1.1 Identification Methods 7

3.1.2 Project Baselines 7

3.2 Configuration and Change Control 7

3.2.1 Change Request Processing and Approval 7

3.2.2 Change Control Board (CCB) 7

3.3 Configuration Status Accounting 8

3.3.1 Project Media Storage and Release Process 8

3.3.2 Reports and Audits 8

4. Milestones 8

5. Training and Resources 8

6. Subcontractor and Vendor Software Control 8

Configuration Management Plan

# Introdução

## Propósito

[Specify the purpose of this **Configuration Management Plan.**]

## 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.

No 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:** 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.

## Definições, acrónimos e abreviaturas

|  |  |
| --- | --- |
| **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. |

## Referências

* IEEE Standard for Software Unit Testing
* SWEBOK

## Overview

[This subsection describes what the rest of the **Configuration Management Plan** contains and explains how the document is organized.]

# Software Configuration Management

## Organização, responsabilidades e interfaces

[Describe who is going to be responsible for performing the various Configuration Management (CM) activities described in the CM Process Discipline.]

## Ferramentas, ambiente e Infraestruturas

As ferramentas que iremos utilizer neste projeto são:

* A linguagem de programação java
* O jUnit- framework utilizado para a codificação dos testes especificados
* Jenkins
* Git- repositório do Código fonte (<https://github.com/rebeccateix20/TrabalhoESII>)
* Youtrack
* Upsource

A infraestrutura utilizada neste projeto é uma máquina virtual , fornecida pelos docentes da disciplina, que tem como objetivo reunir todas as ferramentas enumeradas anteriormente.

# The Configuration Management Program

## Configuration Identification

### Identification Methods

[Describe how project or product artifacts are to be named, marked, and numbered. The identification scheme needs to cover hardware, system software, Commercial-Off-The-Shelf (COTS) products, and all application development artifacts listed in the product directory structure; for example, plans, models, components, test software, results and data, executables, and so on.]

### Project Baselines

[Baselines provide an official standard on which subsequent work is based and to which only authorized changes are made.

Describe at what points during the project or product lifecycle the baselines are to be established. The most common baselines would be at the end of each of the Inception, Elaboration, Construction, and Transition phases. Baselines could also be generated at the end of iterations within the various phases or even more frequently.

Describe who authorizes a baseline and what goes into it.]

## Configuration and Change Control

### Change Request Processing and Approval

[Describe the process by which problems and changes are submitted, reviewed, and dispositioned.]

### Change Control Board (CCB)

[Describe the CCB membership and the procedures for processing change requests and approvals to be followed by the CCB.]

## Configuration Status Accounting

### Project Media Storage and Release Process

[Describe retention policies, and the back-up, disaster, and recovery plans. Also describe how the media is to be retained—online, offline, media type, and format.

The release process describes what is in the release, who it is for, and whether there are any known problems and any installation instructions.]

### Reports and Audits

[Describe the content, format, and purpose of the requested reports and configuration audits.

Reports are used to assess the “quality of the product” at any given time in the project or product lifecycle. Reporting on defects based on change requests may provide some useful quality indicators and, thereby, alert management and developers to particularly critical areas of development. Defects are often classified by criticality (high, medium, and low) and could be reported on the following basis:

* Aging (Time-based Reports): How long have defects of the various kinds been open? What is the “lag time’’ between when defects are found in the lifecycle and when they are fixed?
* Distribution (Count Based Reports): How many defects are there in the various categories by owner, priority or state of fix?
* Trend (Time-related and Count-related Reports): What is the cumulative number of defects found and fixed over time? What is the rate of defect discovery and fix? What is the “quality gap” in terms of open as opposed to closed defects? What is the average defect resolution time?]

# Milestones

[Identify the internal and customer milestones related to the project or product CM effort. This section includes details on when the **Configuration Management Plan** itself is to be updated.]

# Training and Resources

[Describe the software tools, personnel, and training required to implement the specified CM activities.]

# Subcontractor and Vendor Software Control

[Describe how software developed outside of the project environment will be incorporated.]