P.PORTO
ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO
ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Engenharia de Software II

Software Configuration Management Plan

Realizado por:

8180378-Hugo Silva

8200306-Fábio Costa

Conteúdo

1. Introdução

1.1Objetivo

Este documento foi elaborado para documentar as atividades de SCM que serão aplicadas na realização de um projeto. O documento terá os responsáveis por efetuar determinadas tarefas, regras de elaboração, e ferramentas a utilizar para o desenvolvimento do projeto.

1.2 Âmbito

O SCMP será aplicado ao enunciado do trabalho de ES2. O projeto proposto é uma API capaz de processar e otimizar a atividade de uma plataforma que permite facilitar a transação de produtos entre empresas a nível nacional.

Neste documento serão registadas todas as atividades individuais ou de grupo, normas a seguir para alterações de configurações e ferramentas a utilizar.

A metodologia utilizada será a metodologia SCRUM. O Scrum baseia-se nos principios e fundamentos da metodologia Agile e distingue-se pelo desenvolvimento do produto de forma progressiva.

1.3 Abreviações e Glossário

SCM – Software Configuration Management

SCMP – Software Configuration Management Plan

CR – Change Requests

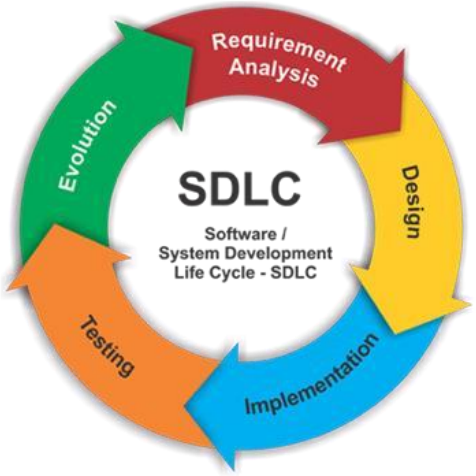
1.4 Referências

IEEE. Standard for Software Configuration Management Plans

1.5 Tarefas no processo de SCM

- Identificação de Configuração.
- Baselines.
- Controlo de alterações.
- Estado da configuração.
- Auditorias e análises de configuração.

1.6 Atividades SLDC



Stage 1: Planning and Requirement Analysis

Stage 2: Design

Stage 3: Build

Stage 4: Test

Stage 5: Product Release

Stage 6: Maintenance

O SCM Plan baseia-se na fase de plano e estruturação do projeto. Pode-se então afirmar que o plano SCMP está contido nos padrões do ciclo de vida de desenvolvimento de software.

2. Configuration Management System

Gestão de configuração (CMS) é um processo de engenharia para estabelecer e manter a consistência de um produto (inclui por exemplo atributos funcionais, requisitos e *design*). Para efetuar esta gestão será usado o *GITLAB*.

2.1 Configuration Identification

Processo de identificação dos itens e respetivos atributos

- **Test case**: os test cases devem ser todos escritos e documentados de forma que possam ser trabalhados e compreendidos por várias pessoas (Ex: testNOMEFUNÇÃO).

- **User Stories**: Todas as User Stories definidas para o projeto devem estar devidamente identificadas e descritas. Devem descrever funcionalidades da API.

- **Identificação dos requisitos:** Todos os requisitos do projeto devem estar identificados com um código único e devem ser separados por módulos (EX: Módulo de transações).

- **Codelines*:** Cada *codeline* deve ser devidamente identificada pelo nome do módulo que se está a desenvolver. Cada módulo deve ter a sua própria *codeline*.

*Um codeline é uma sequência de versões de código-fonte de um determinando *branch*.

Source code scripts: Os nomes dos métodos devem respeitar o *style* camelCase (*style* já utilizado na API fornecida).

- **Issues:** As issues devem possuir o título da user story (sempre que possível)a que pertence, a epic e o conjunto de todas as tarefas que serão realizadas .

- **Branchs:** Cada issue deve corresponder a uma branch nova, quando efetuado um merge request a branch deve ser eliminada.

  Nota: documentação, configuração do ficheiro yml pode ser realizado diretamenta na branch master.

- **Tag:** Descrição do que levou à criação da tag (ex: nova funcionalidade, correção de bug).

2.2 Atividades e responsabilidades

SCM Role: Configuration Manager

Membro: Hugo Silva

Responsabilidades:

- Estabelecer *sprints*, definindo a datas de início e fim dos mesmos.
- Definir funcionalidades que serão colocadas na *branch* master no final do *sprint*.

SCM Role: Integration

Membro: Hugo Silva

Responsabilidades:

- Aprovar Merge Requests.
- Avaliar progresso de acordo com os sprints definidos.
- Confirmar que os registos de Configuration Management identificam corretamente os Configuration Items.
- Rever a estrutura e integridade de itens no sistema de Configuration Management.

SCM Role: Code Reviewer

Membro: Fábio

Responsabilidades:

- Realizar Inspeções de código.
- Analisar pedidos de CR.
- Efetuar revisões para assegurar que alterações a Configuration Items não causem efeitos indesejados nos *sprints* definidos.

SCM Role: Product Owner

Membro: Hugo

Responsabilidades:

- Validar submissões de CRs
- Avaliar progresso de acordo com os *sprints* definidos.
- Definir prioridades de implementação de funcionalidades.

3. Configuration Management Program

3.1 Estados e plano de manutenção

O código e documentação será armazenado num repositório da plataforma *GitLab* que permite automatizar algumas das tarefas realizadas no processo de desenvolvimento.

Deve ser definido um ficheiro yml na *branch master* do repositório do projeto. Este ficheiro deve permitir fazer *build* do projeto e realizar os testes de caixa preta.

Impossibilidade dos *runners* do domínio estg.ipp.pt

Caso os runners não estejam a funcionar corretamente a equipa de desenvolvimento deve criar um repositório no GitLab free. Neste repositório deve ser colocada uma "cópia" do repositório no domínio da ESTG e executado os runners(assim é possibilitado ter a perceção de como funciona a automatização de tarefas no GITLAB).

3.2 Configuration Control

3.2.1 Controlo de alterações

A equipa de desenvolvimento submete um *change request* ao *code review*.

O code review analisa o pedido de change request e faz a comparação do mesmo com o product Backlog e com o estado atual do software.

De seguida, em conjunto com o *Product Owner*, decide-se se o pedido de change request é ou não aceite, a decisão é tomada de acordo com o impacto que a alteração terá na API ou num módulo da API.

Quando uma aprovação de um change request é realizada toda a equipa deve ser notificada. O membro responsável pelo pedido pode então fazer *push* para o repositório de modo que a alteração seja integrada no software.

No caso de um *change request* não ser aprovado deve ser informado o membro que realizou pedido. No contacto deve ser explicado de forma sucinta o motivo(s) da decisão.

O report de um change request deve estar presente na wiki do repositório.

Nota: O pedido de change request por norma são causados pelo cliente do produto. Dado que o produto é um projeto académico não existe um "cliente" a avaliar o estado do software a cada sprint.

3.3 Configuração das auditorias e inspeções

Todos os *reports* dos *runners* serão analisados semanalmente pela equipa de desenvolvimento. Quaisquer medidas corretivas necessárias serão efetuadas, assegurando desta forma estabilidade do SQA, garantido pelos testes de caixa preta, teste de cobertura e inspeções de código.

O processo de analise e alterações deve estar presente no GitLab.

3.4 Identificação e correção erros

1º - Criação de uma issue para a correção do erro (deve estar descrito de forma sucinta o erro);

2º - A DEV Team faz a analise e decide aceitar ou não a Issue;

3º - A issue é atribuída a um developer para que possa ser feita;

4º - A issue é realizada no ambiente de *development*, sendo que deve ser criada uma nova branch;

5º - O developer deve definir um plano de testes para a issue;

6º - Deve ser feito *commit* e *push* para a branch criada;

7º - Fazer Merge Request;

8º - Aprovação do Merge Request por parte do membro responsável pela SCM Role "Integration";

9º - Responsável pela issue realiza o merge e fecha a *issue*;

10º - Responsável pela issue deve documentar o processo realizado na issue para futura manutenção da API.

3.5 Configuração da Baseline do projeto

Baselines são marcos do projeto e permitem fazer uma avaliação do progresso.

Baseline1:

1. Documento de requisitos;
2. Ledger API com bugs encontrados no 1ª trabalho corrigidos(Funcionalidade);
3. Método para obter valor médio das transacções;
4. Documento SCMP;
5. Documentos SCMP e Backlog colocados no wiki do git.

Baseline 2:

1. Tabelas de distâncias entre distritos importada;
2. Importar ficheiro de encomendas;
3. Método para obter número médio de produtos por transacção;
4. Método para obter Valor médio de vendas e compras por distrito;
5. Exportação para JSON das métricas estatísticas 3,4;
6. Registar Encomendas no ledger;

Baseline 3:

1. Método que calcula custo de envio de uma encomenda entre dois distritos;
2. Método para obter número de encomendas por distrito;
3. Método para obter número de produtos vendidos e comprados por distrito;

Basilene 4:

1. Método que agrupa encomendas em contentores de camiões por distrito;
2. Exportação das listagens de contentores para ficheiro Json;

No fim da baseline 4 o produto deve corresponder a todas as users stories identificadas.

4.Ferramentas

Ferramentas para implementação do SCM Plan no projeto a desenvolver:

- **Java** – Linguagem de programação usada para o desenvolvimento do motor de pesquisa e especificação de testes;
- **Gradle** – Ferramenta de configuração e automação de builds de software;
- **jUnit** – Framework para codificação dos testes especificados;
- **Jacoco** – análise de cobertura do código, efetuando a contagem da percentagem de cobertura dos testes, face ao total de instruções, ramos, linhas de código, métodos e classes.
- **PMD** – Ferramenta para análise estática de código e *checktsyle*;
- **Gitlab**, para:
  - Controlo de versões do código fonte;
  - Gradle para a configuração de builds;
  - Issue tracker e utilização *boards* para desenvolvimento usando metodologias ágeis;
  - Armazenamento da documentação do projecto (por exemplo documento de requisitos, SCMP)