P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Laboratório de Desenvolvimento de Software

Software Configuration Management Plan

Versão 1.0 aprovada

Preparado por Hugo Silva (8180378)

Conteúdo

1. Introdução

1.1 Objetivo

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 LDS. O projeto que foi proposto pelo aluno é um mobilizador de frota de camiões.

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

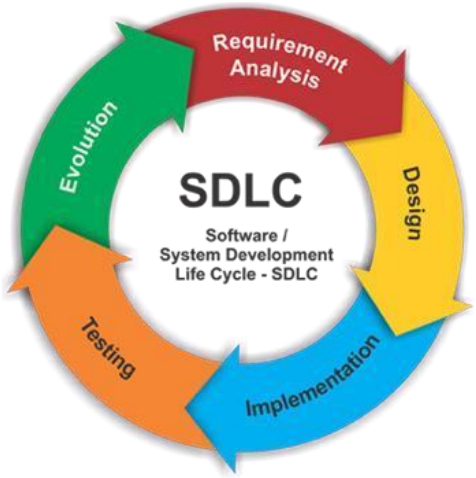
1.4 Referências

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

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

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

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

4. **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*.

5. **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.

6. **Branchs:** Quando efetuado um *merge request* a *branch* deve ser eliminada.
   Nota: documentação, configuração do ficheiro yml pode ser realizado diretamenta na *branch master*.

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

8. **Source code scripts:** Os nomes dos métodos devem respeitar o *style* camelCase.

9. **Base de dados de testes:** Deve ser configurada uma base de dados em mémoria, nunca deve ser usada a base de dados SQL Server.

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: Hugo Silva

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 o estudante deve registar um *runner* local apartir do seu computador para assim proceder a automatização de builds e testes.

3.1.1 Gestão de Branchs

1. O projeto terá uma *branch* Master definida depois da baseline* de configuração inicial.
2. Para cada funcionalidade a ser desenvolvida ou pedido de alteração a ser resolvido deve ser criada uma *branch* apenas para o efeito.
3. *Upload* de documentos e *mockups* pode ser feito diretamente para a branch Master.
4. O nome da *branch* deve seguir o estilo CamelCase.

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 trabalho a realizar será feito de forma indivudual, no entanto é demonstrado o processo de controlo de alterações que normalmente deve ser feito num projeto desenvolvido em equipa. Não será abordado *change requestes* pedidos por clientes 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 pelo estudante. Quaisquer medidas corretivas necessárias serão efetuadas, assegurando desta forma estabilidade do SQA.

O processo de analise e alterações deve estar presente no GitLab. Como será usado SCRUM as auditorias e revisões aos processos serão realizadas através das *retrospective* e *review meeting*.

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 - O estudante faz a analise e decide aceitar ou não a Issue;

3º -Procede à definição ao planeamento da issue, para que esta possa ser feita;

4º - A issue é realizada no ambiente de *development*, sendo que deve ser criada uma *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 - O estudante aprova o Merge Request;
9º - Realiza o merge e fecha a *issue*;

10º - Processo deve ser documentado 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 (Configuration Initial):

1. Documento de requisitos;
2. Documento SCMP;
3. Documentos SCMP e Backlog colocados no wikis do git.

Baseline 2 (Product baseline):

Contém a documentação funcional e física selecionada, necessária para os diferentes tipos de teste dos itens de configuração.

Baseline 3 (Functional baseline):

Deverá definir os requisitos funcionais do sistema ou as especificações do sistema e as características da interface. Deverá documentar a capacidade do sistema (o que o sistema é capaz de concretizar), as funcionalidades presentes no mesmo e a performance caso seja um item de medição necessário;

No fim da baseline 3 o produto deve corresponder a todas as *users stories* identificadas.

4.Ferramentas

Ferramentas para implementação do SCM Plan no projeto a desenvolver:

- **Visual Studio-** para implementação do *front-end*;
- **NUnits-** Biblioteca de testes, desenvolvida para C#.
- **Microsoft Visual Studio-** para realização de testes e desenvolvimento do *back-end*;
- **Gitlab**, para:
  - Controlo de versões do código fonte;
  - Runners para a configuração de builds e realização de testes;
  - Issue tracker e utilização *boards* para desenvolvimento usando metodologias ágeis;
  - Armazenamento da documentação do projecto (por exemplo documento de requisitos, SCMP)