**Centro de Ciências Tecnológicas**

**Disciplina: Análise e Projeto de Sistemas I**

**Curso: Ciência da Computação**

**Prof. Everton Guimarães**

**Parte 1 – Revisão Teórica**

Conceitos abordados na disciplina

- Incluir descrições formais e informais utilizando exemplos.

**1) Gestão e Itens de Configuração**

A **Gestão de Configuração** de Software consiste na *coordenação* do desenvolvimento de software para minimizar entropia, se tornando um importante elemento para garantia da qualidade de software. No ciclo de desenvolvimento de software existem artefatos que representam as mais diferentes informações do processo de desenvolvimento. Estes artefatos podem ser entendidos como **itens de informação**. Um subconjunto dos **itens de informação** são classificados como **itens de configuração**, uma vez que são escolhidos para terminar a **configuração de software**. Os **itens de configuração** pode ser de dois tipos: (i) produto de software, que está relacionado a programas, procedimentos, documentação e informações designadas *a serem entregues a um cliente ou usuário final*; e (ii) produtor de desenvolvimento de software, que está relacionado a descrição de processos, procedimentos, planos, programas e demais documentos, *que podem ou não ter finalidade de ser entregue ao cliente ou usuário final*. Em suma, diz-se que um conjunto de **itens de configuração** de software que estejam inter-relacionados compõe a **configuração de software**. Para que a **gestão de configuração de software** seja executada, algumas tarefas são definidas:

* Identificação – Determina como uma organização *identifica* e *administra* as diferentes versões existentes dos itens de configuração, possibilitando assim que mudanças sejam realizadas de forma mais eficiente.
* Controle de Mudanças – Determina responsáveis pela aprovação e define quais são as prioridades de mudanças.
* Controle de Versão – texto.
* Auditoria de Configuração – texto.
* Relato de Situação – Determina que mecanismos podem ser utilizados para notificar demais integrantes de um time de desenvolvimento sobre as mudanças realizadas nos itens de configuração.
* Controle de Interface – Determina como ocorre gerenciamento de mudanças nos itens de configuração que são afetados por demais itens que não estejam sendo controlados. Por exemplo, pode-se determina como é realizado o gerenciamento causado por alterações externas ao sistema.
* Controle de Subcontratos e Fornecedores – Preocupa-se em definir meios para garantir que módulos / componentes construídos por terceiros estejam corretos e coerentes com o sistema em fase de desenvolvimento. Ou seja, realiza-se do controle para determinar a forma como os itens de configuração, desenvolvidos por solicitação a outras empresas ou foram adquiridos já prontos, deverão ser incorporados ao repositório do projeto.

É importante salientar que existem algumas **tarefas preliminares** que devem ser executadas anteriormente as atividades inerentes a **gestão de configuração**:

1. Seleção os itens a serem gerenciados – neste tarefa deve-se selecionar quais itens de informação serão incorporados como **itens de configuração** de software. Por exemplo, em processos de desenvolvimento mais complexos pode haver uma ampla gama de documentos. No entanto, somente alguns desses documentos são realmente imprescindíveis para atividades que vão, por exemplo, desde a fase de concepção até o produto final do software. Dessa forma, nesta tarefa devemos definir quais itens devem ser priorizados adotando-se critérios como: (i) itens mais usados no ciclo de vida; (ii) itens mais genéricos que contem informações transversais a todo o processo de desenvolvimento; (iii) itens mais importantes para segurança; (iv) itens projetados para reuso; e (v) itens passíveis de atualização simultânea. Somente os artefatos como **itens de configuração** do software deverão ser controlados (ex. Mudanças, versionamento).
2. Para a seleção dos **itens de configuração** podem ser adotados diferentes critérios, tais como: (i) Identificação dos relacionamentos entre itens – Esta atividade é de suma importância para a manutenção de todos os itens de configuração do sistema, uma vez que permite localizar rapidamente os itens afetados por cada alteração; e (ii) Criação de classes de relacionamento – Como exemplos de classes de relacionamentos entre **itens de configuração**, podemos citar equivalência, dependência, derivação, sucessão, dentre outros. Podemos citar, por exemplo, a derivação de um código fonte a partir de um documento de projeto de software ou ainda um relacionamento de dependências, se considerarmos a descrição do projeto modular que é dependente da especificação da arquitetura do sistema.
3. Planejamento das **linhas de base** – A criação das linhas de base pode considerar como marco do projeto, por exemplo, o término de cada ciclo de interação do ciclo de vida do processo de desenvolvimento de software. Dessa forma, a especificação dos **itens de configuração** devem ser revisados e armazenados/atualizados em casa uma das linhas de base.
4. Descrição do Arquivamento e recuperação dos **itens de configuração** do repositório – Nesta parte define-se a estruturação do repositório e permissões de acesso a operações no repositório do sistema de software.

**2) Linha de Base**

**Conceito:** A linha de base é um ponto de referencia no ciclo de vide do desenvolvimento de software marcado pelo término ou aprovação formal de um conjunto de produtos predefinidos.

**Objetivo:** O objetivo de uma **linha de base** consiste na redução da vulnerabilidade a mudanças não controladas por meio realização do controle de mudança formal dos principais entregáveis do sistema (ou itens de configuração). Em outras palavras, uma **linha de base** ajuda a realizar o controle de mudanças sem impedir que aquelas mudanças que são justificáveis em determinados **itens de configuração** sejam realizadas.

**O propósito das linhas de base pode ser resumido em:**

* Definir pontos de progresso mensuráveis dentro do ciclo de vida de desenvolvimento de software.
* Um ponto base para o controle de mudanças em frases subsequentes do projeto
* Uma referencia estável para o trabalho futuro.
* Pontos intermediários e finais para mensurar a aptidão para o propósito dos produtos provenientes do projeto de software.

A definição de **linhas base** eficazes deve conter características: (i) a **linha de base** deve estar associada com a produção e aprovação formal de um entregável físico, tais como um documento ou componente de software; e (ii) todos os itens associados com a **linha de base** devem ser colocados sob um controle de mudanças formal.

Como exemplo de linhas base tipicamente definidas de acordo com o ciclo de desenvolvimento de software, podemos citar:

* Especificação dos requisitos do sistema (linha de base funcional)
* Projeto de alto nível (linha de base preliminar)
* Projeto detalhado (linha de base de projeto)
* O produto de software gerado após a realização dos testes do sistema (linha de base do produto)
* O produto de software em seu ambiente operacional (linha de base operacional)

**Obs.** Um item de configuração de software que passam por uma linha base, diz-se que ele “*tornou-se”* ou “*faz* *parte*” de uma determinada linha base. Logo, quando um **item de configuração** faz parte de uma linha base, ele certamente possui as seguintes características: (i) O item foi revisto formalmente pelos responsáveis da equipe de desenvolvimento e houve acordo entre as partes para sua aprovação; (ii) O item é armazenado em um repositório de itens de configuração; e (iii) somente por ser modificado por meio de procedimentos de controle de mudanças.

Para especificação de uma **linha de base**, pode-se definir algumas informações:

* Os nomes dos itens físicos que constituem a linha de base (documento de projeto, especificação da arquitetura, código fonte).
* O marco em um projeto onde espera-se o cumprimento / finalização de uma linha de base (e.x. fim de uma das fases definidas no processo de desenvolvimento).
* O método a ser adotado para aprovar os itens de configuração definidos na linha de base.
* Os indivíduos responsáveis pela aprovação dos itens da linha de base (e.x. autoridades da aprovação)

**3) Repositório de Itens de Configuração**

O repositório de itens de configuração consiste em um local de acesso onde são armazenados os **itens de configuração** de software, depois de liberados por uma linha base. De acordo com as linhas de base estabelecidas, os **itens de configuração** devem ser *identificados*, *analisados*, *corrigidos*, *aprovados*, *armazenados* no repositório. Os **itens de configuração** armazenados no repositório somente poderão sofrer alguma alteração mediante uma solicitação de modificação formalmente aprovada pelo gerente de configuração. Dessa forma, permite-se manter o controle sobre cada um dos **itens de configuração**, de forma a evitar inconsistências.

**Parte 2 – Utilizando Controle de Versões e Mudanças com Git / GitHub**

O seguinte tutorial tem por objetivo exercitar os conceitos relativos a Gestão de Configuração, mais especificamente Controle de Versões e Controle de Mudanças. O cenário criado para execução da prática é similar ao adotado em ambientes de Fabrica de Software.

Alguns passos de como usar o Git no Eclipse.

1. Crie seu usuário no GitHub
2. Acesse o repositório criado pelo professor através do link abaixo.

* <https://github.com/eguimaraes15/unifor-gestao-configuracao>

1. Crie um Fork do repositório para seu usuário no GitHub (veja figura abaixo). Dessa forma, você estará criando uma ramificação do projeto para sua conta do GitHub de forma que agora você pode colaborar com o repositório de outros usuários.

INSERIR FIGURA

1. Uma vez criado um Fork do repositório que se deseja colaborar, você deve agora criar uma cópia local na sua máquina. Lembre-se que o GitHub permite a criação de uma cópia local, onde são feitas as alterações nos **itens de configuração**, após o que cada alteração é salva por meio do commit. Quando se finaliza um conjunto de alterações, pode-se enviar tais mudanças para o servidor remoto onde está hospedado o repositório. Esta operação é realizada por meio do comando de “*pull request*”.

**Parte 3 – Preparação de Ambiente – Usando Git no Eclipse**

**<***Esta seção do documento encontra-se em construção***>**

1. Instalação
   * Instalar o EGit para realizar integração
   * “Help -> Install New Software
   * Download: <http://download.eclipse.org/releases/indigo>
   * Depois que carregar os pacotes selecione:
     1. General Purpose Tools e marque Marketplace Client.
     2. Clique em “Next”
     3. No campo de busca digite Egit
2. Explorando a perspectiva do Git -> “Git Repository Exploring”

* No painel de controle, selecione a opção “Clone Git Repository”
* Configure os dados de conexão com o repositório usando o projeto hospedado no GitHub.
* Configure o diretório de importação do repositório.
* Selecione o Branch de desenvolvimento
* Configuração remota - Origin (Master)
* Realiza importação do projeto
* Uma vezes importado o projeto do repositório, ele ficará disponível na perspectiva Git, mas ainda falta vincular o projeto há alguma perspectiva de edição de código.

1. Importando um projeto do Git para área de trabalho (Eclipse Workspace)

* Clique com o botão direito repositório, importar projeto do repositório
* Importar projetos existentes ou “Use the New Projects Wizard”.
* Selecione o Working Directory.
* Selecionar o Wizard (ex. Java Project)

P.S. Todas as informações de gerenciamento de repositório estarão integradas pelo menu “Team”. Você pode explorar todas as possibilidades.

INSERIR FIGURA

1. Download de Softwares

* Link para Download do cliente GitHub
  + Windows: <https://windows.github.com>
  + MAC: <https://mac.github.com>

Outras Anotações

**(Parte 1)**

* Criar um fork do projeto (ou executar o comando *clone*)
* Criar um branch (branch)
* Verificar modificações (status/diff)
* Saber o que foi alterado (adicionar arquivo)
* Salvar alterações no repositório (realizar commit)
* Empacotar software/itens de configuração (criar Tag / Release)
* Enviar modificações para o repositório máster (push / fetch -> alterar o título -> merge request no Git)
* Recuperar versão atual do repositório (no final, o branch principal acaba sendo o máster)

**Parte 4 – Outros Recursos GitHub (Milestones, Issues, Labels, Releases)**

**<***Esta seção do documento encontra-se em construção***>**

1. Milestones

<Inserir texto>.

1. Issues

<Inserir texto>.

1. Labels

<Inserir texto>.

1. Releases / Tags

<Inserir texto>.