

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | CENTRO UNIVERSITÁRIO DE ANÁPOLIS - UniEVANGÉLICA |  |
|  |  |  | ENGENHARIA DE COMPUTAÇÃO – 7º PERÍODO |  |
|  |  |  | SEMESTRE SELETIVO 2019/1 |  |
| **DISCIPLINA:** |  | Gerência de Configuração de Software | |  |
|  |  |
| **PROFESSORA:** |  | Dr.ª Renata Dutra Braga | |  |
| **TEMA DA AULA:** |  | Nº 19 – Revisão de conteúdo para a 3ª V.A. | |  |
| **DIA:** |  | 23/05/2019 | |  |
| **DURAÇÃO** |  | 2 horas/aula | |  |
|  |  |  | **LISTA DE ATIVIDADES COMPLEMENTARES** |  |

**Questão 01**

Na engenharia de software, o gerenciamento de configuração de software tem como objetivo

1. gerenciar mudanças e seu impacto no desenvolvimento do software.
2. gerenciar a carga dos dados oriundos de sistemas legados e controlar a consistência desses dados.
3. fornecer uma interface de gerenciamento, onde o ad- ministrador e os usuários, com o devido privilégio, podem alterar a forma como o sistema se comporta.
4. planejar a elaboração dos pacotes de instalação e desinstalação automáticos do software e de seus componentes nos computadores onde o sistema será executado.
5. documentar todo o processo de instalação do software, incluindo os pré-requisitos de software e hardware para o correto funcionamento do sistema.

**QUESTÃO 02**

O termo baseline está associado ao gerenciamento de configurações e corresponde

1. à primeira versão reprovada de uma configuração.
2. às sequências de versões de itens de configuração.
3. aos artefatos de software ainda não verificados
4. à base de dados de teste do software.
5. à versão formalmente aprovada de uma configuração.

**QUESTÃO 03**

Sobre o gerenciamento de configurações e mudanças, é incorreto afirmar que:

1. num processo tradicional de desenvolvimento de software baseado no modelo cascata, o software é entregue para a equipe de gerenciamento de configuração depois que o desenvolvimento está completo e os componentes individuais do software foram testados.
2. a definição e o uso de padrões de gerenciamento de configuração são essenciais para a certificação da qualidade para os padrões CMM e CMMI.
3. procedimentos de gerenciamento de mudança dizem respeito à análise de custo e benefício das mudanças propostas, à aprovação das mudanças viáveis e à rastreabilidade de quais componentes do sistema foram alterados.
4. o gerenciamento de configurações nas abordagens de desenvolvimento ágil deve estar baseado em procedimentos e burocracias rígidos.
5. o ponto de partida para o desenvolvimento do plano de gerenciamento de configurações pode ser o levantamento de um conjunto de padrões de configuração que devem ser adaptados para se atender aos requisitos e às restrições de cada projeto específico.

**QUESTÃO 04**

Analise as afirmações a seguir e assinale V (Verdadeiro) ou F (Falso) para cada uma delas. Justifique as afirmações falsas.

* V ) O MPS.Br caracteriza a Gerência de Configuração de Software em três sistemas principais: (a) controle de modificações (mudanças, alterações); (b) controle de versões; e, (c) gerenciamento da construção.

Pág. 1 de 3

* F ) O processo Gerência de Configuração de Software está incluído no nível de maturidade E do MPS.Br.
* V ) O comando “git clone” cria uma cópia de trabalho em um repositório local.
* F ) A correta definição de release está relacionada aos conjuntos de itens de configuração que são formalmente aprovados e que servem de base para as etapas seguintes de desenvolvimento.

**QUESTÃO 05**

Considere as seguintes afirmativas:

1. Seu escopo deve abordar a definição do que será gerenciado, além de especificar as características

que serão utilizadas para identificar os itens de configuração.

1. Deve especificar as ferramentas usadas para o gerenciamento de configurações e os respectivos processos de uso, porém não deve entrar no mérito de definição de políticas a serem adotadas no

controle de mudanças.

1. Deve identificar quem é o responsável por monitorar a execução do, além de descrever como as

mudanças desse Plano serão avaliadas e aprovadas.

Um plano de gerenciamento de configurações de software em desenvolvimento deve estar de acordo com o que consta em

1. I e III, apenas.
2. I, II, e III.
3. I, apenas.
4. II, apenas.
5. III, apenas.

**QUESTÃO 06**

A norma ISO/IEC 12207 provê um conjunto de processos para facilitar a comunicação entre adquirentes, fornecedores e outros stakeholders no ciclo de vida do produto de software. Com relação ao processo Gerência de Configuração definido pela ISO/IEC 12207, avalie as afirmações a seguir.

1. É uma atividade do processo Gerência de Configuração de Software planejar a gerência de

configuração, tendo como tarefas: (a) definir a estratégia de gerência de configuração e, (b) identificar os itens de configuração que serão controlados.

1. A Gerência de Configuração de Software é um processo que está incluído na categoria “Contexto do

Sistema” e na subcategoria “Processos de Apoio ao Software”.

* 1. São saídas do processo Gerência de Configuração de Software: (a) uma estratégia da gerência de

configuração é definida; (b) baselines de configuração são definidas; (c) os itens que necessitam de gerência de configuração são definidos.

* correto apenas o que se afirma em:

1. I.
2. II.
3. I e II.
4. I e III.
5. II e III.

**QUESTÃO 07**

Estabelecer uma política de sincronização de mudanças para cada item de configuração de software, bem como disponibilizar e manter cada versão produzida dos itens de configuração, são atividades típicas

1. da gerência de mudança.
2. do controle de versão.
3. da auditoria de configuração.
4. do repositório de banco de dados.
5. da construção de software.

**QUESTÃO 08**

Pág. 2 de 3

Os mecanismos de controle de versão, integrados com o processo de controle de modificações, implementam dois elementos importantes do controle de modificação. São eles:

I. Determina quais engenheiros de software podem acessar e modificar um determinado objeto de configuração.

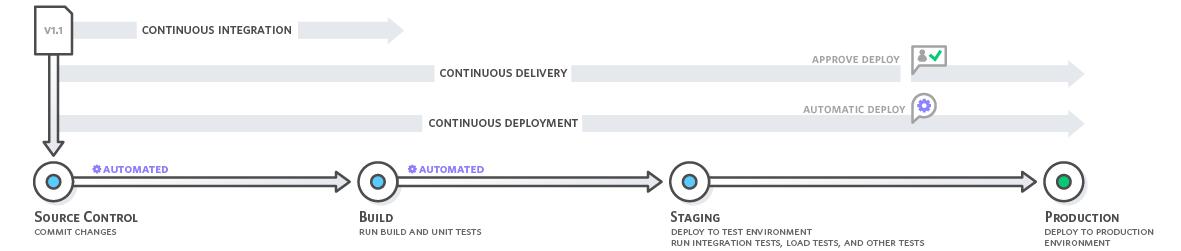
1. Ajuda a garantir que modificações paralelas, realizadas por duas pessoas diferentes, não se sobreponham.

Os elementos I e II são, respectivamente:

1. Controle de segurança e Controle de atualização.
2. Controle de configuração e Controle de modificação.
3. Controle de acesso e Controle de sincronização.
4. Gerência de acessibilidade e Gerência de modificabilidade.
5. Controle de versão e Controle de modificação.

**QUESTÃO 09**

Com a integração continuada, os desenvolvedores frequentemente confirmam um repositório compartilhado usando um sistema de controle de versão, como o Git. Antes de cada confirmação, os desenvolvedores podem escolher executar testes de unidade locais em seus códigos como uma camada de verificação extra anterior à integração. Um serviço de integração contínua cria e executa automaticamente testes de unidade nas novas alterações de código para destacar imediatamente todos os erros.



A integração contínua é referente aos estágios de criação e teste de unidade do processo de lançamento de software. Cada revisão confirmada aciona criação e teste automatizados. Com a distribuição contínua, as alterações de código são criadas, testadas e preparadas automaticamente para que a produção seja liberada. A distribuição contínua expande com base na integração contínua ao implantar todas as alterações de código em um ambiente de teste e/ou ambiente de produção, após o estágio de criação.

Fonte: https://aws.amazon.com/pt/devops/continuous-integration/

**Importância**

A integração contínua é uma técnica de desenvolvimento que busca diagnosticar os bugs (defeitos) e problemas de integração mais rapidamente, utilizando um repositório de código único e executando o build (construção) do software diariamente.

**Vantagens**

Imagine vários desenvolvedores codificando módulos de um mesmo projeto, é natural que colisões de código possam surgir. É bem provável que, em algum momento, um desses módulos seja alterado, simultaneamente, por dois ou mais desenvolvedores ocasionando problemas

A integração contínua atua exatamente antecipando tais situações, permitindo que o time reaja imediatamente e garanta uma evolução muito mais segura do produto. Afinal, “é melhor prevenir que remediar”’.

-possibilidade de restaurar versões anteriores do sistema;

-comparar códigos;

-gestão de alteração, etc.

**Ferramentas**

Jenkins

É uma ferramenta de integração contínua, automatizada, que traz diversos benefícios. A principal funcionalidade dela é construir o projeto por completo de forma automática, rodando os testes disponíveis, a fim de detectar antecipadamente os erros, reduzindo riscos.

CVS

-TortoiseCVS

-CVS - Sistema de Versões Concorrentes

Subversion

-Tigris.org

-Apache Subversion

Git

-Site Git

-GitHub

-GitHub Profissional Services

Pág. 3 de 3