Perguntas Teóricas

Object Constraint Language (OCL)

**Identifique os três ‘Collection types’ (tipos de dados ‘coleções’) pré-definidos na OCL e explique quais as diferenças existentes entre esses tipos de dados que identificou.**

Set: cada elemento é único (ocorre apenas uma vez)

Bag: podem existir elementos repetidos

Sequence: é como “bag” mas com os elementos ordenados

**Tendo em consideração o que estou na unidade curricular “Engenharia de Software II, explique o que entende por OCL (Object Constraint Language).**

OCL – Object Constraint Language – é uma linguagem declarativa que permite definir restrições em um modelo orientado a objeto que não pode ser especificado através de diagramas.

Object Oriented Software Quality Metrics

**No contexto da avaliação da qualidade do software orientado a objetos, explique o que entende por identity harmony (harmonia de identidade no desenho de software), collaboration harmony (harmonia de colaboração no desenho de software) e classification harmony (harmonia de classificação no desenho de software).**

Identity Harmony: qual a justificação para a sua existência. Implementa um conceito específico. Como é feito.

Collaboration Harmony: Cada artefacto colabora com outros para concluir tarefas. Faz tudo sozinho ou usa outros. Como é a interação com os outros.

Classification Harmony: Definido com respeito a ancestrais/descendentes dos artefactos. Combina elementos de identity e collaboration harmony no contexto da herança.

Code Smells/Refactoring

**De entre as seguintes quatro atividades realizadas sobre código fonte de uma aplicação, indique quais identifica como tratando-se de refactoring e porquê: 1) Debugging da aplicação; 2) Implementação de novas funcionalidades na aplicação; 3) Mudança do código fonte da aplicação para melhoria do desempenho (rapidez); 4) Mudança do nome de um atributo para melhorar a compreensão do código fonte.**

Refactoring é: alteração feita na estrutura interna do software para tornar mais fácil entender e tornar mais barato modificar sem mudar o seu comportamento observável. Logo, apenas o ponto 4) é refactoring.

**Indique três métricas que são a base para determinar a qualidade de uma classe e explique porquê que essas métricas podem ser usadas para esse efeito.**

Lazy Class – Classes que não fazem o suficiente devem ser eliminadas;

Data Class – São classes que têm variáveis, têm get’s e set’s mas não têm mais nada;

God Class – Quando uma classe tenta fazer muito, mostra que muitas variáveis.

**Um dos métodos de deteção de code smells é através do uso de regras. Indique, justificando, quais as métricas que usava nas regras para deteção de um God Class.**

Métrica a usar para a God Class – Weighted Methods Count Of Not Acessor Or Mutator Methods

**O code smell Duplicate Code é mais complexo do que o nome indica. Explique a que se deve este facto, explicando o que é o code smell Duplicate code e os seus tipos.**

Duplicate Code – quando se encontra a mesma estrutura da código em mais do que um sítio.

Vários tipos:

* Mesma expressão aparece em 2 métodos na mesma classe;
* Mesma expressão está em 2 subclasses;
* Mesmo código em 2 classes que não estão relacionadas.

Continuous Delivery

**No contexto do que estudou na UC Engenharia de Software II, explique o que entende por cada um dos seguintes conceitos: Continuous Integration, Continuous Delivery e Continuous Deployment.**

Continuous Integration (Jenkins): Significa que as cópias de trabalho do developer são sincronizados com uma branch compartilhada várias vezes ao dia. Integração e testes são feitos automaticamente.

Continuous Delivery (DevOps): Ser sempre capaz de pôr em produção.

Continuous Deployment (Docker): Faz automaticamente deploy do produto na produção, sempre que passar na garantia de qualidade.

**Identifique o nome de uma ferramenta de suporte à entrega contínua de software (Continuous Delivery) que conheça e explique as funcionalidades que essa ferramenta possui no suporte à entrega contínua de software.**

Ferramenta a usar é o DevOps. Cadeia de ferramentas DevOps:

1. Plan – Desenvolvimento e revisão de código
2. Create – Ferramentas de integração contínua, estado de compilação
3. Verify – Teste contínuo que fornecem feedback sobre riscos de negócio
4. Package – Repositório de artefacto, etapa pré-implementação de aplicação
5. Release – Gerar a mudança, aprovar a release, automação da release
6. Configura – Configuração e gerenciamento infraestrutura
7. Monitor – Monitoramento de desempenho de aplicações, experiência do utilizador final.

Software Evolution

**Identifique quatro causas que estão na origem/necessidade de realizar manutenção/evolução do software.**

Causas para a manutenção do sistema:

* Mudar o sistema para corrigir bugs/vulnerabilidades
* Atualização do sistema operacional
* Modificação do sistema para satisfazer novos requisitos
* Corrigir deficiências de forma a atingir os seus requisitos

**Tendo como referência a fase de manutenção/evolução de um projeto de software, explique o que entende pelos conceitos Evolution, Servicing e Phase-out.**

Evolution: Os engenheiros ampliam as capacidades e funcionalidades do sistema para ir ao encontro das necessidades do cliente.

Servicing: O software está sujeito a pequenos reparos de defeitos e mudanças muito simples na função.

Phase Out: Não é realizado mais servicing e os proprietários procuram gerar receita com o uso do software durante o maior tempo possível.