Definição de Construção de software(CS)

É a referência aos detalhes sobre a criação de um software funcional através de combinação de programação, de testes unitários, testes de interação e debugging.

CS.1 Quais são os 5 Fundamentos de CS?

1. Minimizar complexidade
2. Antecipar mudanças
3. Construção para verificação
4. Reuso
5. Padrões na construção

CS.2 O que é refatoração?

Refatoração (do inglês Refactoring) é o processo de modificar um sistema de software para melhorar a estrutura interna do código sem alterar seu comportamento externo.

CS.3 Considerações Práticas

1.O que é um Framework?

É uma coleção de classes abstratas, concretas e interfaces entre elas.

CS.4 Qual ou Quais dos 5 fundamentos de CS é ou são mais importantes? Desenvolva o por que.

Acho que todos os fundamentos são essenciais para a construção de um software, minimizar complexidade é importante, pois vai ajudar a desconstruir a complexidade de um software, diminuindo tempo e recursos. Antecipar mudanças auxilia na tomada de decisões, evitando problemas e possíveis alterações no projeto. Construção para verificação identifica as falhas durante a construção e teste, utilizando teste unitário e automatizado. Padrões na construção auxiliam as equipes a trabalharem de forma ordenada, cada um fazendo sua parte conforme planejado, evitando perda de tempo, desenvolvendo softwares limpos, práticos e reutilizáveis. Reuso creio que seja um fundamento opcional e não essencial para construção, pois reutiliza códigos-fontes e bibliotecas utilizadas em outros projetos.

CS.5 Como você diferenciaria complicado de complexo? De um exemplo.

Complicado: Muitas partes operam entre si de maneiras previsíveis, mas o “como funciona” não é visto tão facilmente, a não ser que você seja um especialista no assunto. Exemplo: dirigir um carro, onde, os comandos dados são respondidos, portanto, você tem uma previsibilidade sobre o que vai acontecer. O que você tem que fazer é dar as “ordens certas”.

Complexo: São imprevisíveis, até por que as interações entre as partes mudam constantemente e com as mudanças surgem novos resultados. Exemplo: o tráfego de

carros, mudando constantemente em reação ao tempo, acidentes, acontecimentos, entre outras mil coisas.

CS.6 Testes Automatizados auxiliam em quais fundamentos de CS?

[ ] Minimizar Complexidade
[ ] Antecipar Mudanças
[x] Construção para Verificação
[ ] Reuso
[ ]Padrões na Construção

CS.7 O que é reuso? Como podemos alcançar reuso na construção de software?

É a utilização de um recurso existente para solucionar problemas diferentes, alguns recursos são: bibliotecas, código-fonte, módulos e Comercial off-the-shelf.

CS.8 O que é TDD? Você concorda ou discorda sobre TDD?

TDD é a sigla para Test-Driven Development, que em bom português quer dizer "Desenvolvimento Guiado a Testes". A própria tradução já entrega o que está por trás dessa técnica. Em se tratando de "Desenvolvimento Guiado a Testes", a premissa trazida por esse conceito é que codifiquemos o teste antes mesmo de codificar a aplicação, ou seja, sendo curto e grosso: a aplicação nem existe e já há um teste esperando por ela.

O TDD (Test Driven Development) baseia-se em três passos, vermelho-verde-refatora. O vermelho é a escrita do primeiro teste antes mesmo da lógica existir. O verde é o ponto em que a lógica para que o teste previamente criado passe. Esta lógica deve ser desenvolvida da forma mais simples possível eliminando complexidades desnecessárias fazendo com que a evolução do código ocorra de forma segura. O refatora é a melhoria do código. Neste ponto são removidas duplicações, múltiplas responsabilidades e o código fica cada vez mais próximo de sua versão final.

Concordo com essa técnica, pois o programador perde em linhas de código por hora, mas ganha horas, dias, quem sabe até semanas na prevenção de novos erros e correção dos que ocorrem durante a implementação de uma nova ferramenta. Detalhe que o método TDD parte do princípio que o teste sempre deve ser implementado antes do código de produção.