Resumo prova 01

Fernanda Guimarães - 2016058166

DCC/UFMG - Engenharia de Software - Prof. Marco Tulio Valente (mtov@dcc.ufmg.br)
Revisão para a Prova 1:

1 Qual a diferença entre requisitos funcionais e não-funcionais?

- Funcionais: "o que"um sistema deve fazer; quais funcionalidades ou serviços ele deve implementar. Exemplo: quando uma nota de uma prova for lançada, os alunos devem ser notificados por e-mail
- Não-funcionais: "como"um sistema deve operar, sob quais "constraints"e qual qualidade de serviço ele deve oferecer. Exemplo: o e-mail acima deve chegar em até um minuto.

2 Descreva cinco tipos de requisitos não-funcionais.

- Desempenho: "deve dar o saldo da conta em menos de 5 segundos"
- Disponibilidade: "deve estar no ar 99.99% do tempo"
- Capacidade: "deve ser capaz de armazenar dados de 1M de clientes"
- Tolerância a falhas: "deve continuar operando mesmo se São Paulo cair"
- Segurança: "deve criptografar todos os dados trocados com as agências"

3 Descreva o principal benefício de um bom projeto modular:

- Paraleliza o desenvolvimento (cada "time"trabalha em um módulo)
- Facilita o entendimento (você pode começar a contribuir após dominar um único módulo)
- Facilita manutenções (que tendem a ficar "isoladas"em módulo)

4 Todo defeito ou bug em um software causa uma falha? Sim ou Não? Justifique.

Nem todo defeito/bug causa falhas; pois o código defeituoso pode nunca vir a ser executado/testado.

5 Defina refactoring. Dê três exemplos de refactoring.

Refactoring: manutenção (perfectiva) realizada exclusivamente para incrementar manutenibilidade; ou seja, não corrige bug, não implementa nova funcionalidade etc
Exemplos clássicos:

- Rename variable/class/etc
- Extract function/class/interface/package
- Move function/class

6 Defina monorepos.

Um único repositório monolítico (isto é, um monorepo), gerenciado por um sistema de controle de versão proprietário

7 Essencialmente, qual a principal motivação (ou inspiração) de processos de desenvolvimento do tipo Waterfall. Isto é, por que eles foram os processos propostos inicialmente para desenvolvimento de software?

Muito usado quando os custos de uma falha de design ou implementação podem ser enormes, até em termos de vidas humanao. Parte do sucesso pode ser explicada pela sua "padronização"pelo Departamento de Defesa Norte-Americano, em 1985.

8 Liste três fatores de qualidade externa de software. Liste três fatores de qualidade interna.

Qualidade Externa: Correção, robustez, reusabilidade. Qualidade Interna: Modularidade, legibilidade, testabilidade.

9 Explique e descreva a classificação de sistemas de software do tipos A, B, C.

Sistemas C (casuais):

- não existe pressão para níveis altos de qualidade
- podem ter bugs
- 1 ou 2 engenheiros
- sistemas pequenos, algo sem importância
- tipo mais comum

Sistemas B (business):

- sistemas críticos (geram lucro, etc)
- risco: não usarem técnicas de ES e se tornarem um "passivo"ao invés de um "ativo"para a empresa

Sistemas A (acute):

- sistemas onde nada pode dar errado, pois o custo é imenso, em termos de vidas humanas e/ou grandes montantes financeiros
- exemplos: sistemas de transporte, médicos, aviação, espaciais etc
- requerem certificações normalmente

10 Qual a principal diferença entre XP e Scrum? Qual a principal diferença entre Kanban e Scrum? Qual a principal diferença entre Lean e Scrum?

Diferença XP e Scrum:

- Scrum é um método ágil para gerenciamento de projetos, mas não necessariamente de software
- Scrum não tem preocupação com práticas, pois seu objetivo é mais amplo.

Diferença Kanban e Scrum: Kanbam e Scrum:

- Scrum - sprints + "quota"de tarefas em cada estágio
- Também não tem scrum master, product owner, pelo menos explicitamente

Lean e Scrum:

- Lean é um método "ágil"para gerenciamento de startups, com ciclos curtos e feedback, ou seja, é mais específico que o Scrum.

11 Descreva um valor de XP? Descreva dois princípios de XP? Descreva três práticas de desenvolvimento propostas por XP?

Valor: feedback constante: pois em sistemas de software, dada à complexidade dos mesmos, pode ser difícil ter a solução "certa", logo de início;

Princípios:

- Flow: priorizar um fluxo contínuo e produtivo de atividades.
- Baby Steps: uma feature de cada vez (a de maior prioridade para o cliente); uma sub-feature de cada

vez etc.

Práticas:

- Pair programming: toda tarefa de programação (design, codificação, testes) deve ser feita por dois programadores, em conjunto;
- Iteração: planeje seu trabalho semanalmente, de forma que ao fim de toda semana deve-se produzir "deployable software".
- TDD: não só ter muitos testes, mas escrever esses testes antes da fase de codificação (e claro, são sempre testes automatizados).

12 O que é uma slack no contexto de XP?

Introduza nos ciclos algumas "folgas", isto é, tarefas menos importantes, que possam ser descartadas, caso o projeto fique atrasado

- Refatorações
- Pesquisa e prospecção de novas tecnologias
- Alguma documentação ou manual de uso
- Seminários internos etc

13 Basicamente, quando ocorre um conflito de integração?

- Ao implementar F, você alterou um arquivo X
- Ao mesmo tempo, outro desenvolvedor alterou as mesmas linhas de X e integrou o código dele! Antes de você integrar ....
- O que ocorre quando você for integrar o seu código? Um conflito

14 Por que XP advoga integrações contínuas?

(1) feedback rápido; (2) evitar "integration-hell"(isto é, quando a integração é mais custosa que o desenvolvimento)

15 Qual a diferença entre integração contínua, continuous delivery e continuous deployment?

- CI: continuous integration - prática na qual membros do time integram seus trabalhos frequentemente. Testes automáticos. Everyone merging code changes to a central repository multiple times a day.
- CY: continuous delivery: do CI, plus automatically prepare and track a release to production (anyone with sufficient privileges can do this in a few clicks).
- CD continuous deployment: é uma evolução de Continuous Integration (CI)- logo após integrado, código é também liberado para uso. Continuous deployment is like continuous delivery, except that releases happen automatically.

16 Cite duas práticas propostas por XP que acabaram não sendo largamente adotadas.

Story points e pair programming.

17 Em projetos XP, não existe a figura de um arquiteto de software, já que o projeto/arquitetura não são definidos up front (isto é, eles também são incrementais). Verdadeiro ou Falso. Justifique.

Falso, podem existir arquitetos. A arquitetura também é definida de forma incremental; o sistema vai sendo particionado, à medida que evolui

18 Quem define o tamanho das estórias em times XP? O que é a velocidade de um time?

São estimadas pelos programadores; que definem quanto tempo leva-se para implementar cada estória (porém, não podem ser muito complexas).

Velocidade: capacidade de trabalho de um time, em uma iteração Exemplo: a velocidade de um time = 26 (significa que ele é capaz de implementar 26 story points em uma iteração)

19 No contexto de Scrum, o que é grooming?

Tarefas de "cuidar"do product backlog, incluindo: (1) criar e refinar estórias; (2) estimar estórias; (3) priorizar estórias. O time todo realiza, chefiado pelo product owner.

20 Qual a diferença entre sprint review e sprint retrospectiva, em Scrum.

Sprint Review:

- Reunião que marca o fim de um sprint, com participação inclusive de stakeholders envolvidos com o resultado do sprint

Retrospectiva:

- Reunião interna para refletir sobre o processo e melhorá-lo. Última atividade do sprint.

21 Suponha que você está fazendo um sistema para um público externo e amplo (por exemplo, um sistema de Q&A, como o do trabalho), usando Scrum. Como escolher o product owner neste caso?

- Product Owner pode ser um cliente externo (por exemplo, quando foi o cliente externo que contratou o desenvolvimento do sistema).
- Ou Product Owner pode ser alguém da área de marketing ou Vendas da organização que está desenvolvendo o software (isto é, alguém que vai representar os clientes reais).

22 Por que cada atividade de um quadro Kanban possui duas colunas? Qual o significado destas duas colunas?

Com exceção do Backlog, demais fases possuem duas colunas:

- 1a coluna: items em andamento na fase (ex.: em implementação)
- 2a coluna: itens que já terminaram essa fase, mas que ainda não foram movidos para a fase seguinte (ex.: implementados e aguardando entrarem em validação; é uma espécie de buffer para a fase seguinte)

23 Essencialmente, por que o modelo em Espiral não é classificado como um modelo de desenvolvimento ágil?

Most Spiral models of development still insist on big, up-front design. The emphasis is on knowing as much as you can about how the system will be used; discovering all the use cases. Once you know these, then you design the system and break it down into phases that follow an iterative detail-design, implementation, test, refactor-design loop.

24 Quais as fases propostas pelo Método RUP?

- Inception: análise de viabilidade, orçamentos e definição de escopo
- Elaboração: especificação de requisitos (via casos de uso) e da arquitetura
- Construção: projeto de mais baixo nível, implementação e testes
- Transição: disponibilização do sistema para produção
- Essas fases podem ser divididas em iterações (ex.: construção pode ter 4 iterações); pode-se também iterar sobre as 4 fases (como em Espiral)

25 Quando não usar métodos ágeis?

- Requisitos estáveis
- Design conhecido e simples
- Desenvolvedores dominam a área do projeto
- Baixo risco
- Custos de mudanças é alto