Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AD2 1° semestre de 2018.

Atenção: Como a AD é individual, caso seja constatado que provas de alunos distintos são cópias uma das outras, independentemente de qualquer motivo, a todas será atribuída a nota ZERO. As soluções para as questões podem sim, ser buscadas por grupos de alunos, mas a redação final das respostas para as questões da prova tem que ser individual!
Os tópicos tratados como pesquisa na AD2 podem ser questionados na AP2 ou AP3!

1. Ao discutirmos os princípios de projeto de software, observamos um ciclo onde requisitos geram a demanda por mais software, que entra em operação, provoca mudanças no ambiente de trabalho, mudanças estas que acabam gerando novos requisitos. Explique porque este ciclo ocorre e como normalmente é encerrado. (Valor: 1,0 ponto; máximo: 10 linhas)

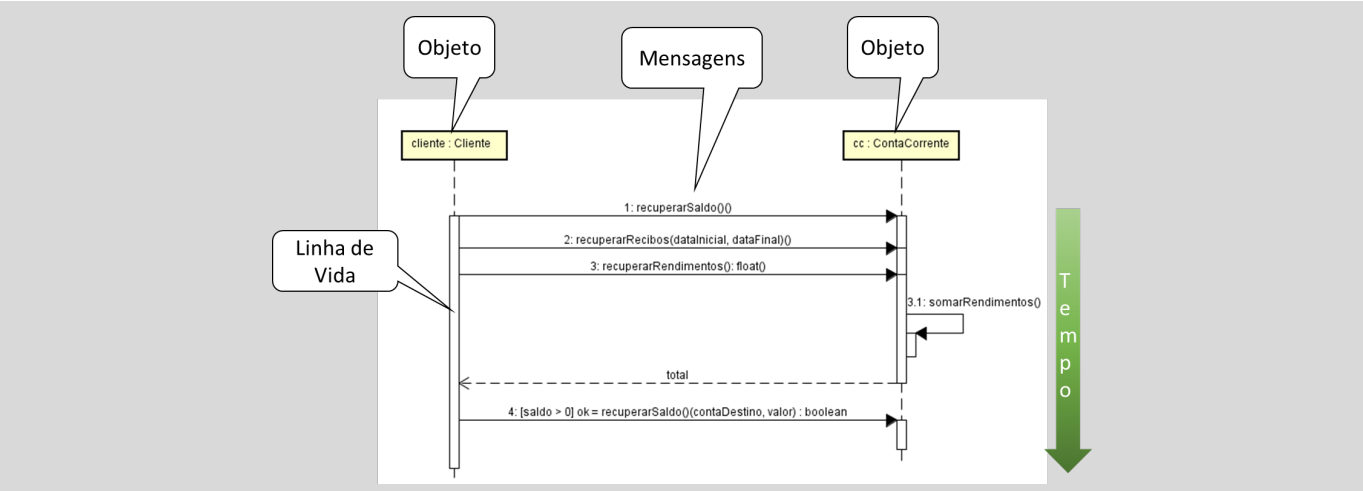
   Um software normalmente é construído de acordo com os requisitos de seus futuros usuários. No entanto, quando o software entra em operação, ele altera o ambiente de trabalho, mudando a forma com que seus usuários trabalham. Desta forma, o software geralmente precisa ser adaptado para a nova realidade, da qual ele participou na formação. Esta é uma das razões pela qual um software usado está em constante alteração, sendo ajustado à medida que os processos que ele suporte evoluem. Este ciclo é encerrado no momento em que o software deixa de ser utilizado.

2. Quais são os componentes de um diagrama de sequência da UML? Mostre um diagrama de sequencia e explique como estes componentes se relacionam e o quê cada um representa no diagrama. (Valor 1,0 ponto; máximo: 10 linhas)

   O Diagrama de Sequência permite representar como os objetos de um sistema OO trocam mensagens entre si para cumprir com algum objeto (por exemplo, realizar um caso de uso).

   Os principais componentes de um diagrama de sequência são Objetos, Linha de Vida e Mensagens.

   A Figura a seguir apresenta um exemplo de um diagrama de sequência. Perceba que a linha do tempo é representada na vertical.



3. O que representa o conceito de coesão em software. Que níveis de coesão podem ser observados no software? Quais os adequados? Como medir coesão? (Valor: 1,0 ponto; máximo 10 linhas)

É uma característica de um elemento que compõe o software que diz que respeito à diversidade de tarefas executadas por este elemento. Por exemplo, um módulo altamente coeso realiza uma única tarefa, sem precisar executar rotinas de outros módulos.

Os níveis de coesão encontrados são Baixo (Coincidental, Lógico, Temporal, Procedural) ou Alto (Comunicacional, Sequencial, Funcional). Em um software, é preferível a coesão Funcional, seguido pelos níveis Sequencial e Comunicacional.

A métrica LCOM (Perda de Coesão em Métodos) pode ser utilizada para medir a coesão.

4. Relacione as afirmações com os conceitos associados a elas (1.0 ponto):

(a) Estilos Arquiteturais	(1) Preocupações com o desenvolvimento de um produto de software, incluindo a concepção, implementação, entrega, uso e manutenção.
(b) Componentes	(2) Reduzir as consequências com eventos inesperados no desenvolvimento ou manutenção do software.
(c) Gerenciamento de Riscos	(3) Solução em alto nível de abstração envolvendo a definição da estrutura e do comportamento sistêmico do software.
(d) Casos de Uso	(4) Abstração correspondente a processos computacionais ou de estruturas de armazenamento.
(e) Ciclo de vida	(5) Explicita a interação a interação com o sistema por meio de cenários e passos bem definidos.

(a) -> (3)

(b) -> (4)

(c) -> (2)

(d) -> (5)

(e) -> (1)

5. Atividade de pesquisa (não vale copiar e colar! Você deve pesquisar e explicar com suas palavras. Indique fontes alternativas que usar!): (valor 2,0 pontos)

O que diz o Código de ética do Profissional de Informática da Sociedade Brasileira de Computação (http://www.sbc.org.br/institucional-3/codigo-de-etica) ? Como ele se compara ao Código de ética e de Prática Profissional da Engenharia de Software, segundo as recomendações da *ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices* (https://www.computer.org/cms/Computer.org/professional-education/pdf/doc.pdf)? O que tem em um que não tem no outro?

Tanto o código de ética da SBC quanto o da ACM/IEEE-CS fornecem diretrizes para a conduta dos profissionais de informática. O código da SBC consiste em um conjunto de 12 artigos que descrevem os deveres dos profissionais de informática. Já o código da ACM/IEEE-CS consiste em um conjunto de 8 artigos.

Uma diferença entre os códigos é que o ACM/IEEE-CS, em sua versão completa, possui um detalhamento dos 8 artigos propostos.

6. Marque verdadeiro (V) ou falso (F) para as afirmações abaixo (valor até 1 ponto – 0,1 por marcação correta. Atenção: cada (2 erros) OR (2 brancos) OR (1 erro + 1 branco) eliminam 1 acerto!):

    a. Especificação de Requisitos incompleta representa uma das causas mais comuns de problemas em projetos de software.
    b. A deterioração da qualidade do software é consequência das alterações de hardware realizadas no ambiente de produção.
    c. Os defeitos que mais aparecem em especificações de requisitos são Informação Estranha e Fato Incorreto.
    d. Sistemas de software devem apresentar alto acoplamento e baixa coesão para reduzir a possibilidade de problemas em tempo de execução e facilitar a manutenção.
    e. O espectro de gerenciamento dos projetos de software deve envolver as pessoas, o problema e o processo.
    f. O custo para correção de um defeito muda muito pouco ao longo do ciclo de vida do software.
    g. Definição, Desenvolvimento e Manutenção representam as fases básicas da engenharia do software.
    h. O projeto é o processo criativo de transformar o problema em uma solução. Entretanto, a descrição de uma solução não pode ser chamada de projeto.

i. Na orientação a objetos, a estrutura e o comportamento dos objetos estão descritos na representação da classe.

j. O número de linhas de comunicação em um projeto depende do número de participantes na equipe. Uma equipe com 10 desenvolvedores implica em 52 linhas de comunicação que necessitam ser gerenciadas.

a. V
b. F
c. F
d. F
e. V
f. F
g. V
h. V
i. V
j. F

7. Uma abordagem orientada a objetos pode ser utilizada para desenvolver qualquer sistema? Quais são os "pontos fortes" da orientação a objetos? Quais são os "pontos fracos"? Dê um exemplo de um sistema em que a orientação a objetos não seria uma estratégia de desenvolvimento apropriada. (Valor até 1,0 ponto, máximo 15 linhas)

A abordagem OO pode ser utilizada para desenvolver qualquer sistema. Entretanto, nem sempre é a mais adequada. O ponto forte da OO é fornecer uma maneira adequada de abstrair o mundo real para representá-lo em um sistema de software.

Entretanto, um software OO tende a consumir mais recursos de hardware devido às camadas de abstração utilizadas pelo paradigma OO. Dessa forma, a orientação a objetos não é indicada para softwares que exigem alto desempenho.

8. Explique a diferença entre teste funcional (caixa fechada), teste estrutural (caixa aberta) e teste baseado em erros, apresentando pelo menos dois exemplos de critérios que podem ser utilizados para projetar casos de teste em cada uma destas perspectivas. (Valor: 1,0 ponto)

Os testes funcionais, também chamados testes de caixa fechada, enxergam o software de forma macro, na perspectiva do usuário, sem perceber detalhes de sua estrutura interna. Os testes funcionais baseiam-se na especificação do software para derivar os casos de testes. Exemplos de critérios que podem ser aplicados para testes funcionais são particionamento em classes de equivalência e Análise de Valor Limite.

Os testes estruturais ou de caixa aberta, baseiam-se na estrutura interna da implementação (por exemplo, código fonte). Geralmente, os critérios utilizados nos testes estruturais são baseados no fluxo de controle do

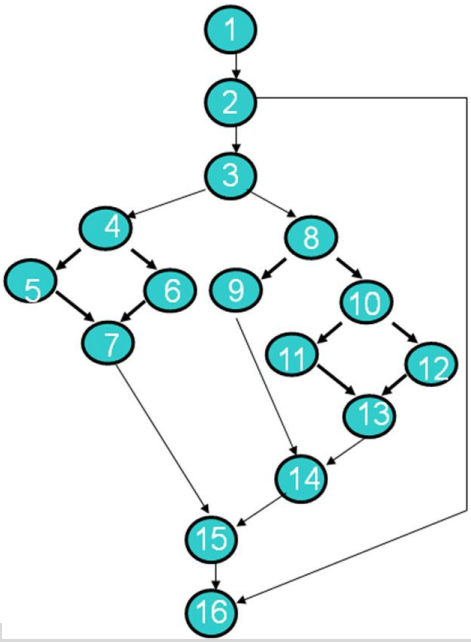
programa. Por exemplo, exercitar todos os comandos (sentenças) ou todas as possibilidades de das instruções condicionais.

Os testes baseados em erros baseiam-se em defeitos já conhecidos e avaliam o software procurando identificar a existência destes tipos de defeitos. Os critérios relacionados aos testes baseados em erros é cobrir todos os erros conhecidos ou apenas erros com determinado nível de complexidade.

9. Calcule a métrica Complexidade Ciclomática para o grafo de fluxo de controle abaixo (mostre como você calculou!) e indique como seu valor pode ser usado para apoiar a tomada de decisão no gerenciamento e desenvolvimento do projeto de software? (valor: 1,0 ponto)

   (fonte: https://guimaraesdani.wordpress.com/tecnicas-de-teste-parte-ii/)

```
1. { int n, lastc, c; n = 1; lastc = getchar();
2.  while(lastc != ENDFILE)
3.          {if ((c = getchar()) = = ENDFILE) {
4.                  if ((n > 1 | | (latsc = = warning))
5.                      putrep (n, lastc);
6.                  Else putchar (lastc);
7.                  }
8.              Else {if (c = = lastc)
9.                          n+ +;
10.                     elseif ((n > 1) | | (lastc = = warning)) {
11.                             putrep (n, lastc) ; n = 1; }
12.                     Else putchar (lastc);
13.             }
14.             lastc = c;
15.         }
16.     }
```



A complexidade ciclomática pode ser calculada através no número de áreas formado pelo grafo resultante do fluxo de controle, incluindo a área externa. Dessa forma, a complexidade ciclomática é 6.

A complexidade ciclomática ajuda a definir a quantidade mínima de casos de teste necessários para cobrir todos os caminhos do fluxo de execução.