Fundação CECIERJ - **Vice Presidência de Educação Superior a Distância**

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AP3 1° semestre de 2016.

Nome –

Assinatura –

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

1) Para as métricas (a-e) abaixo, indique seu nível de precisão (objetiva, subjetiva), escala (nominal, ordinal, intervalar, razão) e objeto de medição (processo, produto). Apresente para cada uma delas 1 exemplo de comportamento do software que é possível observar no contexto do projeto de software. (valor: 2.0 Pontos)

a. Número de Defeitos: objetiva, razão, processo/produto, útil para observar a qualidade dos artefatos, calcular a chance de falha ou apoiar o cálculo da confiabilidade, dentre outros.

b. Tempo de Desenvolvimento (horas): objetiva, razão, processo, permite capturar a duração do projeto, apoiar a observar esforço ou ser usada para o cálculo da produtividade, dentre outros.

c. Nível de Satisfação do Usuário: subjetiva, produto, pode ser ordinal ou intervalar (usualmente *LIkert*), utilizada para apoiar a observação de quão satisfeito os usuários estão com alguma característica do software (sucesso do projeto).

d. Perda de Coesão em Métodos: objetiva, razão, produto. Permite identificar uma eventual perda de qualidade no software por conta de sobrecarga de responsabilidades dos métodos. Pode também apoiar na previsão de falhas.

e. Esforço (Homem/Hora): objetiva, razão, processo. Utilizada como indicador de esforço (Carga de trabalho) no projeto de software.

2) Relacione cada elemento da coluna da esquerda com um e somente um elemento da coluna da direita. (valor 1.0 ponto)

(a) diagrama de classes (1) Explicita os processos de negócio do cliente.

(b) diagrama de casos de uso	(2) Explicita as possibilidades de interação entre os usuários e o sistema.
(c) diagrama de transição de estados	(3) Detalha o comportamento de um objeto no decorrer da sua vida.
(d) diagrama de sequencia	(4) Detalha uma determinada possibilidade de interação entre o usuário e o sistema.
(e) diagrama de atividades	(5) Explicita a estrutura estática interna do sistema.
(f) descrição de casos de uso	(6) Detalha a interação entre diferentes objetos do sistema para atender a uma funcionalidade específica.

A-5; B-2; C-3; D-6; E-1; F-4

3) Explique que diferenças existem entre atividades de manutenção de correção e melhoria, e indique três fatores que podem afetar o esforço de manutenção de software e três responsabilidades da equipe de manutenção. (valor: 2,0 pontos).

Atividades de manutenção tem a intenção de manter o software em funcionamento durante seu ciclo de vida e após sua entrega. Espera-se que apenas manutenções de melhoria ocorram (evolução do software). Entretanto, defeitos no software podem levar a necessidade de correção. Portanto, atividades de correção intencionam corrigir defeitos identificados no software após sua entrega. E atividades de melhoria intencionam aprimorar o software visando estender sua vida útil.

Fatores: Tipo de aplicação, Novidade do sistema, Rotatividade e disponibilidade do pessoal de manutenção, Duração da vida útil do sistema, Dependência de um ambiente que se modifica, Característica de hardware, Qualidade do projeto, Qualidade do código, Qualidade da documentação, Qualidade dos testes.

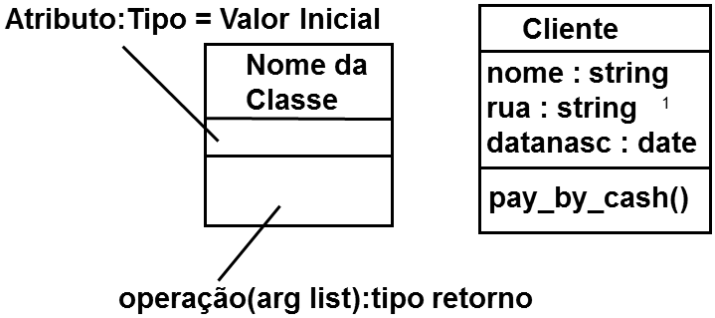
Responsabilidades da Equipe: entender o sistema, localizar informação na documentação do sistema, manter a documentação do sistema atualizada, estender as funções existentes para acomodar novos requisitos ou modificações nos requisitos, acrescentar novas funções para o sistema, encontrar a fonte de falhas ou problemas no sistema, localizar e corrigir faltas, responder questões sobre a forma como o sistema funciona, reestruturar o projeto e codificação dos componentes, reescrever o projeto e código dos componentes, apagar os projetos e códigos de componentes que não são mais úteis, gerenciar trocas para o sistema.

4) Mostre (através de um desenho) quais são as fases e seu encadeamento no ciclo de vida do software em cascata. Cite duas limitações deste modelo de ciclo de vida. Explique como os ciclos de vida chamados incrementais tentam resolver estas limitações (valor: 1,0 ponto)

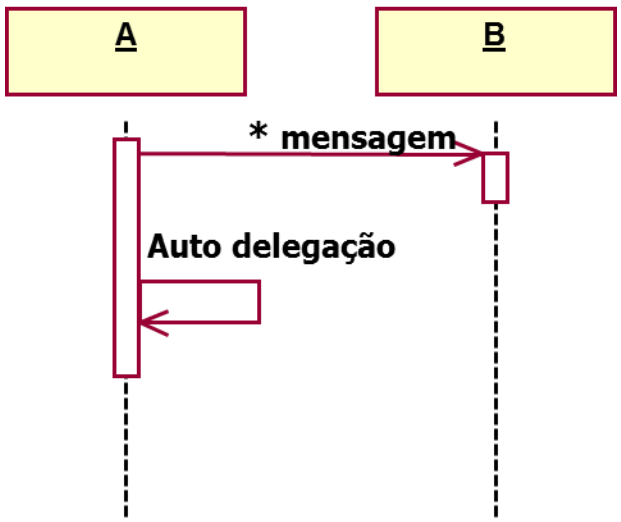
Ciclos de vida cascata organizam a atividade de desenvolvimento de forma sequencial, implicando que uma fase só possa ser iniciada após o término completo da anterior. Por exemplo, atividades de projeto só poderiam ser iniciadas após todos os requisitos e casos e uso serem completamente especificados. Assim, temos as limitações de sequencialidade pura e a eventual demora causada pela expectativa do cliente em receber algo concreto que só ocorrerá ao final do processo. Ciclos incrementais amenizam este problema ao “quebrar” o processo de desenvolvimento em pedaços (incrementos) menores que são entregues à medida que vão sendo liberados.

Uma possível representação:

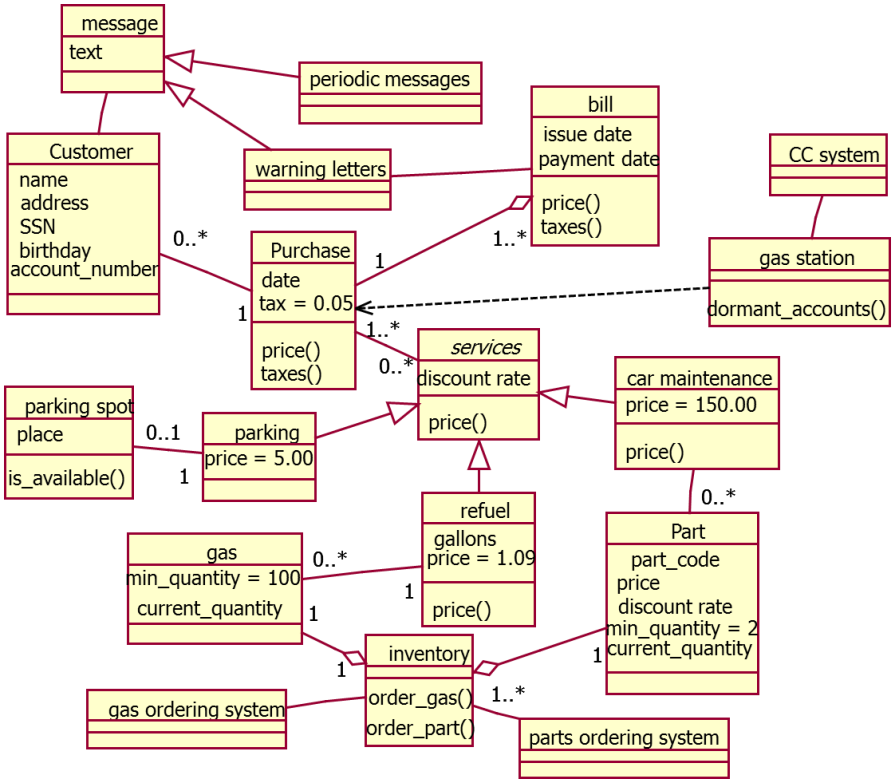
5) Na notação UML, uma classe é representada por um retângulo dividido em 3 regiões. Que informações são inseridas em cada região da classe? Como estas informações aparecem no Diagrama de Sequência? Apresente um exemplo relacionando os dois tipos de diagrama. (valor: 2,0 pontos).

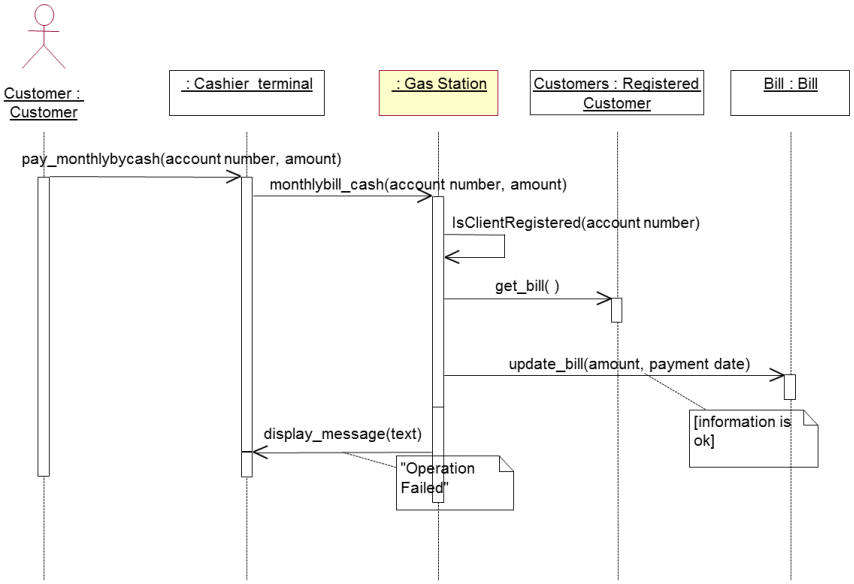


As classes concretas aparecem como retângulos nos diagrama de sequencia, com a linha de vida do objeto representada. As operações/métodos das classes se apresentam como as mensagens trocadas entre os objetos na linha do tempo.



Um possível exemplo (Não precisava ser tão complexo, obviamente!):





6) Marque Verdadeiro (V) ou Falso (F) para as afirmações abaixo (Até 3 pontos – 0,2 por marcação correta. **Atenção: cada 2 erros/2 brancos/1 erro + 1 branco eliminam 1 acerto!)**:

1. Inspeções de Software encontram defeitos sem alterar a produtividade, qualidade e estabilidade dos projetos de software.
2. O melhor testador para o código é o próprio programador.
3. Os resultados dos testes provam que o software está completamente livre de defeitos.
4. Ferramentas como analisadores de código, comparadores de arquivos, repositórios de controle de configuração auxiliam no estágio de manutenção do software.
5. O esforço de manutenção independe do tipo de sistema, seu tempo de vida, e da qualidade do projeto (*design*), uma vez que a primeira lei de evolução afirma que o software vai mudar continuamente.
6. O perfil de experiência do inspetor afeta diretamente a eficiência da inspeção *ad-hoc*.
7. É relevante considerar as necessidades de diferentes *stakeholders* no planejamento e elaboração de materiais de treinamento e documentação.
8. Remover Defeitos é melhor que Prevenir Defeitos.
9. O Critério de Cobertura dos Testes influencia no esforço do teste.
10. O fenômeno de rejuvenescimento do software representa o aumento da qualidade de um sistema existente por meio de atividades como reestruturação, engenharia reversa e reengenharia.
11. Inspeções de Software podem ser aplicadas a diferentes artefatos do ciclo de vida. Por isso, ao usar inspeções não é necessário aplicar testes.
12. O esforço despendido em manutenção perfectiva é proporcionalmente equivalente ao esforço de manutenções preventiva, adaptativa e corretiva juntas.
13. O manual do usuário explica as configurações de hardware e software, os métodos para permitir e negar o acesso de um usuário, os procedimentos para adicionar ou excluir periféricos de um sistema e as técnicas para duplicar ou realizar o backup de arquivos e documentos.
14. As técnicas de inspeção *ad-hoc*, apenas quando aplicadas por inspetores inexperientes, não garantem a cobertura do documento como um todo.
15. O conhecimento adquirido no treinamento pode ser facilmente esquecido para aquelas funções que não são executadas regularmente.

1-F; 2-F; 3-F; 4-V; 5-F; 6-V; 7-V; 8-F; 9-V; 10-V; 11-F; 12-F; 13-F; 14-F; 15-V