UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

PROCESSO SELETIVO – 04/2023

Área de Conhecimento: Engenharia de Software

PROVA ESCRITA – PADRÃO DE RESPOSTA

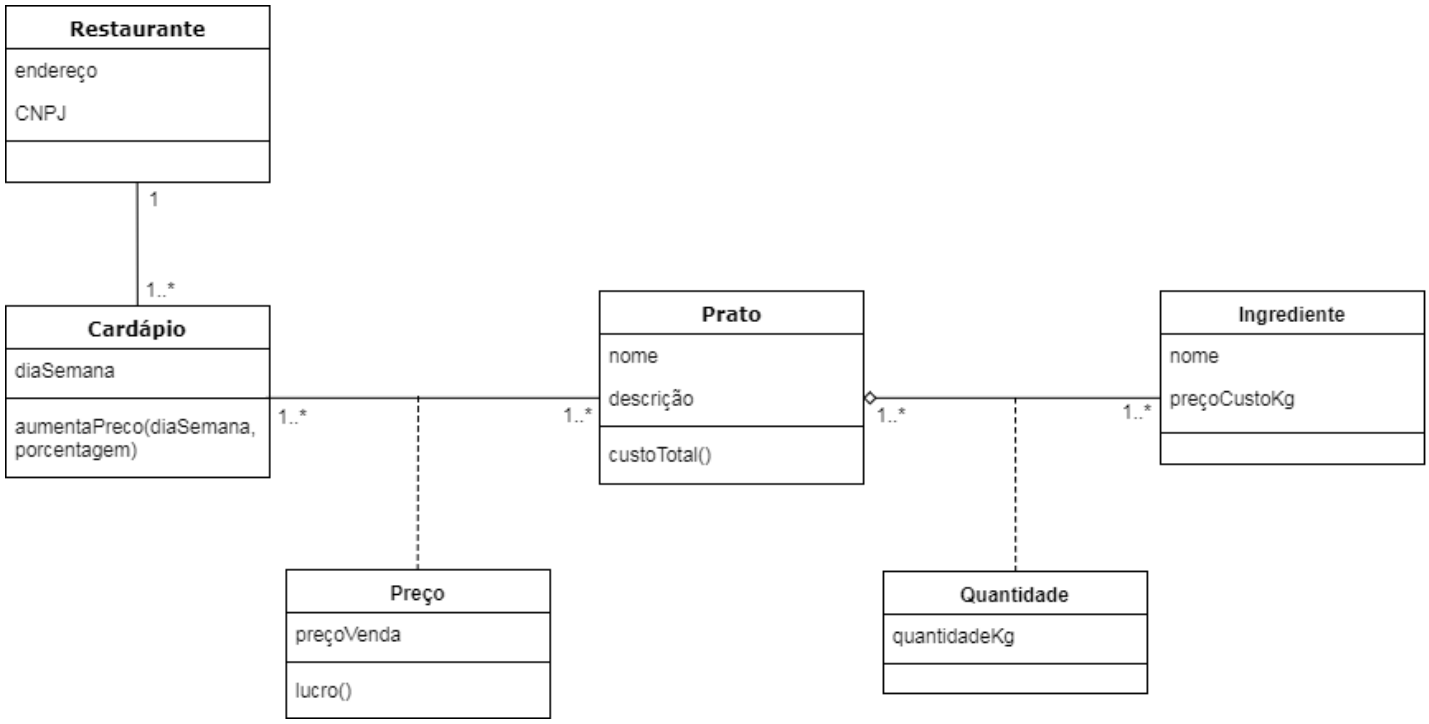
QUESTÃO 1: Ciclo de Vida de Software

Itens que devem ser considerados na resposta para atender ao enunciado, conforme PRESSMAN (2001), SOMMERVILLE (2003) e VALENTE (2020):

- Sobre os métodos tradicionais/prescritivos:
  - Modelo em cascata:
    - Vantagens:
      - produz documentação completa
      - primeiro modelo a disciplinar o processo de software
    - Desvantagens:
      - mudanças nos requisitos não são acomodadas facilmente pelo processo pois os mesmos são tratados apenas no início do processo
      - projeto reais raramente seguem o fluxo sequencial que o modelo propõe
      - uma versão executável do software se torna disponível apenas ao final do processo
  - Modelo Incremental:
    - Vantagem: entregas mais rápidas que o modelo cascata
    - Desvantagem: os incrementos devem ser pequenos quanto mais rápido precisar ser o ciclo incremental, entretanto muitas vezes é difícil mapear requisitos em incrementos adequados
  - Modelo Evolucionário de Prototipagem:
    - Vantagens:
      - desenvolvimento de versões cada vez mais completas a medida que novos ciclos de desenvolvimento são entregues
      - conseguem trabalhar com requisitos instáveis ou que não foram plenamente elicitados no início do projeto
    - Desvantagens:
      - gerenciamento pode ser complexo pois o número de ciclos é incerto
      - cliente enxerga o protótipo como uma versão definitiva, o que pode gerar expectativas errôneas quanto ao prazo de entrega pelo cliente.
  - Outros modelos que podem ser citados como tradicionais/prescritivos: modelo V, modelo W, modelo RAD e Espiral
- Sobre os métodos ágeis:
  - Vantagens:
    - envolvimento do cliente é constante, sendo este parte da equipe
    - entregas incrementais em ciclos iterativos
    - pessoas estão acima do processo
    - mudanças são facilmente acomodadas
    - equipes são auto ajustáveis
  - Desvantagens
    - mudanças na equipe podem comprometer o desenvolvimento de um projeto em andamento pois o processo é fortemente dependente do engajamento de membros da equipe
    - é difícil manter a constância de colaboração de todos os membros da equipe
  - Exemplos: Scrum, XP e Kanban

QUESTÃO 2: Modelagem de Sistemas

Segue abaixo um diagrama mínimo viável que atende ao enunciado desta questão:



QUESTÃO 3: Padrões de Projetos

Segundo o Capítulo 5 de Valente (2020), classes pequenas tendem a ser mais coesas, pois elas apresentam menos responsabilidades. Logo, as responsabilidades presentes na classe tendem a ser mais relacionadas. Por outro lado, classes pequenas costumam depender de outras classes, o que significa que seu acoplamento é maior.

Em resumo:

- Classes ou arquivos grandes apresentam acoplamento menor (o que é bom), mas a coesão também é menor (o que é ruim)
- Classes pequenas apresentam acoplamento maior (o que é ruim), mas a coesão maior (o que é bom).

Portanto, é possível afirmar que existe um "trade-off" entre acoplamento e coesão. Quando se melhora uma dessas propriedades, a outra tende a piorar e vice-versa. Por isso, é importante balancear e combinar coesão e acoplamento, sem privilegiar apenas uma destas.

QUESTÃO 4: Engenharia de Requisitos

As principais atividades a serem realizadas estão envolvidas nas etapas de levantamento e validação de requisitos.
Entre os principais papéis estão, do lado da empresa de desenvolvimento, o Product Owner (PO), que é aquele que capta as necessidades do cliente e serve como ponte entre o cliente e a equipe de desenvolvimento, a equipe de desenvolvimento, composta por um gerente e uma squad de desenvolvimento com desenvolvedores e testadores. No lado do cliente tem-se o stakeholder, que variam desde os usuários finais, passando pelos gerentes do sistema até stakeholders externos, como reguladores, que certificam a aceitabilidade do sistema (Sommerville, 2003).

Dentre as atividades, pode-se destacar primeiramente as usadas para elicitação de requisitos, que incluem entrevistas com stakeholders, aplicação de questionários, leitura de documentos e formulários da organização que está contratando o sistema, realização de workshops com os usuários, implementação de protótipos e análise de cenários de uso. Existem ainda técnicas de elicitação de requisitos baseadas em estudos etnográficos. Etnografia designa a técnica de elicitação de requisitos que recomenda que o desenvolvedor se integre ao ambiente de trabalho dos stakeholders e observe — normalmente, por alguns dias (Valente, 2020).

Requisitos definem o que um sistema deve fazer e sob quais restrições. Requisitos relacionados com a primeira parte dessa definição — o que um sistema deve fazer, ou seja, suas funcionalidades — são chamados de Requisitos Funcionais. Já os requisitos relacionados com a segunda parte — sob que restrições — são chamados de Requisitos Não-Funcionais (Valente, 2020).

No caso de desenvolvimento ágil, a documentação de requisitos é feita de forma simplificada, por meio de histórias do usuário. Por outro lado, em alguns projetos, ainda se exige um Documento de Especificação de Requisitos, o qual inclui todos os requisitos do software que se pretende construir — incluindo requisitos funcionais e não-funcionais – especificados como Casos de Uso (Valente, 2020).

Casos de uso (use cases) são documentos textuais de especificação de requisitos. Casos de uso são escritos na perspectiva de um ator que deseja usar o sistema com um objetivo. Tipicamente, esse ator é um usuário humano (embora possa ser um outro sistema de software ou hardware). Ou seja, normalmente, o ator é uma entidade externa ao sistema (Valente, 2020).

Após sua especificação, os requisitos devem ser verificados e validados. O objetivo é garantir que eles estejam corretos, precisos, completos, consistentes e verificáveis.
Para realizar a verificação e validação dos requisitos, um técnica muito utilizada é a prototipagem, sejam eles throw-away ou incremental. Os protótipos incrementais são também denominados por MVPs (Minimal Viable Products). Um MVP consiste em implementar um sistema simples, com um conjunto de requisitos mínimos, mas que sejam suficientes para validar a viabilidade de continuar investindo no seu desenvolvimento (Valente, 2020).

Por fim, os requisitos devem ser priorizados. Às vezes, o termo requisitos é interpretado de forma literal, isto é, como uma lista de funcionalidades e restrições obrigatórias em sistemas de software. No entanto, nem sempre aquilo que é especificado pelos clientes será implementado nas releases iniciais. Por exemplo, restrições de prazo e custos podem postergar a implementação de certos requisitos (Valente, 2020).

Elicitar e compreender os requisitos dos stakeholders do sistema é um processo difícil por várias razões (Sommerville, 2003):
1. Exceto em termos gerais, os stakeholders costumam não saber o que querem de um sistema computacional; eles podem achar difícil articular o que querem que o sistema faça, e, como não sabem o que é viável e o que não é, podem fazer exigências inviáveis.
2. Naturalmente, os stakeholders expressam requisitos em seus próprios termos e com o conhecimento implícito de seu próprio trabalho. Engenheiros de requisitos, sem experiência no domínio do cliente, podem não entender esses requisitos.

3. Diferentes stakeholders têm requisitos diferentes e podem expressar essas diferenças de várias maneiras. Engenheiros de requisitos precisam descobrir todas as potenciais fontes de requisitos e descobrir as semelhanças e conflitos.
4. Fatores políticos podem influenciar os requisitos de um sistema. Os gerentes podem exigir requisitos específicos, porque estes lhes permitirão aumentar sua influência na organização.
5. O ambiente econômico e empresarial no qual a análise ocorre é dinâmico. É inevitável que ocorram mudanças durante o processo de análise. A importância dos requisitos específicos pode mudar. Novos requisitos podem surgir a partir de novos stakeholders que não foram inicialmente consultados.

Referências:

SOMMERVILLE,I. Engenharia de Software. Addison Wesley, 6ª ed. 2003. MALDONADO, José Carlos. Qualidade de software: teoria e prática. São Paulo:Prentice Hall, 2001. xvi, 303p. : ISBN 8587918540 (broch.) Capítulo 4.
VALENTE, M.T. Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade, Editora: Independente, 395 p., 2020. Capítulo 3.

Membros da Banca:

Claudiomir Selner — **Luciana Rita Guedes**

Avanilde Kemczinski

Assinaturas do documento

Código para verificação: **9DIO613R**

Este documento foi assinado digitalmente pelos seguintes signatários nas datas indicadas:

LUCIANA RITA GUEDES (CPF: 684.XXX.129-XX) em 10/07/2023 às 09:08:27
Emitido por: "SGP-e", emitido em 30/03/2018 - 12:48:12 e válido até 30/03/2118 - 12:48:12.
(Assinatura do sistema)

AVANILDE KEMCZINSKI (CPF: 751.XXX.569-XX) em 10/07/2023 às 09:17:38
Emitido por: "SGP-e", emitido em 30/03/2018 - 12:39:24 e válido até 30/03/2118 - 12:39:24.
(Assinatura do sistema)

CLAUDIOMIR SELNER em 10/07/2023 às 09:22:23
Emitido por: "SGP-e", emitido em 30/03/2018 - 12:40:51 e válido até 30/03/2118 - 12:40:51.
(Assinatura do sistema)

Para verificar a autenticidade desta cópia, acesse o link https://portal.sgpe.sea.sc.gov.br/portal-externo/conferencia-documento/VURFU0NfMTIwMjJfMDAwMjgzMTdfMjgzNDFfMjAyM185RElPNjEzUg== ou o site https://portal.sgpe.sea.sc.gov.br/portal-externo e informe o processo **UDESC 00028317/2023** e o código **9DIO613R** ou aponte a câmera para o QR Code presente nesta página para realizar a conferência.