Capítulo 1 do SWEBOK 4.0 — Requisitos de Software

1. O que são requisitos de software?

Requisitos de software são condições ou capacidades necessárias para um usuário resolver um problema ou atingir um objetivo; eles especificam o que um sistema deve possuir ou fazer para satisfazer contratos, padrões ou documentos formais. Requisito pode ser uma propriedade que o software deve exibir para resolver um problema do mundo real, envolvendo automatização de tarefas, suporte a políticas e processos de negócios, ou controle de dispositivos.

2. Quais são as principais categorias de requisitos de software?

As principais categorias são requisitos de produto (funcionais e não funcionais) e requisitos de projeto (que restringem aspectos do projeto como custo, cronograma e equipe). Requisitos funcionais descrevem comportamentos e políticas que o software deve cumprir, enquanto requisitos não funcionais impõem restrições tecnológicas ou de qualidade de serviço, como desempenho, confiabilidade, segurança e escalabilidade

3. Qual a diferença entre requisitos funcionais e não funcionais?

Requisitos funcionais referem-se às funcionalidades que o sistema deve executar, especificando processos e políticas observáveis. Já os requisitos não funcionais são restrições sobre como o sistema deve funcionar, incluindo restrições tecnológicas (plataformas, linguagens) e restrições de qualidade de serviço (tempo de resposta, confiabilidade).

4. O que são requisitos derivados?

São requisitos não diretamente impostos por um stakeholder externo, mas gerados internamente durante o desenvolvimento, muitas vezes como decisões de design que, para certas subequipes, funcionam como requisitos.

5. Que técnicas são comuns para elicitação de requisitos?

Algumas técnicas comuns incluem entrevistas, workshops facilitados (JAD, JRP), análise de protocolo, grupos focais, questionários, prototipagem exploratória e observação. A elicitação também pode usar fontes de requisitos como versões anteriores, sistemas existentes, benchmarking e análise de documentos .

6. Quais são as propriedades desejáveis de requisitos bem analisados?

Requisitos devem ser inequívocos (uma única interpretação), testáveis, quantificáveis, vinculativos (aceitos pelas partes interessadas), atômicos (indivisíveis), representativos da necessidade real, usar vocabulário das partes interessadas, aceitos por todos, completos, concisos, internamente e externamente consistentes, e economicamente viáveis.

7. Como a validação de requisitos é normalmente realizada?

Validação é obtida através de revisões (inspeções), simulação e execução de modelos formais, e prototipagem que ajuda a demonstrar a intenção dos requisitos para as partes interessadas, garantindo alinhamento com as necessidades reais.

8. Quais abordagens existem para especificação de requisitos?

Especificação pode ser em linguagem natural não estruturada, estruturada (ex: formato ator-ação), baseada em critérios de aceitação (ex: ATDD e BDD), ou baseada em modelos (ex: UML, SysML), com graus variados de formalidade para reduzir ambiguidades.

9. Por que a priorização de requisitos é importante?

A priorização ajuda a focar no desenvolvimento das funcionalidades de maior valor, auxilia em decisões de trade-off, gerenciamento de escopo e ajuda a orientar correções e manutenção focadas naquilo que tem maior impacto para os usuários e stakeholders.

10. Quais são algumas considerações práticas no gerenciamento de requisitos?

O processo de requisitos é iterativo, pois nem tudo pode ser definido completamente de primeira. Requisitos possuem volatilidade e estabilidade variáveis. O rastreamento dos requisitos é essencial para garantir consistência e suportar análise de impacto em mudanças. Controle de mudanças e correspondência do escopo garantem a viabilidade do projeto.

11. Quais tipos de ferramentas existem para suporte a requisitos?

Existem ferramentas de gerenciamento de requisitos (armazenamento, rastreamento, controle de mudanças), ferramentas de modelagem (criação, análise e publicação de modelos de requisitos) e ferramentas de geração de casos de teste funcionais (derivação automática de testes a partir das especificações).

Arquitetura e Design de Software

1. Qual a abordagem recomendada pelo SWEBOK para documentar a arquitetura de um sistema complexo?

A recomendação do SWEBOK é usar visões separadas, como visão física e visão de desenvolvimento, para atender às preocupações de diferentes stakeholders, ao invés de criar um documento técnico massivo ou focar apenas em padrões de design .

2. Qual a diferença fundamental entre arquitetura de software e design de software?

A arquitetura de software (Capítulo 2) trata das decisões estruturais de alto nível que afetam todo o sistema, enquanto o design de software (Capítulo 3) envolve detalhes de implementação e refinamentos que restringem a arquitetura. A arquitetura estabelece restrições que influenciam como o design detalhado será realizado

3. O que é um trade-off na arquitetura de software?

Trade-off é uma compensação entre diferentes atributos desejáveis do sistema, como segurança versus desempenho. Um exemplo comum é equilibrar o grau de criptografia (segurança) com a velocidade de processamento (desempenho).

4. O que caracteriza um requisito arquitetonicamente significativo (ASR)?

Um requisito que influencia fortemente as decisões estruturais do sistema, como capacidade de suportar número elevado de usuários concorrentes, é considerado um requisito arquitetonicamente significativo.

5. Qual padrão de design é adequado para criar um módulo que notifica outros módulos sobre mudanças de estado, mantendo baixo acoplamento?

O padrão Observer (Observador) é o mais adequado, pois permite que os módulos escutem mudanças sem precisar conhecer todos os dependentes diretamente.

6. Qual é a principal distinção entre Alto Nível (High-Level Design) e Detalhado (Detailed Design)?

O projeto de alto nível define a arquitetura geral e componentes principais do sistema, enquanto o projeto detalhado especifica componentes internos, algoritmos e detalhes de implementação de cada módulo.

7. Quais são os princípios de coesão e acoplamento?

Coesão refere-se à medida em que os elementos de um módulo trabalham juntos para realizar uma única função bem definida, enquanto acoplamento indica a dependência entre módulos. Em um bom design, deseja-se alta coesão e baixo acoplamento, para facilitar manutenção e reutilização.

8. Para que serve o princípio de encapsulamento?

Ele visa esconder os detalhes internos de um módulo, expondo uma interface pública bem definida, promovendo modularidade, facilidade de manutenção e isolamento de mudanças.

Requisitos de Software

1. Como se caracteriza um requisito funcional?

Um requisito funcional descreve uma funcionalidade ou comportamento que o sistema deve realizar, como gerar um extrato bancário detalhado ou permitir transferência de dinheiro.

2. O que é uma restrição de qualidade de serviço (QoS)?

É uma exigência relacionada à qualidade do serviço, como o tempo de resposta do sistema, por exemplo, fornecer o saldo da conta em menos de dois segundos, ou acessibilidade, como compatibilidade com deficiências.

3. Qual atividade do processo de Engenharia de Requisitos envolve coleta de informações?

A atividade principal é a Elicitação de Requisitos, onde a equipe coleta informações das partes interessadas e documentos existentes para definir o que o sistema deve fazer.

4. Qual a diferença entre requisitos funcionais e não funcionais com exemplos para comércio eletrônico?

Requisitos funcionais incluem funcionalidades como permitir login ou compra. Requisitos não funcionais abrangem restrições de desempenho, como tempo de resposta ou segurança. Por exemplo, "o sistema deve permitir transações seguras" (funcional) e "o sistema deve processar pedidos em até 2 segundos" (não funcional).

5. Dê um exemplo de requisito não funcional.

Um exemplo seria: "O sistema deve estar disponível 99,9% do tempo", que é uma restrição de qualidade do serviço.

6. Qual técnica de elicitação de requisitos NÃO é adequada?

Testes de unidade não são técnicas de elicitação, pois são aplicadas na fase de validação do produto, e não na coleta inicial de requisitos.

7. Por que a validação de requisitos é importante?

Ela garante que os requisitos definidos realmente atendem às necessidades dos stakeholders, evitando retrabalho e custos adicionais. Técnicas comuns incluem revisões, prototipagem, análise de modelos e validação com usuários