EDUCAÇÃO, POR SUA ESCOLHA

SISTEMAS DA INFORMAÇÃO

MATERIAL INSTRUCIONAL ESPECÍFICO

TOMO 10

CQA/UNIP – Comissão de Qualificação e Avaliação da UNIP

SISTEMAS DA INFORMAÇÃO

MATERIAL INSTRUCIONAL ESPECÍFICO

TOMO 10

Christiane Mazur Doi

Doutora em Engenharia Metalúrgica e de Materiais, Mestra em Ciências - Tecnologia Nuclear, Especialista em Cálculo e Matemática Aplicada, Especialista Estatística Aplicada e Ciência de Dados, Especialista em Língua Portuguesa e Literatura, Especialista em Recursos Gramaticais para Revisão Textual, Engenheira Química e Licenciada em Matemática, com Aperfeiçoamento em Estatística e MBA em Engenharia Econômica. Professora titular da Universidade Paulista.

José Carlos Morilla

Doutor e Mestre em Engenharia de Materiais, Mestre em Engenharia de Produção, Especialista em Engenharia Metalúrgica, Especialista em Física e Engenheiro Mecânico, com MBA em Gestão de Empresas. Professor adjunto da Universidade Paulista.

Larissa Rodrigues Damiani

Doutora em Ciências pelo programa de Engenharia Elétrica - Microeletrônica, Mestra pelo mesmo programa e Engenheira Elétrica - modalidade Eletrônica (ênfase em Telemática). Professora titular da Universidade Paulista.

Tarcísio de Souza Peres

Mestre em Ciências, Especialista em Gestão de Projetos e Engenheiro da Computação, com MBA em Gestão de TI. Cursou Gestão Estratégica e Competitividade. Professor adjunto da Universidade Paulista.

Tiago Guglielmeti Correale

Doutor em Engenharia Elétrica, Mestre em Engenharia Elétrica e Engenheiro Elétrico (ênfase em Telecomunicações). Professor titular da Universidade Paulista.

Material instrucional específico, cujo conteúdo integral ou parcial não pode ser reproduzido ou utilizado sem autorização expressa, por escrito, da CQA/UNIP – Comissão de Qualificação e Avaliação da UNIP – UNIVERSIDADE PAULISTA.

Questão 1

Questão 1.[1]

Leia o texto a seguir.

A etapa de definição de requisitos é voltada para estabelecer quais as funções são requeridas pelo sistema e as restrições sobre a operação e o desenvolvimento do software. Os requisitos de software podem ser classificados como requisitos funcionais e não funcionais.

SOMMERVILLE, I. *Engenharia de Software*, 10. ed. São Paulo: Pearson Education, 2019 (com adaptações).

Considerando as informações do texto, assinale a alternativa em que o item é um requisito funcional.

A. O software deve ser operacionalizado no sistema Linux.
B. O tempo de desenvolvimento não deve ultrapassar seis meses.
C. O software deve emitir relatórios de compras a cada quinze dias.
D. O tempo de resposta do sistema não deve ultrapassar 30 segundos.
E. A base de dados deve ser protegida para acesso apenas de usuários autorizados.

1. Introdução teórica

A questão testa conhecimentos básicos a respeito de requisitos de sistemas de software. Vamos, a seguir, relembrar os principais conceitos envolvidos nesse tema.

Requisitos funcionais e requisitos não funcionais

Os requisitos estabelecem quais funcionalidades um software deve apresentar, assim como as restrições às quais ele será submetido. Os requisitos de sistema são comumente classificados em funcionais e não funcionais.

Os **requisitos funcionais** são as declarações dos serviços que o sistema deve prestar, de como o software deve reagir a determinadas entradas e de como deve se comportar em determinadas situações. De forma geral, eles dizem o que o software "deve fazer". Em alguns casos, os requisitos funcionais podem, também, estabelecer tarefas ou comportamentos que o sistema deve evitar.

Como exemplo, podemos imaginar um sistema de cadastro de clientes encomendado por uma empresa a um desenvolvedor. Os itens listados a seguir correspondem a alguns dos requisitos funcionais desse sistema, expressos em linguagem natural.

[1]Questão 9 – Enade 2021.

- Cada funcionário que utiliza o sistema deve ser identificado exclusivamente por seu número funcional, que corresponde a um código numérico de 6 dígitos.
- Um funcionário deve poder fazer a busca de clientes por nome e sobrenome.
- O sistema deve gerar, semanalmente, um relatório contendo a lista e a contagem do número de clientes cadastrados no período.

A imprecisão na especificação de requisitos pode levar a conflitos entre clientes e desenvolvedores. Por isso, requisitos devem ser bem especificados, sem ambiguidades.

Os **requisitos não funcionais**, conforme o nome sugere, são aqueles que não têm relação direta com as funcionalidades oferecidas pelo software. Esses requisitos especificam ou restringem as características do sistema como um todo. Eles podem ser relacionados a propriedades do sistema, como confiabilidade, tempo de resposta e uso de memória. Muitos requisitos não funcionais também surgem por necessidades dos usuários relacionadas a restrições orçamentárias, políticas organizacionais, necessidade de interoperabilidade com outros sistemas, normas de segurança ou legislação relativa à privacidade.

Como exemplo, vamos voltar a imaginar o sistema de cadastro de clientes. Os itens listados abaixo correspondem a alguns dos requisitos não funcionais desse sistema.

- O sistema deve ficar disponível para os funcionários de todas as filiais da empresa durante o expediente (de segunda a sexta, das 8h às 17h).
- O tempo que o sistema pode permanecer fora do ar no expediente não deve ultrapassar 5 segundos em qualquer dia.
- O sistema deve implementar providências para garantir a privacidade dos clientes cadastrados.
- O programa deve ser executado em determinado sistema operacional.

Sempre que possível, os requisitos não funcionais devem ser escritos adotando critérios quantitativos, de forma que possam ser testados objetivamente. Isso ajuda a evitar divergências interpretativas entre clientes e equipes de desenvolvimento. Como exemplo, considere o requisito a seguir.

- **Os funcionários devem ser capazes de utilizar todas as funções do sistema com pouco tempo de treinamento.**

Perceba que o significado de "pouco tempo" é subjetivo. Para o cliente, pode significar algumas horas de treinamento. Para os desenvolvedores, pode significar algumas semanas.

Como fazemos, então, para contornar essa inconsistência? Devemos tornar o requisito objetivo, conforme exposto a seguir, com um critério quantitativo estipulado.

- **Os funcionários devem ser capazes de utilizar todas as funções do sistema após três horas de treinamento.**

2. Análise das alternativas

A – Alternativa incorreta.

JUSTIFICATIVA. O sistema operacional no qual o software deverá ser operacionalizado corresponde a um requisito não funcional, pois não corresponde diretamente a uma funcionalidade esperada para o software.

B – Alternativa incorreta.

JUSTIFICATIVA. O tempo disponibilizado para o desenvolvimento do software é um requisito não funcional, pois não corresponde diretamente a uma funcionalidade esperada.

C – Alternativa correta.

JUSTIFICATIVA. A emissão de relatórios periódicos é uma possível funcionalidade de um software. Logo, esse requisito é classificado como funcional.

D – Alternativa incorreta.

JUSTIFICATIVA. O tempo de resposta do sistema é um requisito não funcional, pois não corresponde diretamente a uma funcionalidade.

E – Alternativa incorreta.

JUSTIFICATIVA. O nível de proteção da base de dados é um requisito não funcional, pois não corresponde diretamente a uma funcionalidade.

3. Indicações bibliográficas

- HIRAMA, K. *Engenharia de software: qualidade e produtividade com tecnologia*. Rio de Janeiro: Elsevier, 2011.

- PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software: uma abordagem profissional*. Porto Alegre: AMGH, 2021.
- SOMMERVILLE, I. *Engenharia de software*. São Paulo: Pearson Education do Brasil, 2018.

Questão 2

Questão 2.[2]

Leia o texto a seguir.

A engenharia de requisitos é uma área que inclui quatro subprocessos relacionados de alto nível. Esses subprocessos são:

1) *avaliação se o sistema será útil para a empresa (estudo de viabilidade);*
2) *obtenção de requisitos (elicitação de requisitos);*
3) *conversão desses requisitos em alguma forma padrão (especificação);*
4) *verificação se os requisitos realmente definem o sistema que o cliente deseja (validação).*

SOMMERVILLE, I. *Engenharia de Software*. São Paulo: Pearson Addison-Wesley, 2017 (com adaptações).

Uma equipe de Tecnologia da Informação (TI) de uma empresa de consultoria desenvolverá um software de Suporte Técnico para uma grande empresa fornecedora de equipamentos eletrônicos. O estudo de viabilidade do software já foi realizado e aprovado. A equipe de Tecnologia da Informação seguirá os três subprocessos seguintes de alto nível de engenharia de requisitos descritos no texto de Sommerville, ou seja, os subprocessos de elicitação de requisitos, especificação e validação. Para esses três subprocessos, quais são os artefatos que podem ser utilizados por essa equipe de Tecnologia da Informação?

A. Documento de entrevista com usuários; modelo de caso de uso para os requisitos funcionais; prototipação de telas.
B. Documento de estudo de viabilidade; modelo de caso de uso para os requisitos funcionais; prototipação de telas.
C. Matriz de rastreabilidade; modelo de caso de uso para os requisitos não-funcionais; prototipação de telas.
D. Documento de entrevista com usuários; modelo de caso de uso para os requisitos não-funcionais; matriz de rastreabilidade.
E. Documento de estudo de viabilidade; modelo de caso de uso para os requisitos funcionais; matriz de rastreabilidade.

1. Introdução teórica

A questão avalia conhecimentos a respeito das atividades envolvidas na engenharia de requisitos, abordando técnicas e notações utilizadas em cada uma dessas atividades. Vamos, a seguir, estudar o tema.

[2]Questão 18 – Enade 2021.

1.1. Atividades da engenharia de requisitos

Os requisitos de software são o conjunto de funcionalidades, características e restrições que um sistema deve atender para satisfazer às necessidades dos usuários e às metas do projeto.

A engenharia de requisitos pode ser apresentada como uma área composta por subprocessos relacionados, destinados a:

- compreender a necessidade do sistema;
- definir seus requisitos;
- verificar se eles expressam adequadamente aquilo que o cliente deseja.

Entre esses subprocessos, podem ser destacados:

- o estudo de viabilidade;
- a elicitação;
- a especificação;
- a validação de requisitos.

O estudo de viabilidade tem como objetivo avaliar se o sistema proposto é útil para a organização e se há justificativa técnica, econômica e operacional para prosseguir com o projeto. Por isso, a análise deve se concentrar nos três subprocessos seguintes mencionados no enunciado:

- elicitação;
- especificação;
- validação.

A elicitação corresponde à obtenção e à compreensão dos requisitos por meio da interação com os stakeholders. A especificação consiste na conversão desses requisitos em uma forma documentada, organizada e mais precisa. A validação, por sua vez, verifica se os requisitos definidos realmente descrevem o sistema que o cliente deseja.

1.2. Elicitação de requisitos

A **elicitação de requisitos** é a primeira atividade fundamental da engenharia de requisitos e envolve um contato direto com os stakeholders do sistema. Os **stakeholders** são as partes interessadas no sistema, como usuários finais, gerentes de produto, equipe de desenvolvimento, equipe de marketing, acionistas e autoridades reguladoras que certificam a aceitabilidade do sistema.

Os objetivos da elicitação de requisitos são compreender o trabalho que os stakeholders realizam e entender como eles usariam o novo sistema. As tarefas envolvidas nesse processo são elencadas a seguir.

- **Descoberta e compreensão dos requisitos**: é o processo de interagir com os stakeholders para "descobrir" os seus requisitos de usuário.
- **Classificação e organização dos requisitos**: é o processo de pegar um conjunto não estruturado de requisitos, agrupar os requisitos relacionados e organizá-los em grupos coerentes.
- **Priorização e negociação de requisitos**: quando há o envolvimento de muitos stakeholders, os requisitos provavelmente entrarão em conflito. Este estágio envolve a resolução desses possíveis conflitos.
- **Documentação dos requisitos**: um rascunho da documentação dos requisitos de usuário é feito neste estágio.

Na figura 1, vemos um diagrama que ilustra a atividade de elicitação de requisitos. Note que a figura sugere tarefas iterativas, com feedback contínuo de cada tarefa para as demais. O ciclo do processo começa com a descoberta dos requisitos e termina com a sua documentação, mas a compreensão que o analista tem dos requisitos aumenta a cada rodada do ciclo, que só deve terminar quando o documento de requisitos de usuário for produzido de forma definitiva. Os requisitos de usuário quase sempre são escritos em linguagem natural, complementados por diagramas e tabelas apropriados.

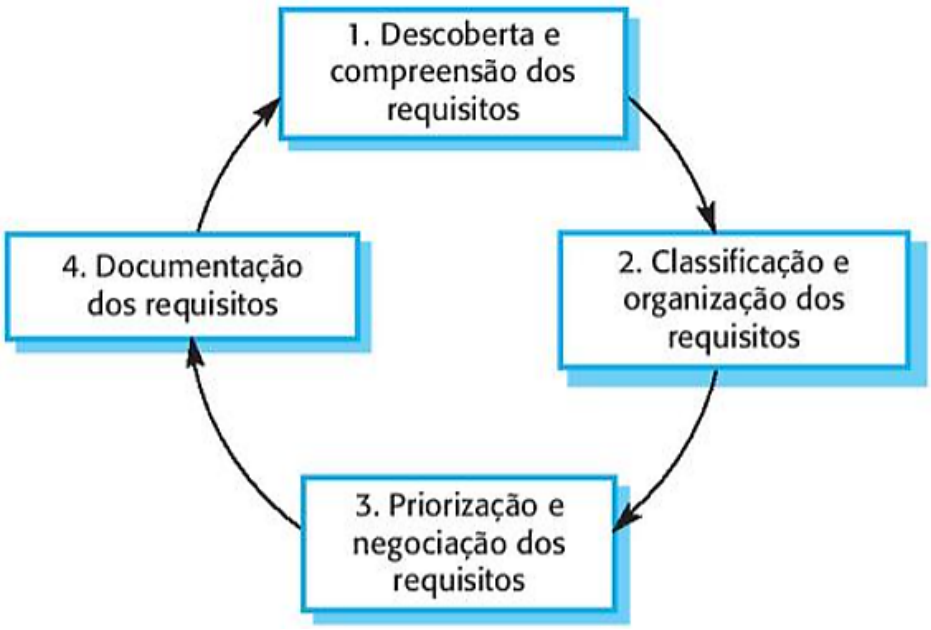

Figura 1. O processo de elicitação de requisitos.
SOMMERVILLE, I. *Engenharia de software*. São Paulo: Pearson Education do Brasil, 2018.

As **técnicas de elicitação dos requisitos** envolvem encontros com os stakeholders para descobrir informações sobre o sistema proposto. É necessário investir tempo para entender como as pessoas trabalham e o que elas produzem. Existem duas abordagens fundamentais para a elicitação de requisitos, elencadas a seguir.

- **Entrevistas**: nesta abordagem, há uma conversa com as pessoas a respeito do que elas fazem.
- **Observação ou etnografia**: nesta abordagem, observam-se as pessoas executando seus trabalhos para ver quais ferramentas elas usam, como usam etc.

1.3. Especificação de requisitos

A **especificação de requisitos** é a segunda atividade fundamental da engenharia de requisitos. A atividade é destinada a escrever os requisitos de usuário do sistema em um documento de requisitos de sistema. Esses requisitos de sistema devem ser claros, completos e consistentes.

Os requisitos de usuário quase sempre são escritos em linguagem natural. Os requisitos de sistema são versões ampliadas dos requisitos de usuário, que os engenheiros de software utilizam como ponto de partida para o projeto do sistema de software.

Os requisitos de sistema também podem ser escritos em linguagem natural, mas outras notações baseadas em formulários, em gráficos ou em modelos matemáticos podem ser aplicadas. Essas notações garantem maior precisão e maior nível de detalhamento aos requisitos, quando comparados aos escritos em linguagem natural.

No quadro 1, são mostradas as possíveis notações que podemos utilizar para escrever requisitos de sistema.

Quadro 1. Notações pelas quais podemos escrever requisitos de sistema.

Notação	Descrição
Sentenças em linguagem natural	Por meio dessa notação, os requisitos são escritos usando frases numeradas em linguagem natural. Cada frase deve expressar um requisito.
Linguagem natural estruturada	Os requisitos são escritos em linguagem natural em um formulário ou template. Cada campo fornece informações sobre um aspecto do requisito.
Notações gráficas	Modelos gráficos, suplementados por anotações em texto, são utilizados para definir os requisitos funcionais do sistema. São usados, com frequência, os diagramas de casos de uso e de sequência da UML (Unified Modeling Language).

Especificações matemáticas	Essas notações baseiam-se em conceitos matemáticos, como as máquinas de estados finitos ou conjuntos.

SOMMERVILLE, 2018 (com adaptações).

1.4. Validação de requisitos

A **validação de requisitos** é a terceira atividade fundamental da engenharia de requisitos. A atividade é destinada a conferir se os requisitos, tanto de usuário quanto de sistema, definem o sistema que o cliente realmente quer. Essa atividade sobrepõe-se às etapas de elicitação e de análise, já que é voltada a encontrar problemas em cada uma delas. A validação de requisitos é muito importante, pois erros em um documento de requisitos podem levar a grandes custos de retrabalho, quando esses problemas são descobertos durante o desenvolvimento ou após o sistema entrar em funcionamento.

Algumas técnicas de validação de requisitos podem ser utilizadas individualmente ou em conjunto. Essas técnicas são resumidas no quadro 2.

Quadro 2. Técnicas de validação de requisitos.

Técnica	Descrição
Revisões de requisitos	Os requisitos são sistematicamente analisados por uma equipe de revisores, que procuram por erros e por inconsistências.
Prototipação	Essa técnica envolve o desenvolvimento de um modelo executável do sistema e o uso desse modelo com os usuários finais e clientes para verificar se ele satisfaz suas necessidades e expectativas. Os stakeholders experimentam o sistema e opinam sobre mudanças nos requisitos.
Geração de casos de teste	Os requisitos devem ser testáveis. Se os testes dos requisitos forem concebidos como parte do processo de validação, eles frequentemente revelarão os problemas existentes nos requisitos.

SOMMERVILLE, 2018 (com adaptações).

2. Análise das alternativas

A – Alternativa correta.

JUSTIFICATIVA. A equipe de Tecnologia da Informação seguirá as três atividades de engenharia de requisitos: elicitação, especificação e validação. O documento de entrevista com usuários é um artefato da etapa de elicitação. O modelo de caso de uso para os requisitos

funcionais é uma notação da etapa de especificação. Já a prototipação de telas é uma técnica da etapa de validação. Logo, esses artefatos podem ser utilizados por essa equipe.

B – Alternativa incorreta.

JUSTIFICATIVA. O documento de estudo de viabilidade não corresponde aos três subprocessos solicitados no enunciado: elicitação, especificação e validação. Veja que o próprio enunciado informa que o estudo de viabilidade já foi realizado e aprovado.

C e D – Alternativas incorretas.

JUSTIFICATIVA. As mudanças no ambiente de negócios, nas organizações e nas tecnologias nos levam a mudanças nos requisitos de um sistema de software. O gerenciamento de requisitos é o processo de gerenciar e controlar essas mudanças. Nesse contexto, podem ser utilizadas as matrizes de rastreabilidade, que associam os requisitos às suas origens e os rastreia durante todo o ciclo de vida do projeto. Logo, as matrizes de rastreabilidade não são um artefato específico das etapas de elicitação, de especificação e de validação de requisitos. Além disso, diagramas de casos de uso, de forma geral, descrevem as funcionalidades propostas para o sistema. Logo, seu foco são os requisitos funcionais.

E – Alternativa incorreta.

JUSTIFICATIVA. O documento de estudo de viabilidade não corresponde aos três subprocessos solicitados no enunciado: elicitação, especificação e validação. Veja que o próprio enunciado informa que o estudo de viabilidade já foi realizado e aprovado. Além disso, as matrizes de rastreabilidade são artefatos associados ao gerenciamento e ao controle de requisitos, e não correspondem a um artefato característico desses três subprocessos.

3. Indicações bibliográficas

- HIRAMA, K. *Engenharia de software: qualidade e produtividade com tecnologia*. Rio de Janeiro: Elsevier, 2011.
- PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de Software: uma abordagem profissional*. Porto Alegre: AMGH, 2021.
- SOMMERVILLE, I. *Engenharia de software*. São Paulo: Pearson Education do Brasil, 2018.

Questão 3

Questão 3.[3]

Leia o texto a seguir.

Arquitetura de software é uma representação que permite analisar a efetividade do projeto no entendimento dos requisitos declarados. Durante a fase de concepção da arquitetura, podem-se considerar alternativas de arquitetura em um estágio em que mudanças ainda são realizadas com menor esforço, diminuindo riscos associados à construção do software ainda na fase inicial.

PRESSMAN, R. S. *Engenharia de Software*: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016 (com adaptações).

A respeito dos estilos e padrões arquiteturais contidos na engenharia de software, avalie as afirmativas.

I. Em arquiteturas orientadas a objetos, a comunicação entre os componentes do software é realizada por intermédio da troca de mensagens.
II. As arquiteturas monolíticas consistem de um sistema dividido em pequenas partes, possibilitando que estas tenham sua manutenção, execução e evolução individual.
III. No padrão arquitetural Modelo-Visão-Controle (MVC), a camada de Modelo representa os dados e a lógica de domínio da aplicação, podendo ser consultada pela Visão e atualizada a partir de ações processadas pelo Controle.
IV. Nas arquiteturas microsserviços, o software possui componentes altamente acoplados, dificultando a manutenção.

É correto apenas o que se afirma em

A. I e III.
B. II e III.
C. II e IV.
D. I, II e IV.
E. I, III e IV.

1. Introdução teórica

A questão testa conhecimentos a respeito de alguns modelos arquiteturais de software, estudados em disciplinas correlatas à área de engenharia de software. Vamos, a seguir, relembrar o tema.

[3]Questão 19 – Enade 2021 (com adaptações).

1.1. Arquitetura de software

Do mesmo modo como a planta baixa de uma casa é apenas uma representação do edifício físico, a representação da arquitetura de software não é o software em si. Ela é apenas uma representação, que permite:

- analisar o atendimento dos requisitos;
- considerar alternativas de arquitetura em um estágio em que fazer mudanças de projeto ainda é fácil;
- reduzir os riscos associados ao desenvolvimento do software.

A arquitetura de software oferece uma ampla visão do sistema que será construído. Ela representa a estrutura e a organização dos componentes de software, suas propriedades e as conexões entre eles. Os componentes de software incluem módulos de programas e as várias representações de dados manipuladas por eles.

De forma geral, um estilo de arquitetura fornece diretrizes gerais para o design do sistema, e um padrão de arquitetura oferece uma solução mais específica para um problema conhecido, de forma que um padrão pertença a um estilo. A seguir, vamos estudar alguns estilos de arquiteturas.

1.2. Arquiteturas monolíticas

Nas arquiteturas monolíticas, o sistema conta com baixa modularização, sendo enxergado e implementado como um único projeto. Todos os componentes de software em um sistema monolítico são interdependentes, devido aos mecanismos de troca de dados dentro do sistema.

Como exemplo, vamos imaginar um sistema para uma loja de calçados. Todos os usuários utilizarão o mesmo sistema, e será preciso que ele esteja dividido em: autenticação e perfis de usuários, compra de produtos, estoque e vendas.

No padrão monolítico, todos os módulos mencionados acima estarão em um único projeto. Se é feita uma alteração no serviço de vendas, todos os outros módulos do projeto também terão que subir novamente para o servidor junto com essa alteração.

Em tempo de execução, todos os módulos de um sistema são executados, pelo sistema operacional, como um processo único. A figura 1 ilustra um sistema de nove módulos (de M1 a M9) que segue o estilo arquitetural monolítico. Em tempo de execução, o sistema executa como um único processo, que é representado pelo quadrado que envolve os nove módulos.

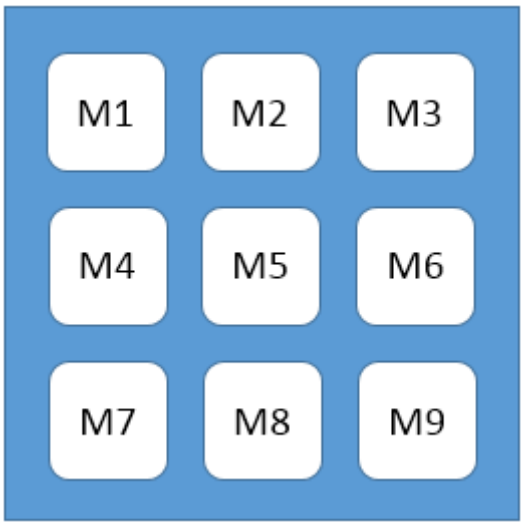

Figura 1. Sistema de nove módulos que segue o estilo arquitetural monolítico.
VALENTE, [s.d.] (com adaptações).

Em geral, um sistema monolítico é mais simples de ser desenvolvido, pois a organização fica concentrada em um único sistema. No entanto, é trabalhoso modificá-lo, pois pequenas mudanças afetam grandes áreas da base de código. Além disso, não há flexibilidade em relação à linguagem de programação: aquela que for escolhida no início do projeto terá que ser seguida para todos os módulos a serem implementados.

1.3. Arquiteturas em microsserviços

Para contornar as desvantagens dos sistemas em arquiteturas monolíticas, algumas empresas passaram a migrar os seus "monólitos" para arquiteturas baseadas em microsserviços. Nessas arquiteturas, fazemos com que certos módulos do sistema sejam executados em processos independentes, sem compartilhamento de memória entre eles. Nesse caso, o sistema é altamente modularizado não apenas em tempo de desenvolvimento, mas também em tempo de execução. Com isso, é menos provável que mudanças em um módulo causem problemas em outro.

O nome desse estilo de arquitetura refere-se a cada módulo como um serviço. Além disso, os serviços são "micro" porque cada um deles implementa apenas uma funcionalidade simples.

Na figura 2, podemos ver o diagrama que ilustra um sistema de nove módulos implementados por meio de uma arquitetura em microsserviços. Esses módulos são executados por seis processos independentes, representados pelos retângulos externos aos módulos. Os módulos M1, M2, M3 e M6 são executados individualmente, ou seja, cada um

em um processo independente. Os módulos M4 e M5 são executados em dupla. Por fim, os módulos M7, M8 e M9 são executados, em conjunto, em um processo.

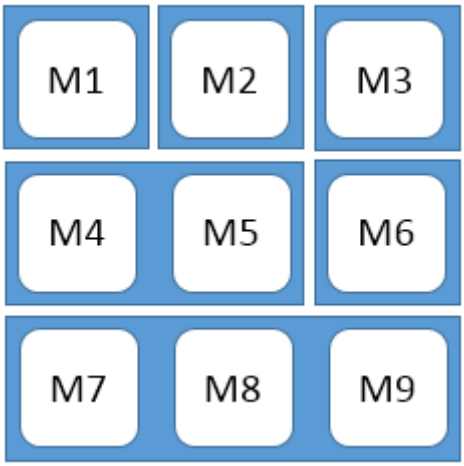

Figura 2. Sistema de nove módulos que segue o estilo arquitetural em microsserviços.
VALENTE, [s.d.] (com adaptações).

1.4. Arquiteturas orientadas a objetos

Nas arquiteturas orientadas a objetos, os componentes de um sistema encapsulam dados e as operações que devem ser aplicadas para manipular esses dados. A comunicação entre componentes é realizada por meio da passagem de mensagens.

Arquiteturas orientadas a objetos utilizam os princípios e os conceitos da programação orientada a objetos. Cada objeto tem uma interface definida pelos seus métodos, e a comunicação ocorre por meio da chamada desses métodos.

1.5. Padrão Modelo-Visão-Controle

O padrão arquitetural Modelo-Visão-Controle (MVC, do termo em inglês Model-View-Controller) pode ser classificado como um padrão de arquitetura orientado a objetos, que define que as classes de um sistema devem ser organizadas em três grupos, elencados a seguir.

- **Visão**: grupo de classes responsáveis pela apresentação da interface gráfica do sistema.
- **Controle**: conjunto de classes que tratam e interpretam eventos gerados por dispositivos de entrada, como o teclado ou o mouse. Como resultado da interpretação desses eventos, as classes de controle podem solicitar uma alteração no estado do modelo ou da visão.

- **Modelo**: grupo de classes que armazenam os dados manipulados pela aplicação e que têm a ver com o domínio do sistema, ou seja, com a lógica do negócio.

O relacionamento entre esses grupos de classes pode ser exemplificado conforme exposto na figura 3, que ilustra um sistema construído no modelo MVC.

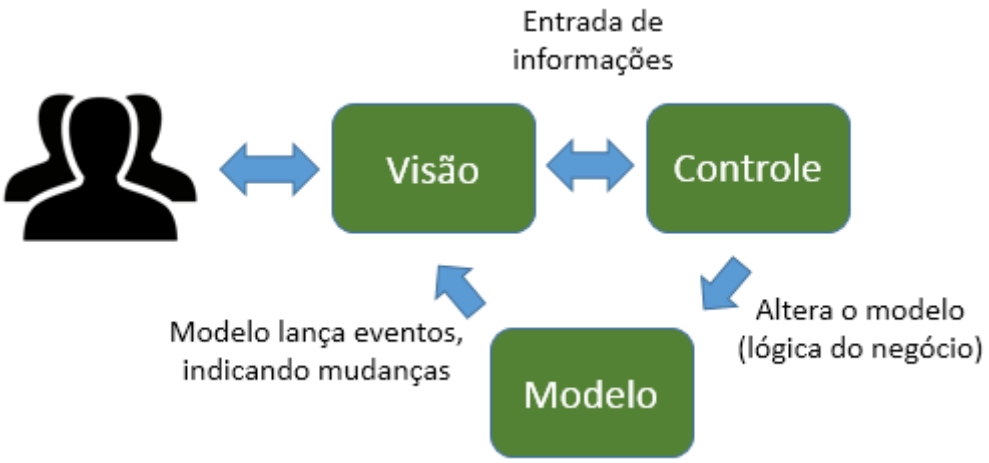

Figura 3. Fluxo de eventos e de informações em um sistema construído em modelo MVC. UNIVERSIDADE FEDERAL DE CAMPINA GRANDE, [s.d.]. Disponível em <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/arqu/mvc/mvc.htm >. Acesso em 12 dez. 2025 (com adaptações).

2. Análise das afirmativas

I – Afirmativa correta.

JUSTIFICATIVA. Nas arquiteturas orientadas a objetos, os objetos interagem entre si trocando mensagens. Cada objeto tem uma interface definida pelos seus métodos, e a comunicação ocorre por meio da chamada desses métodos.

II – Afirmativa incorreta.

JUSTIFICATIVA. Em arquiteturas monolíticas, o sistema é geralmente construído como uma única unidade. Se uma parte do sistema precisar de atualização, todo o sistema deve ser implantado novamente. Isso pode dificultar a manutenção de partes específicas do software.

III – Afirmativa correta.

JUSTIFICATIVA. No padrão MVC, o Modelo representa os dados e a lógica de negócios da aplicação; o Controle processa as interações do usuário e pode solicitar atualizações no Modelo; e a Visão apresenta ao usuário os dados obtidos a partir do Modelo.

IV – Afirmativa incorreta.

JUSTIFICATIVA. Nas arquiteturas de microsserviços, os módulos são projetados para serem independentes. Cada microsserviço é uma unidade que pode ser desenvolvida, implantada e escalada separadamente. Isso facilita a manutenção, pois as alterações em um microsserviço não devem afetar os outros.

Alternativa correta: A.

3. Indicações bibliográficas

- HIRAMA, K. *Engenharia de software: qualidade e produtividade com tecnologia*. Rio de Janeiro: Elsevier, 2011.
- MORAIS, I. S. de (org.). *Engenharia de software*. São Paulo: Pearson Education do Brasil, 2017.
- PENNA, W. *Arquitetura monolítica e microsserviços*. Disponível em <https://www.zappts.com.br/arquitetura-monolitica-e-microsservicos/>. Acesso em 12 dez. 2025.
- PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software: uma abordagem profissional*. Porto Alegre: AMGH, 2021.
- SOMMERVILLE, I. *Engenharia de software*. São Paulo: Pearson Education do Brasil, 2018.
- UNIVERSIDADE FEDERAL DE CAMPINA GRANDE. Disponível em <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/arqu/mvc/mvc.htm>. Acesso em 12 dez. 2025.
- VALENTE, M. T. *Engenharia de software moderna*. Disponível em <https://engsoftmoderna.info/>. Acesso em 12 dez. 2025.

Questão 4

Questão 4.[4]

Leia o texto a seguir.

No decorrer de um projeto de desenvolvimento de software, é compreensível que mudanças venham a ocorrer, sejam por novos entendimentos dos atores envolvidos ou até mesmo de novas demandas apresentadas pelos clientes. Nessa perspectiva, a Gestão de Configuração de Software estabelece um conjunto de atividades para gerenciar alterações através de todo o ciclo de vida de um software.

PRESSMAN, R. S. *Engenharia de Software*: uma abordagem profissional. 8. ed. Porto Alegre: Artmed Editora S.A. e McGraw-Hill Education, 2016 (com adaptações).

Com base no texto, assinale a opção correta acerca das atividades de gestão de configuração de software.

A. Identificação de itens na configuração de software, gerenciamento de alterações, validação de versões e controle de qualidade.
B. Identificação de objetos na configuração de software, controle de versão, controle de alterações, auditoria de configuração e relatório de status.
C. Gerência de projeto, gerência de configuração de software, ciclo de vida do projeto, processo de software e evolução das configurações.
D. Construção de modelo de requisitos, métricas para modelo de projeto, métricas de controle de versão e relatório de manutenção.
E. Identificação de alterações, controle de versão, mapeamento de alteração e notificação da alteração aos envolvidos.

1. Introdução teórica

A questão testa conhecimentos a respeito das atividades envolvidas na gestão de configuração de software. É importante frisar que essas atividades podem ser especificadas de diferentes formas por diferentes autores da área de engenharia de software. A nomenclatura presente nas alternativas da questão corresponde à utilizada por Roger Pressman e Bruce Maxim, no livro intitulado "Engenharia de Software: uma abordagem profissional". Vamos, a seguir, estudar um pouco desse conteúdo.

[4]Questão 21 – Enade 2021.

Gestão de configuração de software

Na construção de um sistema computacional, alterações são inevitáveis. Essas mudanças podem causar confusão quando não são analisadas antes de serem implementadas, ou quando não há boa comunicação entre os membros da equipe responsável pelo projeto.

À medida que o trabalho de engenharia de software avança, a equipe de software cria uma hierarquia de **itens de configuração de software** (SCIs, do termo em inglês software configuration items). Um SCI é uma entidade única identificável, que é gerenciada e documentada durante o ciclo de vida de um projeto. Essas entidades podem incluir códigos-fonte, documentos, scripts e outros elementos relacionados ao sistema. Os SCIs são chamados coletivamente de configuração de software.

Nesse cenário, a **gestão de configuração de software** (SCM, do termo em inglês software configuration management) é responsável por identificar, organizar e controlar modificações ao longo do ciclo de vida do software. O objetivo da SCM é maximizar a produtividade e minimizar possíveis erros.

O processo de gestão de alterações de software define uma série de tarefas que têm quatro objetivos principais:

- identificar todos os itens que definem a configuração do software;
- gerenciar alterações de um ou mais desses itens;
- facilitar a construção de diferentes versões de uma aplicação; e
- assegurar que a qualidade do software seja mantida conforme a configuração evolui.

Com o intuito de atingir esses objetivos, são definidas **cinco tarefas SCM**:

- identificação;
- controle de versão;
- controle de alteração;
- auditoria de configuração;
- elaboração de relatórios.

Essas tarefas são mostradas no diagrama da figura 1. Conforme a disposição da figura, as tarefas SCM podem ser vistas como camadas. Os SCIs "fluem" para fora através dessas camadas por toda a sua vida útil, tornando-se, finalmente, parte da configuração do software de uma ou de várias versões do sistema de software.

Figura 1. Camadas do processo SCM.
PRESSMAN e MAXIM, 2021 (com adaptações).

A atividade de **identificação** envolve a identificação clara dos SCIs que estão sujeitos à configuração. A gestão de configuração requer a capacidade de identificar e de rastrear esses elementos específicos dentro do sistema.

A atividade de **controle de alterações** pode combinar procedimentos humanos e ferramentas automatizadas, proporcionando um mecanismo para o controle de alterações. Resumidamente, nessa atividade, uma solicitação de alteração é apresentada e avaliada. Os resultados da avaliação são apresentados como um relatório de alterações e, por fim, uma ordem de alteração é gerada para cada alteração aprovada.

Na atividade de **controle de versão**, é criada uma atualização do arquivo original logo que a alteração for feita. Alternativamente, os objetos a serem alterados podem ser "retirados" do banco de dados do projeto e, assim que a alteração é feita, os objetos são então "colocados" novamente no banco de dados, e são usados mecanismos de controle de versão apropriados para gerar a próxima versão do software. O controle de versão permite o acompanhamento das alterações feitas, facilitando o trabalho colaborativo e a gestão de diferentes versões do software.

A **auditoria de configuração** complementa a revisão técnica, avaliando o objeto de configuração quanto a características que, em geral, não são consideradas durante a revisão. A revisão técnica focaliza a exatidão técnica do objeto de configuração modificado. Os revisores avaliam o SCI para determinar a consistência com outros SCIs, omissões ou efeitos colaterais em potencial. A auditoria entra como complemento, verificando se a configuração

do software está em conformidade com as políticas, com os padrões e com os requisitos estabelecidos.

É produzido, também, o **relatório de status de configuração**, que reporta o que aconteceu no processo de modificações, quem fez as alterações, quando elas foram feitas e o que será afetado por elas. O relatório não necessariamente é gerado ao final do processo. Ele pode ser gerado em diferentes fases do desenvolvimento, para informar sobre o estado da configuração.

2. Análise das alternativas

A – Alternativa incorreta.

JUSTIFICATIVA. A identificação de itens na configuração de software é uma das atividades da SCM, assim como o gerenciamento de alterações. No entanto, o controle de qualidade não é uma atividade típica da SCM, pois esse é um conceito muito amplo e abrangente.

B – Alternativa correta.

JUSTIFICATIVA. As tarefas de identificação de objetos na configuração de software, controle de versão, controle de alterações, auditoria de configuração e relatório de status compõem o conjunto das cinco tarefas SCM.

C – Alternativa incorreta.

JUSTIFICATIVA. Nesta alternativa, temos conceitos muito amplos, como gerência de projeto e ciclo de vida, que não são atividades exclusivas da SCM.

D – Alternativa incorreta.

JUSTIFICATIVA. Nesta alternativa, temos aspectos como modelo de requisitos e métricas, que estão fora do escopo da gestão de configuração de software.

E – Alternativa incorreta.

JUSTIFICATIVA. As etapas de identificação de alterações, controle de versão, mapeamento de alteração e notificação da alteração aos envolvidos, de certa forma, descrevem superficialmente a sequência de atividades envolvidas na SCM. No entanto, a alternativa não especifica e não nomeia as cinco atividades estabelecidas.

3. Indicações bibliográficas

- MORAIS, I. S. de (org.). *Engenharia de software*. São Paulo: Pearson Education do Brasil, 2017.
- PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software: uma abordagem profissional*. Porto Alegre: AMGH, 2021.
- SOMMERVILLE, I. *Engenharia de software*. São Paulo: Pearson Education do Brasil, 2018.

Questão 5

Questão 5.[5]

Um ponto importante para fábricas de software é a garantia de que seus produtos e/ou serviços acompanhem as diversas mudanças ocasionadas pelo mundo corporativo. As restrições de orçamento ou cronograma, reestruturação do negócio, modificações de regras de negócio ou novas necessidades do cliente, geram mudanças inevitáveis. Nesse cenário, a utilização de um processo de Gerência de Configuração de Software (GCS) que contemple características necessárias para gerenciar e controlar a evolução de um software, através do controle formal de versão e solicitação de mudanças, é extremamente importante.

Sobre o processo de Gerência de Configuração de Software (GCS), avalie as afirmativas.

I. O controle de versão possibilita o compartilhamento de dados e a edição colaborativa, permitindo a gestão de diferentes ramos de desenvolvimento e possibilitando a existência de diferentes versões de forma simultânea.

II. A auditoria de configuração é uma atividade de garantia de qualidade do sistema que tem por objetivo assegurar que a qualidade do software seja mantida quando feitas alterações requisitadas.

III. O relatório de status de configuração possui informações sobre o GCS, tendo por objetivo manter o cliente informado sobre as alterações, e deve ser gerado ao final do processo.

IV. A GCS proporciona um conjunto de atividades de acompanhamento e controle de mudanças que é iniciado logo depois que o software for fornecido ao cliente e colocado em operação.

É correto apenas o que se afirma em

A. I e II.

B. II e III.

C. III e IV.

D. I, II e IV.

E. I, III e IV.

[5]Questão 22 – Enade 2021.

1. Introdução teórica

Gestão de configuração de software

A questão testa conhecimentos a respeito das atividades envolvidas na gestão de configuração de software. A nomenclatura presente nas alternativas da questão corresponde à utilizada por Roger Pressman e Bruce Maxim, no livro intitulado "Engenharia de Software: uma abordagem profissional".

Conforme abordamos na introdução teórica da questão 4, a **gestão de configuração de software** (SCM, do termo em inglês "software configuration management") é responsável por identificar, organizar e controlar modificações ao longo do ciclo de vida do software. No enunciado da presente questão, é usado o termo **gerência de configuração de software** (GCS), que corresponde ao mesmo conceito.

Se você não está confortável com esse assunto, consulte a Introdução Teórica da questão 4, onde as atividades envolvidas na GCS são detalhadas.

2. Análise das afirmativas

I – Afirmativa correta.

JUSTIFICATIVA. Na atividade de controle de versão, é criada uma atualização do arquivo original logo que a alteração for feita. Alternativamente, os objetos a serem alterados podem ser "retirados" do banco de dados do projeto e, assim que a alteração é feita, os objetos são então "colocados" novamente no banco de dados, e são usados mecanismos de controle de versão apropriados para gerar a próxima versão do software. O controle de versão permite o acompanhamento das alterações feitas, facilitando o trabalho colaborativo e a gestão de diferentes versões.

II – Afirmativa correta.

JUSTIFICATIVA. A auditoria de configuração é uma das atividades da GCS que verifica se a configuração do software está em conformidade com as políticas, com os padrões e com os requisitos estabelecidos. Desse modo, a auditoria de configuração tem o objetivo de assegurar que a qualidade do software seja mantida quando feitas alterações requisitadas.

III – Afirmativa incorreta.

JUSTIFICATIVA. O relatório de status de configuração não necessariamente é gerado apenas ao final do processo. Ele pode ser gerado em diferentes fases do desenvolvimento, para informar sobre o estado da configuração.

IV – Afirmativa incorreta.

JUSTIFICATIVA. As atividades de GCS geralmente acontecem desde as fases iniciais do desenvolvimento e continuam por todo o ciclo de vida do software. Portanto, tais atividades não são limitadas apenas ao período após o fornecimento do software ao cliente.

Alternativa correta: A.

3. Indicações bibliográficas

- MORAIS, I. S. de (org.). *Engenharia de software*. São Paulo: Pearson Education do Brasil, 2017.
- PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software: uma abordagem profissional*. Porto Alegre: AMGH, 2021.
- SOMMERVILLE, I. *Engenharia de software*. São Paulo: Pearson Education do Brasil, 2018.

ÍNDICE REMISSIVO