UNIP
EDUCAÇÃO, POR SUA ESCOLHA

CIÊNCIA DA COMPUTAÇÃO

MATERIAL INSTITUCIONAL ESPECÍFICO

TOMO 15

CQA - COMISSÃO DE QUALIFICAÇÃO E AVALIAÇÃO

CIÊNCIA DA COMPUTAÇÃO

MATERIAL INSTRUCIONAL ESPECÍFICO

TOMO 15

Christiane Mazur Doi

Doutora em Engenharia Metalúrgica e de Materiais, Mestra em Ciências - Tecnologia Nuclear, Especialista em Cálculo e Matemática Aplicada, Especialista em 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.

Anderson Aparecido Alves da Silva

Doutor em Engenharia Elétrica – Sistemas Eletrônicos, Mestre em Engenharia da Computação e pós-graduado em Administração de Empresas com ênfase em Análise de Sistemas. Professor titular da Universidade Paulista.

Tiago Guglielmeti Correale

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

Questão 1

Questão 1.[1]

A Lei Geral de Proteção de Dados Pessoais (LGPD) está em vigência desde o final de 2018 e tem por objetivo regulamentar o tratamento de dados pessoais de clientes e usuários de empresas públicas e privadas.

Sobre a LGPD, avalie as afirmativas.

I. A lei reprime o uso indiscriminado de dados pessoais considerados sensíveis, como origem racial ou étnica, convicção religiosa e opinião política, informados em cadastros pelos cidadãos.

II. Os dados anonimizados não serão considerados pessoais, mesmo que, utilizando-se de recursos próprios ou tecnológicos avançados, o processo de anonimização possa ser revertido.

III. O indivíduo poderá exigir que uma empresa informe se possui dados pessoais dele bem como solicitar formalmente que eles sejam corrigidos, atualizados ou eliminados.

IV. A Autoridade Nacional de Proteção de Dados (ANPD) é responsável pela fiscalização e regulação da LGPD, prestando esclarecimentos, averiguando possíveis denúncias e modificando a legislação pertinente quando necessário.

É correto apenas o que se afirma em

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

1. Introdução teórica

1.1. Privacidade

A privacidade refere-se à salvaguarda de um ativo pessoal, em geral uma informação, que tem um indivíduo como proprietário. Esse ativo pessoal pode ser um dado, uma imagem, um histórico médico ou um número de CPF, por exemplo. Assegurar a privacidade dessas informações implica manter o controle sobre sua utilização e sua divulgação, conferindo esse controle ao seu legítimo proprietário. Não se trata de confidencialidade, visto que uma informação privada não é necessariamente secreta, mas sim de garantir que o proprietário da informação mantenha controle sobre a sua distribuição e o seu uso.

[1]Questão 12 – Enade 2021.

Embora o conceito seja compreensível em sua simplicidade, a implementação prática pode ser complexa. O desafio principal reside no fato de que o próprio detentor da informação, muitas vezes, é o seu divulgador. Compartilhar opiniões, fotos e mensagens e outras informações pessoais em plataformas como redes sociais e e-mails frequentemente resulta na perda total de controle sobre o uso e a distribuição dessas informações. Um eventual resultado prejudicial é que as postagens podem ser utilizadas contra o próprio detentor da informação ou podem prejudicar terceiros.

1.2. Lei Geral de Proteção de Dados (LGPD)

A Lei Geral de Proteção de Dados (LGPD) é uma lei federal (Lei Nº 13.709/2018) destinada a salvaguardar a privacidade dos dados pessoais dos indivíduos, protegendo-os contra o uso indevido por parte das empresas. O foco central está nos dados pessoais dos indivíduos.

Um dado pessoal é uma informação relacionada ao titular (pessoa natural identificada ou identificável como o proprietário da informação), como o prontuário médico, a idade, uma foto ou a opinião de um indivíduo. Dados pessoais diretos são aqueles que identificam diretamente uma pessoa, como uma foto. Dados pessoais indiretos são informações que só permitem o rastreamento de alguém de forma indireta, como ações para encontrar uma pessoa a partir da placa de um carro ou de um CPF.

A LGPD também se debruça sobre dados sensíveis, que são dados pessoais mais íntimos, coletados apenas com necessidade legítima, como informações sobre a origem racial ou étnica, a convicção religiosa, a opinião política, a filiação a sindicato, a organização de caráter religioso, filosófico ou político, dados referentes à saúde ou à vida sexual e material genético ou biométrico, entre outros. Os dados sensíveis também incluem dados pessoais de crianças e adolescentes. Há ainda o dado pessoal anonimizado, que não permite a identificação de um indivíduo específico. Um exemplo desse caso está nas pesquisas de opinião pública, que não expõem a forma como cada entrevistado respondeu a uma pergunta ou expressou um parecer sobre um assunto.

A ideia é que qualquer tratamento realizado com o uso de dados pessoais tenha autorização do titular. O artigo 5º X da LGPD (BRASIL, 2018) define tratamento como

> *toda operação realizada com dados pessoais, como as que se referem a coleta, produção, recepção, classificação, utilização, acesso, reprodução, transmissão, distribuição, processamento, arquivamento, armazenamento, eliminação,*

> *avaliação ou controle da informação, modificação, comunicação, transferência, difusão ou extração [de dados]* (BRASIL, 2018).

Há alguns elementos envolvidos no tratamento, como o controlador, o operador e o agente de tratamento. Basicamente, esses elementos podem ser definidos como aqueles que estão de posse dos dados pessoais realizando tratamentos. Podem ser empresas de posse de cadastros de pessoas ou provedores de serviços de redes sociais que contêm diversos dados sobre os clientes.

Além de garantir que o titular é o único que deve permitir (ou não) o tratamento dos próprios dados pessoais, a LGPD exige dos controladores exatidão, clareza, relevância, segurança e atualização dos dados, de acordo com a vontade do titular e com a exigência necessária.

A abrangência da lei estende-se apenas ao país, envolvendo dados pessoais coletados ou gerados no Brasil, relacionados a indivíduos presentes em nosso território (ainda que sejam estrangeiros) ou oferecidos ao público brasileiro. A transferência internacional de dados para países que aplicam legislações semelhantes à LGPD está prevista nos termos da lei. Contudo, a LGPD não se aplica a dados de pessoas jurídicas, gerados fora do país ou de pessoas já falecidas. Também entram nas exceções da abrangência da lei os dados jornalísticos, públicos, corporativos (de empresas), acadêmicos, relacionados à segurança pública ou que não tenham objetivo econômico – um fã que colecione dados pessoais de um artista está isento dos termos da lei.

A LGPD é mais rigorosa para os dados sensíveis, que devem ser tratados apenas quando isso for indispensável e exigem autorização especial para o tratamento. O consentimento relativo a dados de crianças e de adolescentes deve vir só dos responsáveis.

Em relação à segurança e à qualidade dos dados, a responsabilidade é dos controladores/operadores. Eventuais incidentes, como o vazamento de dados ou o tratamento inadequado sem autorização do titular, estão sujeitos aos rigores da lei. A Autoridade Nacional de Proteção de Dados (ANPD) é a responsável pela aplicação das sanções, que, em geral, envolvem advertências, divulgação pública das infrações, bloqueio dos dados e multas.

2. Análise das afirmativas

I - Afirmativa correta.

JUSTIFICATIVA. Nem empresas nem indivíduos podem fazer uso indiscriminado de dados pessoais sensíveis, como origem racial ou étnica, convicção religiosa e opinião política, muitas

vezes informados em cadastros pelos cidadãos. No entanto, o Art. 11º cita que há situações legais que permitem exceções:

- quando há consentimento do titular ou do responsável;
- no cumprimento de obrigação legal ou regulatória;
- em situações ligadas às políticas públicas;
- nos estudos via órgão de pesquisa;
- no direito, no contrato ou no processo;
- na preservação da vida ou da integridade física de uma pessoa;
- na prevenção de fraudes contra o titular;
- na tutela de procedimentos feita por profissionais das áreas da saúde ou sanitária.

Qualquer uso fora dessa lista é considerado ilegal, principalmente se tiver como intenção a prática de discriminação ilícita ou abusiva.

II - Afirmativa incorreta.

JUSTIFICATIVA. O processo de anonimização é aquele pelo qual um dado perde a possibilidade de associação, direta ou indireta, a um indivíduo, e não precisa mais ser regido pela LGPD. De acordo com artigo 5º, III da LGPD (BRASIL, 2018), dado anonimizado é aquele que não identifica o indivíduo. Por exemplo, dados relacionados à votação são anonimizados, pois não é possível identificar, apenas pelos percentuais resultantes das votações, qual foi o candidato em que cada eleitor votou. Adicionalmente, é importante lembrarmos o conceito de "dado pseudonimizado". Esse caso ocorre quando um dado perde a possibilidade de associação direta ou indireta a um indivíduo, mas permanece pessoal com o uso de informação adicional mantida separadamente pelo controlador de dados em ambiente controlado e seguro. Um exemplo desse cenário ocorre quando um autor usa pseudônimos para escrever um livro. Dessa forma, sabemos que dados anonimizados não são considerados pessoais. Porém, a partir do momento em que são utilizadas informações complementares ou recursos que permitam a reversão da anonimização, o dado é considerado pseudonimizado e está sujeito à LGPD.

III - Afirmativa correta.

JUSTIFICATIVA. Um dos aspectos destacados por vários especialistas na criação da LGPD é justamente garantir o consentimento do titular dos dados. É fundamental que exista a clareza de que os dados estão sendo coletados. O titular também deve ser informado da finalidade da coleta de dados e de como (e quais) os dados serão processados e armazenados. Além

disso, a lei garante que o titular deve ser capaz de solicitar a correção, a atualização e a exclusão dos dados coletados e armazenados.

IV - Afirmativa incorreta.

JUSTIFICATIVA. A Autoridade Nacional de Proteção de Dados (ANPD) tem como missão institucional assegurar a mais ampla e correta observância da LGPD no Brasil e, nessa medida, garantir a devida proteção aos direitos fundamentais de liberdade, privacidade e livre desenvolvimento da personalidade dos indivíduos. A ANPD é responsável pela fiscalização e pela regulação da LGPD, prestando esclarecimentos, averiguando possíveis denúncias e modificando a legislação pertinente quando necessário. O art. 55-J da LGPD estabelece diversas competências da ANPD. Contudo, o órgão não realiza especificamente investigação de crimes, mas sim de infrações administrativas, podendo aplicar aos infratores as sanções previstas na LGPD. Eventuais modificações da legislação cabem aos deputados, ao governador e, em alguns casos, ao Tribunal de Justiça e ao Procurador Geral de Justiça.

Afirmativa correta: B.

3. Indicações bibliográficas

- BRASIL. Lei Nº 13.709, de 14 de agosto de 2018. *Lei Geral de Proteção de Dados Pessoais (LGPD)*. Diário Oficial da União, Brasília, DF, 15 ago. 2018. Disponível em <http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm>. Acesso em 19 mar. 2026.
- BRASIL. PRESIDÊNCIA DA REPÚBLICA. *Autoridade Nacional de Proteção de Dados (ANPD)*. Disponível em <https://www.gov.br/anpd/pt-br>. Acesso em 12 out. 2025.
- BRASIL. PRESIDÊNCIA DA REPÚBLICA. *Perguntas Frequentes*. Disponível em <https://www.gov.br/anpd/pt-br/acesso-a-informacao/perguntas-frequentes-2013-anpd#b2>. Acesso em 19 mar. 2026.
- MULHOLLAND, C. A. *LGPD e o novo marco normativo no Brasil*. Porto Alegre: Arquipélago Editorial, 2020.
- RAPÔSO, C. F. L. et al. LGPD - Lei geral de proteção de dados pessoais em tecnologia da informação: revisão sistemática. *Revista de Administração do Cesmac*, v. 4, p. 58-67, 2019.

Questão 2

Questão 2.[2]

Leia o texto a seguir.

O desenvolvimento de sistemas iterativo e evolutivo é uma abordagem que estabelece ciclos de desenvolvimento, com duração fixa, chamados iterações. O produto de cada iteração é um sistema parcial, executável, testável e integrável. Cada iteração inclui suas próprias atividades de análises de requisitos, projeto, implementação e teste. O ciclo de vida iterativo é baseado em refinamentos e incrementos sucessivos de um sistema por meio de múltiplas iterações, com realimentação e adaptação cíclicas como principais propulsores para convergir para um sistema adequado.

CRAIG, L. *Utilizando UML e Padrões: uma Introdução à Análise e ao Projeto Orientados a Objetos*. 3. ed. Porto Alegre: Bookman, 2007 (com adaptações).

Considerando o texto apresentado, assinale a opção correta sobre o desenvolvimento iterativo e evolutivo.

A. A mudança nos requisitos do sistema é algo que gera atraso no desenvolvimento, por isso é aconselhável evitá-la.
B. O ciclo de desenvolvimento possui duração fixa, porém, durante o desenvolvimento, poderá ser alterado no caso de sistemas críticos.
C. O teste de usabilidade deve ser realizado no último ciclo, pois será o momento em que o usuário consegue testar todas as funcionalidades.
D. O subsistema gerado pela implementação dos requisitos no fim de uma iteração poderá ser utilizado pelo cliente como protótipo.
E. O documento de teste de usabilidade deve contemplar os critérios de acessibilidade para atender a todos os usuários do sistema.

1. Introdução teórica

1.1. Desenvolvimento de sistemas

O termo "desenvolvimento de sistemas" refere-se ao processo de criação, design, implementação, teste e manutenção de sistemas de software. Esse campo abrange ampla gama de atividades relacionadas à produção e à melhoria de sistemas de software, desde pequenas aplicações até sistemas complexos e integrados.

O desenvolvimento de sistemas envolve várias etapas, incluindo a análise dos requisitos do cliente, o design da arquitetura do sistema, a codificação do software, os testes para garantir a qualidade e a manutenção contínua para atualizações e correções. Durante esse

[2]Questão 13 – Enade 2021.

processo, os desenvolvedores usam linguagens de programação, ferramentas de desenvolvimento e métodos específicos para transformar os requisitos do usuário em software funcional e eficiente.

1.2. Análise de requisitos

A análise de requisitos envolve a coleta, a documentação e a compreensão detalhada dos requisitos do cliente para o sistema que está sendo desenvolvido. O objetivo é estabelecer clareza sobre o que o sistema deve fazer, como deve funcionar e quais são as expectativas e as necessidades dos usuários finais.

A análise de requisitos fornece a base para o desenvolvimento do sistema. Um entendimento preciso dos requisitos é essencial para evitar problemas como custos excessivos, atrasos no projeto e insatisfação do cliente. É um processo iterativo e contínuo que, muitas vezes, envolve colaboração próxima entre analistas de sistemas, desenvolvedores e stakeholders do projeto.

Há, basicamente, dois tipos principais de requisitos, funcionais e não funcionais, que serão detalhados nas próximas seções.

1.2.1. Requisitos funcionais

Os requisitos funcionais descrevem o que um sistema deve oferecer ao usuário. Eles definem o que o sistema deve fazer em termos de comportamento, operações e interações com seus usuários. Em outras palavras, tais requisitos especificam as funções e as características que o sistema deve fornecer para atender às necessidades dos usuários e alcançar os objetivos do projeto.

Os requisitos funcionais são essenciais para orientar o desenvolvimento do sistema, ajudando a equipe de desenvolvimento a entender o que precisa ser implementado. Eles:

- orientam as diversas atividades de um projeto;
- influenciam as decisões da equipe de arquitetura;
- definem o que deve ser feito pela área de implementação;
- são utilizados como base para as etapas de testes, validação e verificação do sistema.

1.2.2. Requisitos não funcionais

Requisitos não funcionais são critérios que descrevem características do sistema que não se referem diretamente às suas funcionalidades específicas, mas sim às qualidades globais, às restrições ou às propriedades que o sistema deve ter. Enquanto os requisitos funcionais definem o que o sistema deve fazer, os requisitos não funcionais especificam como o sistema deve realizar suas funções. Esses requisitos são importantes para o sucesso do projeto, uma vez que impactam aspectos como desempenho, segurança, usabilidade e confiabilidade do sistema.

Exemplos comuns de requisitos não funcionais são o desempenho, a usabilidade, a confiabilidade, a segurança, a manutenibilidade, a compatibilidade, a escalabilidade e a disponibilidade dos sistemas.

Todos esses requisitos são essenciais para garantir que o sistema atenda não apenas às expectativas funcionais dos usuários, mas também a uma série de critérios que impactam sua utilidade, sua confiabilidade e sua aceitação no ambiente em que será utilizado.

1.3. Fases da análise de requisitos

A análise de requisitos geralmente inclui as fases a seguir.

- **Coleta de requisitos**: os analistas de sistemas interagem com os *stakeholders*, com os clientes, os usuários finais e outros interessados, para entender as necessidades e as expectativas em relação ao sistema.
- **Documentação**: os requisitos são documentados de maneira clara e detalhada para garantir que todos os envolvidos no processo tenham uma compreensão comum. Isso pode envolver a criação de documentos, diagramas, protótipos ou modelos, dependendo da complexidade do sistema e do método utilizado.
- **Análise e priorização**: os requisitos coletados são analisados para garantir que sejam claros, completos e não contraditórios. É comum priorizar os requisitos para ajudar na fase subsequente de *design* e implementação.
- **Validação**: os requisitos são revisados pelos *stakeholders* para garantir que representem com precisão as necessidades. Isso ajuda a evitar mal-entendidos e retrabalho posterior no processo de desenvolvimento.
- **Gerenciamento de mudanças**: com o tempo, é possível que os requisitos inicialmente especificados mudem, por diferentes razões. É importante que essas mudanças sejam

avaliadas e ocorra o gerenciamento desse processo, fazendo com que as mudanças aconteçam de forma controlada (não caótica).

1.4. Iteração e incremento no desenvolvimento de software

O desenvolvimento iterativo é um processo no qual algumas atividades são feitas repetidas vezes ao longo do ciclo de desenvolvimento, e não uma única vez, como no modelo em cascata. Em um processo iterativo, à medida que o software é construído, ele é avaliado e ajustado para melhorias. Não se espera que o resultado seja perfeito nas primeiras iterações.

É possível dividir o planejamento do projeto em uma série de períodos de duração fixa, chamados de "iterações".

O incremento é um pouco diferente da iteração. Enquanto a ideia básica de iteração é a de repetição de um processo, o incremento é um aumento gradativo de novas partes.

Quando se pensa em desenvolvimento de software, o desenvolvimento incremental é a adição de novas soluções ao longo do tempo. O software é construído em partes, com iterações e incrementos não sequenciais, fato que diminui o tempo de desenvolvimento. Isso permite uma forma de paralelismo no projeto. Por exemplo, a equipe de desenvolvimento pode trabalhar em uma parte do projeto (aquela que já tem alguns requisitos definidos) ao mesmo tempo em que novos requisitos são levantados.

Em contraste com o modelo incremental de desenvolvimento, existe o modelo em cascata, que costuma ser sequencial – ao término de uma tarefa, outra é iniciada. É um modelo mais simples de se gerenciar, mas, em alguns ambientes, como os dedicados ao desenvolvimento ágil, quase não é mais utilizado. O modelo iterativo e incremental é muito empregado no desenvolvimento ágil devido à velocidade das mudanças. Apesar de entregar menos funcionalidades iniciais, a ideia é proporcionar, de forma rápida, uma primeira versão do software ao cliente. Isso permite correção de rumos e solicitações de melhorias mais rápidas.

2. Análise das alternativas

A – Alternativa incorreta.

JUSTIFICATIVA. Os requisitos de um software podem não permanecer imutáveis ao longo do desenvolvimento. Esse cenário de mudança dos requisitos é chamado de volatilidade. A volatilidade nos requisitos aumenta a probabilidade de atrasos em um projeto. Essa

volatilidade ocorre principalmente porque, na fase inicial do projeto, os requisitos ainda não estão bem definidos e, na maioria das vezes, alguns detalhes da especificação só são conhecidos durante a implementação do sistema. Além disso, existem diversos cenários legítimos que podem levar a alterações nos requisitos de um projeto (por exemplo, uma empresa pode ter sido vendida durante o desenvolvimento de um produto).

B – Alternativa incorreta.

JUSTIFICATIVA. Em um projeto de software que utiliza um ciclo de vida iterativo, a ideia principal é que o produto vá crescendo e ganhando funcionalidades com o tempo, em pequenos incrementos. Cada iteração pode ser considerada um miniprojeto de duração fixa, sendo que cada um inclui suas próprias atividades de análise de requisitos, projeto, implementação e testes. Isso leva à realimentação rápida.

Como a duração de cada iteração é relativamente pequena, não podemos esperar um produto completo ao final de cada uma. Em termos de qualidade do que foi implementado, o objetivo é ter algo muito próximo do que se pretende entregar, e não apenas um programa similar a um protótipo.

Há especialistas que afirmam que as iterações devem ter duração previamente estabelecida e que a quantidade de funcionalidades escolhidas para serem implementadas esteja de acordo com esse período. O ideal é que o tempo das iterações seja relativamente curto, dada a característica incremental desse método. Assim, se as iterações são muito longas, a aplicação das ideias principais das metodologias ágeis fica prejudicada. Dessa forma, a alteração do tempo pré-determinado durante o desenvolvimento pode ser algo considerado problemático, especialmente no caso de sistemas críticos.

C – Alternativa incorreta.

JUSTIFICATIVA. De acordo com os conceitos de *Human-Computer Interaction* (Interação Humano-Computador) e de *User Experience* (Experiência do Usuário), a usabilidade normalmente se refere à simplicidade e à facilidade com que uma interface (como um *site*, um aplicativo, um programa de computador ou um jogo) pode ser utilizada e é um fator decisivo no projeto de desenvolvimento. Portanto, é necessária uma avaliação sobre qual o momento ideal para o teste de usabilidade. A fase mais indicada para aplicar esse teste é a construção de um projeto, visto que esse é o momento no qual a equipe de desenvolvimento tende a direcionar os esforços para melhorar a experiência do usuário e evitar possíveis problemas de usabilidade.

D – Alternativa incorreta.

JUSTIFICATIVA. O objetivo de um protótipo é expor ao usuário uma versão simplificada de um sistema, como entendido pela equipe de desenvolvimento, com a finalidade de avaliação. O objetivo é que a equipe de desenvolvimento verifique se o seu entendimento das necessidades e dos requisitos do cliente está correto ou na direção certa. Em alguns casos, um protótipo pode ser uma versão simplificada de um produto, mas, em outros, pode ser algo totalmente "descartável". De qualquer forma, um protótipo não deve ser visto como alguma coisa que será utilizada pelo cliente de forma definitiva.

Adicionalmente, nos modelos de desenvolvimento iterativo-incrementais, o objetivo no final das iterações é obter um sistema funcional, ainda que limitado em relação ao número de funcionalidades implementadas, e não apenas um protótipo.

E – Alternativa correta.

JUSTIFICATIVA. Com o aumento da utilização da comunicação e de sistemas online, associado à busca contínua por mais qualidade, menos riscos e melhores resultados, incluir o teste de software no ciclo de vida de desenvolvimento torna-se cada vez mais importante. Um teste de software é uma operação ampla e é composto por outros pequenos testes, como:

- o teste de usabilidade, que avalia até que ponto a interface do software é fácil e intuitiva;
- o teste de confiabilidade, que busca monitorar e garantir a integridade da solução e dos dados;
- o teste de portabilidade, que verifica o grau de portabilidade da aplicação;
- o teste de acessibilidade, que garante que o software possa ser acessado por qualquer usuário, inclusive por aqueles que tenham algum tipo de deficiência ou limitação.

Há, portanto, uma clara ligação entre a usabilidade desejada para um sistema e a acessibilidade que esse mesmo sistema deve oferecer.

3. Indicações bibliográficas

- AGILE ALLIANCE. *Manifesto for agile software development*. Disponível em <https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/>. Acesso em 19 mar. 2026.
- ENGHOLM, H. *Engenharia de software na prática*. São Paulo: Novatec, 2010.
- IBM. *Desenvolvimento iterativo*. Disponível em <https://www.ibm.com/docs/pt-br/elm/6.0.5?topic=scenarios-iterative-development>. Acesso em 19 mar. 2026.

- MACEDO, P. C.; SBROCCO, T. C.; Henrique, J. *Metodologias ágeis: Engenharia de Software sob medida*. São Paulo: Érica, 2012.
- PRESSMAN, R. S. *Engenharia de Software*. 6. ed. São Paulo: McGrawHill, 2006.
- SOMMERVILLE, I. *Engenharia de Software*. 8. ed. São Paulo: Addison-Wesley, 2007.
- VALENTE, M. T. *Engenharia de Software Moderna - Princípios e práticas para desenvolvimento de software com produtividade*. Belo Horizonte: UFMG, 2020.

Questão 3

Questão 3.[3]

Leia o texto a seguir.

O primeiro computador criado foi o ENIAC (Electronic Numerical Integrator And Computer), desenvolvido por Eckert e Mauchly na Universidade da Pennsylvania, Estados Unidos. O projeto iniciou-se em 1943, financiado pelo governo americano. O período era da Segunda Guerra Mundial e o objetivo era poder calcular de forma mais ágil as melhores trajetórias para transporte de armas e mantimentos em meio aos exércitos inimigos. Esse é o tipo de cálculo que pequenos aparelhos celulares fazem hoje para encontrar rotas nas cidades por meio do sistema de GPS (Global Positioning System) e a análise de mapas. O projeto só foi concluído em 1946, tarde demais para ser utilizado para a Segunda Guerra, mas foi bastante utilizado até 1955.

Muitos projetos surgiram depois do ENIAC, mas eles eram barrados por algumas dificuldades e limitações, como, por exemplo, o fato de não serem programados e trabalharem com números decimais. O problema de trabalhar com decimais é que cada algarismo armazenado possui 10 estados possíveis, representando os números de 0 a 9. Dentro de um sistema eletrônico, isso é complicado porque a carga de cada dispositivo, seja transistor, seja válvula, deveria ser medida para se verificar que número ela estava representando. Os erros eram muito frequentes. Bastava que uma válvula estivesse fora da temperatura ideal para que os resultados das operações começassem a sair errado. Von Neumann recomendou, então, que, em sua arquitetura, os dados e instruções passassem a ser armazenados em código binário, facilitando a análise dos mesmos e reduzindo a quantidade de erros.

BRITO, A. V. *Introdução a Arquitetura de Computadores*. UFPB Virtual, 2020. Disponível em <http://producao.virtual.ufpb.br/>. Acesso em 05 mai. 2020 (com adaptações).

Acerca da arquitetura de Von Neumann, avalie as asserções e a relação proposta entre elas.

I. Embora as arquiteturas de computadores tenham evoluído muito do ENIAC aos modernos notebooks de hoje, a arquitetura de Von Neumann, conceito da década de 1950, tem se mantido até os dias atuais.

PORQUE

II. A arquitetura de Von Neumann permite que a CPU realize a busca de uma ou mais instruções além da próxima a ser executada; essa técnica é utilizada para acelerar a velocidade de operação da CPU, uma vez que a próxima instrução a ser executada está normalmente armazenada nos registradores da CPU e não precisa ser buscada da memória principal, que é muito mais lenta.

A respeito dessas asserções, assinale a opção correta.

A. As asserções I e II são proposições verdadeiras, e a asserção II justifica a I.
B. As asserções I e II são proposições verdadeiras, e a asserção II não justifica a I.
C. A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.
D. A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.
E. As asserções I e II são proposições falsas.

[3]Questão 14 – Enade 2021.

1. Introdução teórica

1.1. Arquitetura de Von Neumann

A Arquitetura de Von Neumann é um modelo fundamental para o design de computadores e sistemas de processamento de informações. Ela é formada, basicamente, pela Unidade Central de Processamento (UCP), também conhecida pelo termo em inglês *Central Processing Unit* (CPU), pela memória e pelos dispositivos de entrada e saída. De forma geral, podemos destacar os componentes dessa arquitetura a seguir.

- ***Central Processing Unit*** (CPU ou Unidade Central de Processamento, em português): responsável pelo processamento das instruções. Internamente, a CPU pode ser subdividida em:
    - unidade de controle, responsável por controlar e coordenar operações do computador, como operações de busca, decodificação e execução de instruções;
    - unidade lógica e aritmética (ULA), responsável por realizar operações aritméticas e lógicas, como adição, subtração, comparação e operações booleanas.
- **Memória:** responsável pelo armazenamento de dados e instruções. De forma conceitual, a memória pode ser subdividida nos elementos a seguir.
    - Memória principal: armazena dados e instruções temporariamente enquanto o computador está em execução. Por ser volátil, o conteúdo dessa memória é perdido quando o computador é desligado. A memória principal é chamada de *Random Access Memory* (RAM) e corresponde à área na qual o computador guarda os dados e as instruções que estão sendo processados.
    - Memória secundária: armazena dados de forma permanente e não volátil. Um disco rígido (no inglês *Hard Disk Drive* ou HD) é um exemplo de memória permanente.
    - Memória cache: é uma pequena e rápida memória de acesso aleatório que atua como um intermediário entre a RAM e a CPU. Seu objetivo principal é armazenar temporariamente dados e instruções frequentemente acessados, reduzindo o tempo de acesso à memória principal e melhorando o desempenho geral do sistema. A memória cache opera com base no princípio da localidade, aproveitando a tendência de que os dados recentemente acessados têm maior probabilidade de serem acessados novamente no curto prazo. Existem geralmente diferentes níveis de cache (L1, L2, L3) com diferentes tamanhos e velocidades, cada um servindo para otimizar o desempenho do sistema.

- **Unidade de Entrada e Saída (E/S):** controla os dispositivos usados para dados de entrada (os que recebem dados para processamento, como teclados e mouse) e dados de saída (dispositivos para onde os dados processados são enviados, como monitores ou impressoras).
- **Registradores**: funcionam como pequenas memórias temporárias, internas à CPU, utilizadas para armazenar dados e instruções utilizados em tempo real pelo processador. Em geral, apresentam desempenho superior ao desempenho das memórias RAM e cache. Além disso, o fato de essas memórias estarem localizadas internamente à CPU torna o seu acesso muito mais rápido do que os outros tipos de memória. Normalmente, um processador tem diversos tipos de registradores, mas, de forma geral, os principais tipos são os mencionados a seguir.
    - Registrador de dados (ou acumulador): armazena temporariamente os dados durante as operações da ULA.
    - Registrador de instrução: armazena a instrução atual que está sendo executada pela CPU.
    - Contador de programa: mantém o endereço da próxima instrução a ser buscada na memória.
- **Barramentos**: funcionam como "estradas" que ligam os diversos componentes do computador, pelas quais passam dados e instruções. Um computador pode ter vários barramentos, alguns com finalidades específicas. De forma geral, os principais barramentos são os mencionados a seguir.
    - Barramento de dados: responsável pela transferência de dados entre a CPU e a memória.
    - Barramento de endereço: responsável pela transferência de informações sobre os endereços de memória dos dados, ou seja, identifica onde os dados devem ser lidos ou escritos.
    - Barramento de controle: transmite sinais de controle que coordenam as operações entre a CPU, a memória e os dispositivos de E/S.

A operação básica de um computador digital baseado na Arquitetura de Von Neumann segue um ciclo de instruções. Nesse ciclo, a CPU busca uma instrução na memória, decodifica essa instrução, executa a operação e, se necessário, armazena o resultado de volta na memória. Esse modelo de computação sequencial com um armazenamento único de dados e instruções é uma característica fundamental da Arquitetura de Von Neumann.

1.2. Outras arquiteturas

Embora a Arquitetura de Von Neumann tenha sido crucial para o desenvolvimento dos primeiros computadores e continue sendo a base para computadores atuais, existem outras arquiteturas utilizadas em sistemas digitais. Essas arquiteturas são exemplos de como os projetistas de computadores podem explorar diferentes abordagens para otimizar o desempenho em diferentes cenários e para atender a requisitos específicos de aplicativos. Cada uma tem suas vantagens e suas desvantagens, dependendo das necessidades particulares de uma aplicação ou de um sistema. Como exemplo de arquiteturas computacionais e de CPUs, podemos citar as que seguem.

- **Arquitetura de Harvard**: há separação física entre a memória que armazena dados e a memória que armazena instruções. Isso permite que a CPU acesse simultaneamente dados e instruções, melhorando a eficiência em certos cenários, como em sistemas embarcados e microcontroladores.
- **Arquitetura RISC (*Reduced Instruction Set Computing*):** estilo de design de arquitetura do conjunto de instruções (no inglês, *Instruction Set Architecture* - ISA) de uma CPU. Esse tipo de arquitetura utiliza um conjunto reduzido e simples de instruções, com foco no desempenho e na eficiência. Exemplos de arquiteturas RISC são a arquitetura MIPS (*Microprocessor without Interlocked Pipeline Stages*) e a arquitetura ARM (*Acorn RISC Machine*).
- **Arquitetura CISC (*Complex Instruction Set Computing*)**: em contraste com a arquitetura RISC (*Reduced Instruction Set Computing*), que busca simplificar o conjunto de instruções disponíveis, a arquitetura CISC usa um conjunto de instruções maior e mais complexo. Essa abordagem representa um foco diferente do ponto de vista de projeto de um processador. Enquanto na arquitetura RISC a ideia é desenvolver o processador mais eficiente possível para um conjunto restrito de instruções, na arquitetura CISC o foco está em oferecer um conjunto muito mais amplo de instruções com desempenho satisfatório (mesmo que eles não sejam ótimos). Muitas dessas instruções podem realizar tarefas complexas em uma única operação, o que pode reduzir a quantidade de código necessário para executar certas tarefas e oferecer uma vantagem em termos de desempenho e uso de memória.
- **Arquitetura SIMD (*Single Instruction, Multiple Data*)**: uma única instrução é usada para realizar a mesma operação em múltiplos dados simultaneamente. Isso é

especialmente útil em aplicações que lidam com processamento paralelo, como gráficos e processamento digital de sinais.

- **Arquitetura MIMD (*Multiple Instruction, Multiple Data*)**: diferentemente do modelo de computação tradicional, de natureza sequencial, em que uma única instrução é executada de cada vez, a arquitetura MIMD permite a execução simultânea de várias instruções em diferentes conjuntos de dados. Isso é comumente encontrado em sistemas com vários processadores.
- **Arquitetura VLIW (*Very Long Instruction Word*)**: procura explorar diretamente o recurso de paralelismo em nível de instrução (*Instruction Level Parallelism – ILP*), utilizando informações vindas do compilador. Dessa forma, não é necessário utilizar hardware especial para alguns recursos de paralelismo, como é o caso dos processadores convencionais nos dias de hoje.
- **Arquitetura EPIC (*Explicitly Parallel Instruction Computing*):** desenvolvida pela Intel com a Hewlett-Packard, a arquitetura EPIC é usada na família de processadores Itanium. Baseia-se na execução paralela de instruções, utilizando uma abordagem similar à da arquitetura VLIW.

2. Análise das asserções

I – Asserção verdadeira.

JUSTIFICATIVA. A "programação" dos primeiros computadores exigia um processo trabalhoso, começando com fluxogramas e cédulas de papel, seguidos de desenhos detalhados de engenharia, além de um penoso processo baseado na conexão física de painéis, com a utilização de enorme quantidade de cabos e conectores. A arquitetura de John von Neumann simplificou drasticamente esse processo e é referência nos estudos sobre microprocessadores e arquitetura de computadores. Muitas melhorias surgiram nos últimos anos, ampliando a proposta original da arquitetura de Von Neumann, mas essa arquitetura foi a precursora de boa parte dos sistemas computacionais atuais. Muitos computadores atualmente empregam essa estrutura. Ou seja, o conceito mantém-se atual, mesmo após muitos anos.

II – Asserção falsa.

JUSTIFICATIVA. Um dos problemas com a afirmativa é que o registrador de instrução, presente na maioria dos microprocessadores, armazena a instrução atual em execução, e não a próxima instrução a ser executada. Além disso, para que uma instrução seja armazenada

em um registrador, em algum momento ela deve ser buscada na memória, passando pelo barramento. Na arquitetura de Von Neumann, o mesmo barramento é utilizado para instruções e dados, o que significa que, muito provavelmente, após a CPU ter solicitado uma instrução, ela deve utilizar o mesmo barramento para buscar os dados a serem utilizados pela instrução, inviabilizando a estratégia descrita na asserção. Contudo, se pensarmos que o conceito descrito na asserção corresponde ao uso de pipelines, a sua utilização não se limita à arquitetura de Von Neumann, sendo bastante comum em diversos outros tipos de arquiteturas utilizadas em vários microprocessadores, como a arquitetura de Harvard, por exemplo.
Um dos motivos pelo qual a arquitetura de Von Neumann se tornou tão popular é que ela simplifica o modelo de programação. Um modelo unificado e homogêneo de memória com instruções e dados facilita tanto o trabalho do programador (que não precisa se preocupar com detalhes nem limitações específicas de uma máquina) quanto o trabalho do sistema operacional. Contudo, existem diversos outros modelos de arquiteturas computacionais criados ao longo do tempo. Muitos foram feitos para explorar possibilidades de aumento de desempenho, especialmente o paralelismo. De qualquer forma, mesmo nesses novos modelos, a ideia geral da arquitetura de Von Neumann continua sendo utilizada, devido à sua simplicidade e à suaelegância.

Alternativa correta: C.

3. Indicações bibliográficas

- BARTIK, J. J. *Pioneer Programmer: Jean Jennings Bartik and the Computer that Changed the World*. Kirksville: Truman State University Press, 2013.
- BRITO, G. S.; PURIFICAÇÃO, I. *Educação e novas tecnologias: um (re)pensar*. 3. ed. Curitiba: IBPEX, 2011.
- LORIN, H. *Introdução à arquitetura e organização de computadores*. Rio de Janeiro: Campus, 1985.
- MONTEIRO, M. A. *Introdução à organização de computadores*. 5. ed. Rio de Janeiro: LTC, 2007.
- NULL, L.; LOBUR, J. *Princípios básicos de arquitetura e organização de computadores*. 2. ed. Porto Alegre: Bookman, 2010.
- SALGANIK, M. J. *Bit by Bit: Social research in the digital age*. New Jersey: Princeton University Press, 2017.

- STALLINGS, W. *Arquitetura e organização de computadores*. 8. ed. São Paulo: Pearson Prentice Hall, 2010.
- TANENBAUM, A. S. *Organização estruturada de computadores*. 5. ed. São Paulo: Pearson Prentice Hall, 2010.
- WEBER, R. F. *Fundamentos de arquitetura de computadores*. 3. ed. Porto Alegre: Bookman; Instituto de Informática da UFRGS, 2008.

Questão 4

Questão 4.[4]

Leia o texto a seguir.

O surgimento das metodologias ágeis eliminou o gerenciamento baseado em planos, substituindo-o pelo planejamento incremental. A documentação de projeto foi reduzida ao mínimo e deixou de ser previsto um gerente de projeto. Infelizmente, esse tipo de abordagem não atende as necessidades das organizações, em que gerentes de negócio necessitam acompanhar o andamento dos projetos, controlar orçamento, estabelecer prioridades e atualizar seus planos de negócio. Nesse contexto, foi desenvolvido o SCRUM, um framework para a organização de projetos ágeis. O SCRUM prevê dois indivíduos: o Scrum Master e o Product Owner, que são responsáveis por atuar como interface entre a equipe de desenvolvimento e a organização.

SOMMERVILLE, I. *Engineering Software Products: An Introduction to Modern Software Engineering*. Boston: Pearson, 2019 (com adaptações).

Em relação à metodologia *SCRUM*, avalie as afirmativas.

I. O papel do *Scrum Master* é guiar a equipe no uso efetivo da metodologia *SCRUM*.

II. O papel do *Product Owner* é garantir o foco no produto, evitando que ele se perca em questões técnicas menos relevantes.

III. Tanto o *Scrum Master* como o *Product Owner* têm autoridade direta sobre a equipe.

É correto o que se afirma em

A. II, apenas.

B. III, apenas.

C. I e II, apenas.

D. I e III, apenas.

E. I, II e III.

1. Introdução teórica

1.1. Modelos de desenvolvimento de software

Os antigos modelos de gerenciamento de projeto apresentavam problemas, como falhas nas estimativas dos custos e dos prazos. Por conta disso, alguns produtos eram lançados com atraso e não chamavam a atenção dos consumidores, que, muitas vezes, não se dispunham a pagar pelo resultado.

Além disso, um problema bastante comum era a dificuldade de comunicação entre as diferentes equipes envolvidas nas diversas etapas do desenvolvimento. Dessa forma, um

[4]Questão 15 – Enade 2021.

atraso em uma etapa costumava gerar uma sobrecarga de trabalho na equipe envolvida na etapa seguinte e, frequentemente, um atraso no projeto como um todo.

1.2. Modelo de desenvolvimento em cascata

O modelo de desenvolvimento de software em cascata é um modelo sequencial e linear que organiza o processo de desenvolvimento de software em fases distintas, em que o início de cada fase depende da conclusão da fase anterior. Ele foi proposto como uma abordagem sistemática para o desenvolvimento de software e é conhecido pela sua estrutura hierárquica e sequencial.

Devido à rigidez e à linearidade do modelo de desenvolvimento em cascata, qualquer alteração nos requisitos identificada posteriormente no processo pode exigir a revisão de todas as fases anteriores, tornando o modelo menos flexível em comparação às abordagens mais iterativas, como as metodologias ágeis. Ainda que diferentes empresas possam ter versões específicas desse modelo, de forma geral, as principais fases do modelo em cascata são as que seguem.

- **Levantamento de requisitos**: os requisitos do sistema são identificados e documentados.
- **Análise**: os requisitos levantados na fase anterior são analisados em detalhes. É aqui que os analistas de sistemas elaboram documentos de especificação de requisitos mais detalhados e compreendem a complexidade do sistema.
- ***Design*** **(ou projeto)**: com base nos requisitos especificados, o design do sistema é criado. Isso inclui o design arquitetural, o design de banco de dados e o design de interface do usuário, entre outros.
- **Implementação (codificação):** a codificação do software é realizada com base no design previamente estabelecido. Os programadores escrevem o código-fonte para implementar as funcionalidades e as características planejadas.
- **Testes**: em um projeto que segue um modelo de desenvolvimento em cascata, os testes são feitos após o término da implementação. Os testes são tipicamente vistos como uma atividade de controle de qualidade, mas há vários motivos para a sua execução. Obviamente, um aspecto chave é garantir que ele atenda aos requisitos especificados.
- **Manutenção**: após a conclusão bem-sucedida dos testes, o software é entregue ao cliente para uso. Posteriormente, eventuais correções de bugs, atualizações e melhorias também são realizadas nesta fase.

Um dos problemas apontados por especialistas com relação ao modelo de desenvolvimento em cascata é a dificuldade encontrada na adaptação às mudanças de requisitos durante o desenvolvimento, uma vez que é difícil retroceder e ajustar fases anteriores. Como resultado, em projetos nos quais os requisitos são suscetíveis a alterações, abordagens mais ágeis e iterativas podem ser preferíveis.

1.3. Metodologias ágeis

Os métodos ágeis difundiram-se devido à necessidade de avaliação da situação do projeto em curtos períodos e da comunicação sobre alterações ou imprevistos nas atividades dos demais membros das equipes.

Em 2001, um grupo de desenvolvedores lançou o Manifesto Ágil, no qual deixavam claros os valores dessa nova forma de trabalhar e pensar no desenvolvimento das novas soluções. Entre os principais pontos do Manifesto estão:

- indivíduos e interações mais do que processos e ferramentas;
- software em funcionamento mais do que documentação abrangente;
- colaboração com o cliente mais do que negociação de contratos;
- respostas a mudanças mais do que o seguimento de um plano.

De forma geral, as diversas metodologias ágeis utilizam abordagens de desenvolvimento de software que enfatizam a flexibilidade, a colaboração e a entrega incremental. Essas metodologias foram criadas para superar limitações percebidas em abordagens tradicionais de desenvolvimento de software, como no modelo em cascata.

As metodologias ágeis são caracterizadas por princípios e valores que promovem uma abordagem mais adaptativa e orientada a pessoas. Na sequência, estão alguns dos princípios e das características associados às metodologias ágeis.

- **Entrega incremental**: as funcionalidades são entregues em pequenos incrementos ao longo do tempo, permitindo que partes utilizáveis do software estejam disponíveis mais rapidamente.
- **Colaboração e comunicação**: a comunicação eficaz e a colaboração contínua entre membros da equipe, clientes e stakeholders são priorizadas. Isso inclui reuniões regulares, feedback constante e transparência.
- **Adaptabilidade e flexibilidade**: as metodologias ágeis valorizam a capacidade de adaptação a mudanças nos requisitos e nas condições do projeto. A flexibilidade é essencial para lidar com ambientes de desenvolvimento dinâmicos.

- **Iteração e feedback contínuo**: o desenvolvimento ocorre em ciclos curtos e repetitivos (iterações ou sprints), permitindo a incorporação constante de feedback para melhorar o produto.
- **Priorização de pessoas e de interações**: como explicitado no "Manifesto Ágil", na metodologia ágil, valorizam-se mais as pessoas e a interação entre elas do que o uso de processos e ferramentas. As metodologias ágeis reconhecem a importância da equipe e da colaboração entre pessoas.
- **Foco no cliente**: as necessidades e as expectativas do cliente são fundamentais, e o software é desenvolvido de forma a agregar valor direto ao usuário final.
- **Trabalho em equipe de forma auto-organizada**: equipes ágeis são encorajadas a serem auto-organizadas, ou seja, têm a responsabilidade de organizar e gerenciar seu próprio trabalho.
- **Entrega contínua de software funcional**: o objetivo é entregar software funcional em intervalos regulares, proporcionando valor imediato e capacidade de ajustar as prioridades conforme necessário.

As metodologias ágeis têm sido amplamente adotadas na indústria de software devido à sua capacidade de se adaptarem a mudanças e de gerarem produtos de alta qualidade e de forma eficiente. Elas são especialmente eficazes em projetos em que os requisitos podem evoluir com o tempo ou em ambientes de desenvolvimento dinâmicos. Os métodos ágeis mais conhecidos são o Scrum, o Kanban, o Extreme Programming (XP), entre outros.

O Kanban enfoca o fluxo contínuo de trabalho, permitindo ajustes rápidos nas prioridades, conforme necessário. Introduzido na década de 1990 por Kent Beck, o Extreme Programming (XP) visa a melhorar a qualidade do software e a satisfação do cliente por meio de práticas colaborativas e iterativas.

1.4. Scrum

O Scrum é um arcabouço (framework) para ser usado no gerenciamento do desenvolvimento de produtos complexos, sendo possível o emprego de diferentes processos e técnicas, de acordo com suas regras. Essas regras estão descritas no "Guia do Scrum" (2013), que serve justamente para auxiliar e manter os integrantes do projeto unidos, sendo a base para a definição dos papéis, eventos e artefatos.

O "Guia do Scrum" (2013) define o conceito do "Time Scrum", que engloba os papéis a seguir.

- Product Owner (dono do produto, em português): representa tudo aquilo que o cliente final deseja. Suas funções são acompanhar as entregas do "Time de Desenvolvimento" e certificar-se de que essas entregas correspondem aos requisitos apontados e aos prazos especificados.
- Scrum Master (mestre do Scrum, em português): tem a missão de servir ao "Time de Desenvolvimento" e ao Product Owner, viabilizando que as entregas sejam realizadas conforme os prazos combinados e estipulados de cada sprint. Representa a ligação entre o desenvolvimento e o cliente.
- Time de Desenvolvimento: formado pelos profissionais para realização do trabalho (envolvendo, entre outros profissionais, os programadores). O resultado do seu trabalho implica a entrega de uma das versões utilizáveis e/ou incrementais do produto desenvolvido.

O objetivo do "Time Scrum" é potencializar a utilização dos melhores processos e técnicas para alcançar o objetivo final ao longo dos eventos descritos no "Guia do Scrum", como os mencionados a seguir.

- Sprint: período do desenvolvimento de uma parte incremental do produto.
- Reunião de Planejamento do Sprint: tem como objetivo definir como o trabalho será realizado para garantir a entrega da parte incremental do produto.
- Reunião Diária: períodos para a atualização dos times com relação aos progressos realizados. Na reunião, devem ser inspecionados possíveis pontos de atenção para as próximas 24 horas.
- Revisão do Sprint: executada ao final do sprint, com o objetivo de verificar se há a necessidade de fazer adaptações no backlog (lista de funcionalidades a serem implementadas) do produto que está sendo desenvolvido.
- Retrospectiva do Sprint: momento de identificar pontos que funcionaram no sprint e quais pontos podem ser melhorados, no relacionamento entre as pessoas, nos processos, no uso de ferramentas e na organização do time em si.

2. Análise das afirmativas

I – Afirmativa correta.

JUSTIFICATIVA. Realmente, uma das funções do Scrum Master é servir com um guia efetivo da metodologia Scrum, garantindo que as equipes atuem de forma produtiva e de acordo com a metodologia estabelecida.

II – Afirmativa correta.

JUSTIFICATIVA. O Product Owner representa o cliente final, seus desejos e seus interesses. Por representar o cliente e estar bastante familiarizado com as suas necessidades, ele deve ter conhecimento focado no negócio do cliente, que pode ser bastante diferente dos conhecimentos técnicos da área de desenvolvimento de software. Dessa forma, ele pode ajudar a balancear o conhecimento de uma equipe, que, sem o seu auxílio, poderia ter a tendência em focar apenas em alguns aspectos tecnológicos do software em si, e não nas necessidades do cliente final.

III – Afirmativa incorreta.

JUSTIFICATIVA. Apenas o Scrum Master tem autoridade direta sobre a equipe. O Product Owner tem o papel de representar o cliente, auxiliando a equipe do projeto. Contudo, a autoridade direta sobre a equipe ainda é do Scrum Master.

Alternativa correta: C.

3. Indicações bibliográficas

- BECK, K. *et al*. *Manifesto para Desenvolvimento Ágil de Software*. Disponível em <http://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em 19 mar. 2026.
- SCHWABER, K; SUTHERLAND, J. *Guia do Scrum – Um guia definitivo para o Scrum: as regras do jogo*. Disponível em <https://scrumguides.org/docs/scrumguide/v1/ Scrum-Guide-Portuguese-BR.pdf>. Acesso em 19 mar. 2026.
- STOPA, G. R.; RACHID, C. L. Scrum: Metodologia ágil como ferramenta de gerenciamento de projetos. *CES Revista*, [S.l.], v. 33, n. 1, p. 302-323, ago. 2019. Disponível em <http://seer.uniacademia.edu.br/index.php/cesRevista/article/view/ 2026>. Acesso em 19 mar. 2026.
- SUTHERLAND, J.; SUTHERLAND, J. J. *Scrum: The art of doing twice the work in half the time*. New York: Crown Currency, 2014.

Questão 5

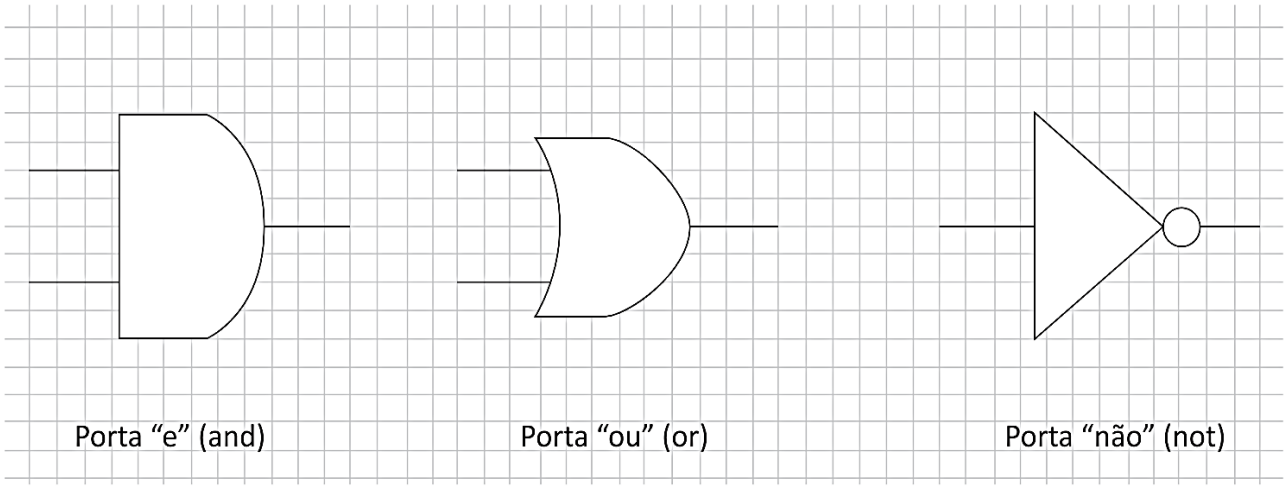
Questão 5.[5]

Leia o texto a seguir.

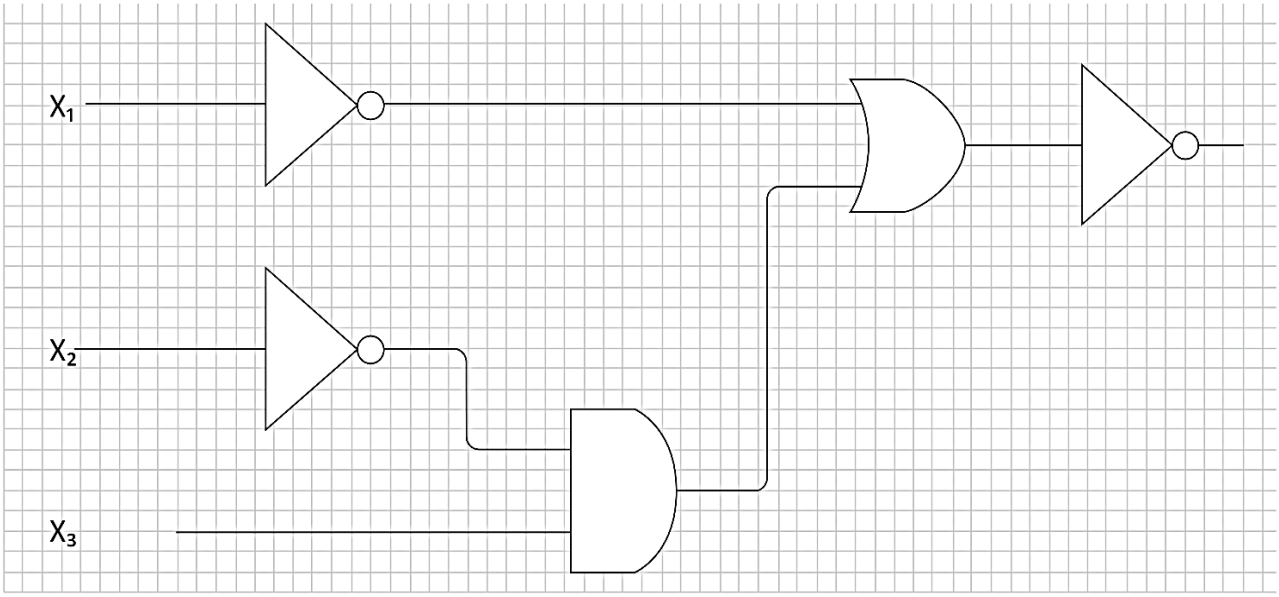
Em 1938, o matemático americano Claude Shannon notou o paralelismo entre a lógica proposicional e a lógica dos circuitos e percebeu que a álgebra booleana teria um papel importante na sistematização deste ramo da eletrônica. Cada um dos conetivos básicos da lógica são instâncias das operações básicas da álgebra booleana ("+", "." e " ' "). Expressões booleanas combinando operações e variáveis podem ser usadas para representar circuitos combinacionais formados por portas lógicas.

GERSTING, J. L. *Mathematical Structures for Computer Science*. New York: W. H. Freeman and Company, 2002.

A figura a seguir apresenta as portas básicas.



A partir das informações apresentadas, considere o circuito combinacional da figura a seguir.



[5]Questão 16 – Enade 2021.

Qual das alternativas apresenta a expressão booleana correspondente?

A. (X3 . X2′) + X1′

B. (X3 . (X2′) + (X1′))′

C. ((X3 . X2)′ + X1′)′

D. (X3 . X2)′ + X1′

E. ((X3 . X2′)′ + X1′)′

1. Introdução teórica

1.1. Claude Shannon

Claude Shannon (1916-2001) foi um matemático e engenheiro eletricista norte-americano, considerado o "pai da teoria da informação". O trabalho dele teve impacto significativo nas áreas de matemática aplicada, telecomunicações (inclusive nas redes de computadores), engenharia elétrica e ciência da computação. Entre as contribuições de Shannon estão a "Teoria da Informação" de 1948, o desenvolvimento da álgebra booleana e a teoria dos códigos de erros em transmissões de dados.

1.2. Eletrônica digital

É uma área da eletrônica que lida com sistemas que utilizam sinais discretos, geralmente representados por números binários (0 e 1). A eletrônica digital desempenha papel crucial em uma variedade de dispositivos modernos, desde smartphones e computadores pessoais até sistemas embarcados, automóveis e muitos outros dispositivos eletrônicos que usamos no dia a dia. Ela envolve o estudo, o design (projeto) e a implementação de circuitos eletrônicos que processam informações de maneira digital, em oposição à eletrônica analógica, que lida com sinais contínuos.

Nos primórdios da eletrônica, todos os problemas eram solucionados por meio de sistemas analógicos. Com o avanço da tecnologia, os problemas passaram a ser solucionados pela eletrônica digital, com a utilização do sistema de numeração binária (que utiliza a base 2) para representar dados. É importante saber que existem outros sistemas possíveis de numeração, como o octal (que utiliza a base 8) e o hexadecimal (que utiliza a base 16), e que esses outros sistemas também são utilizados em muitas aplicações. A conversão entre as

bases 2, 8 e 16 é bastante simples, e calculadoras e programas de computador fazem essa conversão de forma rápida.

Na numeração binária, as informações são representadas por meio de dígitos binários, conhecidos como bits. Cada bit pode ter um valor de 0 ou 1, sendo que 0 representa um valor falso ou desligado, e 1 representa um valor verdadeiro ou ligado. As combinações desses bits (zeros e uns) permitem a representação de números, caracteres e outros tipos de dados.

Nesse contexto, os circuitos digitais são projetados para realizar operações lógicas e aritméticas usando sinais digitais. Assim, elementos de comunicação e processamento, como computadores, processadores de dados, sistemas de controle, codificadores e decodificadores, empregam um pequeno grupo de circuitos lógicos básicos, que são blocos fundamentais dessa infraestrutura. Entre os exemplos, estão as portas lógicas, os flip-flops e os registradores.

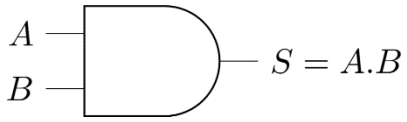
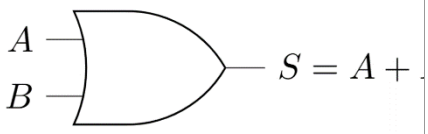
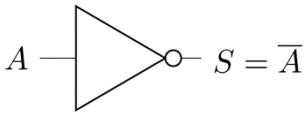
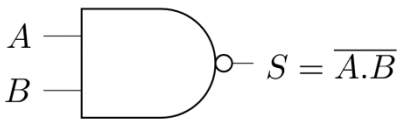
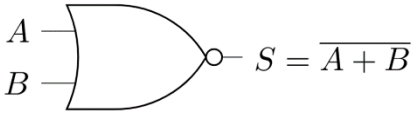
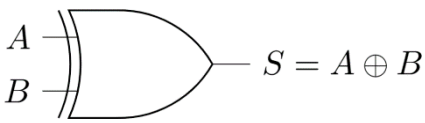
Uma porta digital na álgebra booleana refere-se a um componente eletrônico ou um bloco de construção lógico que realiza uma operação booleana específica. A lógica digital é um aspecto fundamental da eletrônica digital e envolve o projeto de circuitos lógicos que realizam operações booleanas, como as portas lógicas AND (E lógico), OR (OU lógico) e NOT (NÃO lógico ou inversão). Existem vários tipos de portas digitais, cada uma correspondendo a uma operação lógica específica (ou a um conjunto dessas operações). As portas digitais são fundamentais na construção de circuitos lógicos e na implementação de funções booleanas.

Vamos descrever brevemente algumas dessas portas básicas.

- Porta AND (E): produz uma saída verdadeira (1) apenas quando todas as entradas são verdadeiras (1). A representação simbólica é geralmente um ponto de multiplicação (.).
- Porta OR (OU): produz uma saída verdadeira (1) se pelo menos uma das entradas for verdadeira (1). A representação simbólica é um sinal de adição (+).
- Porta NOT (NÃO): inverte o valor de sua única entrada. Se a entrada é verdadeira (1), a saída é falsa (0) e vice-versa. Existem várias representações gráficas para essa operação, entre elas podemos citar o uso uma linha curta sobre a entrada, o uso do símbolo de apóstrofo ('), de um traço acima da operação ($\bar{}$) ou ainda do sinal gráfico til (~) antes de uma variável ou expressão lógica.
- Porta XOR (OU Exclusivo): produz uma saída verdadeira (1) quando o número ímpar de entradas é verdadeiro. A representação simbólica é geralmente um sinal de adição ($\oplus$) com um círculo ao redor.

As tabelas verdade resultantes da aplicação de valores às portas básicas são apresentadas na tabela 1.

Tabela 1. Resumo dos blocos lógicos

Nome	Símbolo Gráfico	Função Algébrica e Tabela Verdade
E (AND)		Entradas: A, B; Saída: S = A.B 0 0 0 0 1 0 1 0 0 1 1 1
OU (OR)		Entradas: A, B; Saída: S = A + B 0 0 0 0 1 1 1 0 1 1 1 1
NÃO (NOT) Inversor		Entrada: A; Saída: S = Ā 0 1 1 0
NÃO-E (NAND)		Entradas: A, B; Saída: S = A.B 0 0 1 0 1 1 1 0 1 1 1 0
NÃO-OU (NOR)		Entradas: A, B; Saída: S = A + B 0 0 1 0 1 0 1 0 0 1 1 0
OU-EXCLUSIVO XOR		Entradas: A, B; Saída: S = A ⊕ B 0 0 0 0 1 1 1 0 1 1 1 0

Baranauskas, 2015 (com adaptações).

3. Resolução da questão

Vamos solucionar a questão, partindo do circuito apresentado no enunciado. Observe as figuras seguintes, com a estrutura passo a passo para a obtenção da expressão booleana correspondente.

Para resolver esse tipo de problema, que envolve circuitos digitais, é conveniente escrevermos a expressão lógica na saída de cada uma das portas lógicas do circuito, começando pelas entradas (no caso, elas estão na parte mais à esquerda da figura), passando por cada uma das portas lógicas intermediárias, chegando até a saída do circuito (no lado mais à direita do circuito).

Na figura 1, observamos a primeira etapa da resolução, com a expressão lógica na saída das primeiras portas lógicas encontradas pelos sinais de entrada X_1, X_2 e X_3.

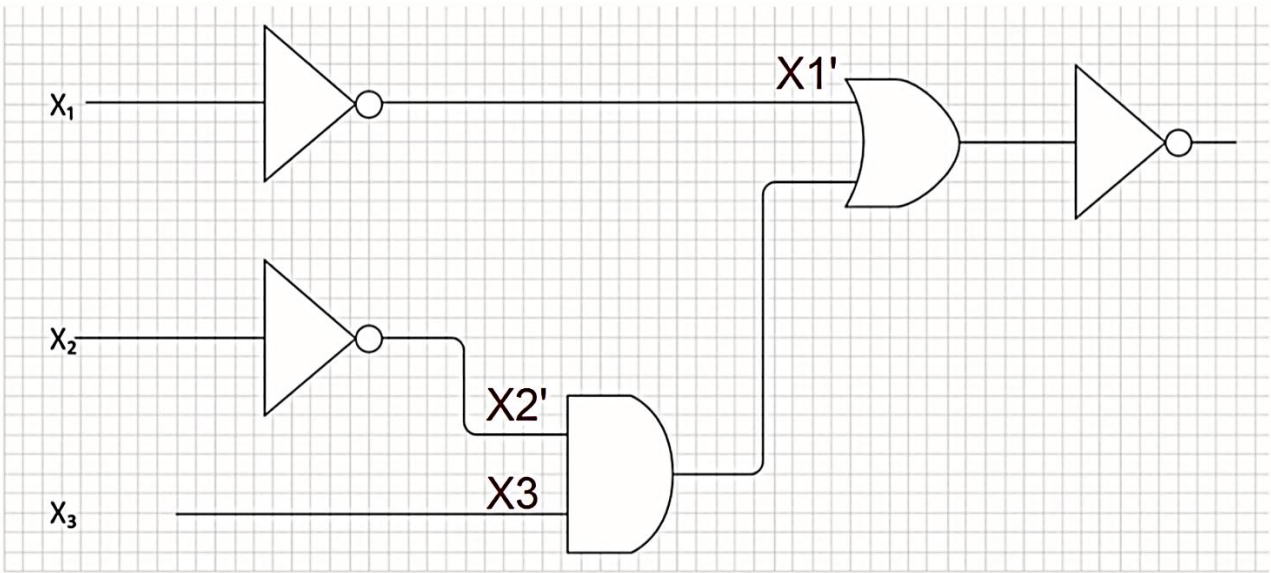


Figura 1. Inversão de X1, na parte superior da figura. Na parte inferior, observe a inversão de X2, que é enviado para uma operação AND com X3.
Enade 2021 – Ciência da Computação. Disponível em <https://download.inep.gov.br/enade/provas_e_gabaritos/2021_PV_bacharelado_ciencia_computacao.pdf>. Acesso em 19 mar. 2026 (com adaptações).

Na figura 2, temos as duas expressões lógicas que servirão de entrada para a porta lógica OU, mostradas na parte superior da figura.

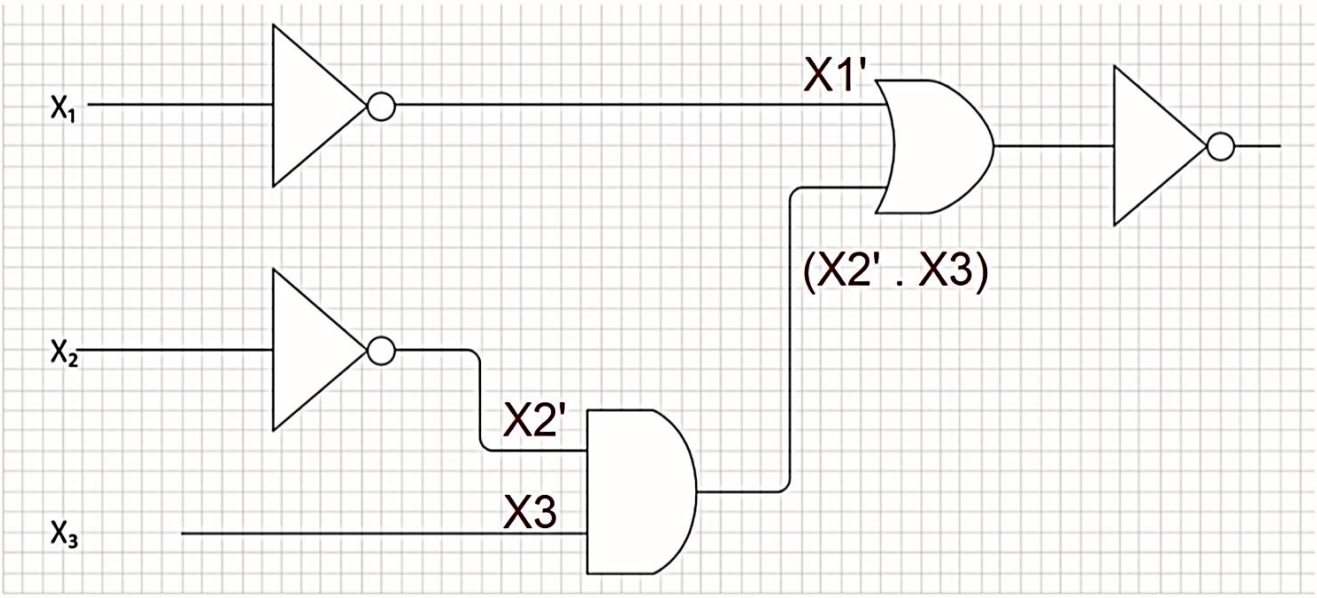


Figura 2. O resultado da operação (X2'. X3) é enviado para uma operação OR com X1'.
Enade 2021 – Ciência da Computação. Disponível em <https://download.inep.gov.br/enade/provas_e_gabaritos/2021_PV_bacharelado_ciencia_computacao.pdf>. Acesso em 19 mar. 2026 (com adaptações).

Finalmente, na figura 3, temos o resultado da operação de OU lógico seguida de uma inversão, que corresponde à última porta lógica do circuito da questão.

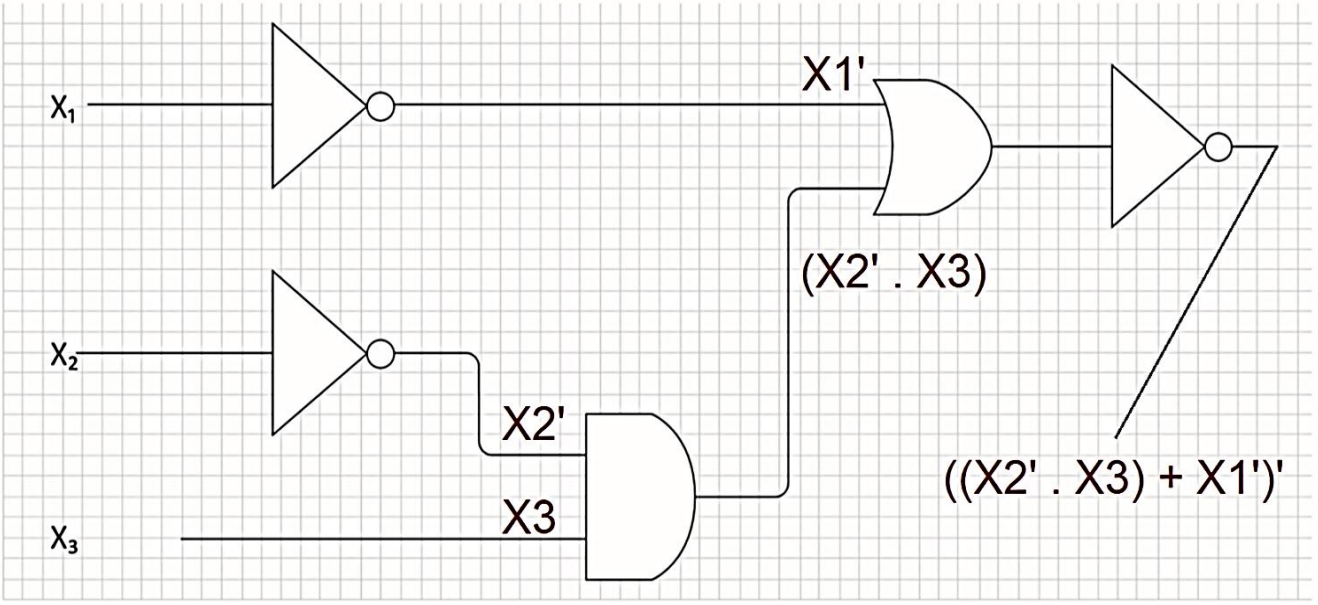


Figura 3. A expressão final, após o último inversor.
Enade 2021 – Ciência da Computação. Disponível em <https://download.inep.gov.br/enade/provas_e_gabaritos/2021_PV_bacharelado_ciencia_computacao.pdf>. Acesso em 19 mar. 2026 (com adaptações).

2. Análise das alternativas

A - Alternativa incorreta.

JUSTIFICATIVA. Observe a ausência do último conector NOT na expressão apresentada, que, por esse motivo, é incorreta.

B - Alternativa correta.
JUSTIFICATIVA. Observe que a expressão apresentada na alternativa corresponde à expressão previamente identificada na figura 3.

C - Alternativa incorreta.
JUSTIFICATIVA. A relação entre X3 e X2 está expressa de forma equivocada, já que apenas X2 deveria ser negada na porta lógica "E". Na expressão, a porta lógica "E" interna que envolve X2 e X3 é erroneamente negada.

D - Alternativa incorreta.
JUSTIFICATIVA. Falta a representação do conector NOT no fim da expressão, o que inverte o resultado.

E - Alternativa incorreta.
JUSTIFICATIVA. A porta lógica "E" que recebe os conectores X3 e X2 não deveria ser negada na expressão.

3. Indicações bibliográficas

- BARANAUSKAS, J. A. *Funções lógicas e portas lógicas*. Departamento de Computação e Matemática – FFCLRP-USP, 2015. Disponível em <https://dcm.ffclrp.usp.br/~augusto/teaching/aba/AB-Funcoes-Logicas-Portas-Logicas.pdf>. Acesso em 19 mar. 2025.
- IDOETA, I. V.; CAPUANO, F. G. *Elementos de eletrônica digital*. 12. ed. São Paulo: Érica, 1987.
- MENDELSON, E. *Álgebra booleana e circuitos de chaveamento*. São Paulo: McGraw-Hill, 1977.

ÍNDICE REMISSIVO