PRATICANDO

MODELAGEM DE SISTEMAS EM UML

Apresentação

Praticar é fundamental para o seu aprendizado. Sentir-se desafiado, lidar com a frustração e aplicar conceitos são essenciais para fixar conhecimentos. No ambiente Praticando, você terá a oportunidade de enfrentar desafios específicos e estudos de caso, criados para ampliar suas competências e para a aplicação prática dos conhecimentos adquiridos.

Objetivo

Ampliar competências e consolidar conhecimentos através de desafios específicos e estudos de caso práticos.

Falhas na Modelagem de Requisitos em Projetos de Software

Caso Prático

Carlos, um desenvolvedor sênior, está liderando um projeto de desenvolvimento de software para uma empresa de tecnologia em rápido crescimento. O projeto visa a criação de um sistema de gestão integrado, utilizando a abordagem orientada a objetos e a Linguagem Unificada de Modelagem (UML) para representar os requisitos e soluções. Durante a fase de levantamento de requisitos, Carlos percebeu que a equipe estava enfrentando dificuldades em capturar e traduzir as necessidades dos usuários finais em modelos adequados. Essas dificuldades resultaram em falhas na comunicação entre a equipe de desenvolvimento e os stakeholders, levando a atrasos no cronograma e aumento dos custos do projeto. O problema ocorre durante a fase de levantamento de requisitos, que é crucial para o sucesso do projeto, e está sendo vivenciado na sede da empresa, localizada em São Paulo. A raiz do problema parece estar na falta de compreensão das técnicas de modelagem e na ausência de uma ferramenta eficaz para validar e confirmar os modelos junto aos usuários finais.

Com base na situação apresentada, analise as possíveis causas das falhas na comunicação entre a equipe de desenvolvimento e os stakeholders. Proponha uma solução que inclua a escolha de técnicas e ferramentas adequadas para melhorar a captura e a tradução dos requisitos em modelos UML, considerando as fases de desenvolvimento do sistema.

Chave de resposta

A principal causa das falhas na comunicação entre a equipe de desenvolvimento e os stakeholders pode ser atribuída à falta de clareza na utilização das técnicas de levantamento de requisitos e na escolha inadequada das ferramentas de modelagem. As técnicas tradicionais, como entrevistas e observações, podem não ter sido suficientes para captar as necessidades dos usuários de forma clara e estruturada, resultando em modelos imprecisos. Além disso, a ausência de uma ferramenta eficaz para validar esses modelos com os usuários finais pode ter agravado o problema, levando a mal-entendidos e retrabalho.

Uma solução eficaz seria aprimorar a aplicação das técnicas de levantamento de requisitos, como entrevistas mais detalhadas, questionários estruturados (formulários) e sessões de brainstorming com os stakeholders. A entrevista, quando bem conduzida, pode ajudar a extrair informações essenciais que podem ter sido omitidas inicialmente. Já o uso de formulários permite uma coleta de dados mais padronizada e direta, ajudando a identificar requisitos específicos. As sessões de brainstorming, por sua vez, podem estimular a criatividade e gerar uma discussão mais ampla sobre as necessidades do sistema, promovendo a participação ativa de todos os envolvidos. Essas técnicas, em conjunto, ajudariam a captar os requisitos de maneira mais precisa e a evitar mal-entendidos. Além disso, a validação desses requisitos pode ser facilitada com o uso de modelos UML durante as reuniões, garantindo que todos os stakeholders compreendam claramente o que está sendo proposto e permitam ajustes antes de prosseguir com o desenvolvimento.

Para saber mais sobre esse conteúdo, acesse:

Tema: Conceitos Básicos de Modelagem de Sistemas

Conceitos Básicos de Modelagem de Sistemas

Desafio 1

Imagine que você foi contratado como Analista de Sistemas em uma empresa de tecnologia para trabalhar no desenvolvimento de uma nova plataforma de e-commerce. Durante uma reunião inicial com a equipe de desenvolvimento, o gerente de projetos destaca a importância de utilizar modelos para orientar a criação do sistema. Ele afirma que esses modelos são fundamentais para a compreensão e estruturação do projeto, garantindo que todas as necessidades sejam atendidas de forma eficaz.

Diante dessa situação, qual das opções a seguir melhor descreve o propósito dos modelos no contexto do desenvolvimento de sistemas?

A

Os modelos são representações detalhadas da interface gráfica que guiam a programação do sistema, garantindo que o design seja implementado corretamente.

B

Os modelos são abstrações da realidade que ajudam a antecipar o comportamento do sistema e a gerenciar a complexidade do projeto.

C

Os modelos são ferramentas exclusivas para documentar as especificações técnicas e requisitos funcionais do sistema após a sua implementação.

D

Os modelos são esquemas que descrevem exclusivamente a estrutura de banco de dados, permitindo uma melhor organização dos dados.

E

Os modelos são diagramas de fluxo que mostram o processo de desenvolvimento do software, desde o início até a entrega final do projeto.

A alternativa B está correta.

A) Incorreta. Embora a interface gráfica seja um componente importante no desenvolvimento de sistemas, os modelos não se limitam a representar apenas o design da interface. O objetivo principal dos modelos é proporcionar uma visão geral e abstrata do sistema como um todo, permitindo que os desenvolvedores compreendam melhor as necessidades e funcionalidades que o sistema deve atender.

B) Correta. Os modelos no contexto do desenvolvimento de sistemas são representações abstratas que permitem antecipar a existência e o comportamento do sistema em construção. Eles ajudam a gerenciar a complexidade inerente ao desenvolvimento de sistemas complexos, fornecendo uma maneira estruturada de visualizar diferentes aspectos do sistema, como funcionalidades, estrutura de dados e interações. Essa abstração é essencial para a tomada de decisões durante o projeto e para garantir que todos os membros da equipe tenham uma compreensão clara e unificada do que está sendo construído.

C) Incorreta. A afirmação de que os modelos são utilizados exclusivamente após a implementação do sistema não é precisa. Os modelos são ferramentas usadas desde as fases iniciais do desenvolvimento para planejar, compreender e validar as decisões de projeto. Eles ajudam a garantir que o sistema atenda às expectativas e requisitos dos usuários antes mesmo de começar a ser codificado.

D) Incorreta. Embora a modelagem de banco de dados seja uma parte do processo, os modelos de sistemas abrangem muito mais do que apenas a organização dos dados. Eles incluem a representação de funcionalidades, interações, e até mesmo o comportamento do sistema em diferentes cenários.

E) Incorreta. Diagrama de fluxo é apenas um dos muitos tipos de modelos que podem ser usados durante o desenvolvimento de software, mas os modelos em si não se restringem a diagramas de fluxo de processo. Eles são usados para representar diferentes aspectos do sistema sob diferentes perspectivas, sendo cruciais para a compreensão e planejamento do projeto em suas várias fases.

Para saber mais sobre esse conteúdo, acesse:

Módulo 1

O QUE SÃO MODELOS E PARA QUE ELES SERVEM

Trecho: “Modelo: representação abstrata e simplificada da realidade. Uma realidade pode demandar diferentes modelos, dependendo da perspectiva com que precise ser observada. Finalidade principal: antecipar a existência de uma realidade de forma a avaliar sua estrutura e comportamento. Modelos são capazes de revelar as propriedades essenciais de um sistema, ajudando no processo de abstração (concentração nos pontos de interesse) e permitindo que foquem no que é relevante.”

Desafio 2

Você é um desenvolvedor de software em uma empresa que está iniciando um novo projeto de um sistema de gestão para uma cadeia de supermercados. Durante a fase inicial, o gerente de projetos enfatiza a importância de seguir um processo de desenvolvimento de sistemas dividido em fases, para assegurar que todos os aspectos do sistema sejam bem compreendidos e implementados com precisão. Diante dessa situação, é crucial entender como o desenvolvimento em fases auxilia na criação de sistemas complexos.

Com base nesse cenário, qual das opções a seguir melhor descreve como o processo de desenvolvimento de sistemas em fases contribui para a construção de um sistema eficiente e alinhado às necessidades dos usuários?

A

Ele permite que cada fase do desenvolvimento foque exclusivamente na codificação do sistema, garantindo que o código seja produzido de forma contínua e sem interrupções.

B

Ele assegura que todas as fases sejam executadas simultaneamente, permitindo uma rápida entrega do sistema, embora sem uma estrutura de validação entre as etapas.

C

Ele organiza o desenvolvimento em fases distintas que vão da compreensão inicial das necessidades até a implementação, diminuindo a complexidade e aumentando o entendimento do sistema ao longo do tempo.

D

Ele elimina a necessidade de revisão durante o processo, pois cada fase já considera todas as possíveis variações do sistema.

E

Ele garante que o desenvolvimento seja linear, sem a necessidade de revisitar fases anteriores, o que acelera o processo de entrega do sistema.

A alternativa C está correta.

A) Incorreta. Focar exclusivamente na codificação em cada fase não reflete a realidade do desenvolvimento em fases. O desenvolvimento de sistemas em fases é caracterizado por uma abordagem estruturada onde diferentes aspectos do sistema são tratados em diferentes momentos, desde o levantamento de requisitos até o design, codificação, testes e implementação. Cada fase tem um propósito distinto e não se trata apenas de produzir código continuamente.

B) Incorreta. Executar todas as fases simultaneamente pode resultar em um sistema mal estruturado, onde o entendimento e a validação de cada fase ficam comprometidos. O processo de desenvolvimento em fases pressupõe uma sequência lógica onde cada fase constrói a base para a próxima, permitindo revisões e validações constantes.

C) Correta. O desenvolvimento em fases é uma abordagem que ajuda a organizar o processo de criação de sistemas complexos. Ele permite que os desenvolvedores comecem com um alto nível de abstração e, à medida que avançam por cada fase, vão detalhando o sistema, diminuindo a complexidade e aumentando o entendimento do que precisa ser implementado. Isso garante que o sistema atenda aos requisitos dos usuários e que qualquer problema seja identificado e corrigido antes de avançar para a próxima fase.

D) Incorreta. A eliminação da necessidade de revisão durante o processo não faz parte do desenvolvimento em fases. Pelo contrário, a revisão e validação são fundamentais em cada fase para garantir que o sistema esteja evoluindo corretamente e atendendo às necessidades do projeto.

E) Incorreta. O desenvolvimento linear e sem revisitar fases anteriores pode ser arriscado e ineficiente. Muitas vezes, é necessário voltar a fases anteriores para ajustar requisitos, design ou outros aspectos conforme novas informações surgem ou problemas são identificados. O processo em fases permite essa flexibilidade e garante a qualidade do sistema final.

Para saber mais sobre esse conteúdo, acesse:

Módulo 1

O PROCESSO DE DESENVOLVIMENTO DE SISTEMAS EM FASES

Trecho: “O desenvolvimento de sistemas computacionais é um processo que envolve pessoas e a necessidade de compreensão de uma realidade muitas vezes complexa e obscura, principalmente no início do desenvolvimento, quando o nível de abstração é alto e pouco se conhece da realidade. Conforme as fases em que o processo de desenvolvimento é particionado se sucedem, o conhecimento sobre o sistema aumenta, diminuindo, consequentemente, o nível de abstração da realidade. Essa complexidade aumenta na medida em que o tamanho do sistema cresce, requerendo um maior planejamento dos recursos a serem usados. O principal recurso no desenvolvimento de um sistema são pessoas, profissionais capacitados.”

Desafio 3

Você foi recentemente contratado por uma empresa de desenvolvimento de software que está migrando seus sistemas de uma arquitetura tradicional baseada em procedimentos para um modelo orientado a objetos. Durante uma reunião de alinhamento, o gerente de desenvolvimento explica que essa mudança trará benefícios significativos em termos de organização, manutenção e reutilização do código. Como desenvolvedor, é essencial que você compreenda os fundamentos do paradigma orientado a objetos e suas vantagens para que possa contribuir de maneira efetiva nesse processo de transição.

Com base nessa situação, qual das alternativas a seguir melhor descreve um dos princípios fundamentais do paradigma orientado a objetos e sua aplicação prática?

A

O paradigma orientado a objetos enfatiza a divisão do sistema em funções independentes, garantindo que cada função seja reutilizada em diferentes partes do código sem a necessidade de criação de objetos.

B

A orientação a objetos permite que as propriedades e comportamentos comuns sejam encapsulados em objetos, que servem como instâncias de classes, promovendo a organização e reutilização de código.

C

O paradigma orientado a objetos organiza o sistema em um conjunto de instruções sequenciais, garantindo que o fluxo do programa siga uma ordem linear e fácil de seguir.

D

A orientação a objetos elimina a necessidade de hierarquia no sistema, uma vez que todos os objetos são tratados de maneira isolada, sem relação com outros objetos ou classes.

E

O paradigma orientado a objetos impede a herança de atributos e métodos, obrigando os desenvolvedores a codificar todos os comportamentos do zero para cada novo objeto.

A alternativa B está correta.

A) Incorreta. Embora a divisão do sistema em funções seja uma prática comum em programação procedural, o paradigma orientado a objetos se concentra em encapsular dados e comportamentos dentro de objetos que pertencem a classes. As funções independentes não são a ênfase principal; em vez disso, a reutilização e a organização do código são alcançadas através da criação de classes e objetos.

B) Correta. No paradigma orientado a objetos, classes servem como moldes para criar objetos, que são instâncias dessas classes. Cada classe encapsula dados (atributos) e comportamentos (métodos) que são comuns a todos os objetos da classe. Isso permite uma organização clara e consistente do código, além de facilitar a reutilização e manutenção, já que mudanças em uma classe afetam todos os objetos derivados dela.

C) Incorreta. A organização em um conjunto de instruções sequenciais é característica da programação procedural, não da orientação a objetos. Na programação orientada a objetos, o foco está na interação entre objetos que realizam tarefas através de métodos, e não em seguir uma sequência linear de instruções.

D) Incorreta. A orientação a objetos faz uso intensivo de hierarquias, principalmente através do conceito de herança, onde classes podem derivar de outras classes, herdando seus atributos e métodos. A relação entre objetos e classes é fundamental para promover a reutilização e a extensão do código.

E) Incorreta. Um dos princípios-chave da orientação a objetos é a herança, que permite que classes filhas herdem atributos e métodos de suas classes pai, evitando a repetição de código e facilitando a manutenção e a expansão do sistema. Impedir a herança seria contrário aos benefícios que o paradigma orientado a objetos oferece.

Para saber mais sobre esse conteúdo, acesse:

Módulo 2

PARADIGMA ORIENTADO A OBJETOS

Trecho: “A orientação a objetos surge como solução a problemas de organização, manutenção e reuso de código, permitindo maior organização, reaproveitamento e extensibilidade de código e, consequentemente, programas mais fáceis de serem escritos e mantidos. O paradigma da orientação a objetos visualiza um sistema de software como uma coleção de agentes interconectados chamados objetos. Cada um deles é responsável por realizar tarefas específicas e, para cumprir com algumas das tarefas sob sua responsabilidade, um objeto pode ter que interagir com outros. É pela interação entre eles que uma tarefa computacional é executada.”

Desafio 4

Você é um desenvolvedor de software em uma equipe responsável pela manutenção e expansão de um sistema de gerenciamento escolar. Durante a implementação de novas funcionalidades, seu gerente de

projeto sugere a aplicação dos conceitos de herança e polimorfismo para maximizar a reutilização de código e facilitar a manutenção. Ele menciona que essas técnicas são cruciais para criar um sistema eficiente e flexível.

Com base nesse cenário, qual das opções a seguir melhor exemplifica o uso de herança e polimorfismo no contexto de um sistema orientado a objetos?

A

Criar uma classe base chamada "Aluno" com métodos para calcular notas e, a partir dela, derivar classes como "AlunoGraduacao" e "AlunoPosGraduacao" que herdam esses métodos, mas reimplementam o cálculo de acordo com regras específicas de cada nível de ensino.

B

Criar várias classes independentes como "AlunoGraduacao" e "AlunoPosGraduacao", sem relação entre elas, e duplicar os métodos de cálculo de notas, para que cada uma tenha seu próprio código exclusivo.

C

Criar uma classe base chamada "Escola" que contém todos os métodos e atributos necessários para gerenciar alunos, professores e turmas, sem a necessidade de classes derivadas para especializar comportamentos.

D

Criar uma única classe "Aluno" sem herança ou polimorfismo, contendo todos os métodos possíveis para qualquer tipo de aluno, e selecionar qual método usar com base em condicionais no código.

E

Criar uma classe base chamada "Nota" que define os métodos de cálculo para diferentes tipos de avaliações, mas impedir que qualquer classe derivada modifique esses métodos, mantendo-os fixos em todas as subclasses.

A alternativa A está correta.

A) Correta. Esta alternativa exemplifica claramente o uso da herança e polimorfismo no contexto orientado a objetos. A classe base "Aluno" encapsula atributos e métodos comuns a todos os tipos de alunos, como o cálculo de notas. As classes derivadas "AlunoGraduacao" e "AlunoPosGraduacao" herdam essas funcionalidades e têm a capacidade de redefinir (polimorfismo) os métodos herdados para adaptar o cálculo de notas às especificidades de cada tipo de aluno. Isso promove a reutilização de código, pois evita duplicação, e aumenta a flexibilidade do sistema, permitindo fácil manutenção e expansão.

B) Incorreta. Criar classes independentes sem herança resulta na duplicação de código, que vai contra os princípios de reutilização e eficiência promovidos pelo paradigma orientado a objetos. Isso dificulta a manutenção, pois qualquer alteração no comportamento comum exigirá mudanças em múltiplos locais, aumentando o risco de inconsistências e erros.

C) Incorreta. Agrupar todos os métodos e atributos necessários para diferentes entidades (alunos, professores, turmas) em uma única classe base viola o princípio da responsabilidade única e pode levar a um design de código complexo e difícil de manter. A herança permite que funcionalidades comuns sejam

corretamente abstraídas e organizadas em hierarquias de classes, o que melhora a modularidade e reutilização.

D) Incorreta. A utilização de uma única classe "Aluno" com condicionais para determinar qual método usar é uma abordagem procedimental, não orientada a objetos. Isso resulta em um código difícil de manter e entender, pois não aproveita os benefícios da herança e polimorfismo, que permitem um design mais limpo e organizado, com cada classe lidando apenas com as responsabilidades relevantes ao seu contexto.

E) Incorreta. Impedir que classes derivadas modifiquem os métodos herdados vai contra o princípio do polimorfismo, que é justamente permitir que esses métodos sejam redefinidos para adaptar comportamentos conforme necessário. A flexibilidade e a extensibilidade do sistema seriam prejudicadas, reduzindo as vantagens oferecidas pela herança.

Para saber mais sobre esse conteúdo, acesse:

Módulo 2

OS PILARES DA ORIENTAÇÃO A OBJETOS

Trecho: "Mecanismo para derivar novas classes a partir da definição de classes existentes, com base em um processo de refinamento. Uma classe derivada ou descendente herda os dados (atributos) e o comportamento (métodos) da classe base ou ancestral ou ascendente. A implementação da herança garante a reutilização de código, que, além de economizar tempo e dinheiro, propicia mais segurança ao processo de desenvolvimento, posto que as funcionalidades da classe base podem ter sido usadas e testadas. [...] Permitindo que [o polimorfismo] se aproveite alguns métodos e se altere (redefina) outros. Dessa forma, um método com mesmo nome em classes distintas pode ter diferentes comportamentos.".

Desafio 5

Você foi designado para liderar uma equipe de desenvolvimento em um projeto que utiliza a UML (Unified Modeling Language) como a principal ferramenta de modelagem de sistemas. Durante uma reunião com sua equipe, é fundamental que você esclareça as funcionalidades e o propósito da UML, garantindo que todos estejam alinhados na utilização dessa linguagem de modelagem.

Considere as afirmativas abaixo sobre a UML:

I. A UML é uma linguagem de modelagem padronizada que pode ser utilizada em diferentes metodologias de desenvolvimento de sistemas, independentemente da tecnologia empregada.

II. A UML define a ordem e a obrigatoriedade dos diagramas que devem ser utilizados em cada fase do desenvolvimento do sistema.

III. A UML permite a construção de modelos que representam as partes essenciais de um sistema, abrangendo diferentes perspectivas, como lógica, implementação e implantação.

Qual das alternativas a seguir está correta?

A

Somente a afirmativa I está correta.

B

Somente as afirmativas I e II estão corretas.

C

Somente as afirmativas I e III estão corretas.

D

Somente a afirmativa II está correta.

E

Todas as afirmativas estão corretas.

A alternativa C está correta.

A) Incorreta. Embora a afirmativa I esteja correta, a alternativa não considera a veracidade da afirmativa III, que também está correta.

B) Incorreta. A afirmativa II está incorreta, pois a UML não define a ordem ou a obrigatoriedade dos diagramas a serem utilizados em cada fase do desenvolvimento do sistema. A UML é flexível e pode ser adaptada conforme as necessidades do projeto e a metodologia de desenvolvimento adotada.

C) Correta. A afirmativa I está correta ao afirmar que a UML é uma linguagem de modelagem padronizada e independente de tecnologia, aplicável a diferentes metodologias de desenvolvimento. A afirmativa III também está correta, pois a UML permite a construção de modelos que representam diferentes aspectos do sistema, como a lógica (estrutura e comportamento), implementação (detalhes técnicos) e implantação (distribuição física). Essas afirmativas refletem o verdadeiro propósito e aplicação da UML no desenvolvimento de sistemas.

D) Incorreta. A afirmativa II está incorreta, pois a UML não impõe uma sequência fixa de uso dos diagramas, sendo essa uma decisão que cabe à equipe de desenvolvimento, baseada nas necessidades específicas do projeto.

E) Incorreta. A afirmativa II é incorreta, logo, todas as afirmativas não podem estar corretas. A UML não dita a ordem e obrigatoriedade dos diagramas, o que torna essa alternativa equivocada.

Para saber mais sobre esse conteúdo, acesse:

Módulo 3

O QUE É UM UML, AFINAL?

Trecho: “A UML é, portanto, uma linguagem padrão para construção de projetos de sistemas, voltada para a visualização, a especificação, a construção e a documentação de artefatos de um sistema. Foi projetada para ser independente do método ou processo de desenvolvimento utilizado. [...] A UML não diz quais diagramas usar e nem em que ordem, pois a metodologia de desenvolvimento ditará essa ordem. A UML disponibiliza diagramas sob diferentes visões ou perspectivas.”

UML Para Modelagem do Domínio

Desafio 1

Você foi contratado para gerenciar o desenvolvimento de um novo sistema de gerenciamento de estoque para uma grande rede de supermercados. Durante as reuniões iniciais com os stakeholders, foi discutido que o sistema deve ser capaz de realizar a busca rápida por produtos, mesmo durante horários de pico, e deve garantir que os dados de estoque estejam sempre atualizados em tempo real. Considerando essas exigências, qual dos aspectos abaixo representa um requisito funcional para o sistema que você está desenvolvendo?

A

O sistema deve suportar até 500 usuários simultâneos sem queda de desempenho.

B

O sistema deve permitir que os usuários busquem produtos por nome, código de barras ou categoria.

C

O sistema deve garantir que a resposta a qualquer consulta seja entregue em menos de 2 segundos.

D

O sistema deve ser compatível com diferentes navegadores, como Chrome, Firefox e Safari.

E

O sistema deve criptografar os dados sensíveis dos usuários antes de armazená-los.

A alternativa B está correta.

A) O sistema deve suportar até 500 usuários simultâneos sem queda de desempenho: Incorreta. Esta alternativa descreve um requisito não funcional relacionado ao desempenho do sistema. Requisitos não funcionais são importantes para definir as qualidades do sistema, como a capacidade de suportar múltiplos usuários simultâneos, mas não se referem diretamente a uma funcionalidade específica que o sistema deve oferecer ao usuário.

B) O sistema deve permitir que os usuários busquem produtos por nome, código de barras ou categoria: Correta. Esta alternativa descreve um requisito funcional. Requisitos funcionais são declarações sobre o que o sistema deve fazer, ou seja, as funcionalidades e serviços que ele deve oferecer. A capacidade de

buscar produtos é uma funcionalidade essencial que o sistema deve prover, sendo um exemplo típico de requisito funcional.

C) O sistema deve garantir que a resposta a qualquer consulta seja entregue em menos de 2 segundos: Incorreta. Esta alternativa também se refere a um requisito não funcional, mais especificamente relacionado ao tempo de resposta do sistema. Apesar de ser crucial para a qualidade da experiência do usuário, o tempo de resposta não define uma funcionalidade específica do sistema, mas sim uma característica de desempenho.

D) O sistema deve ser compatível com diferentes navegadores, como Chrome, Firefox e Safari: Incorreta. Este é outro exemplo de um requisito não funcional, focando na compatibilidade do sistema com diferentes plataformas. Embora a compatibilidade seja essencial para a usabilidade do sistema, não descreve uma funcionalidade que o sistema oferece aos usuários.

E) O sistema deve criptografar os dados sensíveis dos usuários antes de armazená-los: Incorreta. Esta alternativa aborda um requisito não funcional relacionado à segurança. A criptografia de dados é uma medida de segurança para proteger informações sensíveis, mas não representa uma funcionalidade que o sistema oferece diretamente aos usuários.

Para saber mais sobre esse conteúdo, acesse:

Módulo 1

REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

“Requisitos funcionais são declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. [...] Requisitos não funcionais são restrições aos serviços ou funções oferecidos pelo sistema, incluindo restrições de timing, restrições no processo de desenvolvimento e restrições impostas pelas normas. Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo.”

Desafio 2

Você foi designado como analista de sistemas em uma empresa de desenvolvimento de software e precisa criar um Diagrama de Casos de Uso para um sistema bancário que permitirá aos clientes realizar diversas operações, como abrir contas, realizar depósitos, saques e consultar saldos. Considerando as melhores práticas para a criação de Diagramas de Casos de Uso, qual das seguintes opções representa corretamente o papel de um ator no diagrama?

A

Um ator no Diagrama de Casos de Uso deve ser representado por uma caixa retangular com o nome da operação.

B

O ator deve ser um componente interno do sistema, como o banco de dados, que executa as operações automaticamente.

C

Um ator é uma entidade externa que interage com o sistema e pode ser um usuário humano, outro sistema ou hardware.

D

Um ator no Diagrama de Casos de Uso deve ser representado por uma linha pontilhada que liga diferentes operações.

E

O ator é uma representação interna do sistema que define as regras de negócio que ele deve seguir.

A alternativa C está correta.

A) Um ator no Diagrama de Casos de Uso deve ser representado por uma caixa retangular com o nome da operação: Incorreta. No Diagrama de Casos de Uso, um ator não é representado por uma caixa retangular, mas sim por um boneco palito, que simboliza uma entidade externa que interage com o sistema. A caixa retangular é usada para representar sistemas ou subsistemas.

B) O ator deve ser um componente interno do sistema, como o banco de dados, que executa as operações automaticamente: Incorreta. Esta alternativa está incorreta porque o ator no Diagrama de Casos de Uso deve ser uma entidade externa ao sistema. Componentes internos, como banco de dados ou módulos de software, não são considerados atores no diagrama; eles fazem parte do sistema.

C) Um ator é uma entidade externa que interage com o sistema e pode ser um usuário humano, outro sistema ou hardware: Correta. Esta é a definição precisa de um ator no Diagrama de Casos de Uso. Um ator representa qualquer entidade externa ao sistema que interage com ele, como um usuário, outro sistema, ou mesmo um dispositivo de hardware.

D) Um ator no Diagrama de Casos de Uso deve ser representado por uma linha pontilhada que liga diferentes operações: Incorreta. Esta alternativa confunde o conceito de ator com o relacionamento entre casos de uso. Um ator é representado por um boneco palito, e os relacionamentos entre atores e casos de uso são representados por linhas cheias, não pontilhadas.

E) O ator é uma representação interna do sistema que define as regras de negócio que ele deve seguir: Incorreta. O ator não representa as regras de negócio nem é uma parte interna do sistema. Ele é sempre uma entidade externa que interage com o sistema, e as regras de negócio são implementadas dentro do próprio sistema.

Para saber mais sobre esse conteúdo, acesse:

Módulo 1

DIAGRAMA DE CASOS DE USO

“O ator é quem utiliza ou interage em um caso de uso, ou seja, é uma entidade externa que interage com o sistema, mas que não faz parte dele. Interagir significa trocar informações com o sistema, seja fornecendo ou recebendo essas informações. [...] O ator pode representar papéis executados por usuários humanos, um componente de hardware, outros sistemas com os quais o sistema em desenvolvimento se comunica. Desse modo, o nome do ator deve expressar esse papel de forma clara.”

Desafio 3

Você está trabalhando como analista de sistemas no desenvolvimento de um software de gestão hospitalar. Durante a modelagem do sistema, identificou que, ao cadastrar um paciente para uma consulta, o sistema precisa verificar se o prontuário do paciente já está registrado e, caso não esteja, criar um novo prontuário antes de agendar a consulta. Considerando as boas práticas de modelagem usando UML, como você representaria essa necessidade de reutilização de uma parte do processo em outros casos de uso no Diagrama de Casos de Uso?

A

Criando um relacionamento de extensão entre o caso de uso "Cadastrar Prontuário" e o caso de uso "Agendar Consulta".

B

Usando um relacionamento de inclusão entre o caso de uso "Cadastrar Prontuário" e o caso de uso "Agendar Consulta".

C

Ligando diretamente o ator "Paciente" ao caso de uso "Cadastrar Prontuário" sem qualquer relacionamento entre os casos de uso.

D

Definindo "Agendar Consulta" como um caso de uso independente, sem qualquer referência ao "Cadastrar Prontuário".

E

Criando um novo caso de uso "Verificar Prontuário" que substitui a necessidade de "Cadastrar Prontuário".

A alternativa B está correta.

A) Criando um relacionamento de extensão entre o caso de uso "Cadastrar Prontuário" e o caso de uso "Agendar Consulta": Incorreta. O relacionamento de extensão é utilizado quando há a necessidade de adicionar um comportamento opcional ou condicional a um caso de uso principal. No caso descrito, "Cadastrar Prontuário" não é uma funcionalidade opcional, mas sim um passo necessário que pode ser reutilizado em vários outros casos de uso, o que caracteriza um relacionamento de inclusão.

B) Usando um relacionamento de inclusão entre o caso de uso "Cadastrar Prontuário" e o caso de uso "Agendar Consulta": Correta. O relacionamento de inclusão é apropriado aqui porque "Cadastrar Prontuário" é uma parte do processo que deve ser realizada dentro de outros casos de uso, como "Agendar Consulta". A inclusão permite que essa funcionalidade seja reutilizada sem duplicação desnecessária de lógica no diagrama.

C) Ligando diretamente o ator "Paciente" ao caso de uso "Cadastrar Prontuário" sem qualquer relacionamento entre os casos de uso: Incorreta. Embora o ator "Paciente" possa estar diretamente ligado ao caso de uso "Cadastrar Prontuário", essa alternativa não resolve o problema da necessidade de reutilização de parte do processo em outros casos de uso, o que é precisamente o que o relacionamento de inclusão pretende abordar.

D) Definindo "Agendar Consulta" como um caso de uso independente, sem qualquer referência ao "Cadastrar Prontuário": Incorreta. Definir "Agendar Consulta" como um caso de uso independente sem referência a "Cadastrar Prontuário" ignora a necessidade de verificar ou criar um prontuário, o que é uma parte essencial do processo e deve ser incluído para garantir que todas as etapas necessárias sejam seguidas.

E) Criando um novo caso de uso "Verificar Prontuário" que substitui a necessidade de "Cadastrar Prontuário": Incorreta. Embora a verificação do prontuário possa ser parte do processo, o "Cadastrar Prontuário" é um caso de uso específico que representa uma funcionalidade distinta. Simplesmente criar um novo caso de uso para verificação não aborda a necessidade de reutilizar o processo de cadastro em outros contextos, como descrito na questão.

Para saber mais sobre esse conteúdo, acesse:

Módulo 1

RELACIONAMENTO DE INCLUSÃO

"Relacionamento de inclusão tem como objetivo principal a reutilização. Suponha que uma parte dos passos de execução de um caso de uso se repita em diversos outros casos de uso. Podemos criar um caso de uso separado e 'incluí-lo' nos demais. A inclusão funciona de forma semelhante à chamada de uma função em um programa. A representação gráfica do relacionamento de inclusão é uma linha pontilhada com o estereótipo <<include>> apontando do caso de uso inclusor para o incluído.".

Desafio 4

Você está trabalhando no desenvolvimento de um sistema de comércio eletrônico que permite aos clientes realizar compras online. Durante a criação do Diagrama de Casos de Uso, identificou que, em algumas compras, o sistema deve validar o limite de crédito do cliente antes de finalizar a transação. No entanto, essa validação só é necessária quando o valor da compra excede um determinado limite. Qual seria a forma mais adequada de modelar essa necessidade de verificação de crédito no Diagrama de Casos de Uso?

A

Utilizando um relacionamento de inclusão entre o caso de uso "Finalizar Compra" e o caso de uso "Validar Limite de Crédito".

B

Criando um relacionamento de generalização entre o caso de uso "Finalizar Compra" e o caso de uso "Validar Limite de Crédito".

C

Usando um relacionamento de extensão entre o caso de uso "Finalizar Compra" e o caso de uso "Validar Limite de Crédito".

D

Definindo "Validar Limite de Crédito" como um caso de uso separado e independente de "Finalizar Compra", sem qualquer ligação entre eles.

E

Adicionando o processo de "Validar Limite de Crédito" diretamente no diagrama, como um requisito funcional do "Finalizar Compra", sem relacioná-los.

A alternativa C está correta.

A) Incorreta. O relacionamento de inclusão é utilizado quando uma funcionalidade precisa ser obrigatoriamente executada em diversos outros casos de uso. No entanto, no cenário descrito, a validação do limite de crédito é opcional, pois só ocorre em determinadas condições, o que caracteriza melhor um relacionamento de extensão.

B) Incorreta. A generalização é usada quando há uma hierarquia entre casos de uso, onde o caso de uso mais específico herda as características do caso de uso mais genérico. Neste caso, a validação do limite de crédito é uma funcionalidade opcional, não uma especialização da finalização da compra.

C) Correta. O relacionamento de extensão é utilizado para representar funcionalidades que são acionadas de maneira condicional, ou seja, em determinadas situações. Como a validação do limite de crédito só ocorre quando o valor da compra ultrapassa um limite específico, a modelagem correta é a extensão do caso de uso "Finalizar Compra" com o "Validar Limite de Crédito".

D) Incorreta. Embora "Validar Limite de Crédito" seja uma funcionalidade separada, ela não deve ser completamente independente, já que é parte de um processo condicional dentro do fluxo de "Finalizar Compra". Ignorar esse relacionamento tornaria o diagrama impreciso em termos de dependências e condições de execução.

E) Incorreta. Adicionar diretamente a funcionalidade no diagrama como um requisito funcional fixo não seria adequado, pois a validação do crédito ocorre apenas sob certas condições. Isso deve ser tratado com uma extensão, já que não é uma ação sempre presente em todas as finalizações de compra.

Para saber mais sobre esse conteúdo, acesse:

Módulo 1

RELACIONAMENTO DE EXTENSÃO

"O relacionamento de extensão tem como objetivo representar situações em que diferentes sequências de passos possam ser inseridas no mesmo caso de uso. Ou seja, dependendo de uma condição, o caso de uso segue um caminho diferente. Desse modo, um caso de uso estende outro, permitindo um comportamento eventual. Note que a existência do caso de uso estendido deve ser independente da existência de quaisquer casos de uso que o estendam.".

Desafio 5

Você foi encarregado de modelar um sistema de gestão de funcionários para uma grande empresa. Durante a análise, identificou que a empresa possui diferentes tipos de funcionários, como vendedores e supervisores. Ambos compartilham algumas características comuns, como nome e matrícula, mas possuem particularidades em seus cargos, como salário fixo para supervisores e comissão por vendas para vendedores. Qual seria a melhor maneira de representar essa estrutura no Diagrama de Casos de Uso?

A

Criando um relacionamento de inclusão entre os casos de uso "Gerenciar Vendedor" e "Gerenciar Supervisor".

B

Usando um relacionamento de generalização entre os casos de uso "Gerenciar Funcionário" e os casos de uso "Gerenciar Vendedor" e "Gerenciar Supervisor".

C

Definindo "Gerenciar Vendedor" e "Gerenciar Supervisor" como casos de uso independentes, sem qualquer relação entre eles.

D

Aplicando um relacionamento de extensão entre os casos de uso "Gerenciar Funcionário" e "Gerenciar Vendedor".

E

Agrupando todos os funcionários sob um único caso de uso "Gerenciar Funcionário", sem distinção entre vendedores e supervisores.

A alternativa B está correta.

A) Criando um relacionamento de inclusão entre os casos de uso "Gerenciar Vendedor" e "Gerenciar Supervisor": Incorreta. O relacionamento de inclusão é apropriado quando uma funcionalidade é reutilizada em outros casos de uso. No entanto, aqui a necessidade é representar uma hierarquia entre casos de uso relacionados, onde existem funcionalidades comuns e outras específicas para cada tipo de funcionário, o que se encaixa melhor no conceito de generalização.

B) Usando um relacionamento de generalização entre os casos de uso "Gerenciar Funcionário" e os casos de uso "Gerenciar Vendedor" e "Gerenciar Supervisor": Correta. O relacionamento de generalização é adequado quando você tem casos de uso que compartilham comportamentos comuns, mas também possuem diferenças específicas. Neste caso, "Gerenciar Funcionário" é o caso de uso genérico, enquanto "Gerenciar Vendedor" e "Gerenciar Supervisor" são casos de uso específicos que herdam características do caso de uso genérico.

C) Definindo "Gerenciar Vendedor" e "Gerenciar Supervisor" como casos de uso independentes, sem qualquer relação entre eles: Incorreta. Esta abordagem não aproveita a estrutura hierárquica que existe entre os diferentes tipos de funcionários. Ignorar o relacionamento de generalização resultaria em uma modelagem redundante, onde funcionalidades comuns teriam que ser repetidas em vários casos de uso, o que não é eficiente.

D) Aplicando um relacionamento de extensão entre os casos de uso "Gerenciar Funcionário" e "Gerenciar Vendedor": Incorreta. A extensão é usada para representar comportamentos opcionais ou adicionais que podem ocorrer em determinados casos, mas neste contexto, o que se busca é uma hierarquia de funcionalidades comuns com especificidades, o que caracteriza melhor o uso de generalização.

E) Agrupando todos os funcionários sob um único caso de uso "Gerenciar Funcionário", sem distinção entre vendedores e supervisores: Incorreta. Ao agrupar todos os funcionários sob um único caso de uso, as particularidades de cada tipo de funcionário seriam perdidas. A modelagem não refletiria a estrutura real do sistema, onde diferentes tipos de funcionários têm diferentes processos e requisitos.

Para saber mais sobre esse conteúdo, acesse:

MÓDULO 1

“Na generalização entre casos de uso, quando um caso de uso herda de outro, significa que as sequências de passos de execução do caso de uso mais genérico (pai) valem também para o mais específico (filho). Além disso, o caso de uso herdeiro participa de qualquer associação do qual o caso de uso mais genérico participe. Isso significa que um ator que interage com o caso de uso pai pode também interagir em qualquer caso de uso filho.”

Utilizando UML para Projetar o Software

Desafio 1

Imagine que você está trabalhando como analista de sistemas em uma empresa de desenvolvimento de software. Durante a fase de levantamento de requisitos, você percebe que a equipe está confusa sobre as etapas seguintes e a importância de cada uma no processo de desenvolvimento. Como analista, você precisa explicar a seus colegas a importância do ciclo de vida de desenvolvimento de software, especialmente destacando o papel da fase de projeto na qualidade do produto final.

Qual das opções abaixo melhor descreve a função da fase de projeto no ciclo de vida de desenvolvimento de software?

A

A fase de projeto é responsável por levantar e documentar todos os requisitos do sistema, garantindo que todas as necessidades dos usuários sejam atendidas antes do início da implementação.

B

A fase de projeto é onde as funcionalidades são testadas exaustivamente para garantir que o sistema esteja livre de bugs antes de ser entregue ao cliente.

C

A fase de projeto é o momento em que a arquitetura do software é definida, incluindo a criação de diagramas e modelos que orientarão o desenvolvimento e garantirão a qualidade do produto final.

D

A fase de projeto é onde as interfaces gráficas são desenvolvidas e testadas, assegurando que o sistema seja intuitivo e fácil de usar para os usuários finais.

E

A fase de projeto é a etapa em que o sistema é implantado em ambiente de produção e os usuários começam a utilizá-lo, sinalizando o fim do ciclo de desenvolvimento.

A alternativa C está correta.

A) A fase de projeto é responsável por levantar e documentar todos os requisitos do sistema: Incorreta. Esta alternativa confunde a fase de levantamento de requisitos com a fase de projeto. O levantamento de requisitos é realizado na fase inicial do ciclo de vida do desenvolvimento de software, onde são identificadas e documentadas as necessidades e expectativas dos stakeholders. Já a fase de projeto (design) se concentra em traduzir esses requisitos em uma arquitetura técnica que servirá como base para o desenvolvimento do sistema.

B) A fase de projeto é onde as funcionalidades são testadas exaustivamente: Incorreta. Esta alternativa descreve a fase de testes, que ocorre após a implementação do software. Na fase de testes, o foco é validar o funcionamento das funcionalidades desenvolvidas para garantir que o sistema esteja livre de bugs. O objetivo da fase de projeto não é testar, mas sim planejar e definir como o sistema será estruturado e desenvolvido.

C) A fase de projeto é o momento em que a arquitetura do software é definida: Correta. A fase de projeto é crucial no ciclo de vida de desenvolvimento de software, pois nela são criados os diagramas e modelos que guiarão todo o processo de implementação. Durante essa fase, são definidas as estruturas de dados, as interfaces de comunicação, as regras de negócio e outras especificações técnicas que garantirão a qualidade e a eficiência do produto final. O planejamento cuidadoso nesta etapa minimiza o risco de problemas durante a implementação e facilita a manutenção do sistema ao longo do tempo.

D) A fase de projeto é onde as interfaces gráficas são desenvolvidas e testadas: Incorreta. O desenvolvimento e teste das interfaces gráficas fazem parte da fase de implementação e testes. Na fase de projeto, o foco está na concepção da arquitetura geral do sistema e na definição de como cada componente do software interagirá, mas não na criação das interfaces em si.

E) A fase de projeto é a etapa em que o sistema é implantado em ambiente de produção: Incorreta. A implantação do sistema em ambiente de produção ocorre na fase final do ciclo de vida, conhecida como implantação. Esta fase marca o ponto em que o software desenvolvido é disponibilizado para uso pelos usuários finais. O projeto é uma etapa anterior, focada na criação da estrutura que permitirá o desenvolvimento e a futura implantação do sistema.

Para saber mais sobre esse conteúdo, acesse:

Módulo 1

CICLO DE VIDA DE DESENVOLVIMENTO DE SOFTWARE

“As atividades típicas de um processo de desenvolvimento de software compreendem, de modo geral, independentemente do processo adotado, as seguintes etapas: Levantamento de requisitos, Análise, Projeto (design), Implementação, Testes, Implantação. [...] O foco é a etapa de projeto (ou design) do sistema, na qual existe uma dependência de seus artefatos em relação aos artefatos elaborados na fase de análise (ou modelagem). A figura a seguir ilustra os diagramas da UML que, usualmente, são construídos nos modelos de análise e de projeto.”

Desafio 2

Você foi designado como desenvolvedor responsável por um novo sistema de gerenciamento de pedidos em uma empresa de comércio eletrônico. Durante a fase de design, a equipe decidiu utilizar diagramas de sequência para documentar as interações entre os objetos envolvidos no processo de pedido. É fundamental que você compreenda o papel do diagrama de sequência e como ele pode ajudar. Qual das alternativas abaixo melhor descreve a utilidade do diagrama de sequência no desenvolvimento de sistemas?

A

O diagrama de sequência é utilizado para modelar a estrutura hierárquica das classes no sistema, garantindo que cada classe esteja corretamente organizada de acordo com sua função no processo de pedido.

B

O diagrama de sequência é usado para definir os casos de uso do sistema, descrevendo as funcionalidades que serão implementadas e como elas se relacionam com os requisitos dos usuários.

C

O diagrama de sequência ajuda a visualizar a ordem cronológica das interações entre objetos, mostrando como as mensagens são trocadas entre eles para realizar uma funcionalidade específica no sistema.

D

O diagrama de sequência é responsável por documentar as regras de negócio aplicáveis a cada transação de pedido, garantindo que todas as políticas empresariais sejam seguidas durante o processo de desenvolvimento.

E

O diagrama de sequência serve para representar a arquitetura física do sistema, detalhando como os componentes de hardware e software serão distribuídos na rede corporativa.

A alternativa C está correta.

A) O diagrama de sequência é utilizado para modelar a estrutura hierárquica das classes no sistema: Incorreta. Essa descrição é mais adequada para um diagrama de classes. O diagrama de classes mostra as relações estáticas entre classes, como herança, composição e associação, mas não captura o comportamento dinâmico ou a sequência de interações entre os objetos do sistema.

B) O diagrama de sequência é usado para definir os casos de uso do sistema: Incorreta. A definição de casos de uso é geralmente documentada em um diagrama de casos de uso, que descreve as funcionalidades do sistema e suas interações com atores externos. O diagrama de sequência, por outro lado, foca em como os objetos dentro do sistema interagem para realizar um caso de uso específico.

C) O diagrama de sequência ajuda a visualizar a ordem cronológica das interações entre objetos: Correta. O diagrama de sequência é uma ferramenta essencial na modelagem dinâmica de sistemas orientados a objetos. Ele descreve como os objetos interagem em um determinado fluxo de trabalho, destacando a sequência temporal das mensagens trocadas entre eles. Isso permite identificar e validar as operações necessárias para que o sistema funcione conforme esperado, além de garantir que todas as interações ocorram na ordem correta.

D) O diagrama de sequência é responsável por documentar as regras de negócio aplicáveis a cada transação de pedido: Incorreta. Enquanto o diagrama de sequência pode mostrar como as interações entre objetos implementam regras de negócio, ele não é a ferramenta principal para documentar essas regras. As regras de negócio são geralmente capturadas em documentos de requisitos ou em diagramas de atividades, que focam mais nos fluxos de controle e nas decisões lógicas.

E) O diagrama de sequência serve para representar a arquitetura física do sistema: Incorreta. A arquitetura física de um sistema, incluindo a distribuição de componentes de hardware e software, é normalmente documentada em um diagrama de implantação. O diagrama de sequência não se preocupa com a arquitetura física, mas sim com a interação entre objetos ao longo do tempo em um ambiente de software.

Para saber mais sobre esse conteúdo, acesse:

Módulo 1

DIAGRAMA DE SEQUÊNCIA

“Diagramas de sequência recebem esse nome porque descrevem, ao longo de uma linha do tempo, a sequência de comunicações entre objetos de um sistema de informação. Em geral, são construídos logo após os diagramas de caso de uso e de classes, pois têm como principais objetivos: Documentar casos de uso; Mostrar como os objetos do sistema se comunicam através de mensagens em ordem temporal; Validar se todas as operações das classes foram identificadas e declaradas; Validar a existência de um objeto necessário ao funcionamento do sistema.”

Desafio 3

Você está atuando como engenheiro de software em um projeto que utiliza diagramas de sequência para modelar as interações entre objetos durante o desenvolvimento de um sistema de automação industrial. Durante uma reunião, o gerente de projetos pede que você explique alguns dos principais elementos que compõem um diagrama de sequência. Abaixo estão três afirmações sobre esses elementos. Quais delas são verdadeiras?

1. As mensagens em um diagrama de sequência são representadas por linhas horizontais conectando as linhas de vida dos objetos, e essas mensagens podem ser síncronas ou assíncronas.

2. As linhas de vida representam a existência dos objetos ao longo do tempo e são desenhadas como linhas verticais tracejadas que começam quando o objeto é criado e terminam quando ele é destruído.

3. As condições de guarda em um diagrama de sequência são usadas para indicar que uma mensagem só será enviada se uma determinada condição lógica for satisfeita.

A

Apenas as afirmativas 1 e 2 são verdadeiras.

B

Apenas as afirmativas 1 e 3 são verdadeiras.

C

Apenas as afirmativas 2 e 3 são verdadeiras.

D

Todas as afirmativas são verdadeiras.

E

Nenhuma das afirmativas é verdadeira.

A alternativa D está correta.

A) Apenas as afirmativas 1 e 2 são verdadeiras: Incorreta. Esta alternativa está parcialmente correta, pois tanto a afirmativa 1 quanto a afirmativa 2 descrevem elementos corretos de um diagrama de sequência. No entanto, ela ignora a validade da afirmativa 3, que também é correta, tornando essa opção incorreta.

B) Apenas as afirmativas 1 e 3 são verdadeiras: Incorreta. Esta alternativa também está parcialmente correta, reconhecendo a validade das afirmativas 1 e 3. Afirmativa 1 corretamente descreve como as mensagens são representadas e diferenciadas em um diagrama de sequência, enquanto a afirmativa 3 explica corretamente o uso de condições de guarda. No entanto, a afirmativa 2 também é verdadeira, o que torna essa alternativa incorreta.

C) Apenas as afirmativas 2 e 3 são verdadeiras: Incorreta. Embora as afirmativas 2 e 3 estejam corretas, essa alternativa não é a mais completa, pois a afirmativa 1 também é verdadeira. As linhas de vida são, de fato, representadas por linhas verticais tracejadas que mostram a duração de um objeto no diagrama, e as condições de guarda são usadas para controlar o fluxo de mensagens. Mas, como a afirmativa 1 também é correta, essa alternativa está errada.

D) Todas as afirmativas são verdadeiras: Correta. Essa alternativa está correta, pois todas as três afirmativas descrevem elementos precisos de um diagrama de sequência. A afirmativa 1 descreve corretamente a representação das mensagens e a distinção entre síncronas e assíncronas. A afirmativa 2 explica a função das linhas de vida, que mostram a existência temporal dos objetos. A afirmativa 3 descreve corretamente as condições de guarda, que controlam a transmissão de mensagens com base em condições lógicas. Portanto, todas as afirmativas são verdadeiras.

E) Nenhuma das afirmativas é verdadeira: Incorreta. Todas as afirmativas fornecidas estão corretas. Cada uma delas descreve corretamente um aspecto fundamental do diagrama de sequência. Portanto, essa alternativa está incorreta.

Para saber mais sobre esse conteúdo, acesse:

Módulo 1

ELEMENTOS DO DIAGRAMA DE SEQUÊNCIA

“O diagrama de sequência é composto de diversos tipos de elementos. [...] As mensagens podem ser síncronas, assíncronas, de retorno e de criação de objeto. [...] As linhas de vida representam a sequência na qual a vida do objeto transcorre durante a interação. [...] Quando são criadas interações entre os objetos, um retângulo sobre a linha é colocado [...] Muitas vezes é necessário indicar que o termo controle está associado a uma expressão lógica. Nesse caso, a sintaxe para o termo controle pode ser uma condição ou uma iteração.”

Desafio 4

Você está desenvolvendo um sistema para uma clínica médica que precisa gerenciar informações de pacientes, médicos e agendamentos de consultas. Durante a fase de design, você decide utilizar um diagrama de classes para organizar e estruturar os dados do sistema. Qual das alternativas abaixo melhor descreve um dos principais usos do diagrama de classes?

A

O diagrama de classes é utilizado para representar o fluxo de controle e as interações dinâmicas entre os objetos durante a execução do sistema.

B

O diagrama de classes serve para modelar os processos e atividades do sistema, detalhando as etapas necessárias para a conclusão de uma tarefa.

C

O diagrama de classes descreve os atributos e métodos das classes, além das relações entre elas, como associações, heranças e agregações.

D

O diagrama de classes é usado para definir as interfaces gráficas do sistema, especificando o layout e a aparência das telas que os usuários irão utilizar.

E

O diagrama de classes é uma ferramenta para mapear a arquitetura física do sistema, mostrando como os componentes de hardware e software são distribuídos na infraestrutura de TI.

A alternativa C está correta.

A) O diagrama de classes é utilizado para representar o fluxo de controle e as interações dinâmicas entre os objetos durante a execução do sistema: Incorreta. Essa descrição corresponde mais adequadamente ao diagrama de sequência ou ao diagrama de atividades, que são usados para modelar o comportamento dinâmico e o fluxo de controle dos objetos durante a execução do sistema. O diagrama de classes, por outro lado, se concentra em representar a estrutura estática do sistema.

B) O diagrama de classes serve para modelar os processos e atividades do sistema, detalhando as etapas necessárias para a conclusão de uma tarefa: Incorreta. O mapeamento de processos e atividades é o domínio do diagrama de atividades ou dos diagramas de fluxo de trabalho, que capturam o fluxo das tarefas e processos no sistema. O diagrama de classes não é utilizado para este propósito.

C) O diagrama de classes descreve os atributos e métodos das classes, além das relações entre elas, como associações, heranças e agregações: Correta. O diagrama de classes é uma ferramenta essencial para modelar a estrutura estática de um sistema. Ele descreve as classes, seus atributos, métodos e as relações entre elas, como associações (relações entre objetos), heranças (relações de generalização/especialização) e agregações (relações de composição ou agregação). Essa representação ajuda a entender como os diferentes elementos do sistema estão organizados e interconectados.

D) O diagrama de classes é usado para definir as interfaces gráficas do sistema, especificando o layout e a aparência das telas que os usuários irão utilizar: Incorreta. O diagrama de classes não se destina ao design de interfaces gráficas. O layout e a aparência das telas são mais adequadamente modelados com protótipos de UI ou ferramentas de design de interface de usuário. O diagrama de classes foca na estrutura interna do sistema, não na interface do usuário.

E) O diagrama de classes é uma ferramenta para mapear a arquitetura física do sistema, mostrando como os componentes de hardware e software são distribuídos na infraestrutura de TI: Incorreta. A arquitetura física do sistema é representada por diagramas de implantação, que detalham a distribuição dos componentes de hardware e software na rede e infraestrutura de TI. O diagrama de classes, por outro lado, representa a organização lógica do código-fonte e os relacionamentos entre as classes do sistema.

Para saber mais sobre esse conteúdo, acesse:

Módulo 2

DIAGRAMA DE CLASSES DE PROJETO

“O diagrama de classes de projeto apresenta os dados que serão necessários para a construção do sistema de informação. Esse é o diagrama que possui o maior número de símbolos para sua diagramação. O seu principal objetivo é permitir a visualização dos componentes das bases de dados do sistema, ou seja, as

classes e seus objetos, relações entre elas, suas operações, seus métodos, suas interfaces e seus pacotes. Também é papel desse diagrama atender às demandas dos casos de uso, armazenando as estruturas de dados projetadas para o sistema.”

Desafio 5

Você está trabalhando no desenvolvimento de um sistema financeiro que precisa armazenar dados de transações e relatórios temporários. Durante a fase de design, sua equipe está discutindo como gerenciar o ciclo de vida dos objetos no sistema. Alguns dados precisam ser armazenados permanentemente, enquanto outros só são necessários durante a execução do sistema. Considerando o conceito de classes persistentes e transientes, qual das seguintes alternativas melhor explica a diferença entre esses dois tipos de classes?

A

Classes persistentes são aquelas que permanecem na memória durante toda a execução do sistema, enquanto as classes transientes são removidas da memória assim que sua função é concluída.

B

Classes persistentes são aquelas cujos objetos têm seus dados armazenados em banco de dados ou outro meio de armazenamento durável, enquanto classes transientes têm seus objetos destruídos ao final da execução ou quando não são mais necessários.

C

Classes persistentes são utilizadas para armazenar dados temporários e caches de informação, enquanto classes transientes são usadas para dados que precisam ser mantidos entre diferentes execuções do sistema.

D

Classes persistentes só podem ser usadas em sistemas com banco de dados, enquanto classes transientes são específicas para sistemas que não utilizam armazenamento durável.

E

Classes persistentes armazenam dados em arquivos de texto simples, enquanto classes transientes utilizam estruturas complexas de armazenamento em memória.

A alternativa B está correta.

A) Classes persistentes são aquelas que permanecem na memória durante toda a execução do sistema, enquanto as classes transientes são removidas da memória assim que sua função é concluída: Incorreta. Esta alternativa confunde os conceitos de memória com persistência. Classes persistentes têm a ver com o armazenamento durável de dados fora da memória volátil, enquanto classes transientes têm seus objetos destruídos após a execução do sistema, mas isso não é relacionado à sua permanência na memória durante a execução.

B) Classes persistentes são aquelas cujos objetos têm seus dados armazenados em banco de dados ou outro meio de armazenamento durável, enquanto classes transientes têm seus objetos destruídos ao final da execução ou quando não são mais necessários: Correta. Esta alternativa define com precisão a diferença entre classes persistentes e transientes. Classes persistentes são usadas para objetos que precisam ter seus dados armazenados de forma durável, como em um banco de dados, para que possam

ser recuperados em futuras execuções do sistema. Já as classes transientes são usadas para objetos que são temporários e não necessitam ser armazenados permanentemente.

C) Classes persistentes são utilizadas para armazenar dados temporários e caches de informação, enquanto classes transientes são usadas para dados que precisam ser mantidos entre diferentes execuções do sistema: Incorreta. Esta alternativa inverte os conceitos. Na realidade, são as classes transientes que armazenam dados temporários, enquanto as classes persistentes mantêm dados entre diferentes execuções do sistema.

D) Classes persistentes só podem ser usadas em sistemas com banco de dados, enquanto classes transientes são específicas para sistemas que não utilizam armazenamento durável: Incorreta. Embora seja comum que classes persistentes interajam com bancos de dados, elas não são limitadas a esse tipo de armazenamento. Além disso, classes transientes podem existir em sistemas com ou sem armazenamento durável, dependendo das necessidades do sistema.

E) Classes persistentes armazenam dados em arquivos de texto simples, enquanto classes transientes utilizam estruturas complexas de armazenamento em memória: Incorreta. Essa descrição não corresponde à função real dessas classes. O tipo de armazenamento (arquivo de texto, banco de dados, etc.) não define a persistência ou transitoriedade de uma classe. A distinção principal é se os dados do objeto precisam ser preservados após a execução do sistema (persistentes) ou não (transientes).

Para saber mais sobre esse conteúdo, acesse:

Módulo 2

CLASSES PERSISTENTES E TRANSIENTES

“As classes são divididas em: Persistentes: Presumem que seus objetos precisam, de alguma maneira, ser preservados mesmo após o encerramento da utilização do sistema. Não persistentes (transientes): Terão seus objetos destruídos durante ou na finalização do uso do sistema. Estas são diferenciadas por estereótipos <<transient>>.”

Estudo de Caso de Modelagem de Sistemas em UML

Desafio 1

Você foi recentemente contratado como analista de sistemas em uma empresa que está desenvolvendo um novo software de gerenciamento de projetos. Durante a fase inicial do projeto, você foi solicitado a criar um diagrama que descreva como os diferentes usuários do sistema interagem com as funcionalidades principais. Esse diagrama será utilizado pela equipe de desenvolvimento para entender os requisitos funcionais do sistema e planejar as próximas etapas do projeto. Qual diagrama da UML é mais adequado para essa tarefa?

A

Diagrama de Classes

B

Diagrama de Sequência

C

Diagrama de Casos de Uso

D

Diagrama de Atividades

E

Diagrama de Estados

A alternativa C está correta.

A) Incorreta. O Diagrama de Classes é utilizado para descrever a estrutura estática de um sistema, mostrando as classes, atributos, métodos e os relacionamentos entre eles. Ele é importante para o desenvolvimento do sistema, mas não é o diagrama mais apropriado para descrever as interações dos usuários com o sistema.

B) Incorreta. O Diagrama de Sequência é usado para detalhar como os objetos do sistema interagem em uma sequência temporal, ilustrando o fluxo de mensagens entre os objetos em um determinado cenário. Embora útil para entender a lógica de um caso de uso específico, ele não fornece uma visão geral das interações do usuário com o sistema.

C) Correta. O Diagrama de Casos de Uso é ideal para descrever as interações entre os usuários (atores) e o sistema, representando as funcionalidades principais do sistema sob a perspectiva do usuário. Ele ajuda a capturar os requisitos funcionais de maneira clara e facilita a comunicação entre os analistas, desenvolvedores e clientes.

D) Incorreta. O Diagrama de Atividades descreve o fluxo de trabalho ou o comportamento de um sistema em termos de atividades. Ele é útil para modelar processos de negócios ou operações detalhadas, mas não é o mais adequado para descrever as interações entre usuários e o sistema como um todo.

E) Incorreta. O Diagrama de Estados mostra os estados pelos quais um objeto passa durante seu ciclo de vida em resposta a eventos, e como ele reage a esses eventos. Ele é útil para modelar o comportamento dinâmico de um único objeto, mas não é adequado para mostrar as interações dos usuários com o sistema.

Para saber mais sobre esse conteúdo, acesse:

Módulo 2

MODELO DE CASOS DE USO

"O modelo de casos de uso é composto por um ou mais diagramas de casos de uso, artefatos gráficos, e pelas descrições de casos de usos, artefatos textuais. Os componentes do referido diagrama incluem

casos de uso, atores, ou seja, elementos externos que interagem com o sistema, e relacionamentos entre os elementos anteriores.".

Desafio 2

Você foi contratado por uma empresa de tecnologia para desenvolver um novo sistema de gestão de inventário. Durante a fase inicial do projeto, você precisa garantir que o sistema atenda a todas as necessidades dos usuários finais. Para isso, você deve definir quais serviços o sistema deve fornecer e identificar as restrições operacionais. Qual é a etapa do desenvolvimento de software em que essas atividades são realizadas?

A

Projeto de Sistema

B

Implementação

C

Levantamento de Requisitos

D

Testes de Software

E

Manutenção

A alternativa C está correta.

A) Incorreta. O Projeto de Sistema é a fase onde se definem as especificações técnicas do sistema, como arquitetura e design, com base nos requisitos já levantados. Nessa fase, o foco é a estruturação do sistema e como ele será construído, mas não na identificação das necessidades dos usuários.

B) Incorreta. A Implementação é a fase onde o código do sistema é escrito e as funcionalidades especificadas são desenvolvidas. Essa etapa se concentra em transformar as especificações em um software funcional, mas não envolve a definição de requisitos.

C) Correta. O Levantamento de Requisitos, também conhecido como Elicitação de Requisitos, é a etapa inicial onde se identifica o que o sistema deve fazer (requisitos funcionais) e as restrições sob as quais ele deve operar (requisitos não funcionais). Essa fase é crucial para assegurar que o sistema atenda às expectativas e necessidades dos usuários.

D) Incorreta. A fase de Testes de Software ocorre após a implementação e envolve verificar se o sistema funciona conforme especificado, identificando e corrigindo erros. Essa fase é essencial para garantir a qualidade do sistema, mas não se destina à definição de requisitos.

E) Incorreta. A Manutenção é a fase pós-implantação do software, onde são feitas correções, melhorias e atualizações conforme necessário. Embora importante para o ciclo de vida do software, essa etapa não envolve a coleta inicial de requisitos.

Para saber mais sobre esse conteúdo, acesse:

Módulo 1

LEVANTAMENTO DE REQUISITOS

"Considerando as atividades genéricas que compõem o processo de desenvolvimento de software, a atividade de levantamento de requisitos, também denominada elicitação de requisitos, corresponde à etapa de compreensão do problema por parte do engenheiro de software ou equipe. O principal objetivo do levantamento de requisitos é que usuários e equipe de desenvolvimento tenham a mesma visão do problema a ser resolvido.".

Desafio 3

Como parte de um projeto de desenvolvimento de software, você está responsável por definir as estruturas internas do sistema. Seu papel é identificar e representar as principais entidades e seus relacionamentos estáticos para garantir que o sistema atenda aos requisitos levantados. Para isso, você utiliza um diagrama específico da UML. Qual diagrama você deve criar?

A

Diagrama de Sequência

B

Diagrama de Casos de Uso

C

Diagrama de Atividades

D

Diagrama de Classes

E

Diagrama de Estados

A alternativa D está correta.

A) Incorreta. O Diagrama de Sequência é utilizado para modelar a interação entre objetos ao longo do tempo, mostrando a troca de mensagens entre eles em um cenário específico. Embora seja útil para entender o fluxo de eventos, ele não representa a estrutura estática do sistema.

B) Incorreta. O Diagrama de Casos de Uso é empregado para capturar os requisitos funcionais do sistema, representando as interações entre os atores (usuários) e as funcionalidades do sistema. Esse diagrama é mais adequado para entender as necessidades do usuário, mas não para descrever a estrutura interna do sistema.

C) Incorreta. O Diagrama de Atividades é utilizado para modelar o fluxo de trabalho ou processos dentro do sistema, detalhando como as atividades são executadas e em que ordem. Ele se foca no comportamento dinâmico, mas não na estrutura estática das entidades do sistema.

D) Correta. O Diagrama de Classes é o diagrama apropriado para representar a estrutura estática do sistema. Ele descreve as classes, seus atributos, métodos e os relacionamentos (associações, herança, etc.) entre elas. Esse diagrama é fundamental para entender a arquitetura do sistema e como as diferentes partes interagem entre si.

E) Incorreta. O Diagrama de Estados é usado para modelar os estados pelos quais um objeto passa ao longo do tempo em resposta a eventos. Ele é focado no comportamento de um objeto individual, não na estrutura geral do sistema.

Para saber mais sobre esse conteúdo, acesse:

Módulo 3

MODELO DE CLASSES

"Um modelo de classes é composto por um ou mais diagramas de classes, mostrando a existência das classes e os seus relacionamentos numa visão lógica e estática do sistema. [...] Classes se relacionam com outras classes ou mesmo com a própria classe através de associações, que podem ser associações simples ou específicas, como a agregação, a composição e a generalização/especialização.".

Desafio 4

Você está atuando como analista de sistemas em um projeto de desenvolvimento de software para automatizar processos de negócios em uma empresa. Sua tarefa atual é criar um diagrama que mostre o fluxo de trabalho de uma atividade específica, desde o início até o fim, incluindo decisões, paralelismos e possíveis exceções. Qual diagrama da UML você deve usar para realizar essa tarefa?

A

Diagrama de Sequência

B

Diagrama de Estados

C

Diagrama de Atividades

D

Diagrama de Casos de Uso

E

Diagrama de Componentes

A alternativa C está correta.

A) Diagrama de Sequência: Incorreta. O Diagrama de Sequência é utilizado para representar a interação entre objetos em uma sequência temporal, mostrando como as mensagens são trocadas entre eles ao longo do tempo em um cenário

específico. Embora útil para entender a interação entre objetos, ele não é adequado para representar o fluxo completo de atividades dentro de um processo.

B) Diagrama de Estados: Incorreta. O Diagrama de Estados descreve os diferentes estados que um objeto pode assumir ao longo do tempo e as transições entre esses estados em resposta a eventos. Ele é útil para modelar o comportamento dinâmico de um objeto, mas não para representar um processo ou fluxo de trabalho.

C) Diagrama de Atividades: Correta. O Diagrama de Atividades é ideal para representar o fluxo de trabalho ou processo dentro de um sistema. Ele mostra a sequência de atividades, as decisões, o paralelismo e as possíveis exceções, fornecendo uma visão clara de como o processo ocorre do início ao fim. Esse diagrama é crucial para entender o comportamento dinâmico e o fluxo de controle de um processo.

D) Diagrama de Casos de Uso: Incorreta. O Diagrama de Casos de Uso representa as interações entre os usuários e as funcionalidades principais do sistema, ajudando a capturar os requisitos funcionais. No entanto, ele não detalha o fluxo de atividades dentro de um processo específico.

E) Diagrama de Componentes: Incorreta. O Diagrama de Componentes mostra a organização física do código do sistema, incluindo módulos e suas dependências. Ele é importante para a fase de implementação, mas não para modelar o fluxo de atividades dentro de um processo.

Para saber mais sobre esse conteúdo, acesse:

Módulo 3

MODELO DE ATIVIDADES

"O modelo de atividades é composto por diagramas de atividades que, entre outros, descrevem os aspectos dinâmicos de um sistema, ou seja, um diagrama de atividade exibe passos de uma computação. Os referidos diagramas possuem as seguintes aplicações: modelagem de processo de negócio, modelagem da lógica de um caso de uso, modelagem da lógica de uma operação.".

Desafio 5

Você está desenvolvendo um sistema de reservas para um hotel, e uma das funcionalidades envolve gerenciar o ciclo de vida das reservas dos clientes. Durante o desenvolvimento, é necessário modelar como as reservas evoluem desde o momento em que são criadas até serem confirmadas, alteradas ou canceladas. Para representar essas transições e as condições sob as quais elas ocorrem, qual diagrama da UML você deve utilizar?

A

Diagrama de Atividades

B

Diagrama de Sequência

C

Diagrama de Estados

D

Diagrama de Casos de Uso

E

Diagrama de Classes

A alternativa C está correta.

A) Diagrama de Atividades: Incorreta. O Diagrama de Atividades é usado para modelar o fluxo de trabalho ou processos dentro do sistema, mostrando a sequência de atividades e decisões. Ele é útil para entender o fluxo de ações, mas não é o melhor diagrama para representar as transições de um objeto específico ao longo do tempo.

B) Diagrama de Sequência: Incorreta. O Diagrama de Sequência é utilizado para representar a interação entre objetos ao longo do tempo, detalhando a troca de mensagens entre eles. Embora útil para entender a dinâmica entre objetos, ele não é adequado para modelar as transições de um único objeto.

C) Diagrama de Estados: Correta. O Diagrama de Estados é ideal para representar as diferentes condições que um objeto pode assumir durante seu ciclo de vida e como ele transita entre essas condições em resposta a eventos. No contexto do sistema de reservas, esse diagrama é crucial para modelar como uma reserva muda de "Em validação" para "Confirmada", "Cancelada" ou outras situações possíveis.

D) Diagrama de Casos de Uso: Incorreta. O Diagrama de Casos de Uso descreve as interações entre os usuários e as funcionalidades principais do sistema, capturando os requisitos funcionais. Ele não é utilizado para modelar o comportamento de objetos específicos dentro do sistema.

E) Diagrama de Classes: Incorreta. O Diagrama de Classes representa a estrutura estática do sistema, incluindo as classes, atributos, métodos e as relações entre elas. Ele é fundamental para entender a arquitetura do sistema, mas não é utilizado para modelar o comportamento dinâmico de um objeto ao longo do tempo.

Para saber mais sobre esse conteúdo, acesse:

Módulo 3

MODELO DE ESTADOS

"No contexto do paradigma orientado a objetos, um determinado objeto possui um estado particular, mudando de estado quando acontece algum evento interno ou externo ao sistema. Nessa transição de um estado para outro, um objeto realiza determinadas ações dentro do sistema. O modelo de estados é composto por diagramas de estados que descrevem os possíveis estados pelos quais objetos, instâncias de uma determinada classe, podem passar e as alterações dos estados como resultado de eventos que atingem esses objetos.".

Introdução a Padrões de Projeto Design Patterns

Desafio 1

Imagine que você foi designado para liderar a evolução de um sistema de software em uma empresa que passou por várias mudanças ao longo dos anos. O sistema, desenvolvido por diferentes equipes, apresenta diversos desafios de manutenção devido à falta de uma estrutura consistente. Antes de implementar esses padrões, é fundamental entender o que motivou a criação dos padrões de projeto e como eles podem ser aplicados a problemas recorrentes no desenvolvimento de software.

Qual é a principal motivação para a utilização de padrões de projeto no desenvolvimento de sistemas de software?

A

Aumentar a complexidade do sistema, tornando-o mais robusto contra falhas inesperadas.

B

Facilitar a implementação de novas funcionalidades, reduzindo a necessidade de revisão das estruturas existentes.

C

Criar uma estrutura única que funcione para todos os tipos de sistemas, independentemente das suas especificidades.

D

Reutilizar soluções já testadas e comprovadas para problemas recorrentes, promovendo a consistência e a eficiência no desenvolvimento.

E

Substituir completamente a necessidade de um design personalizado, eliminando a criatividade individual dos desenvolvedores.

A alternativa D está correta.

A) Incorreta. Aumentar a complexidade do sistema não é um objetivo dos padrões de projeto. Pelo contrário, padrões de projeto buscam simplificar a solução de problemas recorrentes, evitando a complexidade desnecessária.

B) Incorreta. Embora facilitar a implementação de novas funcionalidades seja um benefício, a principal motivação dos padrões de projeto é a reutilização de soluções eficazes para problemas recorrentes, não apenas a facilidade de expansão.

C) Incorreta. Não existe uma estrutura única que funcione para todos os sistemas. Os padrões de projeto são aplicados conforme a necessidade específica de cada sistema e problema, proporcionando flexibilidade e adequação ao contexto.

D) Correta. Os padrões de projeto foram criados para capturar e reutilizar soluções que já se mostraram eficazes em resolver problemas específicos no desenvolvimento de software. Isso permite que desenvolvedores utilizem essas soluções para criar sistemas bem estruturados, facilitando a manutenção e a evolução do software.

E) Incorreta. Padrões de projeto não substituem a criatividade individual; eles complementam o processo de design, oferecendo soluções comprovadas que podem ser adaptadas conforme a criatividade e as necessidades do projeto.

Para saber mais sobre esse conteúdo, acesse:

Módulo 1

MOTIVAÇÃO E ORIGEM DOS PADRÕES DE PROJETO

"Imagine que você foi alocado em um novo trabalho, ficando responsável por evoluir um sistema desenvolvido por outras pessoas ao longo de anos. Você não se sentiria mais confortável se o software tivesse sido projetado com estruturas de solução que você já conhecesse? [...] É comum um desenvolvedor iniciante se sentir perdido diante da diversidade de respostas possíveis, sendo que algumas podem facilitar, enquanto outras podem dificultar bastante as futuras evoluções do sistema."

Desafio 2

Você foi contratado como desenvolvedor em uma empresa que está criando um sistema de gerenciamento de pedidos para uma loja online. Para melhorar a estrutura da loja online, você decide aplicar os padrões GRASP. Antes de começar, é importante entender o propósito dos padrões GRASP e como eles ajudam em um sistema orientado a objetos.

Qual é o principal objetivo dos padrões GRASP no desenvolvimento de sistemas orientados a objetos?

A

Definir uma maneira única de organizar todas as classes e objetos em um sistema.

B

Reduzir o número de classes necessárias para implementar uma funcionalidade no sistema.

C

Atribuir responsabilidades de forma que as classes tenham alta coesão e baixo acoplamento, facilitando a manutenção e evolução do sistema.

D

Garantir que todas as classes do sistema sejam dependentes umas das outras para evitar inconsistências.

E

Substituir a necessidade de um design personalizado, impondo uma estrutura rígida a ser seguida.

A alternativa C está correta.

A) Incorreta. Os padrões GRASP não prescrevem uma maneira única de organizar todas as classes, mas oferecem diretrizes para atribuir responsabilidades de forma eficaz, promovendo coesão e acoplamento adequado.

B) Incorreta. Reduzir o número de classes não é o objetivo dos padrões GRASP. Em vez disso, eles se concentram em garantir que as responsabilidades sejam atribuídas de maneira que promova a coesão e o acoplamento adequado, independentemente do número de classes.

C) Correta. O principal objetivo dos padrões GRASP é atribuir responsabilidades às classes de maneira que o sistema tenha alta coesão (classes com responsabilidades bem definidas e focadas) e baixo acoplamento (mínima dependência entre as classes). Isso facilita a manutenção, evolução e entendimento do sistema, promovendo uma arquitetura mais robusta e sustentável.

D) Incorreta. Pelo contrário, os padrões GRASP buscam reduzir dependências excessivas entre as classes, promovendo um baixo acoplamento e evitando que mudanças em uma classe impactem negativamente outras.

E) Incorreta. Os padrões GRASP não substituem a necessidade de design personalizado, mas fornecem diretrizes que ajudam a tomar decisões de design que levam a uma arquitetura mais eficiente e fácil de manter.

Para saber mais sobre esse conteúdo, acesse:

Módulo 2

PADRÕES GRASP

"GRASP é o acrônimo para o termo em inglês *General Responsibility Assignment Software Patterns*, termo definido por Craig Larman no livro intitulado *Applying UML and Patterns*, que define padrões gerais para a atribuição de responsabilidades em software. Os padrões GoF tratam de problemas específicos em projeto de software. Os padrões GRASP podem ser vistos como princípios gerais de projeto de software orientado a objetos aplicáveis a diversos problemas específicos.".

Desafio 3

Você está trabalhando como desenvolvedor em uma empresa que mantém um sistema de e-commerce. Recentemente, a equipe decidiu aplicar os princípios SOLID para melhorar a estrutura e a flexibilidade do código. Durante a revisão de um módulo que calcula descontos, você observa que o código atual requer modificações significativas sempre que uma nova regra de desconto é adicionada, o que pode ser uma indicação de que o design atual não segue alguns dos princípios SOLID. Para melhorar esse cenário, é crucial entender como esses princípios podem ser aplicados.

Qual princípio SOLID está sendo violado se o código de um sistema requer modificações sempre que uma nova regra de negócio é adicionada, e como esse princípio pode ser aplicado para resolver o problema?

A

O Princípio da Responsabilidade Única (SRP) está sendo violado; para corrigir, deve-se garantir que cada classe tenha apenas uma razão para mudar, dividindo as responsabilidades.

B

O Princípio da Inversão de Dependências (DIP) está sendo violado; a solução é garantir que módulos de alto nível não dependam de implementações de baixo nível, mas sim de abstrações.

C

O Princípio Aberto Fechado (OCP) está sendo violado; para corrigir, deve-se permitir a extensão do comportamento do sistema através da adição de novos módulos, sem modificar os módulos existentes.

D

O Princípio da Segregação de Interfaces (ISP) está sendo violado; a solução é dividir interfaces grandes em menores e mais específicas para cada cliente.

E

O Princípio da Substituição de Liskov (LSP) está sendo violado; para corrigir, deve-se garantir que subclasses possam ser substituídas por suas superclasses sem alterar o comportamento esperado do sistema.

A alternativa C está correta.

A) Incorreta. Embora o Princípio da Responsabilidade Única (SRP) seja importante, ele não está diretamente relacionado ao problema de modificar o código existente para adicionar novas regras de negócio. O SRP foca em garantir que cada classe tenha apenas uma responsabilidade, mas o problema descrito é mais uma questão de extensibilidade, relacionada ao OCP.

B) Incorreta. O Princípio da Inversão de Dependências (DIP) trata da relação entre módulos de alto e baixo nível, recomendando que ambos dependam de abstrações. Embora seja relevante em muitos contextos, não aborda diretamente a necessidade de modificar código existente ao adicionar novas funcionalidades.

C) Correta. O Princípio Aberto Fechado (OCP) estabelece que uma classe deve estar aberta para extensão, mas fechada para modificação. Isso significa que, ao adicionar novas funcionalidades, como regras de desconto, deve-se fazê-lo através de novos módulos ou classes que estendem o comportamento existente, sem alterar o código das classes já implementadas. Dessa forma, a manutenção e a evolução do sistema são facilitadas, evitando que mudanças em uma parte do código impactem outras partes.

D) Incorreta. O Princípio da Segregação de Interfaces (ISP) sugere que interfaces grandes devem ser divididas em menores e mais específicas, para que os clientes não sejam forçados a implementar métodos que não utilizam. No entanto, isso não resolve diretamente o problema de modificações frequentes no código para adicionar novas regras.

E) Incorreta. O Princípio da Substituição de Liskov (LSP) estabelece que subclasses devem poder substituir suas superclasses sem alterar o comportamento esperado. Esse princípio é crucial para garantir a corretude do sistema, mas não está relacionado ao problema de extensibilidade discutido na questão.

Para saber mais sobre esse conteúdo, acesse:

Módulo 3

PRINCÍPIO ABERTO FECHADO (OCP)

"O Princípio OCP (do inglês Open Closed Principle) estabelece que um módulo deve estar aberto para extensões, mas fechado para modificações, isto é, seu comportamento deve ser extensível sem que seja necessário alterá-lo para acomodar as novas extensões. Quando uma mudança em um módulo causa uma cascata de modificações em módulos dependentes, isso é sinal de que a estrutura do software não acomoda mudanças adequadamente. Se aplicado de forma adequada, este princípio permite que futuras mudanças desse tipo sejam feitas pela adição de novos módulos, e não pela modificação de módulos já existentes."

Desafio 4

Você está desenvolvendo um sistema para uma empresa de logística, onde cada módulo é responsável por uma parte específica do processo de entrega. Durante a análise do código, você percebe que uma das classes responsável pela gestão de pedidos também contém funcionalidades relacionadas à geração de relatórios e à integração com o sistema financeiro. Isso torna a classe complexa e difícil de manter, além de gerar dependências desnecessárias entre diferentes partes do sistema. Para melhorar essa situação, é essencial aplicar um dos princípios SOLID.

Qual princípio SOLID está sendo violado e como esse princípio pode ser aplicado para resolver o problema?

A

O Princípio Aberto Fechado (OCP) está sendo violado; para corrigir, deve-se garantir que a classe possa ser estendida sem a necessidade de modificações diretas.

B

O Princípio da Inversão de Dependências (DIP) está sendo violado; a solução é garantir que a classe dependa de abstrações, e não de implementações concretas.

C

O Princípio da Substituição de Liskov (LSP) está sendo violado; para corrigir, deve-se assegurar que as subclasses possam substituir a classe sem alterar o comportamento esperado.

D

O Princípio da Responsabilidade Única (SRP) está sendo violado; para corrigir, deve-se dividir a classe em várias classes menores, cada uma com uma única responsabilidade bem definida.

E

O Princípio da Segregação de Interfaces (ISP) está sendo violado; a solução é dividir as interfaces grandes em menores, específicas para cada função da classe.

A alternativa D está correta.

A) Incorreta. Embora o Princípio Aberto Fechado (OCP) seja relevante para permitir a extensão de uma classe sem modificá-la, o problema em questão está relacionado à atribuição de múltiplas responsabilidades a uma única classe, o que é tratado pelo SRP.

B) Incorreta. O Princípio da Inversão de Dependências (DIP) lida com as dependências entre módulos de alto e baixo nível, garantindo que ambos dependam de abstrações. No entanto, isso não aborda diretamente o problema de múltiplas responsabilidades em uma única classe.

C) Incorreta. O Princípio da Substituição de Liskov (LSP) trata da substituição de uma classe por suas subclasses sem alterar o comportamento esperado, mas não está relacionado à questão de uma classe ter múltiplas responsabilidades.

D) Correta. O Princípio da Responsabilidade Única (SRP) estabelece que uma classe deve ter apenas uma razão para mudar, ou seja, deve ser responsável por uma única funcionalidade ou parte do sistema. No

caso descrito, a classe que gerencia pedidos também está encarregada de gerar relatórios e integrar o sistema financeiro, o que viola o SRP. Para corrigir essa violação, a classe deve ser dividida em várias classes menores, cada uma focada em uma única responsabilidade, como gestão de pedidos, geração de relatórios, e integração financeira, respectivamente.

E) Incorreta. O Princípio da Segregação de Interfaces (ISP) sugere que interfaces grandes devem ser divididas em menores e mais específicas para cada função. Embora relevante para evitar a sobrecarga de métodos nas interfaces, o ISP não trata diretamente da questão de múltiplas responsabilidades em uma única classe.

Para saber mais sobre esse conteúdo, acesse:

Módulo 3

PRINCÍPIO DA RESPONSABILIDADE ÚNICA (SRP)

"O princípio SRP (do inglês Single Responsibility Principle) estabelece que o módulo deve ser responsável por um – e apenas um – ator, representando o papel desempenhado por alguém envolvido com o software. [...] Essa alocação de responsabilidades viola o princípio SRP, pois define um módulo que atende a diferentes atores. [...] De acordo com o princípio SRP, devemos separar essas responsabilidades em três módulos, o que nos daria a flexibilidade de, por exemplo, separar a busca de produtos e a inclusão de produtos em serviços fisicamente independentes, o que possibilitaria atender diferentes demandas de escalabilidade de forma mais racional."

Desafio 5

Você está trabalhando no desenvolvimento de um sistema de gestão de estoques para uma grande rede de supermercados. Durante a revisão do código, você percebe que o módulo responsável pela consulta de preços dos produtos está diretamente vinculado a um banco de dados específico, o que torna difícil trocar ou atualizar o banco de dados sem fazer alterações significativas no código. Isso também dificulta a realização de testes unitários, pois o código está fortemente acoplado a uma implementação concreta. Para melhorar a flexibilidade e facilitar a manutenção, é necessário aplicar um dos princípios SOLID que lida com a dependência entre módulos de diferentes níveis.

Qual princípio SOLID está sendo violado se um módulo de alto nível, como a consulta de preços, depende diretamente de uma implementação concreta de um banco de dados, e como esse princípio pode ser aplicado para resolver o problema?

A

O Princípio da Responsabilidade Única (SRP) está sendo violado; a solução é garantir que cada classe tenha apenas uma responsabilidade bem definida.

B

O Princípio Aberto Fechado (OCP) está sendo violado; para corrigir, deve-se garantir que o módulo possa ser estendido sem modificar o código existente.

C

O Princípio da Inversão de Dependências (DIP) está sendo violado; para corrigir, deve-se fazer com que o módulo de alto nível dependa de uma abstração, como uma interface, em vez de uma implementação concreta.

D

O Princípio da Substituição de Liskov (LSP) está sendo violado; para corrigir, deve-se garantir que subclasses possam substituir as classes base sem alterar o comportamento esperado.

E

O Princípio da Segregação de Interfaces (ISP) está sendo violado; a solução é dividir interfaces grandes em menores, específicas para cada cliente.

A alternativa C está correta.

A) Incorreta. O Princípio da Responsabilidade Única (SRP) trata de garantir que uma classe tenha apenas uma responsabilidade, mas não aborda o problema de dependência direta entre módulos de diferentes níveis.

B) Incorreta. O Princípio Aberto Fechado (OCP) sugere que um módulo deve estar aberto para extensão, mas fechado para modificações. Embora seja importante, ele não se aplica diretamente ao problema de dependência de implementações concretas.

C) Correta. O Princípio da Inversão de Dependências (DIP) estabelece que módulos de alto nível não devem depender de implementações concretas de baixo nível, mas sim de abstrações, como interfaces. Isso permite que o sistema seja mais flexível e facilita a substituição ou atualização de componentes, como trocar o banco de dados sem a necessidade de alterar o código do módulo de consulta de preços. Aplicando o DIP, você poderia introduzir uma interface que abstraia as operações de banco de dados, permitindo que diferentes implementações possam ser utilizadas conforme necessário, sem alterar o módulo de alto nível.

D) Incorreta. O Princípio da Substituição de Liskov (LSP) trata da capacidade de substituir subclasses por suas superclasses sem alterar o comportamento do sistema. Esse princípio não aborda diretamente a questão das dependências entre módulos de alto e baixo nível.

E) Incorreta. O Princípio da Segregação de Interfaces (ISP) recomenda a divisão de interfaces grandes em menores e mais específicas, mas não lida com a dependência de implementações concretas em módulos de alto nível.

Para saber mais sobre esse conteúdo, acesse:

Módulo 3

PRINCÍPIO DA INVERSÃO DE DEPENDÊNCIAS (DIP)

"O princípio DIP (do inglês Dependency Inversion Principle) estabelece que as entidades concretas devem depender de abstrações e não de outras entidades concretas. [...] Por que essa dependência é inadequada? Como aspectos tecnológicos são voláteis, devemos isolar a lógica de negócio, permitindo que a implementação concreta desses aspectos possa variar sem afetar as classes que implementam a lógica de negócio.".

Considerações finais

Parabéns! Incluir exercícios em sua rotina de estudos te auxilia a fixar, aplicar e desenvolver a capacidade de resolver problemas! Continue, o processo de aprender é contínuo e seu esforço fundamental.

Compartilhe conosco como foi sua experiência com este conteúdo. Por favor, responda a este formulário de avaliação e nos ajude a aprimorar ainda mais a sua experiência de aprendizado!