Carlos Henrique Paisca RA:62030

Thiago Rodrigo Bucalon Ra:68962

Professor(a): Edson Alves de Oliveira Junior

Disciplina: Análise de Sistemas de Software

Lista de exercícios: Orientação de objetos e processo unificado

1. Quais as principais características do paradigma de orientação a objetos?

A principal característica da orientação a objeto é a facilidade para abordar projetos em uma linguagem que flui de acordo com o funcionamento real do sistema, simulando o mundo. Ele utiliza objetos como classes, métodos entre outros o que torna mais fácil para reutilizar trechos de códigos e oculta dados irrelevantes, facilitando a interpretação do código.

2. Quais as principais diferenças entre o paradigma de orientação a objetos e o estruturado?

A diferença é a facilidade de entender os conceitos do estrutural a paradigma POO, do contrário o código é melhor estruturado (mais fácil de entender) em POO, pois o estruturado mistura o tratamento de dados com o comportamento do código. Em projetos grandes, geralmente utilizam o POO.

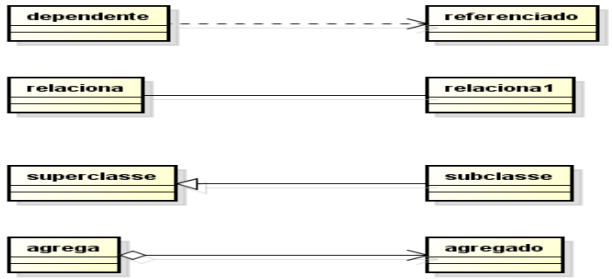
3. Quais as vantagens da utilização de sobrecarga de operadores? Dê exemplos.

Uma das vantagens de usar a sobrecarga de operações é a de diminuir a aparente complexidade do código porque ele permite utilizar um mesmo nome para operações parecidas. Assim, utiliza – se menos memória e diminui erros de programação.

4. Quais os tipos de relacionamento entre classes que podem ser utilizados no paradigma de orientação a objetos? Mostre a sua representação em UML e dê um exemplo de cada um deles.

As classes podem ter a operação: de **herança**, quando cria uma subclasse com componentes do superclasse, ou seja, todos os métodos e atributos da superclasse são ‘copiadas’ para a subclasse; **Associação**, quando uma classe possui uma ligação (influência) com outra classe; **Agregação**, quando uma classe inclui estruturalmente outra classe; **Dependência**, quando uma classe é estritamente relacionada com outra

de forma que se uma classe for modificada a dependente será afetada;



5. Qual a diferença entre o relacionamento de composição e agregação? Dê um exemplo de cada um deles.

A composição e a agregação são relações parecidas, mas diferem no grau de 'dependência'. A relação de agregação é uma ligação em que permite à existência de um objeto de uma classe A sem que exista um objeto da outra classe agregada B.

Exemplos: Classe time e classe atleta podem ter um time ou um atleta, ou seja, pode existir um atleta que não está em um time e pode existir um time sem o atleta relacionado. Este é uma relação de agregação. Já na relação de composição, deve existir um objeto da classe-todo para existir um objeto da classe-parte. Por exemplo: classe pedido e classe itemPedido. Somente existirá itens de um pedido se houver um pedido.

6. Explique os princípios da UML. Quais são as visões de UML?

UML é um modelo de linguagem para retratar graficamente um sistema e tem objetivo de analisar, especificar, visualizar, construir e documentar artefatos de software. Serve para criar uma representação minimizada da realidade, facilitando o entendimento do funcionamento do sistema e proporcionando uma visão ampla. Visões de UML são: A Visão estática, visão de maquinas de estados, de projeto, de casos de usos, atividades e interação.

7. Quais os princípios diagramas da UML? Para que serve cada um deles?

Os principais diagramas são:

- Classe: que mostram as classes que compõe o sistema especificando, também, os seus atributos e métodos. Ele também mostra a relação entre as classes;
- Sequência: que representam todos os passos tomados na utilização do sistema;
- Estados: que mostra as condições para o sistema ficar em certo estado;
- de procedimentos;

0

- Casos de usos: que analisa cada situação em que o sistema é utilizado, mostrando quem interage com o sistema;
- Colaboração: troca de mensagens focadas no objeto e na ação;
- Estruturas internas:
- Componentes: que mostra o sistema funcionalmente, expondo as relações entre seus componentes e a organização de seus módulos;
- Atividades:
- Comunicação: que mostra como as classes chamam um ao outro, ou seja, como que eles se comunicam;
- Instalação:
- Pacotes: mostra a relações entre os pacotes que compõe o sistema.

8. Qual o objetivo do Processo Unificado?

O objetivo do processo unificado é de proporcionar um fluxo de desenvolvimento mais próximo com o comportamento real do desenvolvimento do projeto. Ele se baseia em partições do projeto para pequenos trabalhos utilizando iteração e adaptação para implementar o sistema.

9. Quais os princípios básicos do Processo Unificado? Explique cada uma deles.

No processo unificado existe três princípios. Ele é orientado pelo casos de uso para o levantamento de requisitos. Assim é melhor visualizado pelo cliente, pois ele entende melhor e fica claro as funcionalidades do sistema. Ele é centrado na arquitetura que representa diferentes visões de como o sistema pode ser implementado. Ele é iterativo e incremental. O projeto é dividido em vários partes. Todas essas partes passam por um procedimento de um projeto inteiro, contenho a fase da concepção, transitividade, elaboração, construções. Assim, possibilitando a adaptações e identificar mudanças.

10. Quais os principais workflows do processo unificado?

Os principais workflows do PU são requerimento, analise, implementação, design e teste.

Lista de exercícios: workflow de captura de requisitos

1. Qual o objetivo do workflow de captura de requisitos?

O objetivo é modelar o negócio e especificar requisitos. Ele busca entender os problemas, como a organização funciona, definir os usuários, especificar os casos em que o sistema é utilizado, detalhar os casos de uso.

2. Quais os principais artefatos do workflow de captura de requisitos?

A partir da captura de requisitos são criados: a visão geral do negócio, uma descrição do sistema, modelo conceitual de objetos de negócio demonstrando os principais conceitos;

Comentado [TB1]: Deve melhorar essa resposta.

3. Quais as atividades do workflow da fase de captura de requisitos? Explique cada uma delas?

Primeiramente encontra os atores que interagem ou que fazem uma grande influência com sistema, define os casos de uso, e delimita o ambiente. O sistema pode ter vários casos de usos, então deve priorizar o caso de uso principal para ter foco nessa parte na iteração inicial e para saber quais serão priorizados em outras iterações. Depois detalhamos os casos, detalhando os fluxos de eventos e como os atores interagem com o sistema e criar diagramas. Em Seguida, temos que prototipar a interface, analisando o que ele deve oferecer ao usuário. Por final, deve-se estruturar os casos de uso, adicionando casos opcionais e procurando casos que podem ser reutilizados.

4. O que representa a visão geral do negócio? Explique porque é importante fazer este modelo por meio de um exemplo.

A representação da visão geral do negócio é mostrar como o negócio funciona e as regras de negócio em diagramas. A importância da utilização de exemplos reais é para o melhor entendimento dos problemas e das necessidades da empresa. Assim, o software criado é próprio para o empreendimento no sentido que torna o trabalho prático e já evita futuros problemas do empreendimento.

5. O que é o modelo de casos de uso? Qual a sua importância no processo de desenvolvimento de software?

Modelo de casos de uso é um conjunto de descrições e diagramas que descrevem as metas dos usuários, as interações entre os usuários e o sistema e o comportamento do sistema em certas situações. Ele é importante na hora da priorização de casos e para o levantamento de requisitos. Com ele temos a visão geral do projeto total.

6. Mostre a representação de casos de uso em UML com todas as suas variações. Dê um exemplo para cada uma delas.

O diagrama da UML utilizado na modelagem de casos de uso é o diagrama de casos de uso. Um modelo de casos de uso típico é formado de vários casos de usos composto de:
1. Casos de uso
2. Atores
3.Relacionamentos entre eles
   3.1: Comunicação.
   3.2: Inclusão: Ocorre quando há uma parte de comportamento que é semelhante em mais de um caso de uso
   3.3: Extensão: Utilizado para modelar situações em que diferentes sequências de interações podem ser inseridas em um caso de uso
   3.4: Generalização: Permite que um caso de uso (ou um ator) herde características de um caso de uso (ou ator) mais genérico

7. O que representa o modelo de objetos de negócio (modelo conceitual ou de domínio)? Qual a sua importância no processo de desenvolvimento de software?

Modelo conceitual, modelo mostrando os principais conceitos do sistema real, sem ainda pensarem uma solução de software. Representado por um diagrama de classes sem atributos e operações.

- entender e estruturar o comportamento dinâmico da organização na qual o sistema será implantado (organização-alvo).
- entender os problemas atuais da organização e identificar potenciais melhorias.
- garantir que clientes, usuários finais e desenvolvedores tenham uma compreensão comum da organização-alvo.

8. O que representa a arquitetura de software? Qual a importância de desenvolve-la desde o workflow de requisitos?

O que representa a arquitetura de um software é o diagrama de pacotes. A arquitetura de software é descrição abstrata, na forma de modelos, de diferentes visões do sistema em termos de unidade (partes) que interagem entre si. Arquitetura não é uma fase do desenvolvimento, mas o resultado das decisões de design sobre a estrutura e o comportamento do software. Mesmo que estas atividades não tenham sido deliberadamente realizada.