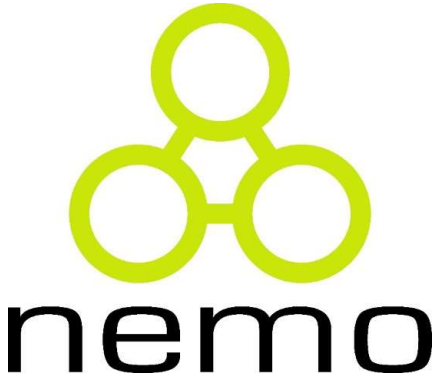


Projeto de Sistemas de Software

Jordana S. Salamon

jssalamon@inf.ufes.br

jordanasalamon@gmail.com

DEPARTAMENTO DE INFORMÁTICA

CENTRO TECNOLÓGICO

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Introdução

- Uma vez definidos e refinados todos os componentes da arquitetura de software, deve-se passar ao projeto detalhado das classes, quando se projetam detalhadamente os atributos, as associações e os métodos de cada classe.

- Neste momento, deverão ser definidos, dentre outros, as interfaces dos métodos, os algoritmos usados para implementá-los e a visibilidade de atributos, associações e métodos.

Introdução

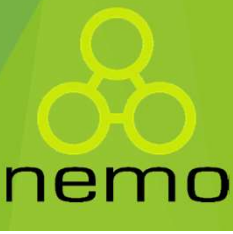
- Inicialmente, uma descrição do protocolo de cada classe deve ser estabelecida, indicando o conjunto de métodos da classe acessíveis a objetos de outras classes, i.e. sua interface pública.

- A seguir, deve-se fazer uma descrição da implementação da classe, provendo detalhes internos necessários para a implementação, mas não necessários para a comunicação entre objetos.

Introdução

- Concluído o projeto de classes, a fase de projeto pode ser considerada finalizada.

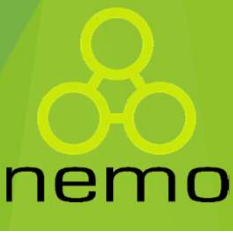
- Contudo, antes de dar como concluída essa fase, é imprescindível avaliar a qualidade do Documento de Projeto (ou Especificação de Projeto), artefato contendo os modelos produzidos e informações relevantes acerca das decisões tomadas.



Introdução

- Na verdade, não é necessário concluir a fase de projeto para avaliar a qualidade do que está sendo produzido nessa fase.

- Essa avaliação pode, e deve, ser conduzida em pontos de controle demarcados previamente, visando avaliar subprodutos da fase de projeto, tais como a arquitetura de software e os modelos dos componentes da arquitetura.



Projetando Atributos e Associações

- Tipicamente, classes de um projeto orientado a objetos são diretamente implementadas na forma de classes em uma linguagem de programação.

- As exceções ficam por conta de páginas Web tradicionais, caso as mesmas tenham sido projetadas como classes do Componente de Interface com o Usuário.

- Atributos e associações são implementados como variáveis de instância da respectiva classe.

Projetando Atributos e Associações

- No caso de atributos monovalorados, o tipo da variável de instância é um tipo básico da linguagem de programação adotada, tal como int, float e string, ou um tipo de dado de domínio previamente estabelecido, conforme definido no projeto do componente correspondente.

- No caso de associações navegáveis com multiplicidade máxima 1, a classe de origem da associação deve ter uma variável de instância do tipo da classe de destino.

Projetando Atributos e Associações

- Recomenda-se fortemente que todos os atributos e associações sejam definidos como privados.

- Como decorrência disso, a classe deve implementar métodos para acessar (get) e alterar (set) os valores de seus atributos. Aconselha-se que esses métodos sejam nomeados com o nome do atributo/associação precedido pelas palavras get e set, respectivamente.

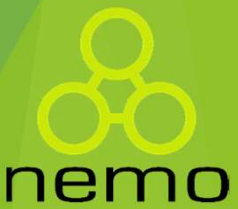
- Esse é um padrão indicado, pois é adotado pela maioria das ferramentas CASE durante a geração de código. Além disso, quando a camada de persistência é construída usando o framework Hibernate, p.ex., essa nomeação é fundamental para o funcionamento do mecanismo de persistência.

Projetando Atributos e Associações

- Wazlawick (2004) reforça que, para viabilizar o funcionamento do mecanismo de persistência, é fundamental que os atributos sejam acessados e modificados exclusivamente pelas operações get e set.

- Em hipótese alguma outro método, mesmo sendo da própria classe, poderá acessar ou modificar tais variáveis diretamente.

- Quando uma classe possui um atributo obrigatório ou uma associação navegável de multiplicidade mínima 1, seu método construtor deve ter um parâmetro do tipo definido a ser atribuída à variável de instância correspondente, de modo a manter a consistência.



Projetando Atributos e Associações

- Quando isso não ocorrer no método construtor, é necessário que a criação do objeto aconteça no contexto de uma transação que envolva também a atribuição do valor (método set).

- No caso de atributos multivalorados ou associações navegáveis com multiplicidade máxima n, a implementação é feita tipicamente por meio de uma variável de instância de um tipo de uma estrutura de dados que comporte uma coleção de elementos, tal como o tipo Conjunto (Set).

Projetando Atributos e Associações

- Para facilitar a identificação no código de que se trata de um conjunto de valores, recomenda-se utilizar o nome da variável no plural.
- Além disso, é importante garantir que a classe usada para implementar o tipo Conjunto tenha métodos para adicionar e remover elementos do conjunto.
- Nesse caso, os métodos get e set são usados para obter e atribuir a coleção inteira e não um elemento da coleção.
- Por fim, caso a coleção seja ordenada, deve-se utilizar uma estrutura de dados que trabalhe a ordenação, tal como uma lista.

Projetando Atributos e Associações

- Quando uma associação for bidirecional, i.e., navegável nos dois sentidos, ela deve ser implementada em ambas as classes (seguindo as diretrizes apresentadas anteriormente).

- Contudo, atenção especial deve ser dada ao controle da redundância, de modo a se manter a consistência.

Projetando Métodos

- No que concerne aos métodos, é necessário definir os tipos e estruturas de dados para as interfaces (parâmetros e retorno), bem como uma especificação do algoritmo de cada método.
- No caso de operações complexas, é uma boa opção dividi-las, criando métodos de nível mais baixo, estes privadas à classe.
- O projeto algorítmico de uma operação pode revelar a necessidade de variáveis locais aos métodos ou de variáveis globais à classe para tratar detalhes internos.
- Definir as estruturas de dados a serem utilizadas para essas variáveis também é parte do projeto detalhado dos métodos.

Projetando Métodos

- Para o projeto dos algoritmos, pode-se utilizar pseudocódigo.

- Contudo, há que se avaliar a utilidade de especificações detalhadas de algoritmos e se há necessidade de fazer essas especificações para todos os métodos.

- Como regra geral, essa estratégia só deve ser aplicada a métodos em que não é intuitivo saber o que se espera deles ou quando há regras de negócio específicas a serem tratadas.

Projetando Métodos

- Durante o projeto detalhado dos métodos, deve-se levar em conta que projetistas muitas vezes não conhecem os recursos da linguagem de programação adotada tão bem quanto os programadores.

- Por exemplo, as linguagens de programação oferecem uma variedade de estruturas de dados, tais como vetores, listas, filas, pilhas, conjuntos, mapas etc., e o projetista pode não fazer a melhor escolha em relação às estruturas a serem utilizadas.

Projetando Métodos

- O mesmo ocorre em relação ao projeto dos algoritmos. De fato, o projeto de métodos está na tênue fronteira entre o projeto detalhado e a implementação.

- Assim, pode ser mais produtivo deixá-lo por conta dos programadores, supervisionados pelos projetistas.

Avaliando a Qualidade do Documento de Projeto

- A qualidade de um produto de software não se atinge de forma espontânea.

- Ela deve ser construída ao longo do processo de desenvolvimento de software. Para tal, é necessário avaliar a qualidade dos diversos produtos intermediários do processo de software, dentre eles o Documento de Projeto de Software ou Especificação de Projeto.

- O Documento de Projeto é um documento valioso. Ele servirá de base para as etapas de implementação e testes. Assim, a qualidade desse documento é vital para a qualidade do produto de software resultante.

Avaliando a Qualidade do Documento de Projeto

- Diversos aspectos do projeto devem ser avaliados. É importante lembrar que há vários projetos possíveis que podem implementar corretamente um conjunto de requisitos.

- Um bom projeto equilibra custo e benefício, de modo a minimizar o custo total do sistema ao longo de seu tempo de vida total.

- Existem na literatura alguns critérios propostos para avaliação da qualidade de projetos orientados a objetos, dentre eles:

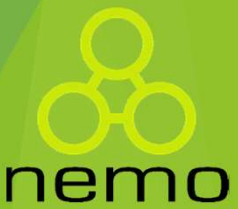
Avaliando a Qualidade do Documento de Projeto

- • *Acoplamento*: acoplamento diz respeito ao grau de interdependência entre componentes de software.

- O objetivo é minimizar o acoplamento, isto é, tornar os componentes tão independentes quanto possível.

- No projeto orientado a objetos, deve ser avaliado tanto o acoplamento entre classes como o acoplamento entre subsistemas.

- A meta é minimizar o número de mensagens trocadas e a complexidade e o volume de informação nas mensagens.



Avaliando a Qualidade do Documento de Projeto

- • *Coesão*: define como as atividades de diferentes componentes de software estão relacionadas umas com as outras. Vale a pena ressaltar que coesão e acoplamento são interdependentes e, portanto, uma boa coesão, geralmente, conduz a um pequeno acoplamento.

- No projeto orientado a objetos, três níveis de coesão devem ser verificados:

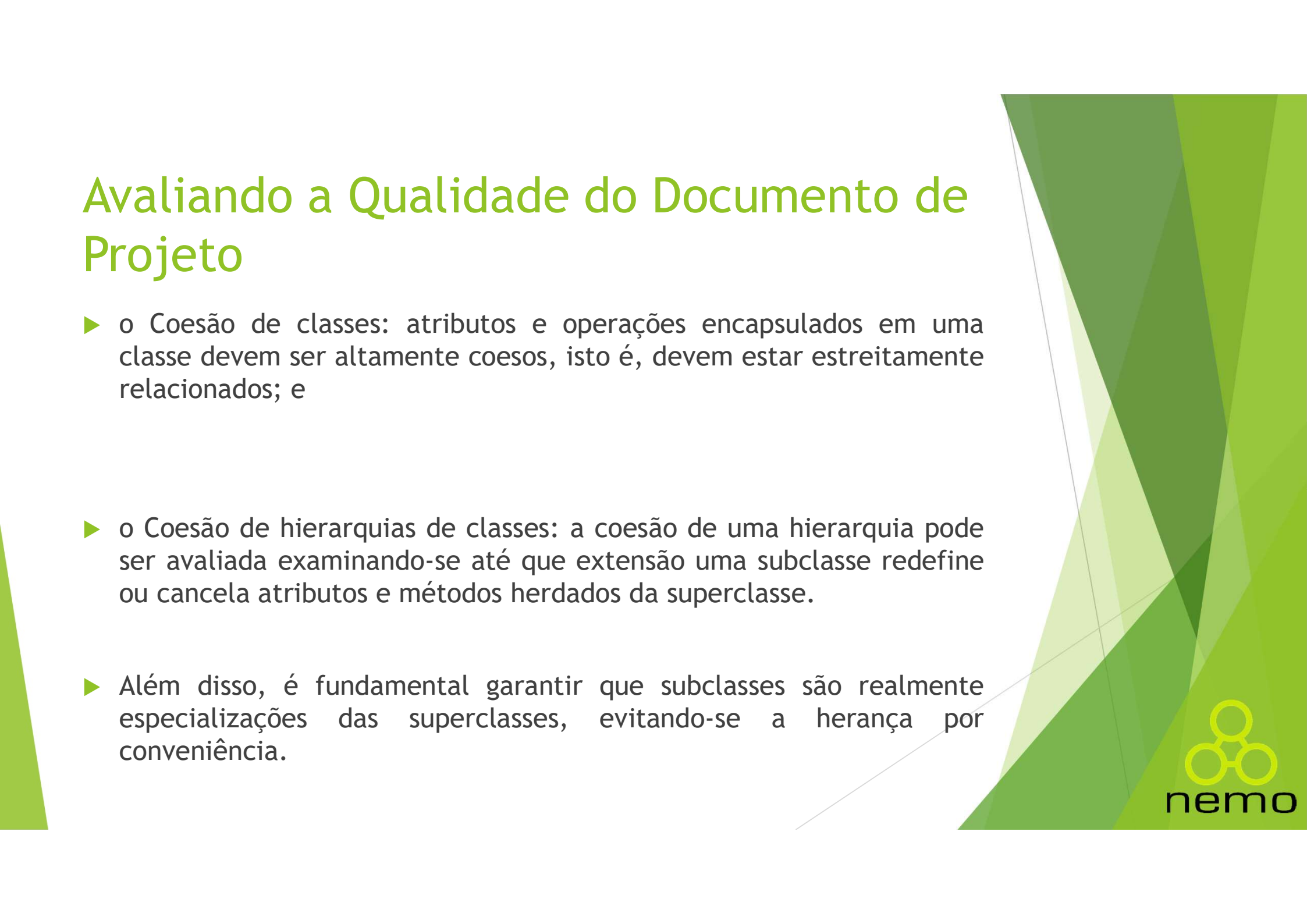
- o Coesão de métodos individuais: um método deve executar uma e somente uma função;

Avaliando a Qualidade do Documento de Projeto

- o Coesão de classes: atributos e operações encapsulados em uma classe devem ser altamente coesos, isto é, devem estar estreitamente relacionados; e

- o Coesão de hierarquias de classes: a coesão de uma hierarquia pode ser avaliada examinando-se até que extensão uma subclasse redefine ou cancela atributos e métodos herdados da superclasse.

- Além disso, é fundamental garantir que subclasses são realmente especializações das superclasses, evitando-se a herança por conveniência.

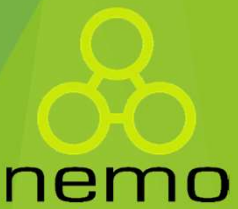


Avaliando a Qualidade do Documento de Projeto

- • *Clareza*: um projeto deve ser passível de entendimento por programadores, testadores e outros projetistas.

- • *Reutilização*: bons projetos devem ser fáceis de serem reutilizados e devem, sempre que possível, reutilizar porções de projeto previamente elaborados e padrões de projeto (padrões arquitetônicos e design patterns) consagrados.

- • *Efetivo Uso da Herança*: para sistemas médios, hierarquias de classe não devem ter mais do que sete níveis de generalização-especialização. Projetos com uso intensivo de herança múltipla devem ser evitados, pois são mais difíceis de serem entendidos e, consequentemente, de serem reutilizados e mantidos.



Avaliando a Qualidade do Documento de Projeto

- • *Protocolo de Mensagens Simples*: protocolos de mensagem complexos são uma indicação comum de acoplamento excessivo entre classes.

- Assim, a passagem de muitos parâmetros deve ser evitada.

- • *Métodos Simples*: os métodos que implementam as operações de uma classe devem ser mantidos pequenos.

- Se um método envolve muito código, há uma indicação de que as operações da classe foram pobremente divididas.

Avaliando a Qualidade do Documento de Projeto

- • *Habilidade de "avaliar por cenário"*: é importante que um projeto possa ser avaliado a partir de um cenário particular escolhido.

- Revisores devem poder representar o comportamento de classes e objetos individuais e, assim, verificar o comportamento dos objetos nas circunstâncias desejadas.

- Essa característica é igualmente importante para testar posteriormente o produto de software.

Avaliando a Qualidade do Documento de Projeto

- Assim como os demais documentos produzidos ao longo do processo de software, o Documento de Projeto deve ser revisado. Durante uma revisão desse documento, os seguintes aspectos devem ser observados:

- • Aderência a padrões de documento de projeto estabelecidos pela organização.
- • Aderência a padrões de nomenclatura estabelecidos pela organização, incluindo nomes de classes, atributos, métodos, pacotes etc.
- • Coerência com os modelos de análise e de especificação de requisitos.

nemo

Avaliando a Qualidade do Documento de Projeto

- Do ponto de vista de coerência entre modelos, os seguintes aspectos devem ser observados:

- • As classes da Camada de Lógica de Negócio devem ser necessárias e suficientes para cumprir as responsabilidades apontadas pelos casos de uso do documento de especificação de requisitos, agora já com uma perspectiva de implementação, i.e., levando-se em conta os requisitos não funcionais.

- • As classes da Camada de Interface com o Usuário devem ser necessárias e suficientes para permitir o acesso e a realização de todos os casos de uso do documento de especificação de requisitos.

Avaliando a Qualidade do Documento de Projeto

- • As classes da Camada de Gerência de Dados devem ser necessárias e suficientes para tratar do armazenamento e recuperação de objetos de todas as classes persistentes do sistema (tipicamente, as classes do Componente de Domínio do Problema).

- • Alterações não decorrentes da tecnologia, mas da detecção de um erro na especificação de requisitos ou na análise devem ser feitas nos correspondentes modelos de especificação e análise de requisitos. Não basta alterar o documento de projeto. Lembre-se que é fundamental manter a coerência entre os modelos de análise e projeto.

That’s all Folks!