UNIVERSIDADE ESTADUAL
VALE DO ACARAÚ

GOVERNO DO
ESTADO DO CEARÁ
Secretaria da Ciência, Tecnologia
e Educação Superior

NEaD UVA
Núcleo de Educação a Distância
Universidade Estadual Vale do Acaraú

Engenharia de Software

Prof. Thales Damasceno

5º Bloco

Engenharia de Software

Projeto no Nível Componentes

TÓPICOS:

- INTRODUÇÃO
- COMPONENTES DE SOFTWARE
- PROJETO DE COMPONENTES BASEADOS EM CLASSES
- CONDUÇÃO DO PROJETO NO NIVEL DE COMPONETES
- CONCLUSÃO

Participação dos alunos:

Francinaldo Pinto

Lauro César

Leandro Xavier

5º Bloco

Engenharia de Software

Introdução

- Como surgiu?
- Por que?
- O Que é?
- Quais os custo?

5º Bloco

Engenharia de Software

Projeto no Nível Componentes

TÓPICOS:

- INTRODUÇÃO
- **COMPONENTES DE SOFTWARE**
- PROJETO DE COMPONENTES BASEADOS EM CLASSES
- CONDUÇÃO DO PROJETO NO NIVEL DE COMPONETES
- CONCLUSÃO

5º Bloco

Engenharia de Software

O que é um componente de Software?

- Estrutura Modular
- Diversas Funcionalidades
- Encapsulamento
- Unidade independente
- Sistemas de *Componentes*

Oblo Spheres

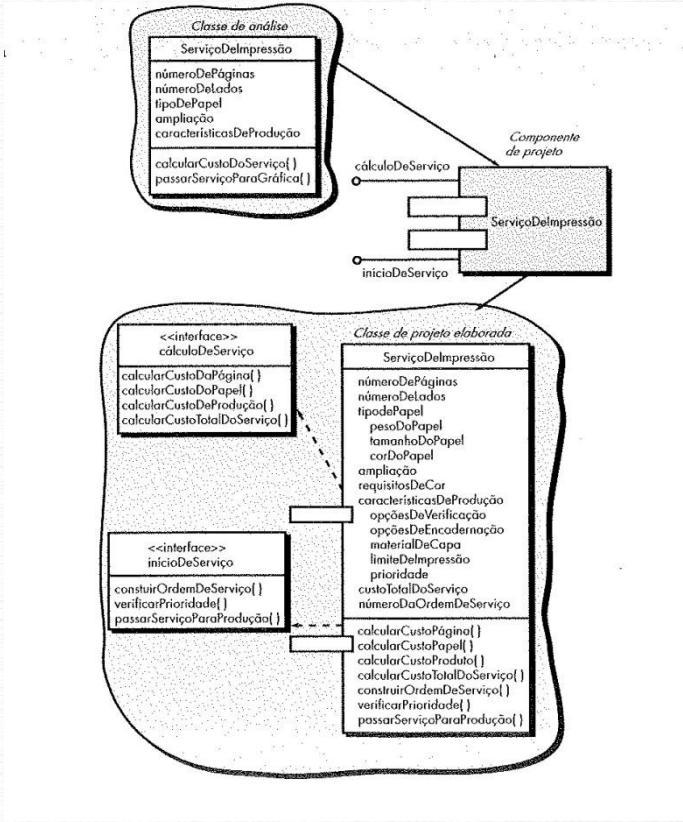
Ex[1]: Programação Orientada a Objeto – É a ***Classe*** que implemente a Interface

5º Bloco

Engenharia de Software

Uma Visão Orientada a Objeto

- Baseado em Classes
- Conjuntos de Classes colaboradoras
- Comunicação entre as interfaces entre as classes
- Componentes que fornecem serviços

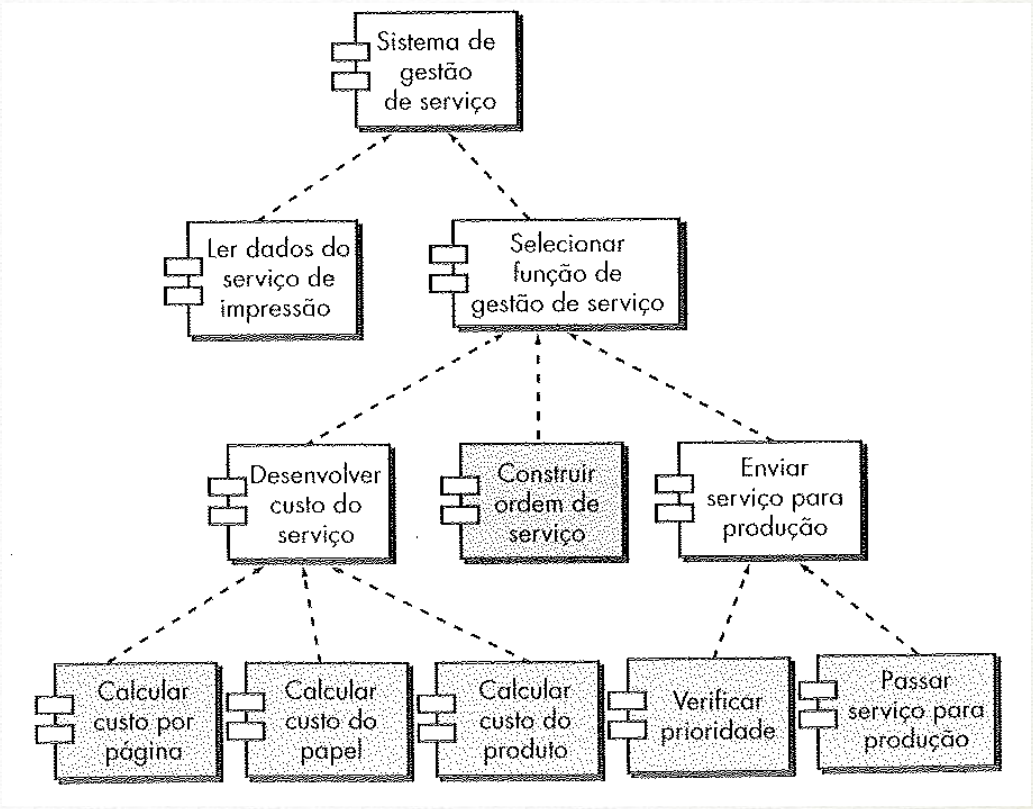


5º Bloco

Engenharia de Software

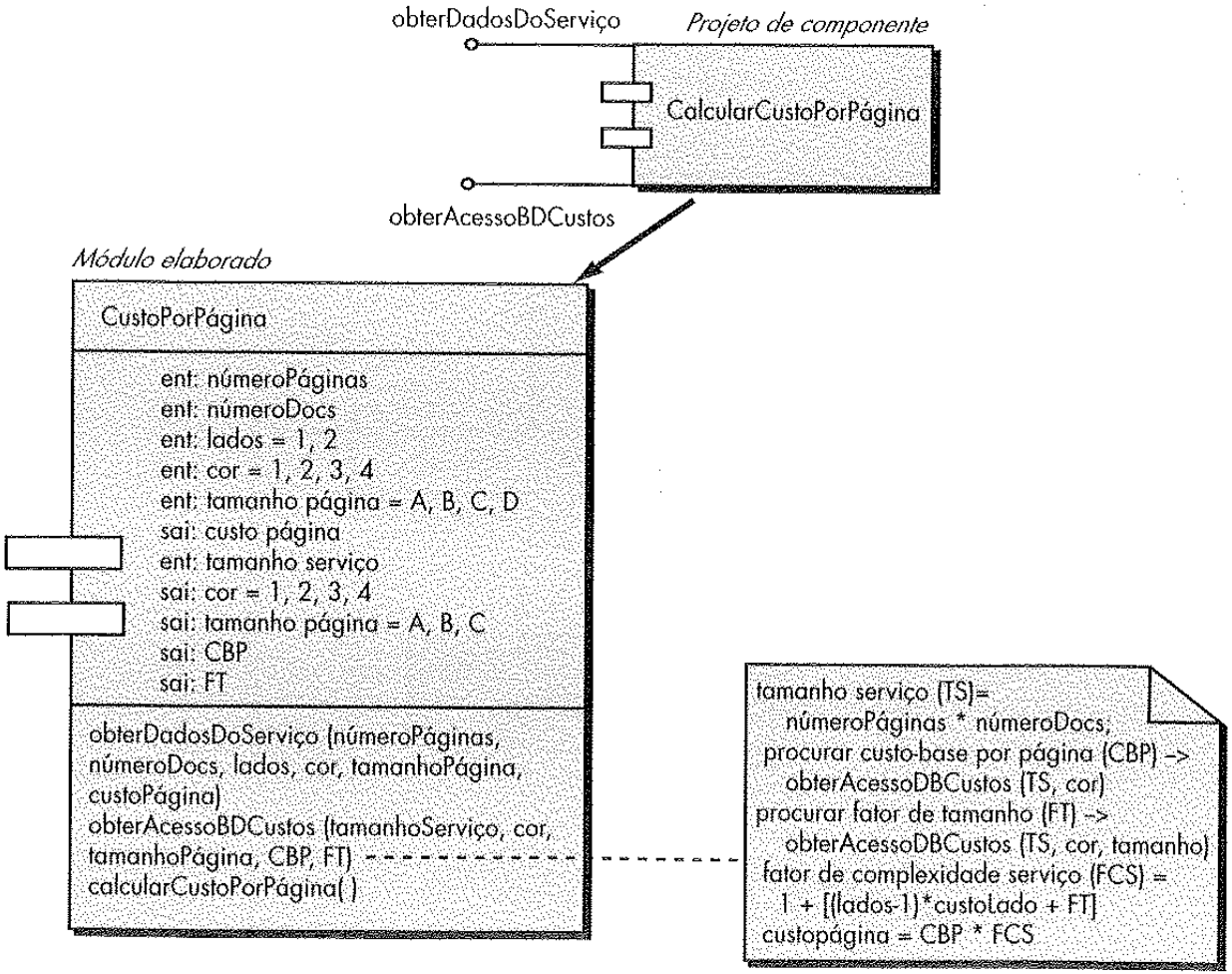
A Visão Convencional

- Elemento Funcional
  - Controle
  - Domínio
  - Infra-Estrutura
- Hierarquia de Módulos



5º Bloco

Engenharia de Software



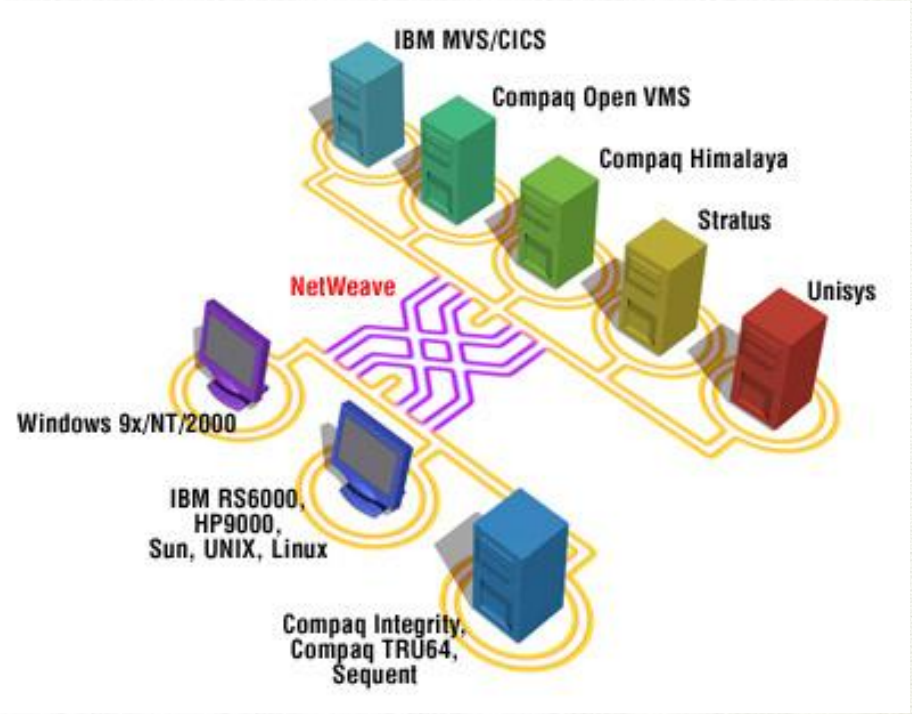
Projeto no Nível Componente

5º Bloco

Engenharia de Software

Uma visão relacionada a Processo

- Modelos anteriores
- Necessidade da reusabilidade
- Middleware
  - CORBA
  - Ginga
  - RemoteSync



Projeto de Componentes Baseado em Classes

- Componentes fornecem funcionalidades-alvo com interface definida;
- Produtos baseados em componentes disponíveis são pesquisados e avaliados para o domínio da aplicação em questão;
- Tópicos de integração de componentes são considerados;
- Uma arquitetura de *software* é projetada para acomodar os componentes;

Projeto de Componentes Baseado em Classes

- Componentes são integrados à arquitetura;
- Testes abrangentes são realizados para garantir a funcionalidade adequada.

O Modelo de Desenvolvimento Baseado em Componentes leva ao reuso de *software*, e a reusabilidade fornece aos engenheiros vários benefício mensuráveis.

5º Bloco

Engenharia de Software

Projeto no Nível Componentes

TÓPICOS:

- INTRODUÇÃO
- COMPONENTES DE SOFTWARE
- **PROJETO DE COMPONENTES BASEADOS EM CLASSES**
- CONDUÇÃO DO PROJETO NO NIVEL DE COMPONETES
- CONCLUSÃO

Diretrizes para projeto no nível de componente

Um conjunto de diretrizes pragmáticas de projeto pode ser aplicado a medida que o projeto no nível de componentes prossegue, essas aplicam-se a componentes, interfaces e características de dependências e de herança que tem um impacto sobre o projeto resultante.

Diretrizes para projeto no nível de componente

- **Componentes**: convenções de nomes devem ser estabelecidas para componentes específicos, os nomes devem ter significado para fácil entendimento de pessoas sem o conhecimento técnico.

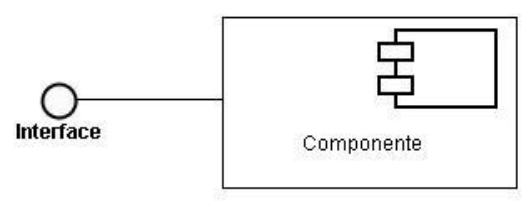
- Também podemos usar **estereótipos** para ajudar a identificar a natureza dos componentes, Por Exemplo, <<infra-estrutura>>, <<bancodedados>> e <<tabela>>.

Diretrizes para projeto no nível de componente

Interfaces fornecem uma importante informação sobre comunicação e colaboração, de acordo com Ambler [AMB02] recomenda-se:

- **O uso da representação pirulito de uma interface no lugar da abordagem UML formal de caixas e setas tracejadas**

ex:



- **Interfaces devem fluir da esquerda para a direita da caixa do componente para caixa do componente.**
- **O uso apenas daquelas interfaces relevantes para determinado componente.**

5º Bloco

Engenharia de Software

Diretrizes para projeto no nível de componente

Dependência e herança para melhorar a legibilidade, é aconselhável modelar dependências da esquerda para a direita, e herança de baixo para cima (subclasse e superclasse respectivamente). Além disso interdependência de componentes devem ser representadas via interfaces em vez de representação de dependência de componente a componente.

5º Bloco

Engenharia de Software

Coesão

Coesão está ligado ao princípio da responsabilidade única, que foi introduzido por Robert C. Martin no inicio dos anos 2000 e diz que uma classe deve ter apenas uma única responsabilidade e realizá-la de maneira satisfatória, ou seja, uma classe não deve assumir responsabilidades que não são suas .

- É uma medida de força funcional relativa de um módulo.
- Um módulo coeso executa uma única tarefa, exigindo pouca interação com outros módulos.
- Alta Coesão é o Desejável

5º Bloco

Engenharia de Software

Coesão

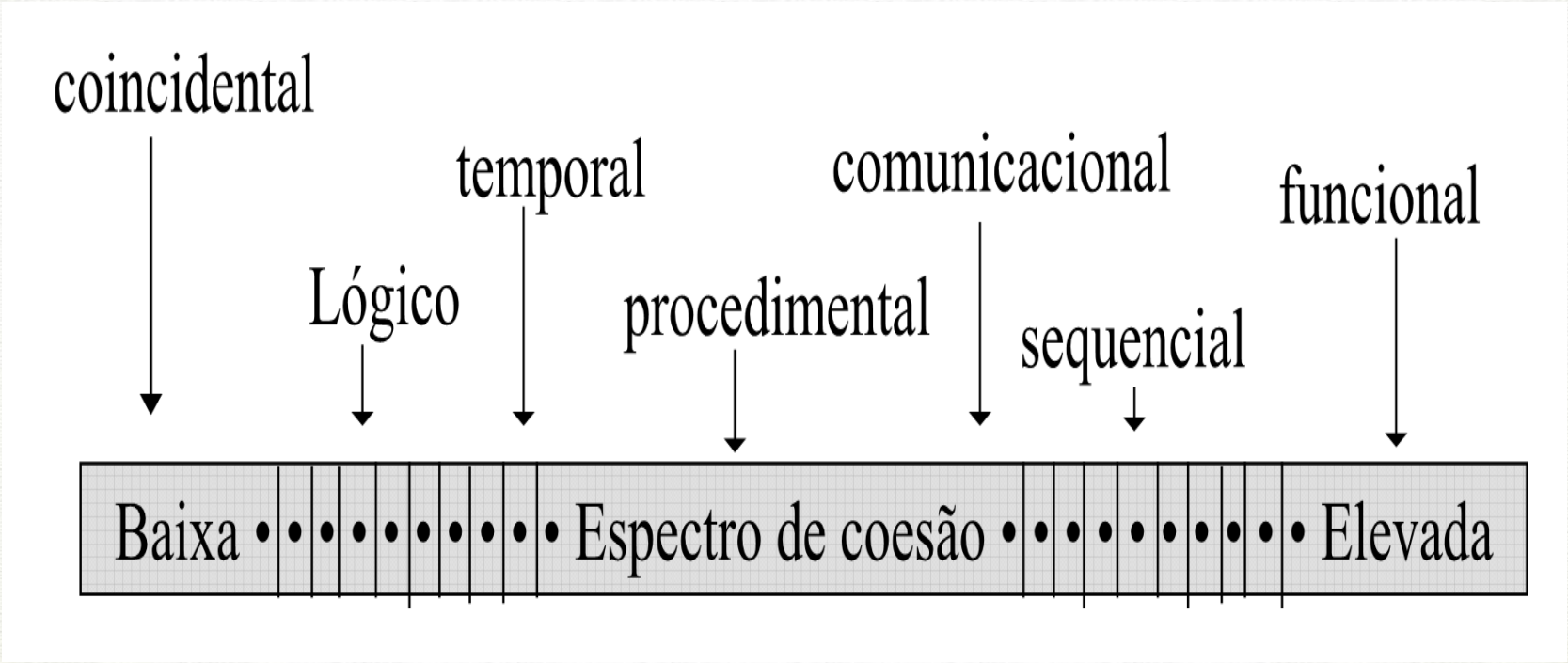
Uma classe com baixa coesão faz muitas coisas não relacionadas e leva aos seguintes problemas:

- Difícil de entender
- Difícil de reusar
- Difícil de manter
- Problemas com mudanças

5º Bloco

Engenharia de Software

Coesão



Fonte: http://pt.scribd.com/doc/48925116/acoplamento-e-coesao

Coesão (exemplo)

A classe Programa tem responsabilidades que não são suas, como obter um produto e gravá-lo no banco de dados. Então, dizemos que esta classe não está coesa, ou seja, ela tem responsabilidades demais, e o que é pior, responsabilidades que não são suas.

```
public class Programa
{
  public void ExibirFormulario()  {
    //implementação
  }

  public void ObterProduto()  {
    //implementação
  }

  public void gravarProdutoDB  {
    //implementação
  }
}
```

Fonte: http://www.devmedia.com.br/entendendo-coesao-e-acoplamento/18538

5º Bloco

Engenharia de Software

Coesão (exemplo)

Neste exemplo, uma clara separação de responsabilidades, o que contribui para um design desacoplado e organizado. O formulário não assume o papel de cadastrar o produto, ele pede a quem tem a responsabilidade para que faça tal tarefa. O que temos que ter em mente é que uma classe deve ser responsável por exercer uma única responsabilidade e fazer outras classes cooperarem quando necessário.

```
public class Programa

{

    public void MostrarFormulario()    {

     //Implementação

    }


   public void BotaoGravarProduto( ) {

     Produto.gravarProduto();

   }


}
```

Fonte: http://www.devmedia.com.br/entendendo-coesao-e-acoplamento/18538

Acoplamento

- O acoplamento significa o quanto uma classe depende da outra para funcionar. Quanto maior for a dependência entre ambas, podemos dizer que as classes estão fortemente acopladas.

- O forte acoplamento nos traz muitos problemas.

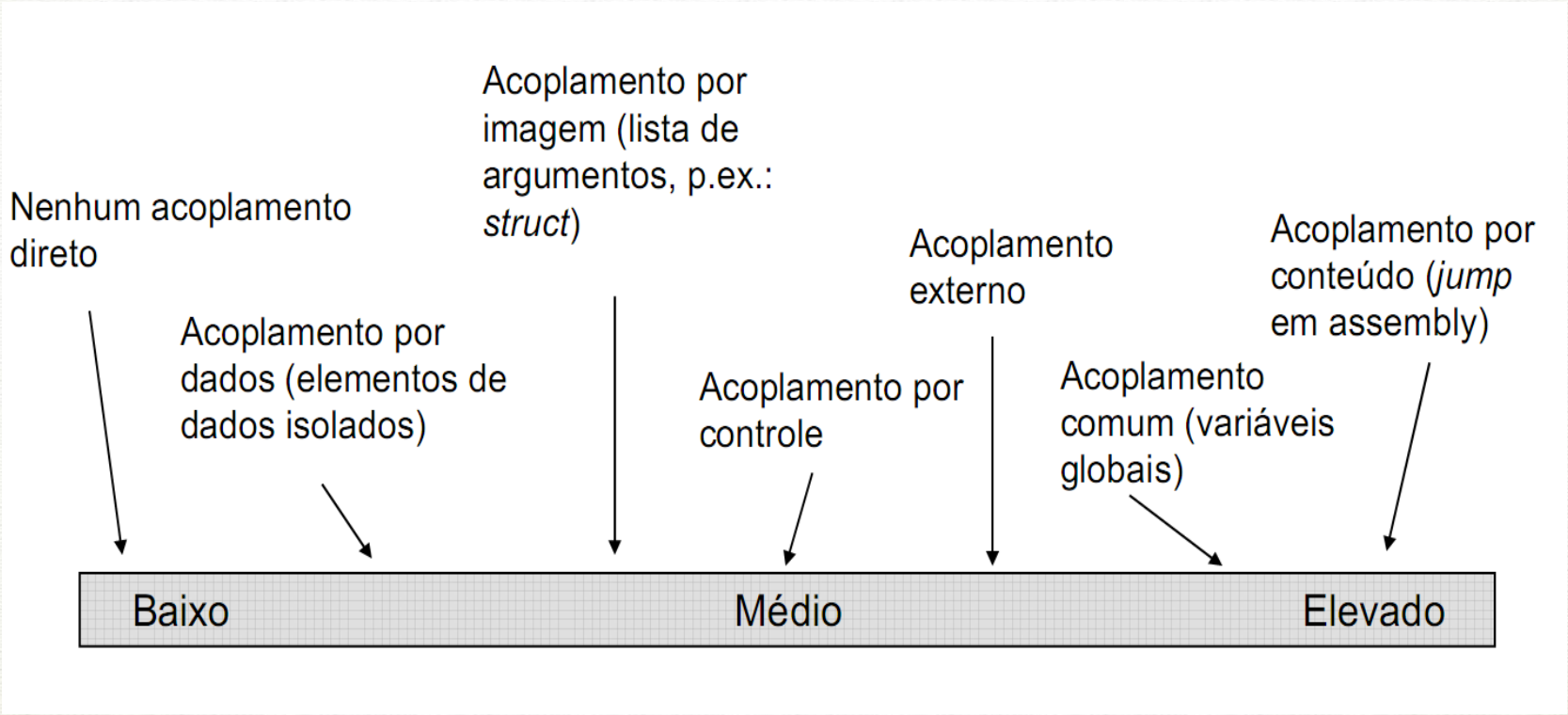
Acoplamento

- O acoplamento significa o quanto uma classe depende da outra para funcionar. Quanto maior for a dependência entre ambas, podemos dizer que as classes estão fortemente acopladas.
- O forte acoplamento nos traz muitos problemas.
- É uma medida da interdependência relativa entre os módulos.
- Depende da complexidade de interface entre os módulos.
- Baixo Acoplamento é o desejável.

5º Bloco

Engenharia de Software

Acoplamento

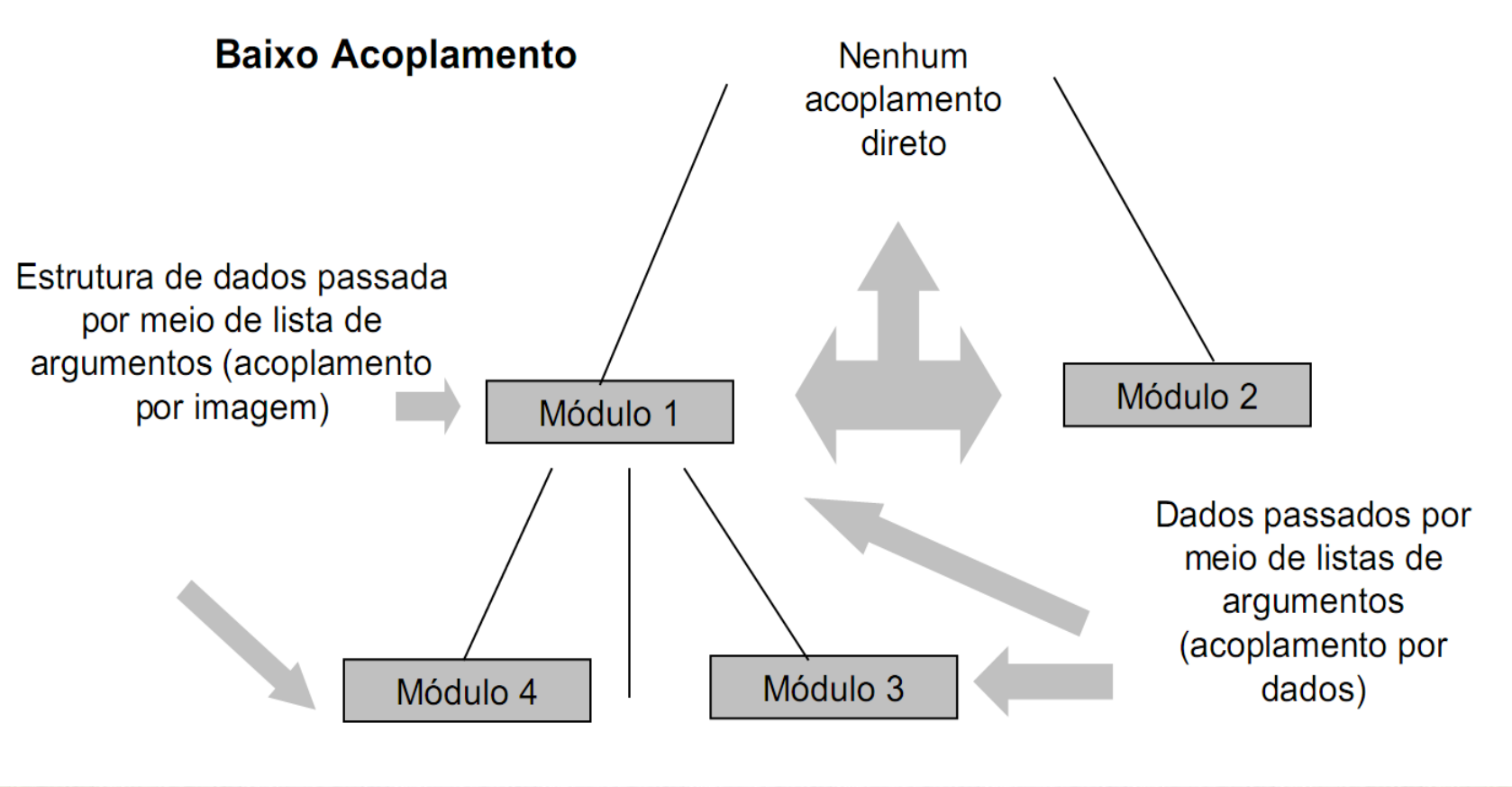


Fonte: http://pt.scribd.com/doc/48925116/acoplamento-e-coesao

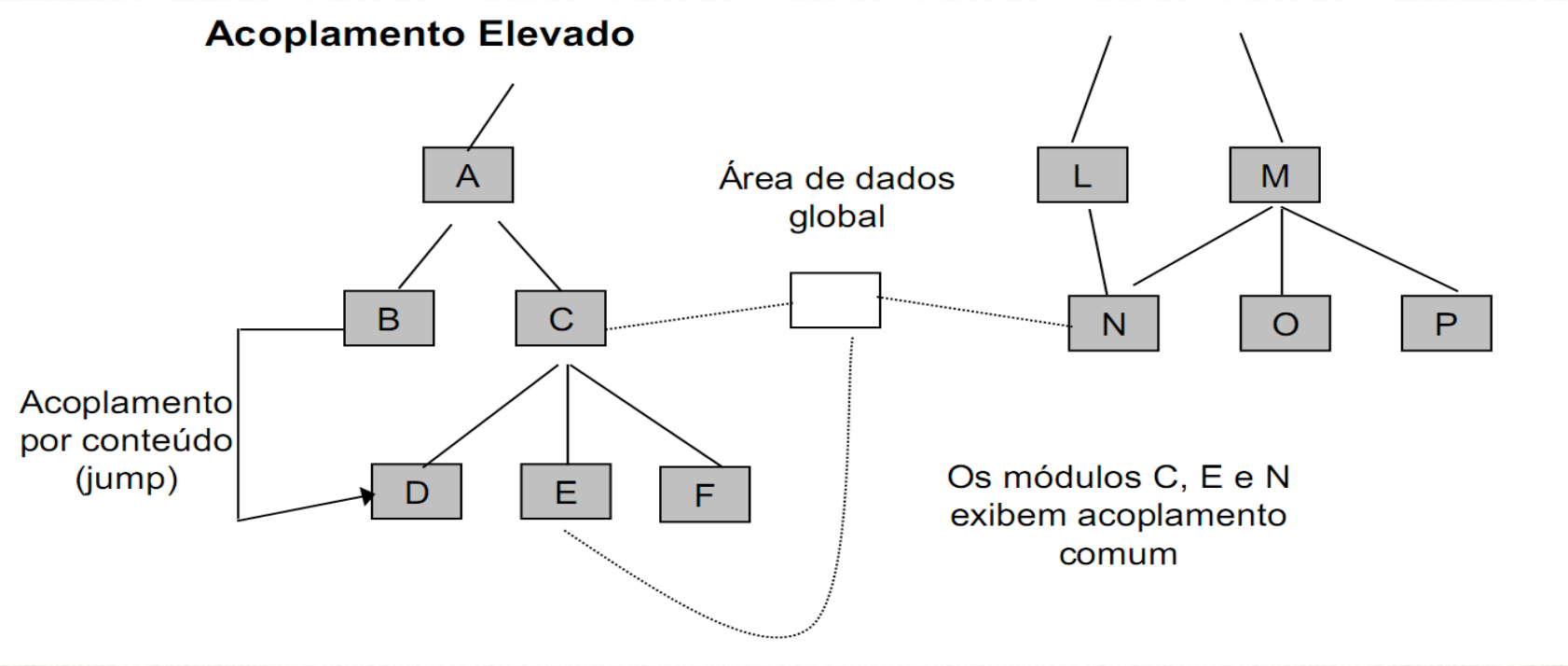
5º Bloco

Engenharia de Software

Acoplamento (exemplos)

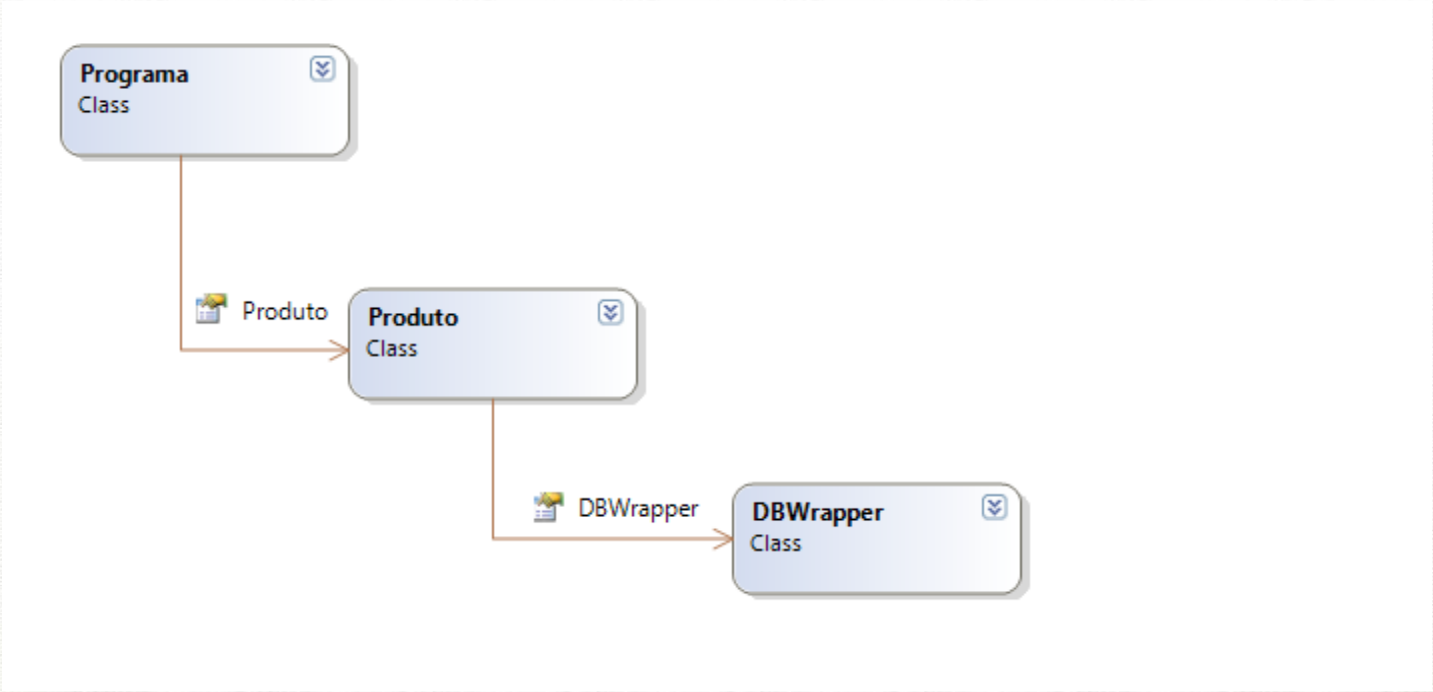


Acoplamento (exemplos)



Fonte: http://pt.scribd.com/doc/48925116/acoplamento-e-coesao

Acoplamento (exemplos)



Fonte: http://www.devmedia.com.br/entendendo-coesao-e-acoplamento/18538

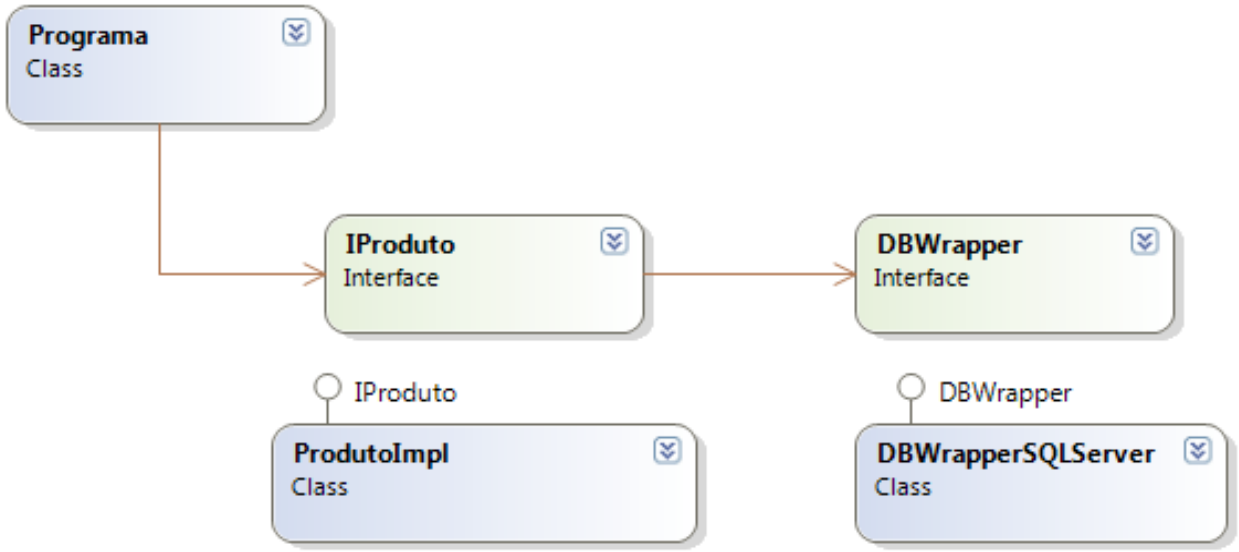
Acoplamento (exemplos)

Nessa cadeia de classes, o forte acoplamento na mesma, torna muito custoso a sua manutenção e o seu gerenciamento, pois qualquer mudança vai afetar toda a cadeia de classes.

Solução:

- A solução para o alto acoplamento é utilizar abstrações, assim se eu precisar mudar alguma coisa eu não quebro a relação, apenas direciono para a nova implementação.
- Um exemplo disso e se eu precisar mudar o banco SQLServer para um Oracle ou outro, eu não tenho medo da mudança como vemos no exemplo.

Acoplamento (Exemplo)



Fonte: http://www.devmedia.com.br/entendendo-coesao-e-acoplamento/18538

5º Bloco

Engenharia de Software

COESÃO X ACOPLAMENTO

É o nível de inter-dependência entre os módulos de um software. O acoplamento está diretamente relacionado com a coesão, pois quanto maior for o acoplamento menor será o nível de coesão. Isto se deve ao fato de que, quando um módulo ou classe possui uma dependência muito forte por outro módulo ou classe, ele não é "forte" o suficiente para desempenhar suas tarefas de forma individual, dificultando, por exemplo, alterações no código da sua dependência (módulo ou classe de serviço). A coesão é a medida da força relativa de um módulo. Quanto maior for a coesão, menor será o nível de acoplamento de um módulo

5º Bloco

Engenharia de Software

Projeto no Nível Componentes

TÓPICOS:

- INTRODUÇÃO
- COMPONENTES DE SOFTWARE
- PROJETO DE COMPONENTES BASEADOS EM CLASSES
- **CONDUÇÃO DO PROJETO NO NIVEL DE COMPONETES**
- CONCLUSÃO

5º Bloco

Engenharia de Software

Condução do Projeto no Nível de Componente

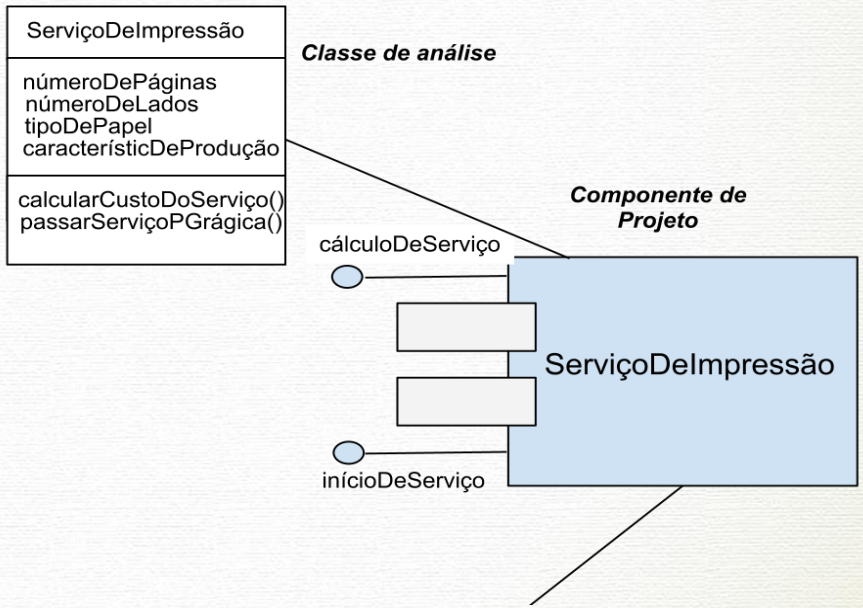
Após ser obtidas os modelos de análise e arquitetural, o projetista deve transformar estas informações em uma representação de projeto que sirva de base durante todo o processo de construção do sistema (codificação e teste).

Os seguintes passos são tarefas típicas quando se é utilizado a um sistema que se utiliza do paradigma de orientação a objetos.

Passo 1: Identificar todas as classes de projeto que correspondam ao domínio do problema.

Como na visão orientada a objetos cada componente contém um conjunto de classes colaborativas, então deve-se elaborar todas as classes de análise que correspondam ao domínio do problema.

Ao lado um exemplo de classe **ServiçoDeImpressão**



Passo 2: Identificar todas as classes de projeto que correspondam ao domínio de infraestrutura.

Essas classes não são elaboradas no modelo de análise, mas neste passo. Classes e componentes de infraestrutura normalmente são componentes de IGU, componentes de sistema operacionais, componentes de gestão de objetos e dados e outros.

Exemplos: classes que fazem chamadas ao sistema operacional para determinar função, classes de acessos a banco de dados, etc.

5º Bloco

Engenharia de Software

Passo 3: Elaborar todas as classes de projeto que não são adquiridas como componentes reusáveis.

- Este passo requer que todas as interfaces, atributos e operações necessárias para implementar a classe sejam descritas em detalhes.
- Deve-se elaborar as classes que não focam o domínio problema, nem classes de infra-estrutura.
- Heurísticas (formas de solucionar um problema) de projetos com coesão e acoplamento de componentes devem ser levados em conta.

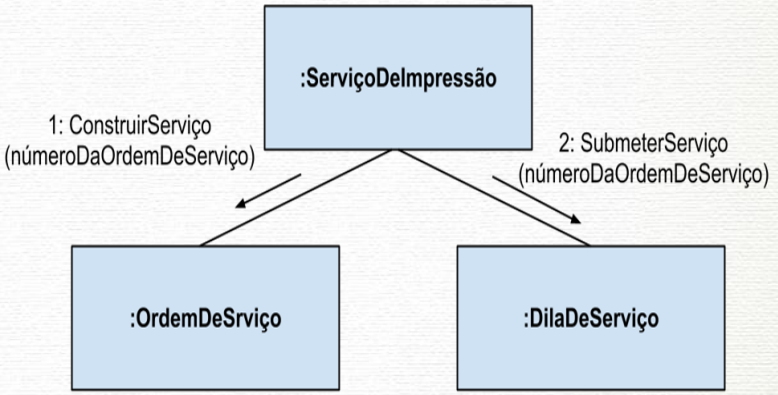
5º Bloco

Engenharia de Software

Passo 3a: Especificar detalhes de mensagens quando classes ou componentes colaboram.

Como o modelo de análise usa diagramas para demonstrar as mensagens que as classes e componentes utilizam para se comunicarem, é necessário à medida que o projeto prossegue mostrar os detalhes dessas mensagens, ou seja, mostrar os detalhes das colaborações entre os objetos de um sistema

A figura abaixo ilustra um exemplo de diagrama de colaboração simples para um sistema de gráfica.



Mensagens são passadas como é ilustrado por setas da figura. Á medida que o projeto prossegue, cada mensagem é elaborada expandindo sua sintaxe da seguinte modo em OCL

[condição de guarda] expressão de sequência (valor de retorno):= nome da mensagem (lista de argumentos)

5º Bloco

Engenharia de Software

Passo 3b: Identificar interfaces adequadas para cada componente.

Uma interface é o equivalente a uma classe abstrata que disponibiliza uma forma de as classes se conectarem através de suas operações.

Todas as operações, ou seja, as formas de conexão em uma classe abstrata (interface) deve ser coesiva; isto é, deve exibir processamento que enfoca funções limitadas, devem ter “exclusividade de enfoque”.

5º Bloco

Engenharia de Software

Passo 3c: Elaborar atributos e definir tipos de dados e estruturas de dados necessários para implementá-los.

Muitas das vezes as estruturas e tipos de dados são implementados de acordo com a liguagem de programação que será utilizada.

Em UML define-se um tipo de dados de atributo por meio da seguinte sintaxe:

nome: tipo-expressão = valor inicial {cadeia de propriedade}
Exemplo: *peso-TipoDoPapel = "A" {contém 1 de 4 valores – A, B, C ou D}*

Nome: *N*ome do atributo.

Tipo-expressão: Tipo de dados.

Valor inicial: *V*alor que o atributo assume quando o objeto é criado.

Cadeia de propriedade: Define uma propriedade ou conjunto de características do atributo.

Passo 3d: Descrever fluxo de processamento em cada operação em detalhe.

Deve ser descrito em detalhes toda a sequência algorítmica das operações, isso pode ser realizado através da utilização de uma linguagem de programação baseada em pseudocódigo (por exemplo: Portugol) ou com um diagrama de atividade UML.

5º Bloco

Engenharia de Software

Passo 4: Descrever fontes de dados persistentes (bancos de dados e arquivos) e identificar as classes necessárias para geri-los.

Em muitos casos esses mecanismos de armazenamento de dados persistentes são inicialmente especificados como parte do projeto arquitetural. No entanto, à medida que o projeto prossegue é útil especificar detalhes sobre a estrutura e organização desses fontes de dados persistentes.

Por exemplo especificar qual a tecnologia de dados utilizados (banco de dados ou arquivos); caso seja banco de dados deve-se indicar qual o SGDB(Sistema Gerenciador de Banco de Dados) utilizado. Além de listar todas as classes responsáveis por acesso ao dados do sistema.

5º Bloco

Engenharia de Software

Passo 5: Desenvolver e elaborar representações comportamentais para uma classe ou componente

O comportamento dinâmico de um objeto, ou seja, uma instância de uma classe de projeto quando o programa é executado, é afetado por eventos externos a ele e pelo seu estado atual.

Pode se entender melhor o comportamento de uma classe examinando todos os casos de uso em que tratam dessa classe de modo relevante.

A transição de um estado para outro que ocorre como consequência de um evento, pode ser representado na forma (em OCL):

nome-evento(lista-parâmetros)[condição-de-guarda]/expressão de ação

nome-evento: identifica o nome

Lista-parâmetros: dados associados ao evento

Condição-de-guarda: condição que deve ser satisfeita para que o evento ocorra

expressão de ação: ação que ocorre quando a condição é verdadeira.

5º Bloco

Engenharia de Software

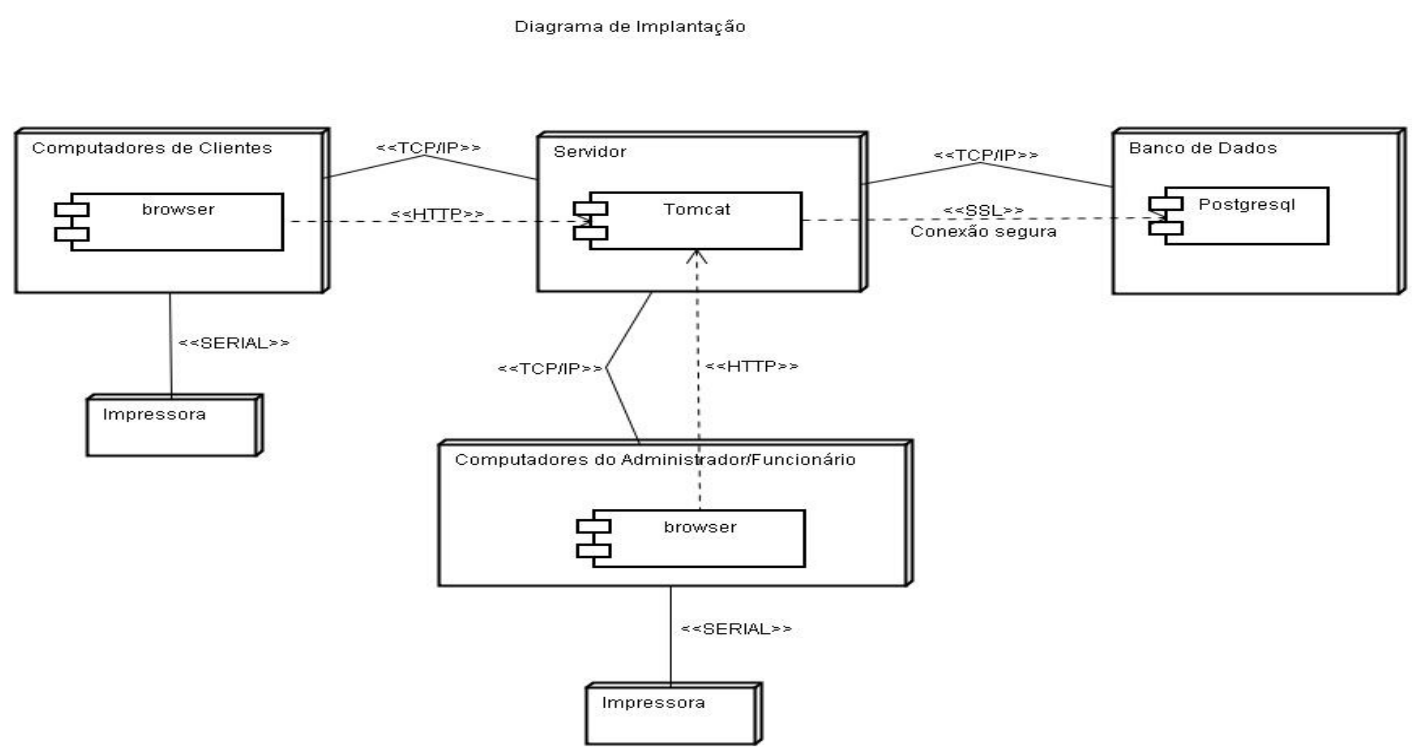
Passo 6: Elaborar diagramas de implantação para fornecer detalhe adicional de implementação.

Diagramas de implantação (como já vimos) são usados como parte do projeto arquitetural. As funções principais do sistema (algumas vezes representado como subsistemas) são representados de acordo com o ambiente específico de hardware e sistema operacional a ser usado e a localização de pacotes nesse ambiente é especificado.

5º Bloco

Engenharia de Software

Abaixo um exemplo de diagrama de implantação, em que mostra quais os tipos de SO, SGBD, dentre outras informações do ambiente computacional de um projeto.



5º Bloco

Engenharia de Software

Passo 7: Fabricar toda a representação do projeto no nível de componente e sempre considerar alternativas.

Nesta fase é feita toda a implementação do projeto.

É essencial também recriar o modelo no nível de componente à medida que o projeto é implementado, pois o primeiro não será tão tão completo, consistente ou preciso .

Além disso, o projetista não deve ter uma visão restrita e definitiva acerca do domínio do problema do sistema. Deve-se sempre haver outras soluções alternativas de projetos.

5º Bloco

Engenharia de Software

Projeto no Nível Componentes

TÓPICOS:

- INTRODUÇÃO
- COMPONENTES DE SOFTWARE
- PROJETO DE COMPONENTES BASEADOS EM CLASSES
- CONDUÇÃO DO PROJETO NO NIVEL DE COMPONENTES
- **LINGUAGEM DE RESTRIÇÕES DE OBJETO**
- CONCLUSÃO

Linguagem de Restrição de Objeto

Apesar de existirem uma grande variedade de diagramas disponíveis, representações gráficas são insuficientes para representar de forma explícita e formal as informações de algum elemento do modelo de projeto.

Uma linguagem mais apropriada para isso deve se apoiar nos conceitos de teoria dos conjuntos e ter uma sintaxe menos matemática do que sintaxes de LPs.

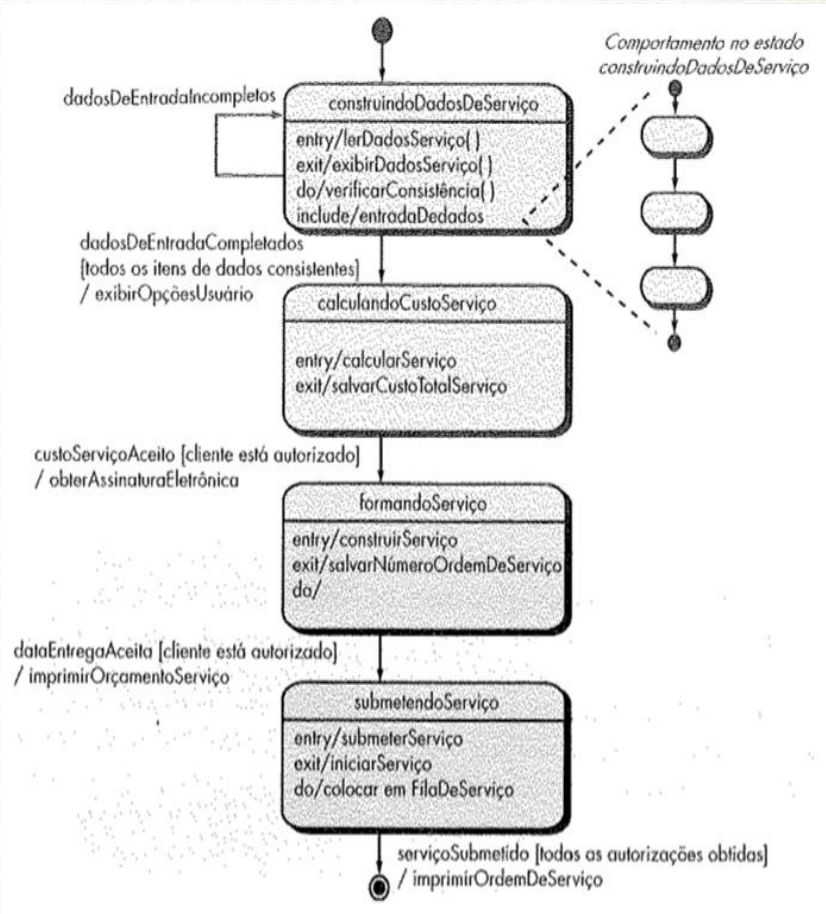
A OCL complementa os diagramas em UML e permite aos engenheiros de software utilizar uma gramática e sintaxe formal que não possibilite haver declarações ambíguas sobre os elementos do projeto, tais como classes, objetos, eventos, mensagens e interfaces.

5º Bloco

Engenharia de Software

Linguagem de Restrição de Objeto

Como exemplo de uma declaração na linguagem OCL, considere a condição de guarda colocada no evento *custoServicoAceito* que causa uma transição entre os estados **calculandoCustoServiço** e **formandoServiço** no diagrama de estados para a classe **ServiçoDeImpressão** como mostrado na figura ao lado:



5º Bloco

Engenharia de Software

Linguagem de Restrição de Objeto

Em OCL, a expressão pode tomar a forma:

cliente

self. AutoridadeAutorização = 'sim'

Em que **autoridadeAutorização** é um atributo booleano da classe **Cliente** que deve ser fixado como **sim** para que a condição de guarda seja satisfeita

A especificação OCL completa pode ser encontrada em

http://www.omg.org/

5º Bloco

Engenharia de Software

Projeto no Nível Componentes

TÓPICOS:

- INTRODUÇÃO
- COMPONENTES DE SOFTWARE
- PROJETO DE COMPONENTES BASEADOS EM CLASSES
- CONDUÇÃO DO PROJETO NO NIVEL DE COMPONETES
- LINGUAGEM DE RESTRIÇÕES DE OBJETO
- **CONCLUSÃO**

5º Bloco

Engenharia de Software

Conclusão

- Tarefas Nível de Abstração de dados

- Visões
  - Orientada a Objeto
  - Convencional

"Os detalhes não são detalhes. Eles formam o Projeto"

Charles Eames

5º Bloco

Engenharia de Software

Referencias Bibliográficas

- **Livros e Artigos**
- PRESSMAN, R. S.. Engenharia de Software. Makron Books. 1995
- [LET01] Lethbrige, T. e Laganiere, R., Object-Oriented Software Engineering: Pratical Software
- Development Using UML and Java, McGraw-Hill, 2001.
- **Sites:**
- http://pt.scribd.com/doc/48925116/acoplamento-e-coesao
- http://ptrecenti.blogspot.com.br/2009/12/coesao-e-acoplamento.html
- http://www.devmedia.com.br/entendendo-coesao-e-acoplamento/18538
- http://www.inf.ufpr.br/andrey/ci221/SOFTua10.pdf
- http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0210477_04_cap_02.pdf
- http://www.devmedia.com.br/entendendo-coesao-e-acoplamento/18538
- http://pt.wikipedia.org/wiki/Componente_de_software
- http://rafaelsakurai.blogspot.com.br/
- http://escola.berlios.de/hp/artifacts/baselined/inf/requerimento.pdf