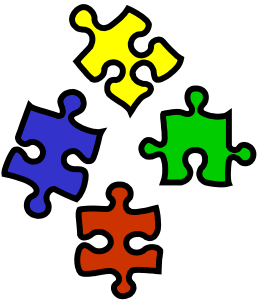
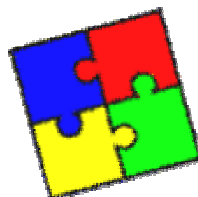
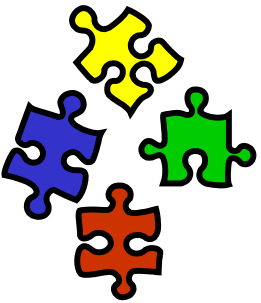


Visão de Código

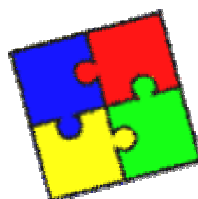
- Descreve como o software que implementa o sistema é organizado
- O objetivo principal desta visão é facilitar a construção, integração, instalação e teste do sistema respeitando a integridade das outras três visões arquiteturais

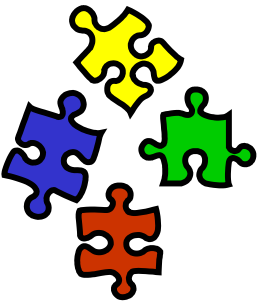




Visão de Código

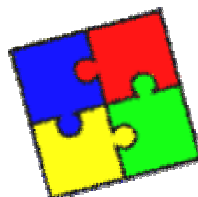
- Descreve como um módulo, suas interfaces e suas dependências são mapeadas a componentes e dependências específicas da linguagem. Estes mapeamentos e convenções podem variar de acordo com a linguagem de programação
- As visões de módulo e de execução são independentes de linguagem de programação

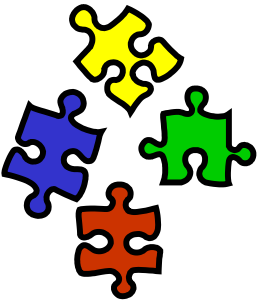




Visão de Código

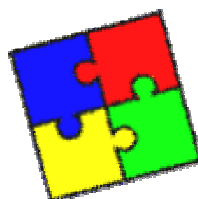
- Com um executável
  - Visão de código reflete a estrutura da visão de módulo
- Com múltiplos executáveis
  - Visão de código torna-se mais complexa e em geral diverge da visão de módulo e da visão de execução

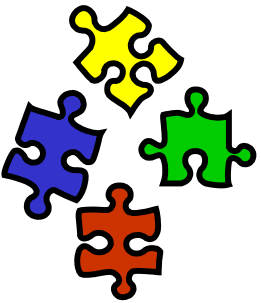




Visão de Código

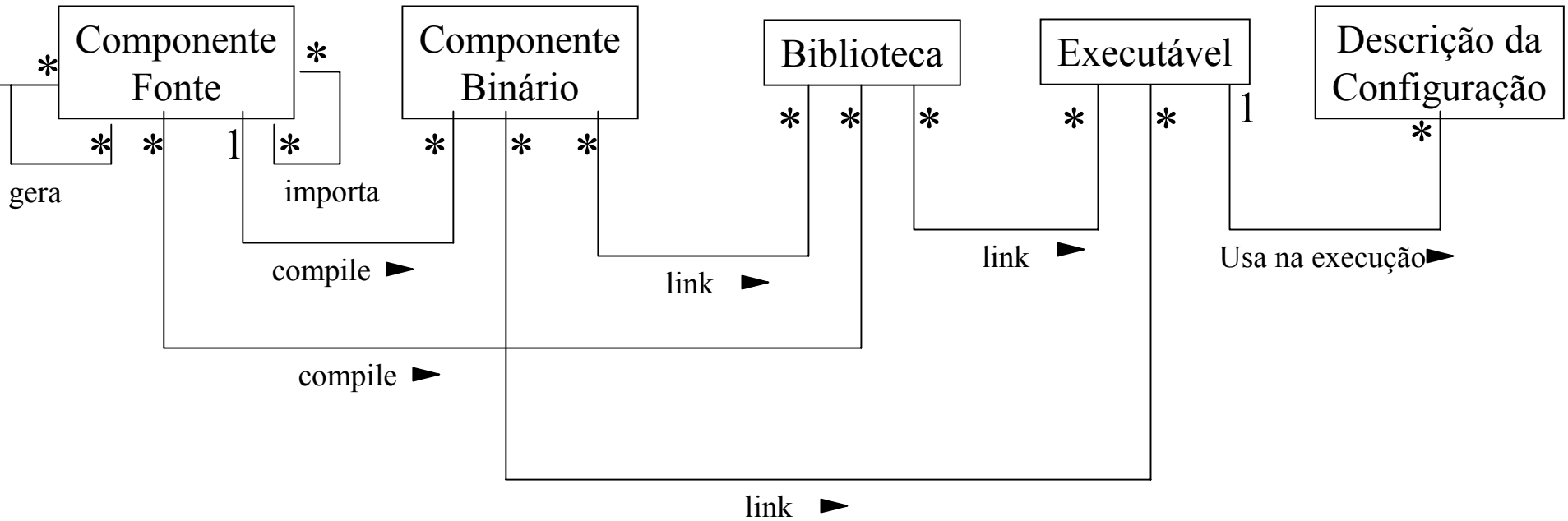
- Tarefas:
  - Mapear os elementos da visão de módulo e da visão de execução para componentes de código:
    - Componentes Fonte
    - Componentes Intermediários (componentes binários,
    - Componentes de Desenvolvimento (executáveis, bibliotecas dinâmicas e descrição da configuração)

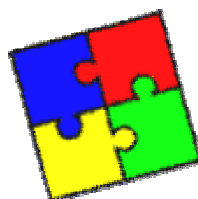


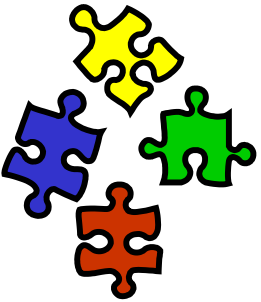


Visão de Código

Meta-modelo dos elementos básicos

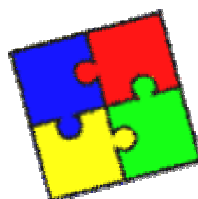


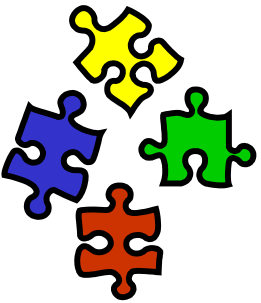




Visão de Código
Componentes Fonte

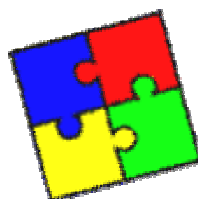
- Tipicamente componentes fontes representam as interfaces e módulos específicos da linguagem
  - Exemplo: em C/C++ interfaces específicas da linguagem são arquivos com nomes como *.h e módulos tem nome com *.c e *.CPP

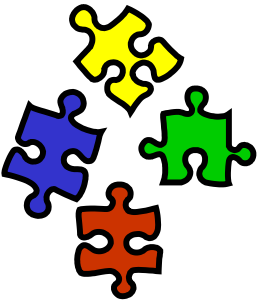




Visão de Código
Componentes Fonte

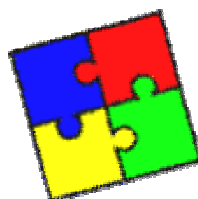
- São relacionados com outros por dois tipos de dependências específicas da linguagem:
  - **Importação**: um componente precisa importar outro. Esta relação é uma dependência de compilação. Por exemplo: a dependência de inclusão entre um componente fonte e um arquivo de cabeçalho
  - **Geração**: quando um componente fonte é gerado a partir de outro componente fonte.

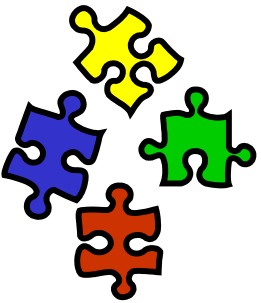




Visão de Código
Componentes Fonte

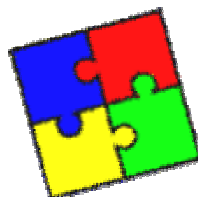
- Para identificar componentes fonte:
  - Identificar componentes fonte e mapear os elementos e dependências da visão de módulo para componentes fonte e dependências
  - Organizar os componentes fonte usando estruturas de armazenamentos como diretórios e arquivos
    - É necessário definir um critério de organização dos arquivos: similaridade de funcionalidade, pessoa responsável pelo desenvolvimento, etc...

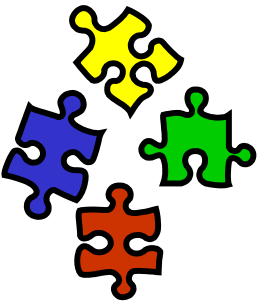




Visão de Código
Componentes Fonte

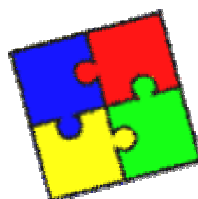
- Elementos da visão de módulo são mapeados em componentes fonte específicos de linguagem que implementa-os. Por exemplo: Mapeando para C++, a interface de um módulo é mapeada em um arquivo .h e o código é mapeado em um arquivo .CPP
- Para módulos complexos pode ser necessário distribuir a implementação em vários arquivos

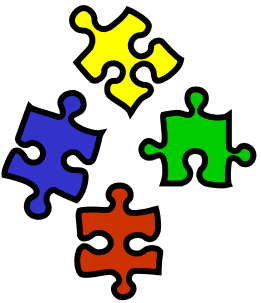




Visão de Código
Componentes Intermediários

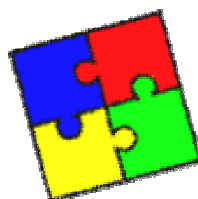
- São específicos da linguagem de programação e do ambiente de desenvolvimento.
  - Exemplo: Em C++, para cada arquivo .CPP tem um componente binário correspondente ou arquivo .obj
- Binários e Bibliotecas Estáticas são componentes intermediários

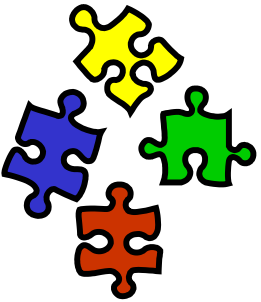




Visão de Código
Componentes de Desenvolvimento

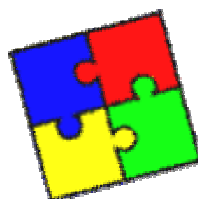
- São componentes necessários para formar um sistema executável
  - Executáveis, bibliotecas dinâmicas e descrições de configuração
- A descrição da configuração pode descrever processos e/ou recursos





Visão de Código
Resumo da Representação

Artefato	**Representação**
Correspondência Módulo-Comp.Fonte	Tabela
Correspondência Executável – entidade de execução	Tabelas
Descrição dos componentes da Visão de código e suas dependências	Diagrama de Componentes UML



Visão de Código
Meta-modelo