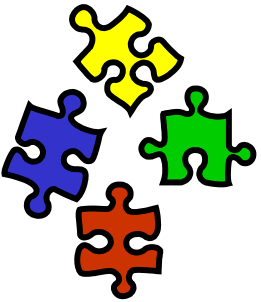
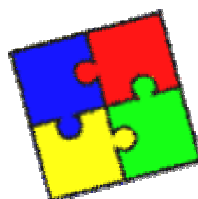
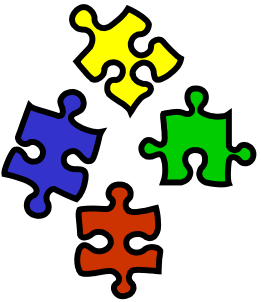


# Visões Arquiteturais

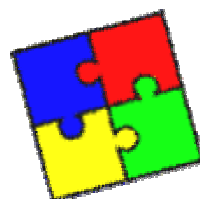
- Separar diferentes aspectos em visões separadas com o objetivo de gerenciar complexidade.
- Cada visão descreve diferentes conceitos da Engenharia.
- Visões permitem reduzir a quantidade de informação que o arquiteto trata em um dado momento
- Muitos arquitetos de software tem usado as diferentes visões sem reconhecê-las como visões arquiteturais separadas.



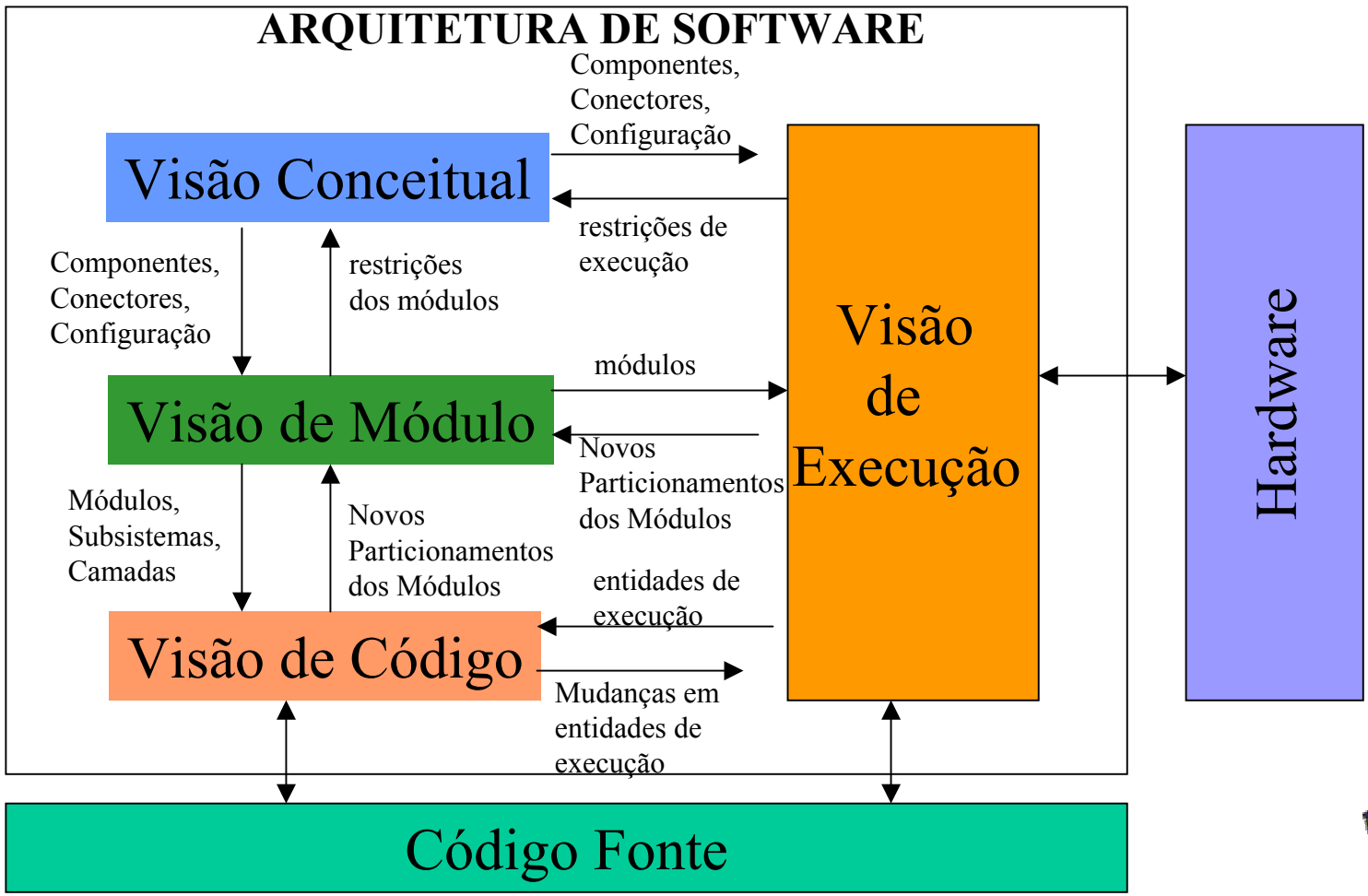


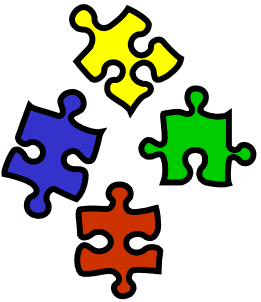
# Visões Arquiteturais

- Quatro visões:
  - **Conceitual**: descreve o sistema em termos dos elementos de design e relacionamentos entre eles
  - **Módulo**: consiste na decomposição do sistema em módulos
  - **Execução**: consiste no mapeamento dos componentes a entidades de execução e de hardware. Este mapeamento deve ser definido na fase de projeto arquitetural e não apenas na fase de desenvolvimento.
  - **Código**: consiste na organização do código fonte em bibliotecas, binários, arquivos, versões e diretórios



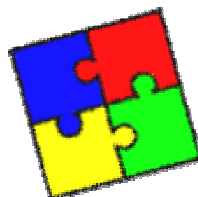
# Visões Arquiteturais

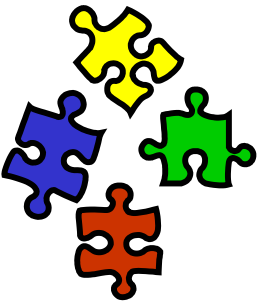




# Visão Conceitual

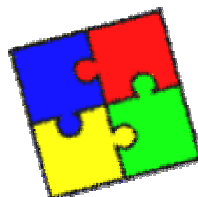
- Mais próxima ao domínio da aplicação
- A funcionalidade do sistema é mapeada em elementos arquiteturais:
  - Componentes, conectores, configuração
- Desafio:
  - Isolar comunicação entre componentes em conectores

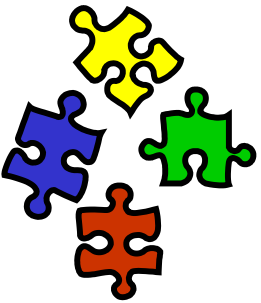




# Visão de Módulo

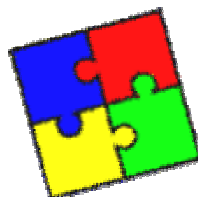
- Componentes e Conectores que formam a visão conceitual são mapeados em subsistemas e módulos
  - Dependências entre módulos devem ser minimizadas
  - Reuso de módulos deve ser maximizado

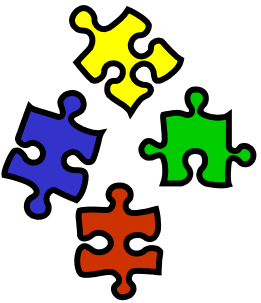




# Visão de Execução

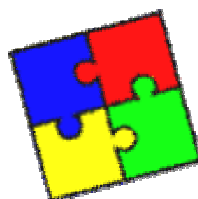
- Descreve como os módulos são mapeados para elementos oferecidos pela plataforma de execução e como estes são mapeados para o hardware
- Define entidades de execução e seus atributos como uso de memória e de elementos de hardware

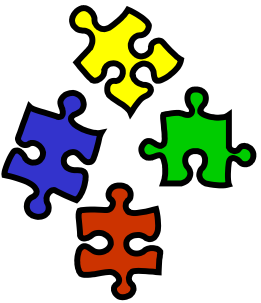




# Visão de Código

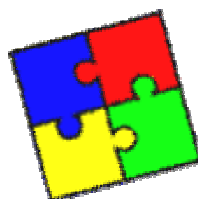
- Determina:
  - como as entidades de execução da visão de execução são mapeadas a componentes a nível de implementação (p.ex. executáveis)
  - como módulos da visão de módulo são mapeados em componentes fonte
  - como componentes a nível de implementação são mapeados a partir dos componentes fonte

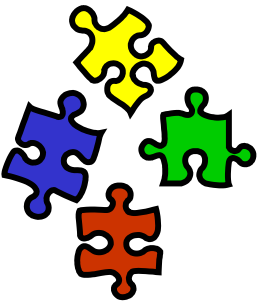




# Relacionamento entre as Visões

- Há necessidade de *feedback* e interação entre as visões
- As visões *módulo, execução e código* auxiliam a diminuir o *gap* entre o que uma ADL modela e o que uma plataforma de execução pode executar.
  - Componentes e conectores da visão conceitual não podem ser diretamente mapeados para uma plataforma de execução
  - As visões módulo e execução definem como componentes e conectores são mapeados para uma plataforma de execução

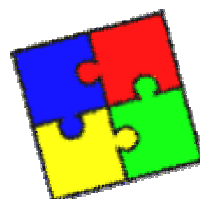


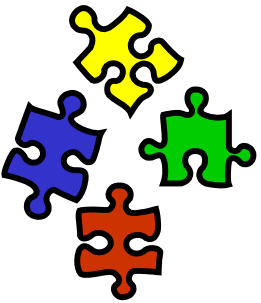


# Visão Conceitual

- Definir componentes, conectores e configuração
  - Mapear comportamento funcional do sistema em componentes
  - Mapear os aspectos de controle e comunicação em conectores
  - Definir as instâncias dos componentes e conectores e como elas são interconectadas

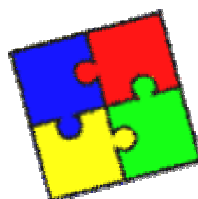
Estilos Arquiteturais podem ser utilizados nesta fase!



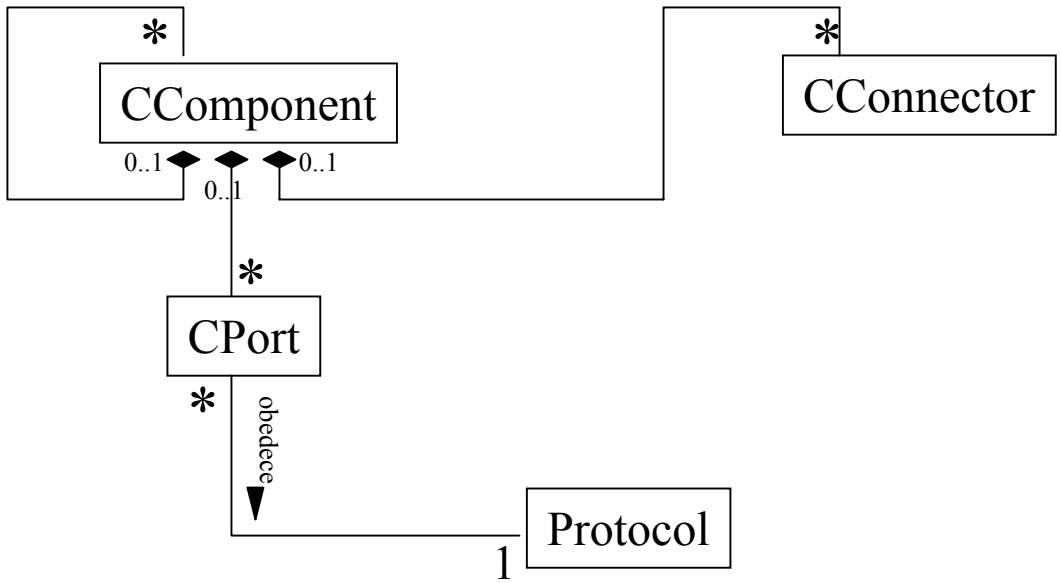


# Visão Conceitual

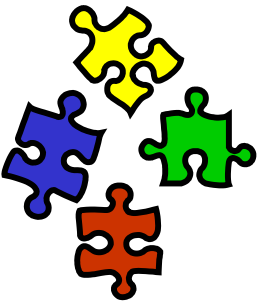
- Componentes Conceituais
  - Contém PORTAS que são pontos de interação
  - PORTAS ≠ INTERFACE
    - Serviços Oferecidos X Serviços Requisitados
      - Interface define serviços e operações que o componente OFERECE não o que ele USA.
      - Portas definem tanto serviços que o componente oferece quanto os que o componente usa.
    - Implementação
      - Cada porta tem um protocolo associado que diz como as operações são encaminhadas



# Visão Conceitual
# Componente Conceitual



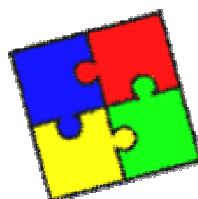

Elementos UML

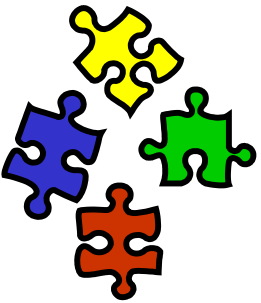


# Elementos UML

- Caixas são classes na notação UML
- Linhas são relações
  - Linhas com o diamante sólido significa composição: as classes ligadas ao diamante é decomposta ou contém as classes da outra extremidade. A multiplicidade da relação é exibida pelo número (ou um asterisco para zero ou mais) próximo ao fim da relação

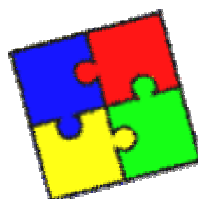
Voltar



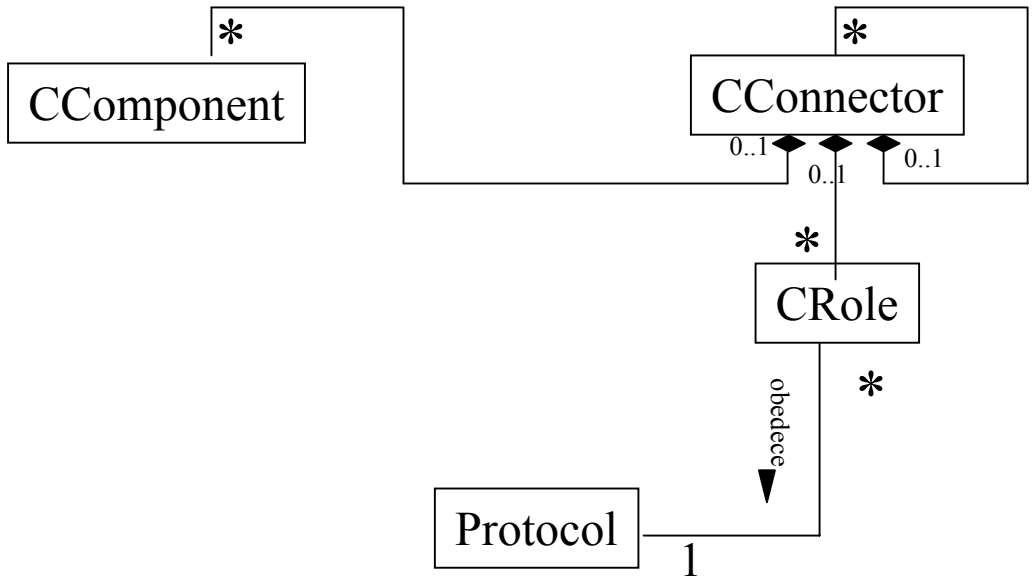


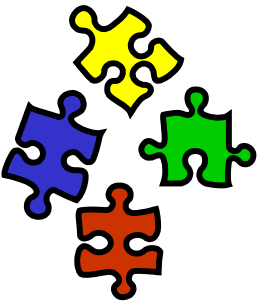
# Visão Conceitual

- Conectores Conceituais
  - Contém PAPÉIS (ROLES) que são pontos de interação
  - Os Papéis obedecem a um protocolo associado
  - Podem ser decompostos em Componentes e Conectores



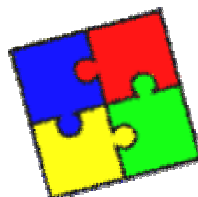
# Visão Conceitual
# Conector Conceitual

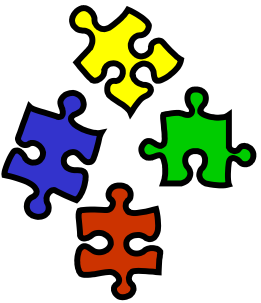




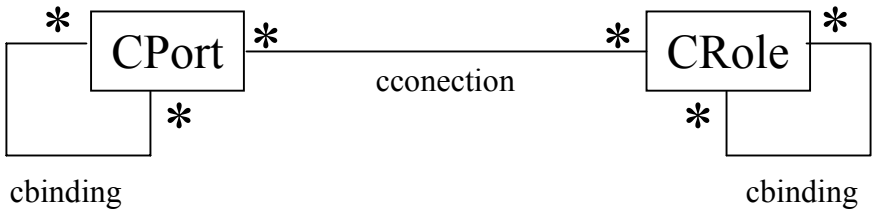
# Visão Conceitual

- Configuração Conceitual
  - Define relações entre componentes e conectores
  - Componentes e Conectores são interconectados através de suas portas e papéis
  - Podem ser decompostos em Componentes e Conectores





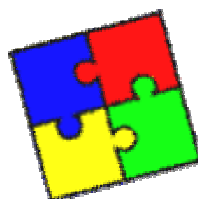
# Visão Conceitual
# Configuração Conceitual

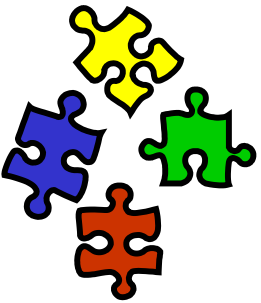



{cconection pode conectar cport e crole apenas quando o respectivo componente e conector são em-capsulados pelo mesmo elemento}

{cconection pode conectar cport e crole apenas quando o eles tem protocolos compatíveis}

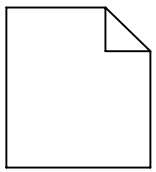
Elemento UML



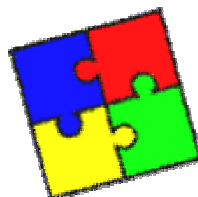


# Elemento UML

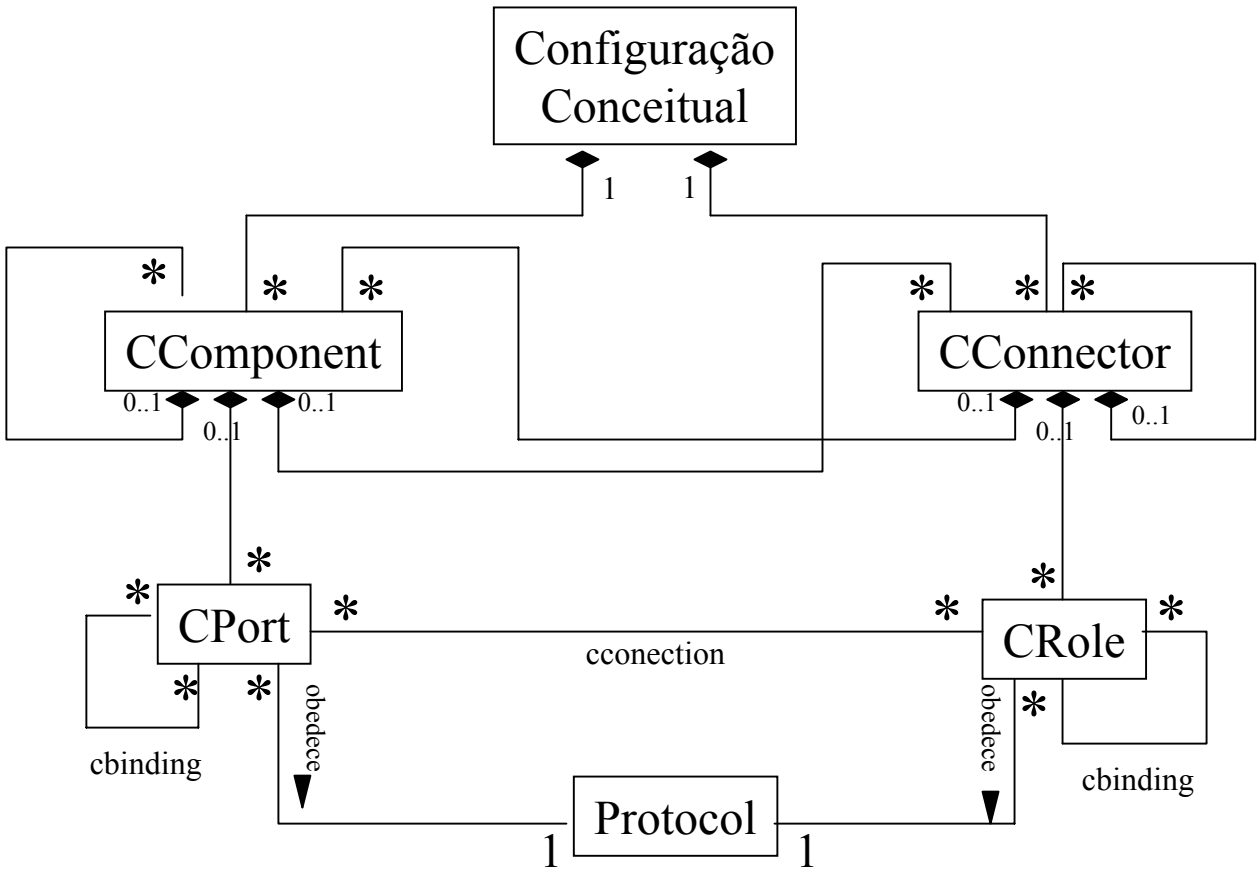
- O elemento NOTE é usado para descrever restrições adicionais que é difícil expressar com a notação gráfica



Voltar

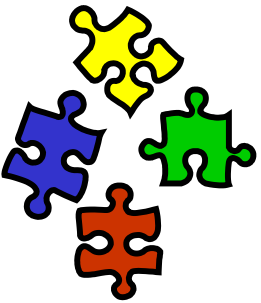


# Visão Conceitual
# Meta-Modelo da Estrutura




{cconection pode conectar cport e crole apenas quando o respectivo componente e conector são em-capsulados pelo mesmo elemento}

{cconection pode conectar cport e crole apenas quando o eles tem protocolos compatíveis}



# Visão Conceitual

- Além de descrever a estrutura de componentes, conectores e da configuração, é necessário descrever os **aspectos comportamentais**
  - UML Statechart Diagram