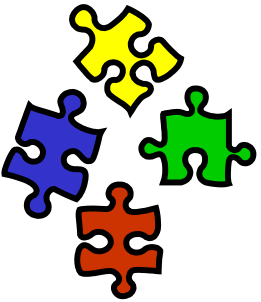
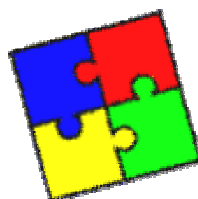
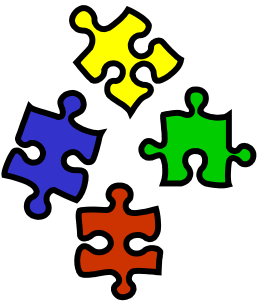


Padrões

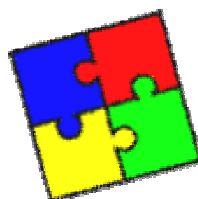
- Endereçam uma classe de problemas recorrentes e apresenta uma solução para eles (podem ser considerados um par problema-solução)
- Permitem a construção de software com propriedades definidas
- Ajudam na construção de arquiteturas de software complexas e heterogêneas
- Ajudam a gerenciar a complexidade do software

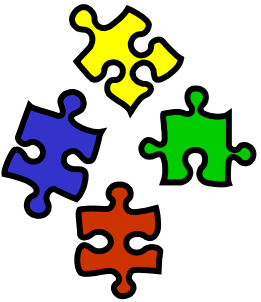




Anti-Padrões

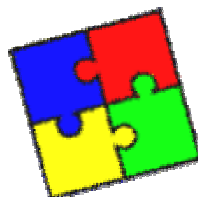
- Mostram o insucesso da aplicação de uma padrão em determinadas classes de problemas
- Podem ser de dois tipos:
  - Descrevem uma solução ruim para um problema
  - Descrevem como escapar de uma solução ruim e a partir dela chegar em uma solução boa para um problema

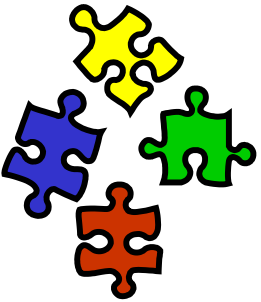




Categorias de Padrões

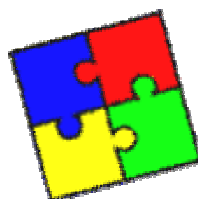
- Padrão Arquitetural ou Estilo Arquitetural
- Padrão de Design (Design Patterns)
- Idiomas

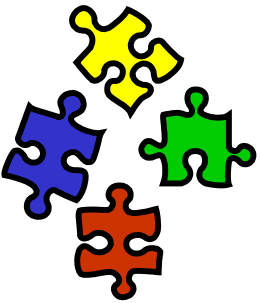




Padrões Arquiteturais

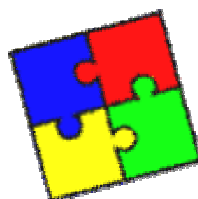
- Expressam um esquema de organização para sistemas de software
- Arquitetura de Software Orientada a Padrões
  - Maneira de documentar arquiteturas de software
  - Oferece *templates* para arquiteturas de software concretas que são um conjunto de regras predefinidas que especificam as responsabilidades e a organização dos relacionamentos entre as partes constituintes

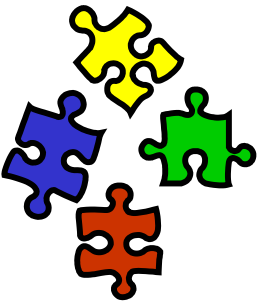




Padrões de Design

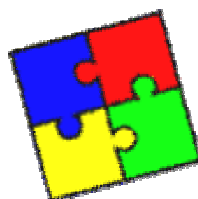
- Os subsistemas da arquitetura de software bem como os relacionamentos entre eles, consistem de várias unidades arquiteturais menores. Tais unidades são descritas usando *design patterns*.
- Oferece um esquema para refinar os subsistemas

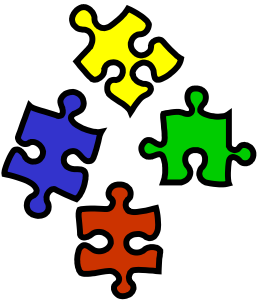




Idiomas

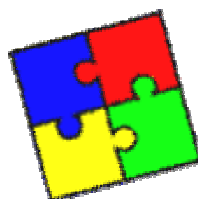
- Tratam com a implementação dos aspectos de design
- É um padrão de baixo-nível específico para uma linguagem de programação
- Descreve como implementar aspectos particulares dos componentes ou relacionamentos entre eles usando as características de uma dada linguagem

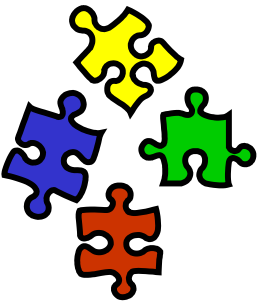




Estilos Arquiteturais

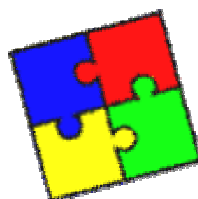
- Em geral sistemas seguem um estilo, ou padrão, de organização estrutural
- Os estilos diferem:
  - nos tipos de componentes que usa
  - na maneira como os componentes interagem com os outros (regras de interação)
- Termos relacionados: padrão arquitetural, estilo arquitetural, idioma arquitetural

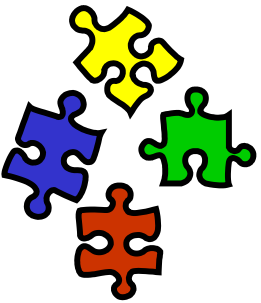




Estilos Arquiteturais

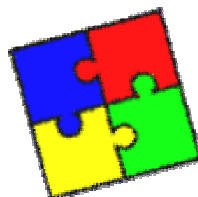
- Um estilo arquitetural define uma família de sistemas em termos de um padrão de organização estrutural [Shaw96]
- Define:
  - um vocabulário de *componentes*
  - tipos de *conectores*
  - conjunto de *restrições* que indica como os elementos são combinados

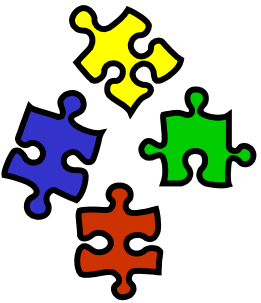




Exemplos de Estilos Arquiteturais

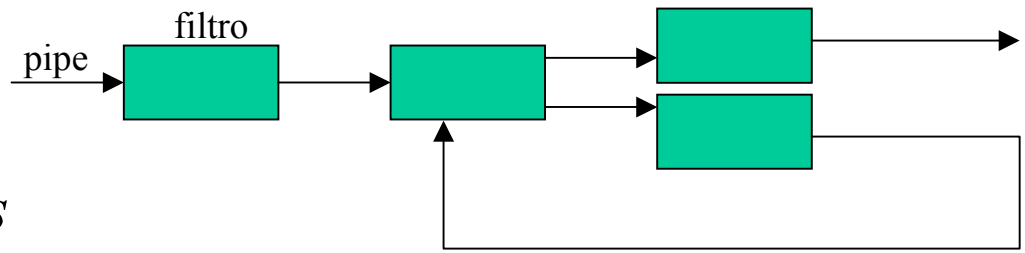
- Pipes e Filtros
- Camadas
- Invocação Implícita (Baseada em Eventos)
- BlackBoard
- Broker
- Reflexão

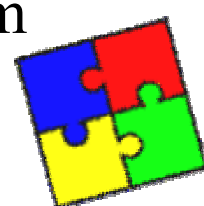


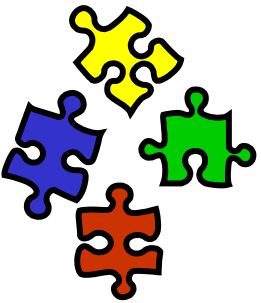


Estilo: Pipe e Filtros

- Tipicamente divide a tarefa de um sistema em vários passos de processamento sequencial
- Componentes:
  - São chamados *Filtros*
  - Tem um conjunto de entradas e um conjunto de saídas
  - Realiza o processamento de um stream de dados
- Conectores:
  - São chamados *Pipes*
  - Servem como condutores, transmitindo as saídas de um filtro para as entradas de outro.

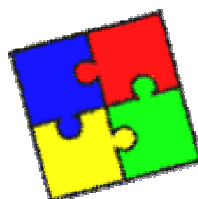


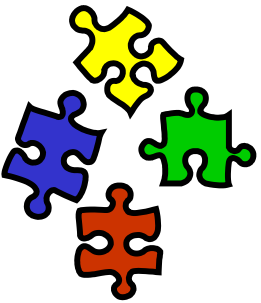




Estilo: Pipe e Filtros

- Especializações do Estilo Pipe e Filtros
  - *Pipelines*: restringe a topologia a sequência linear de filtros
  - *Pipes Tipados* (Typed Pipes): estabelece que os dados passados entre dois filtros tenham um tipo bem definido

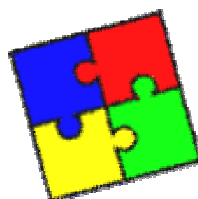


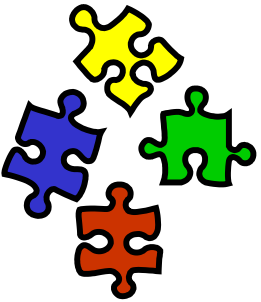


Estilo: Pipe e Filtros

Class Filtro	**Colaboradores** Pipe
Responsabilidade Recebe dados de entrada Realiza uma função nos dados de entrada Fornece dados de saída	

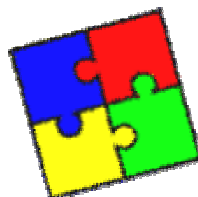
Class Pipe	**Colaboradores** Filtros
Responsabilidade Recebe dados de entrada Realiza uma função nos dados de entrada Fornece dados de saída	

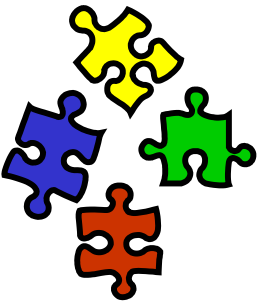




Estilo: Pipe e Filtros Implementação

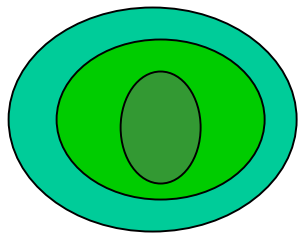
- Divida as tarefas do sistema em uma sequência de estágios de processamento:
  - Cada estágio deve depender apenas da saída do seu predecessor direto
  - Todos os estágios são conectados pelo fluxo de dados
- Defina o formato de dados a ser passado ao longo de cada pipe
- Decida como implementar cada conexão com pipe
  - Implica em decidir se os filtros são componentes ativos ou passivos

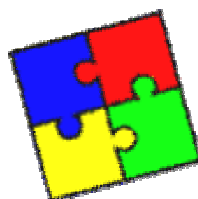


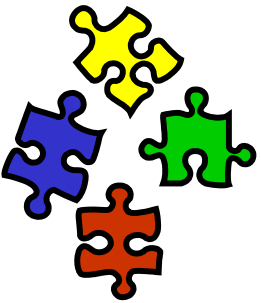


Estilo: Camadas

- Um sistema em *camadas* é organizado hierarquicamente, cada camada oferecendo serviço a camada acima dela e servindo como cliente da camada inferior.
- Componentes:
  - São representados por cada camada
- Conectores:
  - São definidos pelos protocolos que determinam como cada camada irá interagir com outra
  - Limitam as interações a camadas adjacentes

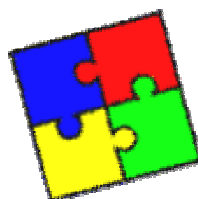


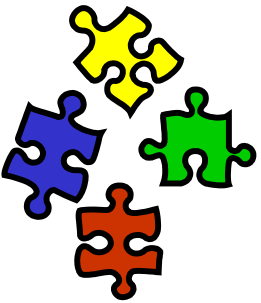




Estilo: Camadas

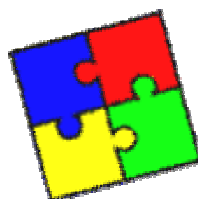
- Protocolos de rede são os melhores exemplos de arquiteturas em camadas
  - Definem um conjunto de regras e convenções que descrevem como programas de computador em diferentes máquinas comunicam-se
  - O formato, conteúdo e significado das mensagens são definidas

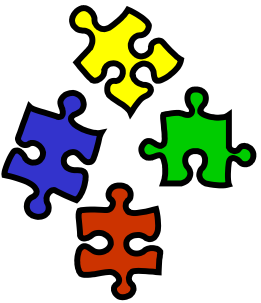




Estilo: Camadas

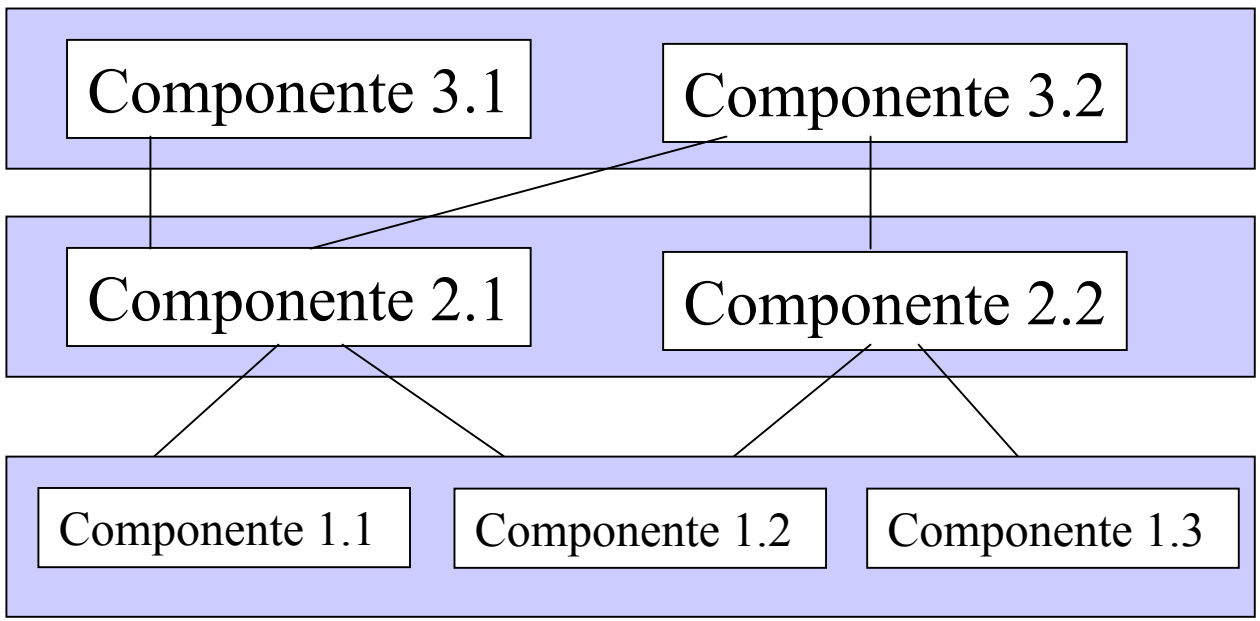
Class Camada J	**Colaboradores** Camada J-1
Responsabilidade Oferecer serviços usados pela camada J+1 Delegar subtarefas a camada J-1	

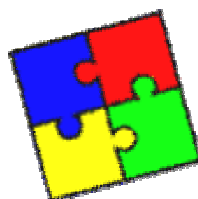


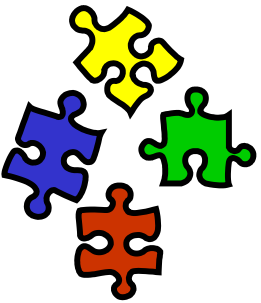


Estilo: Camadas

- Cada camada individual pode ser uma entidade complexa consistindo de diferentes componentes







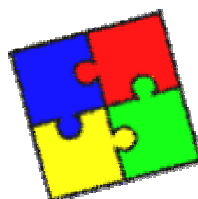
Estilo: Camadas

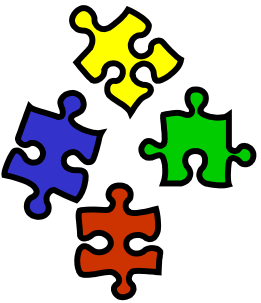
Modelo OSI da ISO

- Aplicação
- Apresentação
- Sessão
- Transporte
- Rede
- Enlace
- Físico

Modelo TCP/IP

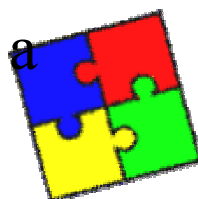
- Aplicação
- TCP
- IP
- Rede de Comunicação

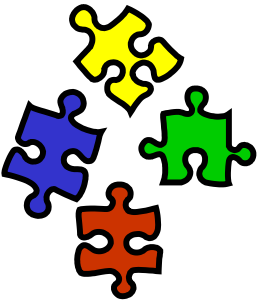




Estilo: Camadas Implementação

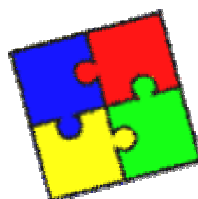
- Defina o **critério de abstração** para agrupar tarefas em camadas
  - Exemplo: a distância do hardware pode formar os níveis mais baixos e a complexidade conceitual os níveis mais altos
- Determine o número de níveis de abstração de acordo com seu critério de abstração
- Nomei as camadas e determine as tarefas de cada uma delas
  - A tarefa da camada mais alta é a percebida pelo cliente
  - As tarefas das demais camadas visam ajudar a realização da tarefa da camada mais alta

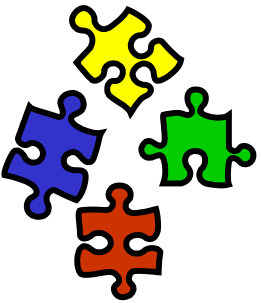




Estilo: Camadas Implementação – cont.

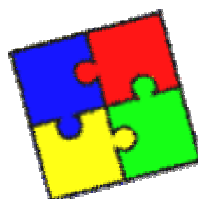
- Especifique os serviços
- Especifique uma interface para cada camada
- Refine cada camada:
  - Estruturação de cada camada individualmente
  - Quando uma camada é complexa ela deve ser separada em componentes individuais e cada componente pode seguir um padrão ou estilo diferente

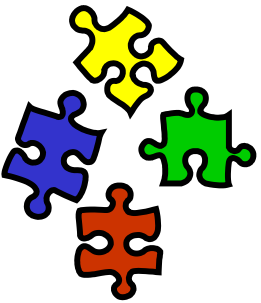




Estilo: Camadas

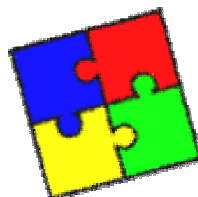
- Variantes do estilo
  - *Sistema de Camadas Relaxadas*: é menos restritivo quanto aos relacionamentos entre as camadas. Cada camada pode usar os serviços de TODAS as camadas abaixo dela não apenas da camada mais próxima.
  - *Camadas através de herança:* algumas camadas são implementadas como classes base. As camadas mais altas herdam a implementação das camadas mais baixas. Comum em sistemas orientados a objetos.

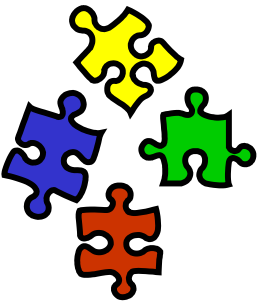




Estilo: Camadas

- Vantagens:
  - Permite projetos baseados em níveis crescentes de abstração
  - Permite particionar problemas complexos em uma sequência de passos incrementais
  - Mudanças em uma camada afetam, no máximo, as duas adjacentes
  - Permite que diferentes implementações da mesma camada possam ser usadas desde que mantenham a mesma interface com as camadas adjacentes





Estilo: Camadas

- Desvantagens:
  - Nem todos os sistemas são facilmente estruturados em forma de camadas
  - É difícil encontrar os níveis de abstração corretos (muitas vezes os serviços abrangem diversas camadas).