Arquitetura de Sofware

- Define *conceitos, padrões e estilos* para a composição de software formado por componentes
- *Framework* é usado muitas vezes como sinônimo de Arquitetura

Componente

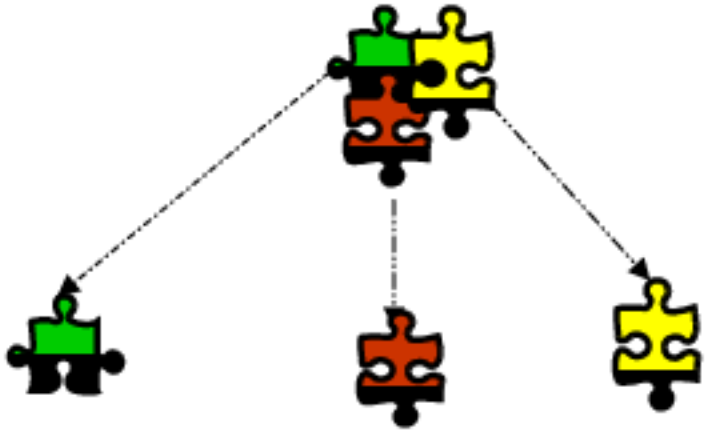
- Definições:
  - *É um elemento de software que segue um **modelo de componentes** e pode ser desenvolvido independentemente e composto através de um **padrão de composição** [B.Council and G. Heineman]*
  - *Componentes são elementos **padronizados** usados para **composição** [C. Szyperski]*

Componente

- Características:
  - Auto-contido
  - Funcionalidade bem definida
  - Definido através de interfaces que possibilita composição sem conhecimento da implementação do componente
  - Definido de acordo com um modelo de componentes

Composição

- União de porções de software "pré-fabricadas" para formar um sistema



Desenvolvimento baseado em Componentes

- Mercado de Componentes
- Menos tempo de desenvolvimento
- Mais confiável (por reusar partes testadas)
- Ideal de possibilitar que o desenvolvimento de software seja uma *linha de produção em massa*

Objetos X Componentes

- A definição de objetos não inclui:
  - Noções de independência
  - Composição

Apesar destes aspectos poderem ser adicionados, a tecnologia de objetos é mais usada para construir aplicações monolíticas [C.Szyperski]

Exemplo do Uso da Idéia de Componentes

- Sistemas Operacionais: aplicações são componentes executando sobre eles (compartilhando arquivos e fazendo composição via pipe e filtros)
- Plug-in: Browsers Netscape
- Visual Basic

Modelo de Componentes

- Determina a forma como um componente deve ser desenvolvido
  - Exemplos
    - COM (Component Object Model) da MicroSoft determina que cada componente ofereça uma interface *IUnknown*
    - CORBA da OMG determina que o componente tenha uma interface escrita em IDL (Interface Definition Language)
- Determina um padrão de interação

Modelo de Componentes

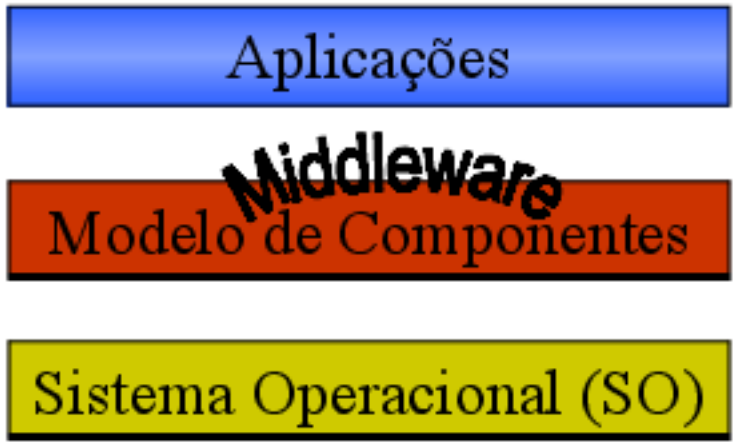
- Define **padrões** para:

Interfaces	Especificação do componente
Identificação	Nomes únicos globais
Interoperabilidade	Comunicação e troca de dados entre componentes implementados em linguagens diferentes

Modelo de Componentes

- A implementação do modelo de componentes:
  - executa no topo de um SO.
  - oferece suporte a execução dos componentes

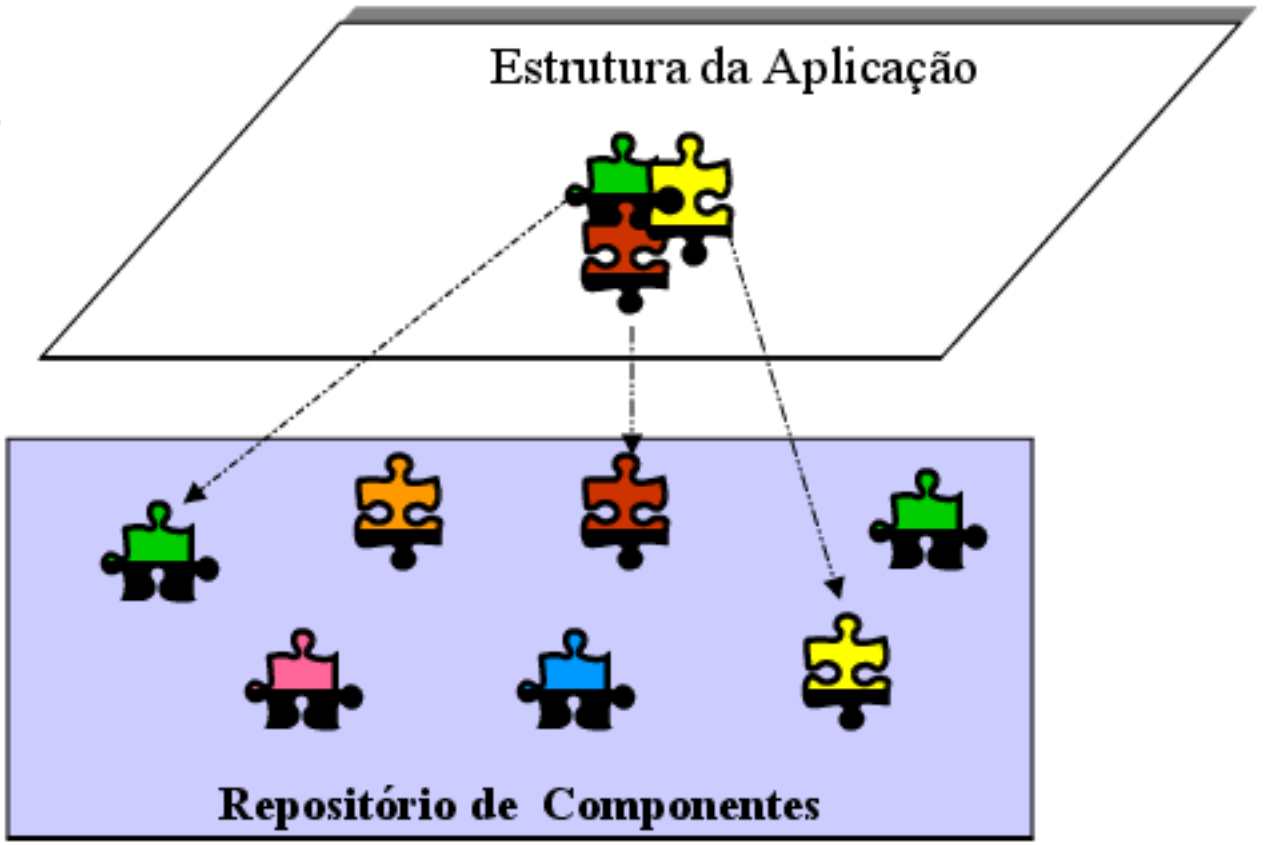
- ***Middleware***: software que situa-se entre a aplicação e o sistema operacional



Arquitetura & Componentes

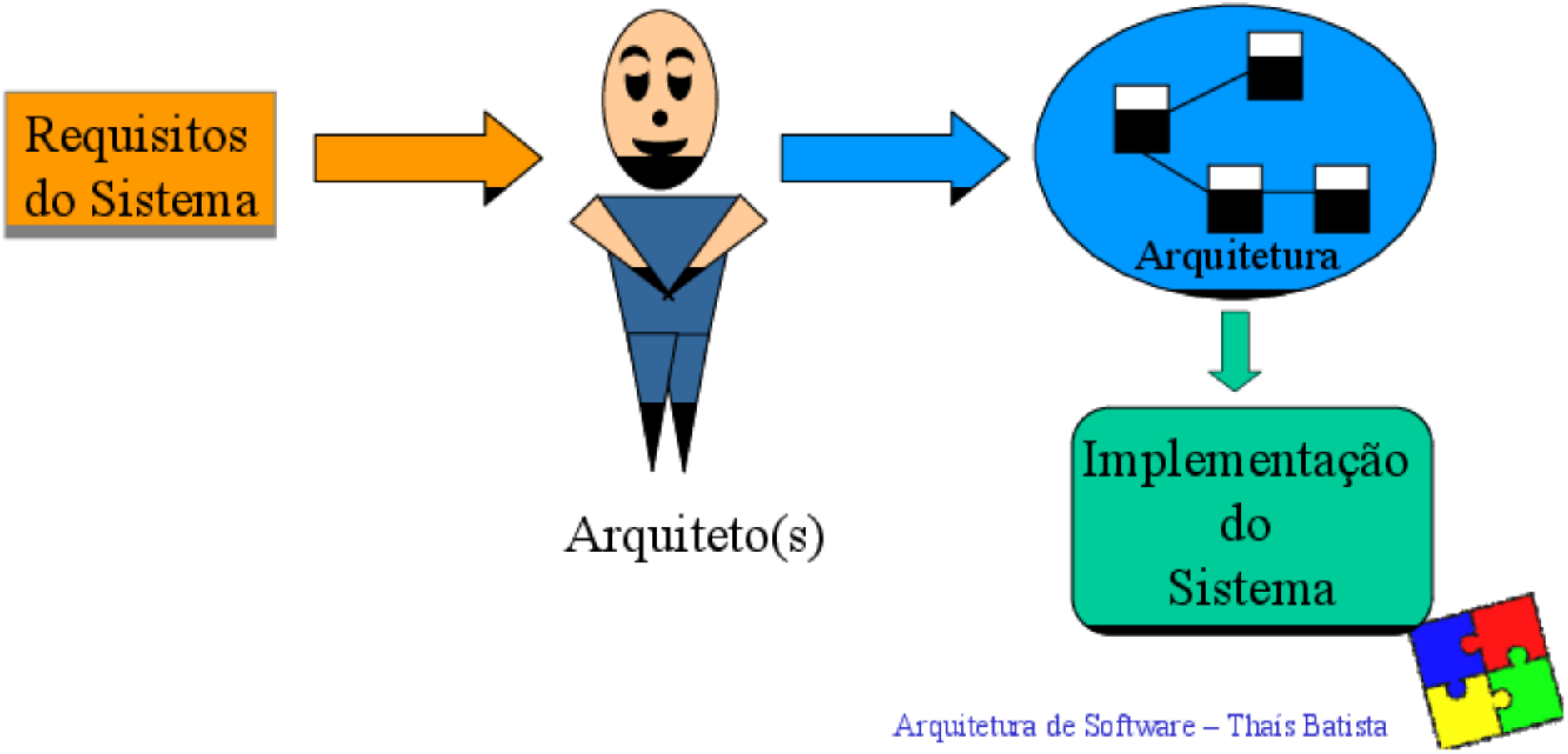
Arquitetura da Aplicação

Modelos de Componentes



Arquitetura de Software

Ponte entre os requisitos do sistema e sua implementação



Arquitetura de Software (AS)

- Abstração que ajuda a gerenciar complexidade
- Projeto Arquitetural do Software
  - Estrutura modular do software
    - Componentes
    - Relacionamento entre componentes
- Linguagem de Descrição Arquitetural (ADL) descreve a arquitetura do sistema

Motivação para AS

- *Programming-in-the-large X Programming-in-the-small (Frank DeRemer and Hans Kron. 1976)*
  - *Programming-in-the-large:* estrutura da aplicação descrevendo a interconexão entre os componentes
    - MILs (Module Interconnection Language): linguagens puramente declarativas para descrever estaticamente a interconexão entre os módulos
  - *Programming-in-the-small*: código interno do componente ou do módulo

Motivação para AS

- Representação de sistemas informalmente através de caixas e linhas
- Gerenciamento da complexidade de sistemas
- Dinamismo
- Redução do *gap* entre especificação e implementação

Terminologias em AS

- **Estilo Arquitetural** ou **Padrão Arquitetural** ou **Estilo** (simplesmente): define os tipos de elementos e como eles interagem
  - Ex.: Pipe-Filtros é um estilo que define:
    - Elementos (dois tipos): pipe e filtro.
    - Interação: um pipe pode ser conectado a um filtro mas pipes não podem ser conectados a pipes nem filtros podem ser conectados a filtros
    - Arquitetura: processamento é mapeado em filtros e os pipes agem como condutores de dados

Terminologias em AS

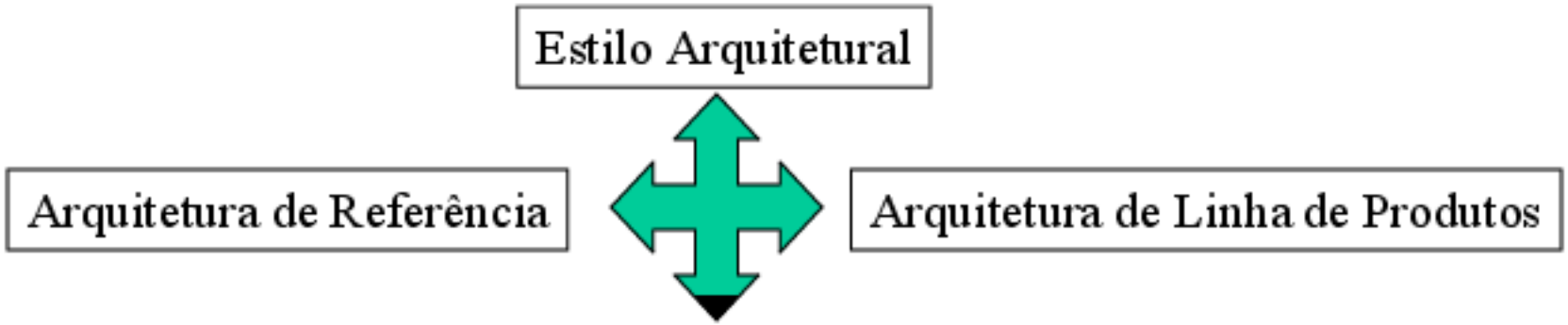
- **Arquitetura de Referência** ou **Arquitetura de Software específica de Domínio**: define tipos de elementos e interações permitidas em um domínio particular de aplicações. Ou seja, define como a funcionalidade de um domínio é mapeada em elementos arquiteturais

Terminologias em AS

- **Arquitetura de uma linha de produtos**:
  - É aplicada a um conjunto de produtos dentro de uma organização. Tais produtos compartilham um design comum e, as vezes, parte da implementação
  - Define tipos de elementos e como eles interagem

Terminologias em AS

- As arquiteturas não são mutuamente exclusivas
  - Exemplo: uma **arquitetura de linha de produtos** pode ser baseada em uma **arquitetura de referência** e esta, por sua vez, em um **estilo arquitetural**



ADLs

- Específicas para descrição de arquiteturas
- Expressam características estruturais e comportamentais dos sistemas
- Permitem reusabilidade
- Oferecem primitivas para composição do sistema
- Em geral incluem um formalismo

ADLs

- Características Desejáveis:
  - Suporte a dinamismo
    - Inserir/Remover Componentes e Conexões entre eles
    - Atualizar Componentes
  - Permitir heterogeneidade de padrões e/ou de linguagens

Elementos Básicos de ADLs

- Componentes
- Conectores
- Configurações

Componentes

- Blocos de alto nível
- Oferece
  - Modularidade
  - Separação dos conceitos
- Publicam a interface
- Escondem a implementação

Componentes

- Interfaces
  - Definem os pontos de acesso ao componente - propriedades visíveis externamente
  - Possibilitam plug-and-play
  - Servem como um contrato entre componentes
  - Interfaces bem definidas tornam fácil a identificação e o propósito do comportamento do componente
  - Podem ser representadas por: Portas, Pontos de Interação, Serviços, etc... (diferentes terminologias e, em alguns casos, diferentes tipos de informações)

Componentes

- Exemplo Interface

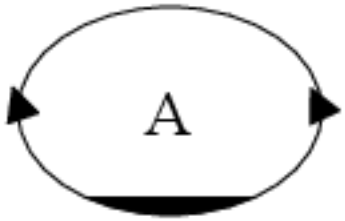
```
Interface pessoa
{
   string nome;
   short identidade;
   long data_de_nascimento;

   void add_pessoa(in string nome, in short id, in long data)
         raises (JaCadastrado)
}
```

Componentes

- Exemplo Interface em MetaH

```
with type package Nav_Types;
process interface A is
    sensor: in port Nav_Types.Rate_Vector;
    position: out port Nav_Types.ECEF_Type;
end A;
```



Conectores

- Descrevem as interações e estabelecem restrições de como as mensagens fluem entre os componentes
  - Exemplo: um conector em um sistema cliente-servidor descreve o comportamento do cliente como uma sequência de solicitações alternadas com a recepção dos resultados
- Concentram a informação sobre interconexão e a torna visível e reusável

Conectores

- ADLs modelam conectores de várias formas:
  - Explicitamente: têm nome e podem ser reusados
    - Exemplos: UniCon, ACME, C2, Wright
  - In-Line: não têm nome e não podem ser reusados
    - Exemplo: Rapide e MetaH

Conectores

- Exemplo

Conector em ACME

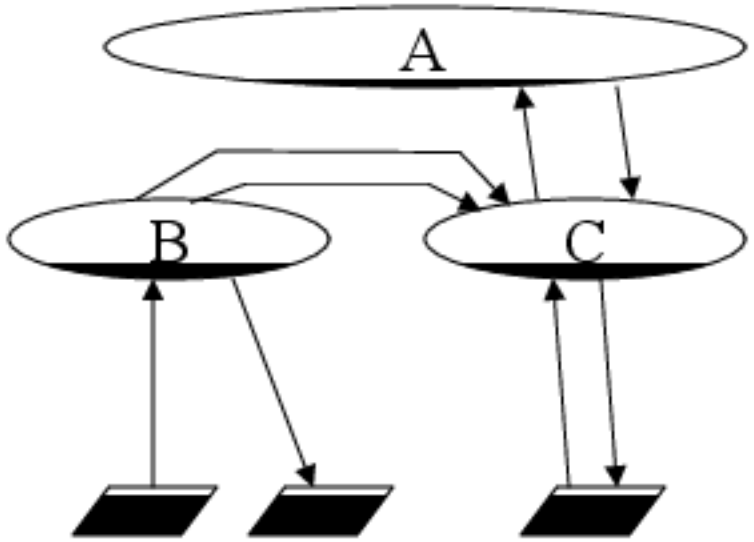
```
Connector rpc = {
Roles {caller, callee}
Properties { synchronous}
```

Conector em Rapide

```
connect
C.Send ||> B1.Receive
```


Configurações

- Descrevem a topologia do sistema: os elementos constituintes (componentes e conectores) e como eles estão interconectados para formar a arquitetura



Configurações

Configuração em C2

```
architecture StackVisualizationArchitecture is
components
      top_most StackADT;
       internal StackVisualization1; StackVisualization2;
       bottom_most GraphicsServer;
connectors
      connector TopConnector is
                  message_filter no_filtering
      end TopConnector;
      connector BottomConnector is
                  message_filter no_filtering
      end BottomConnector;
architectural_topology
      connector TopConnector connections
            top_ports StackADT;
            bottom_ports StackVisualization1; StackVisualization2;
      connector BottomConnector connections
            top_ports StackVisualization1; StackVisualization2;
            bottom_ports GraphicsServer;
end StackVisualizationArchitecture;
```

Exemplo em Wright

- Descrição simplificada de um sistema cliente-servidor em Wright

Style Cliente-Servidor
Component Cliente
Port p
Computation [descrição]

Component Servidor
Port p
Computation [descrição]

Connector Cli-Serv
Role c
Role s
Glue [descrição]
EndStyle

A configuração é uma instância do estilo arquitetural

Configuration Cliente-Servidor
Style Cliente-Servidor
Instances C:Cliente, CS:Cli-Serv, S:Servidor
Attachments
C.p **as** CS.c
S.p **as** CS.s
End Configuration

Exemplos de ADLs

ADL	Organização	Referência	Objetivos de Design
ACME	Carnegie Mellon University	Carnegie Mellon University, 1997	Oferece uma linguagem de intercâmbio para facilitar compartilhamento de componentes arquiteturais
MetaH	Honeywell Technology	Vestal, 1998	Especifica aspectos de tempo real e concorrência
Rapide	Stanford University	Luckham et al. 1995	Oferece uma ADL executável baseada em um modelo de execução orientado a eventos
UniCon	Carnegie Mellon University	Shaw and Garlan, 1996	Suporta contrução arquitetural baseada em estilo interconectando componentes
Wright	Carnegie Mellon University	Allen and Garlan 1997	Oferece um formalismo que foca nos tipos explícitos de conectores

Classificação de ADLs

- De acordo com o relacionamento entre a descrição arquitetural e a implementação do sistema
  - Linguagens independentes de implementação
    - Wright, Rapide
    - Problema: Erosão Arquitetural => não correspondência entre descrição arquitetural e implementação do sistema
  - Linguagens limitadas a implementação:
    - UniCon