INF016 – Arquitetura de Software

05 – Conectores

Sandro Santos Andrade
sandroandrade@ifba.edu.br

Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Tecnologia Eletro-Eletrônica
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas

Introdução

- Projetar e implementar funcionalidades e gerenciamento de dados em sistemas grandes, complexos e distribuídos é algo difícil
- Realizar decisões arquiteturais referentes à efetiva integração e interação entre componentes é uma tarefa de igual importância e talvez mais desafiadora
- Conectores de *software* demandam um estudo exclusivo

Introdução

- Conectores de *software* realizam transferência de controle e dados entre componentes
- Conectores também podem disponibilizar serviços (ex: persistência, invocação, *messaging* e transações) independente da funcionalidade dos componentes envolvidos
- Tais serviços são geralmente considerados "*facilities components*" (ex: no CORBA, DCOM e RMI)
  - Tratar estes serviços, entretanto, como conectores deixa a arquitetura mais clara e mantém os componentes com foco nos aspectos referentes à aplicação e ao domínio

Introdução

- Conectores permitem que arquitetos integrem funcionalidades heterogêneas, desenvolvidas em momentos e lugares diferentes, por diferentes organizações
- Os conectores são “os guardas dos portões do *separation of concerns*”
- Um erro comum é achar que conectores são simplesmente chamadas entre dois componentes (ex: invocação de método)
- Como arquiteturo de *software* você não está limitado a isso, pode-se utilizar *message passing* e RPC, por exemplo

Introdução

- Abordagens tradicionais da Engenharia de *Software* possuem uma visão limitada dos conectores, principalmente se fortemente baseada em componentes *COTS* advindos de múltiplas fontes
- Quando os componentes se tornam maiores e mais heterogêneos a sua correta interação se torna um determinante crítico das propriedades do sistema
- Os conectores são peça chave para integrar componentes com hipóteses divergentes sobre seu ambiente de execução

Introdução

- O que os conectores podem fazer ?
  - Distribuir uma requisição de serviço para componentes especificamente identificados
  - Realizar *broadcast* de uma notificação de evento para qualquer componente interessado (não identificado ou até mesmo desconhecido)
  - Requerer que o componente solicitante suspenda o seu processamento até que um *ACK* seja recebido (síncrono*/blocking)* ou permitir que o componente solicitante continue com o seu processamento (assíncrono/*non-blocking*)
  - Rotear requisições na ordem recebida
  - Ordenar, filtrar e combinar requisições de acordo com alguma regra pré-definida

Introdução

- Um importante papel do trabalho do arquiteto é, portanto:
  - Compreender as necessidades de interação entre componentes
  - Identificar cuidadosamente todos os atributos relevantes desta interação
  - Selecionar os conectores candidatos
  - Avaliar qualquer *trade-off* associado a cada candidato
  - Analisar as propriedades-chave da configuração componente-conector resultante

Introdução

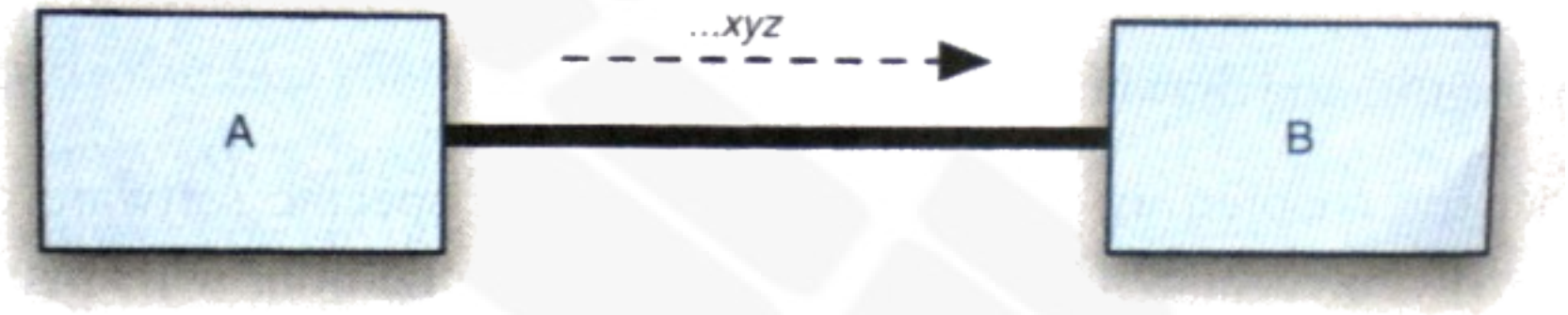
- Serão apresentados:
  - Os diferentes papéis que os conectores podem desempenhar no sistema
  - Os diferentes tipos de conectores que o arquiteto tem a sua disposição e os papéis que cada tipo desempenha
  - Os diversos pontos de variação para cada tipo de conector
  - Um conjunto de dicas e diretrizes gerais sobre a aplicabilidade, vantagens e desvantagens de cada conector

Motivação

- Os conectores são, em sua maioria, elementos arquiteturais independentes de aplicação
- Nos permite entender **como** o sistema realiza sua tarefa sem necessariamente entender exatamente **o que** o sistema faz, ou seja, suportam de forma efetiva abstração e *separation os concerns*
- Conectores permitem o desenvolvimento e uso de um vocabulário específico para interações em *software*
- Habilitam, portanto, uma comunicação facilitada e raciocínio preciso sobre propriedades do sistema que derivam das interações entre componentes

Motivação

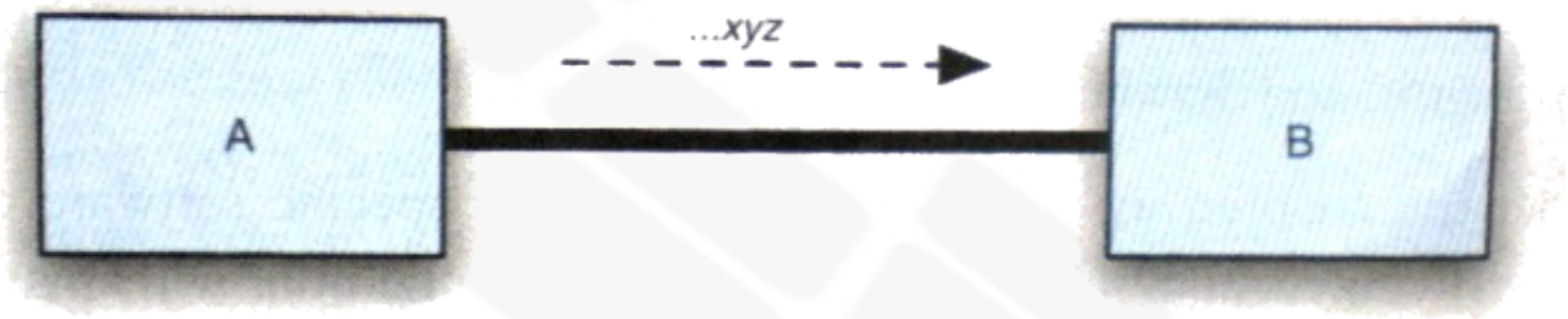
- Exemplo:



  - Visão em alto nível:
    - “Os componentes A e B se comunicam através de um *pipe* (conector) do UNIX”

Motivação

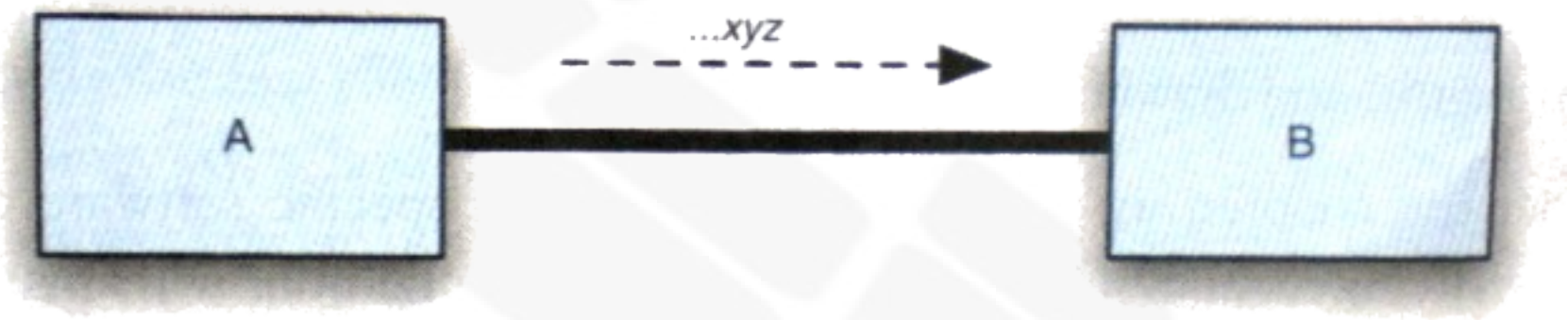
- Exemplo:



  - Visão em nível mais baixo:
    - O *pipe* permite interação por fluxos (*streams*) de dados não-formatados
    - É um conector simples: um único canal de interação (duto) e transferência de dados somente uni-direcional
    - Cardinalidade: um emissor – um receptor
    - Produz fraquíssimo acoplamento entre A e B
    - Não realiza *data buffering* (entrega *at most once*)

Motivação

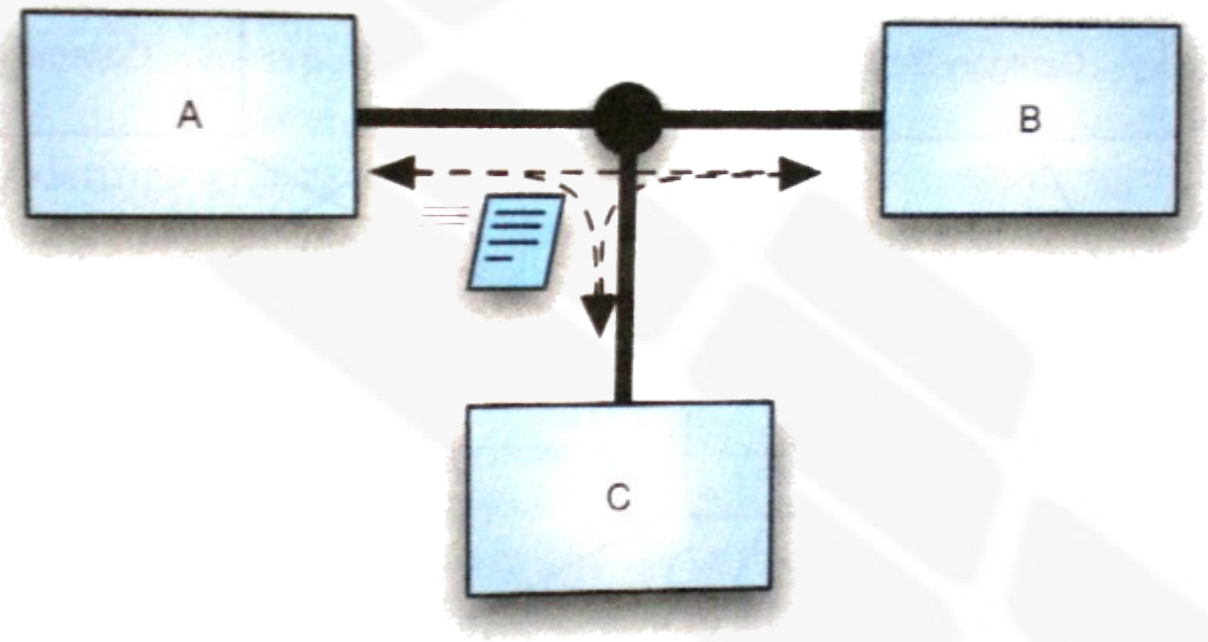
- Exemplo:



  - 1ª modificação: *i*) comunicação de B para A e; *ii*) entrega garantida
    - Adiciona-se um novo *pipe* de B para A
    - Utilização de *pipes* com *data buffering*
    - Adição de novos componentes requerem mais adição ou substituição de *pipes*, possivelmente com substancial *system downtime*
  - 2ª modificação: dados como pacotes discretos e tipados

Motivação

- Exemplo:



  - 2ª modificação: dados como pacotes discretos e tipados
    - Dados agora são processados de forma mais eficiente
    - Utiliza-se um conector *event bus*

Motivação

- Tanto o *pipe* quanto o *event bus* compartilham as propriedades de fraco acoplamento entre componentes, comunicação assíncrona e possível *data buffering*
- O *event bus* é mais adequado para suportar adaptação:
  - É capaz de estabelecer, *on-the-fly*, dutos entre componentes
  - Cardinalidade: um emissor – múltiplos receptores
  - Permitem, em qualquer momento durante a execução do sistema, a adição e remoção de componentes, e o *subscribe*/*unsubscribe* de tipos específicos de eventos

Fundamentos

- Os *building blocks* elementares e subjacentes a todo conector são as primitivas de gerenciamento de fluxo de controle e fluxo de dados
- Tais primitivas fornecem a base conceitual para a construção de conectores sofisticados e complexos
- Em adição a estas primitivas, todo conector mantém um ou mais canais (dutos), utilizados para ligar os componentes envolvidos na interação e suportar o fluxo de dados e controle entre eles
- Um duto é necessário para a existência do conector mas, por si só, não disponibiliza nenhum serviço adicional de interação

Fundamentos

- Os conectores mais simples, tais como o *module linker*, realizam seu serviço simplesmente formando dutos entre componentes
- Outros conectores adicionam aos dutos alguma combinação de controle de dados e de fluxo, com o objetivo de disponibilizar serviços de interação mais ricos
- Os conectores mais complexos podem também possuir uma arquitetura interna que inclui processamento e armazenamento de informação:
  - Ex: conector de balanceamento de carga (direcionamento da requisição baseado na carga dos componentes)

Fundamentos

- Os conectores mais simples estão tipicamente já implementados nas linguagens de programação
- Conectores compostos, por outro lado, formados pela composição de vários conectores (e possivelmente componentes), são geralmente disponibilizados como bibliotecas ou *frameworks*
- Conectores simples disponibilizam somente um tipo de serviço de interação
- Os conectores compostos ajudam a superar as limitações das linguagens de programação modernas

Tipos de Serviços de Interação

- Um conector disponibiliza um ou mais dos seguintes tipos de serviços de interação:
  1) Comunicação
  2) Coordenação
  3) Conversão
  4) Facilitação
- Todo conector disponibiliza serviços de pelo menos uma dessas categorias
- Um conjunto rico de capacidades de interação pode demandar o uso de múltiplos serviços:
  - Ex: *procedure call* (comunicação + coordenação)

Tipos de Serviços de Interação

1) Serviços de Comunicação:
   - Suporta a transmissão de dados entre componentes
   - *Building block* primário para interação entre componentes
   - Exemplos de dados: mensagens, dados a serem processados, resultados de computações

Tipos de Serviços de Interação

2) Serviços de Coordenação:

- Suportam a transferência de controle entre componentes
- A *thread* de execução é passada de um componente para outro
- Exemplos de conectores de coordenação: chamadas de função e invocações de método
- Conectores de mais alta ordem, tais como aqueles utilizados para balanceamento de carga, disponibilizam interações mais ricas e complexas construídas em torno dos serviços de coordenação

Tipos de Serviços de Interação

3) Serviços de Conversão:

- Transformam a interação requerida por um componente naquela disponibilizada por outro
- Permitir que componentes heterogêneos interajam não é uma tarefa fácil, interações divergentes estão sempre presentes
- Tais divergências são causadas por incompatibilidades no tipo, número, frequência e ordem das interações
- Serviços de conversão permitem que componentes que não foram projetados para trabalhar em conjunto possam estabelecer e conduzir interações
    - Exs: conversão de formato de dados e *wrappers* para componentes legados

Tipos de Serviços de Interação

4) Serviços de Facilitação:

- Realizam a mediação e simplificação da interação entre componentes
- Mesmo quando os componentes foram projetados para trabalhar em conjunto pode ser necessário disponibilizar mecanismos para melhor facilitar e otimizar as suas interações
- Mecanismos tais como balanceamento de carga, serviços de escalonamento e controle de concorrência podem ser necessários para atender certos requisitos não-funcionais e para reduzir inter-dependência entre componentes

Tipos de Conectores

- Representam o nível geral de abstração utilizado pelos arquitetos
- Conectores simples são modelados neste nível e detalhes podem ser deixados para o projeto e implementação
- Conectores mais complexos, entretanto, requerem que seus detalhes sejam decididos também no nível arquitetural
- Esses detalhes representam variações nas instâncias do tipo de conector (dimensões do conector)

Tipos de Conectores

- Cada dimensão do conector possui um conjunto do possíveis valores
- A seleção de um único valor para cada dimensão resulta em uma espécie concreto de conector:
  - Instanciando dimensões de um único tipo de conector obtém-se um conector simples
  - Instanciando dimensões de múltiplos tipos de conectores resulta em conectores *composite* (de mais alta ordem)

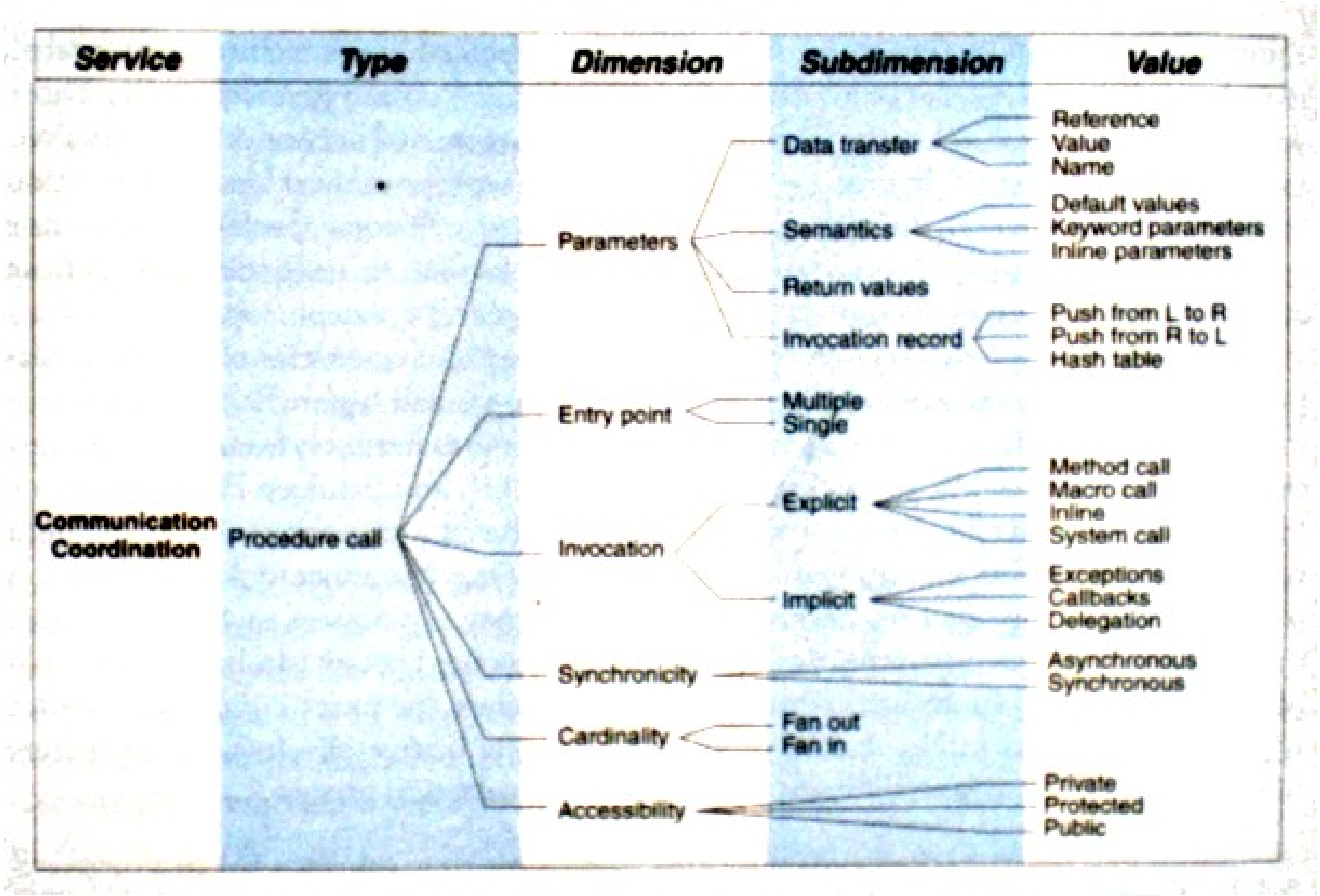
Tipos de Conectores

- Classificação dos conectores quanto à forma de realização dos serviços de interação:
  1) *Procedure Call*
  2) *Event*
  3) *Data Access*
  4) *Linkage*
  5) *Stream*
  6) *Arbitrator*
  7) *Adaptor*
  8) *Distributor*

Tipos de Conectores
Procedure Call

- Modelam o fluxo de controle entre componentes através de diversas técnicas de invocação (serviço de **coordenação**)
- Adicionalmente, podem transferir dados entre os componentes envolvidos através de parâmetros e valores de retorno (serviço de **comunicação**)
- São os mais conhecidos e amplamente utilizados: métodos na orientação a objetos, chamadas de sistema e *callbacks*
- Frequentemente utilizados como base para conectores *composite*, como *Remote Procedure Call* (RPC). Neste caso, também realizando serviços de **facilitação**

Tipos de Conectores
Procedure Call



Tipos de Conectores
Event

Evento: efeito instantâneo da conclusão (normal ou errônea) da invocação de uma operação em um objeto, ocorrendo na localização do próprio objeto

David Rosenblum e Alexander Wolf (1997)

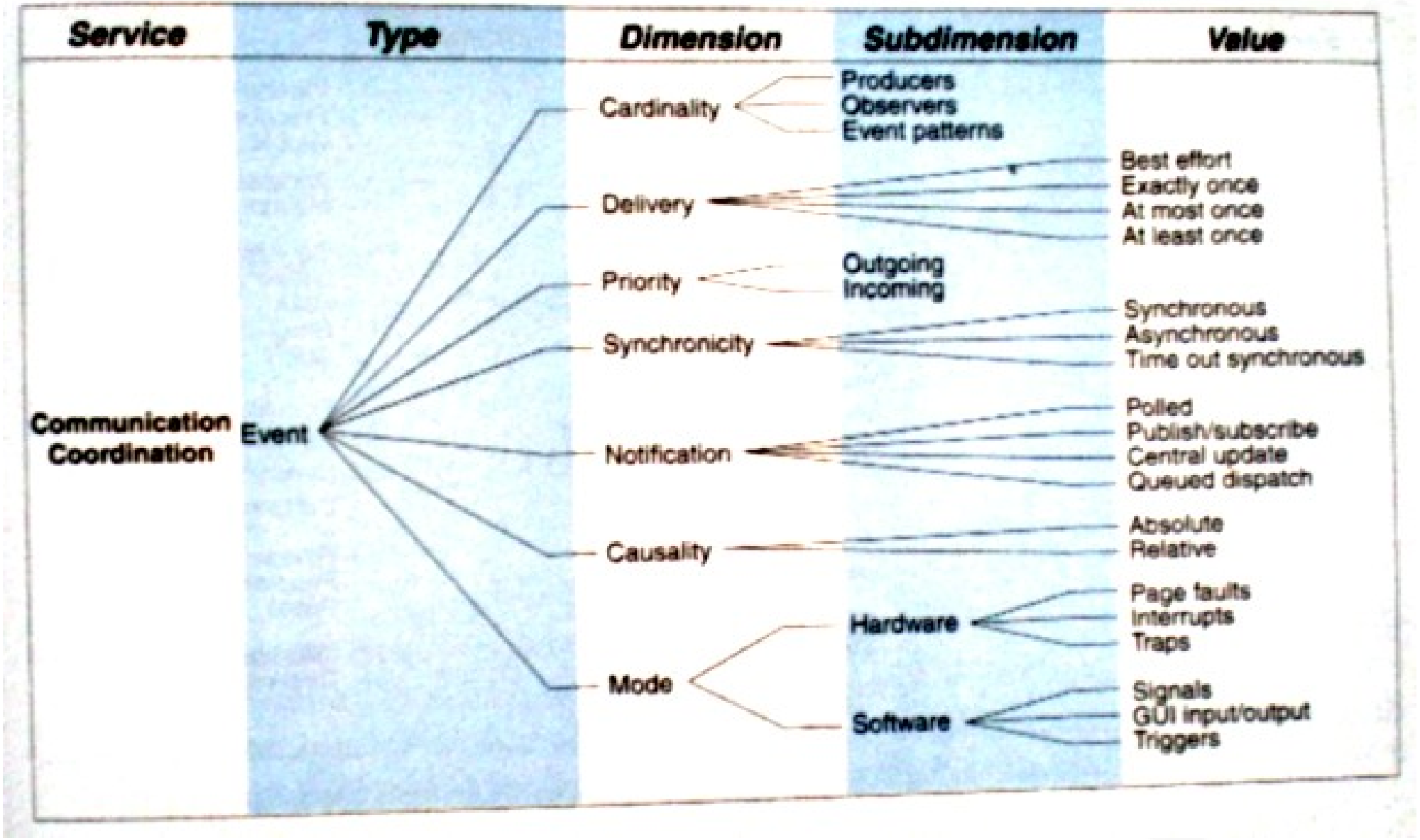
- São similares ao *Procedure Call* pois afetam o fluxo de controle entre componentes (serviços de **coordenação**)
  - O fluxo de controle é iniciado pela ocorrência do evento
  - O conector, uma vez ciente da ocorrência do evento, gera mensagens (notificações de eventos) para todas as partes interessadas e produz controle para que os componentes processem estas mensagens

Tipos de Conectores
Event

- Entretanto, diferem do *Procedure Call* pois:
  - Mensagens podem ser geradas na ocorrência de um único evento ou de um padrão específico de eventos
  - O conteúdo do evento pode ser estruturado para conter informações tais como hora e local da ocorrência do evento e outros dados específicos de aplicação (serviços de **comunicação**)
  - Formam “conectores virtuais” entre os componentes interessados no mesmo evento
  - Tais “conectores virtual” aparecem e desaparecem dinamicamente durante a execução do sistema

Tipos de Conectores

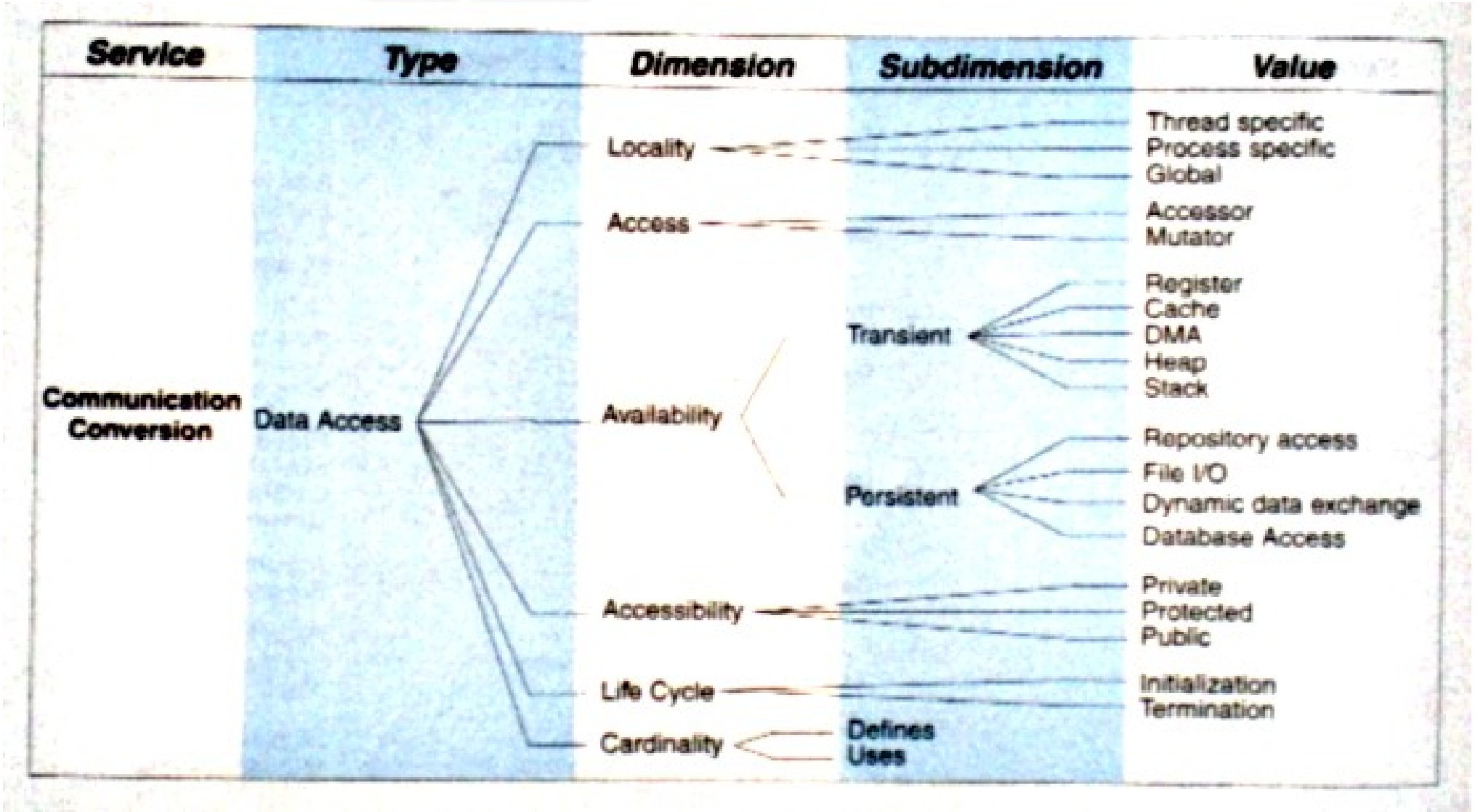
Event



Tipos de Conectores
Data Access

- Permitem que componentes acessem dados mantidos em um componente de armazenamento de dados - *Data Store* (serviços de **comunicação**)
- Tal acesso frequentemente requer inicialização e limpeza do *Data Store* antes e depois da operação, respectivamente
- Caso exista diferença de formato entre o dado requerido e aquele armazenado o conector pode realizar tradução da informação (serviço de **conversão**)
- O *Data Store* pode ser persistente ou temporário, impactando no mecanismo utilizado pelo conector
- Ex: mecanismos de *query* (SQL) e acesso a informação de repositórios tais como os de componentes de *software*

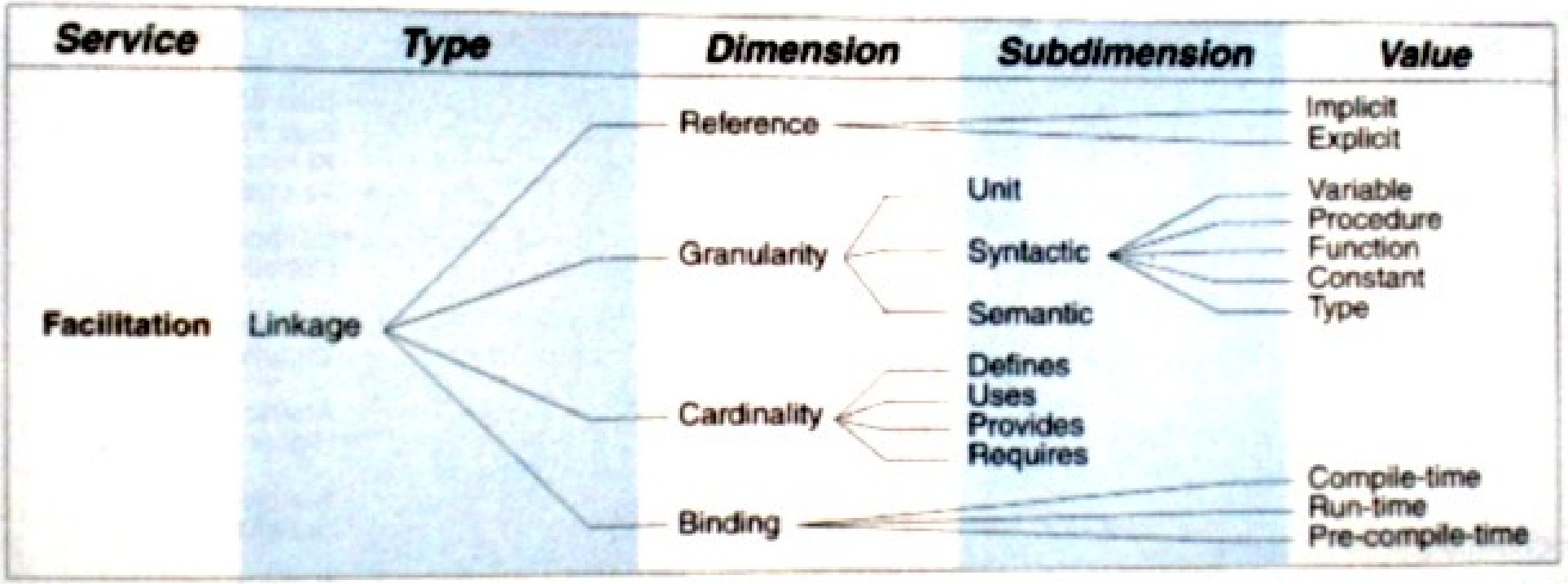
Tipos de Conectores
Data Access



Tipos de Conectores
Linkage

- Utilizados para unir os componentes e mantê-los desta forma durante sua interação
- Viabilizam o estabelecimento de dutos – canais de comunicação e coordenação – que são então utilizados por conectores de mais alta ordem para garantir a semântica de interação
- Realizam serviços de **facilitação**
- Uma vez o duto estabelecido o conector *linkage* pode ser removido do sistema ou nele permanecer para facilitar a evolução do *software*
- Ex: ligações entre componentes e barramentos no C2 e relacionamentos de dependência entre módulos de *software* em *Module Interconnection Languages* (MIL)

Tipos de Conectores
Linkage



Tipos de Conectores
Linkage

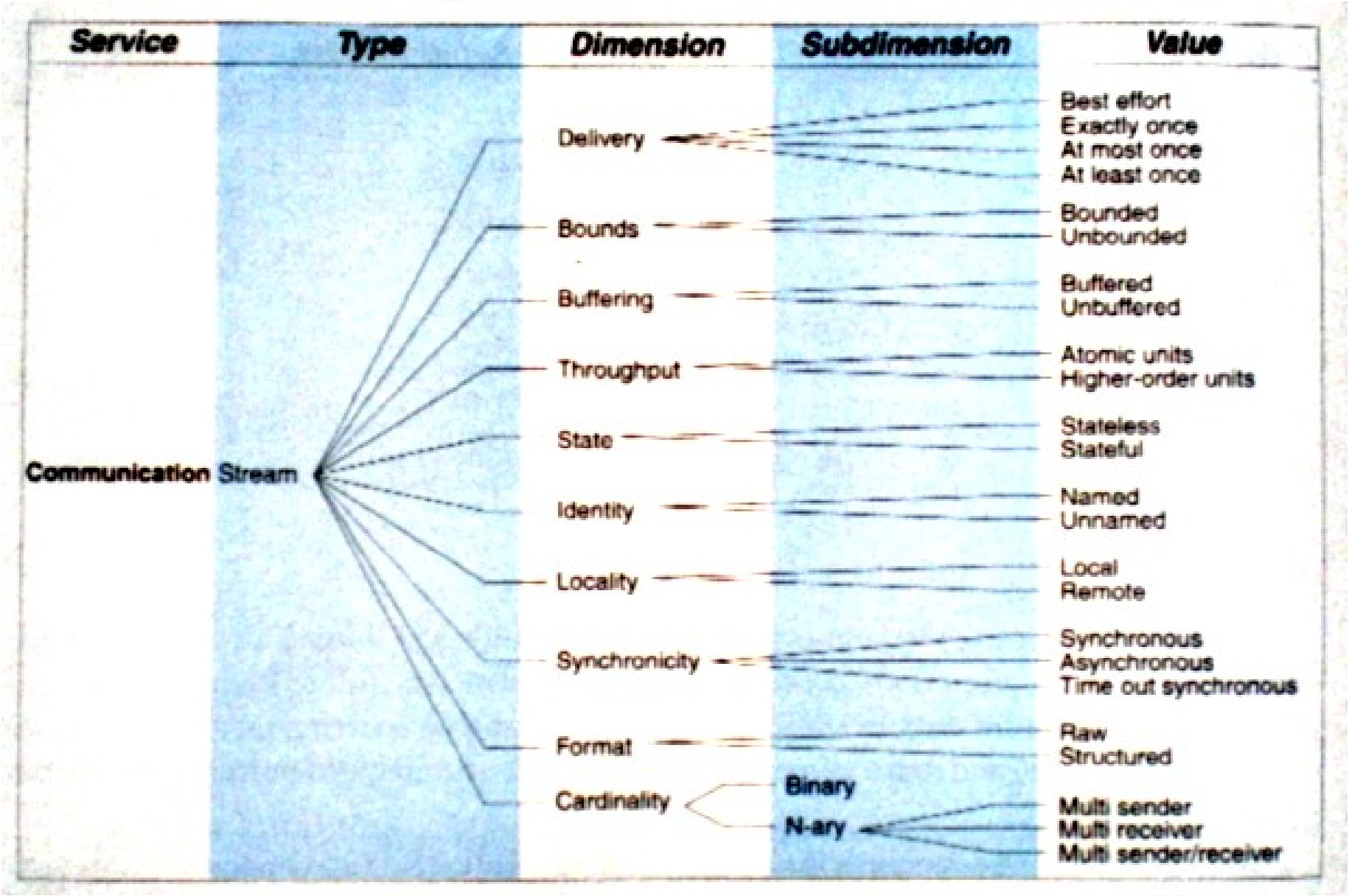
- Sub-dimensões de granularidade:
  - Interconexões de unidade especificam somente que um componente (modulo, objeto, arquivo, …) depende de outro. Ex: *build tools* tais como o *Make*
  - Interconexões sintáticas refinam este relacionamento e estabelecem ligações entre variáveis, *procedures*, funções, constantes e tipos definidos dentro do componente conectado. Ex: análise estática e *smart compilation*
  - Interconexões semânticas especificam **como** os componentes ligados devem interagir. Garantem que os requisitos e restrições da interação são explicitamente afirmados e satisfeitos. Ex: protocolos de interação

Tipos de Conectores
Stream

- Utilizados para realizar transferências de grandes quantidades de dados entre processos autônomos (serviços de **comunicação**)
- São também utilizados em protocolos de transmissão de dados em sistemas *client-server* para realizar a entrega de resultados de computações
- Podem ser combinados com outros tipos de conectores:
  - *Data Access* para definir conectores *composite* de acesso a bancos de dados
  - *Event* para multiplexar a entrega de uma grande quantidade de eventos
- Ex: *pipes* do Unix, *sockets* de comunicação TCP/UDP e protocolos *client-server* proprietários

Tipos de Conectores

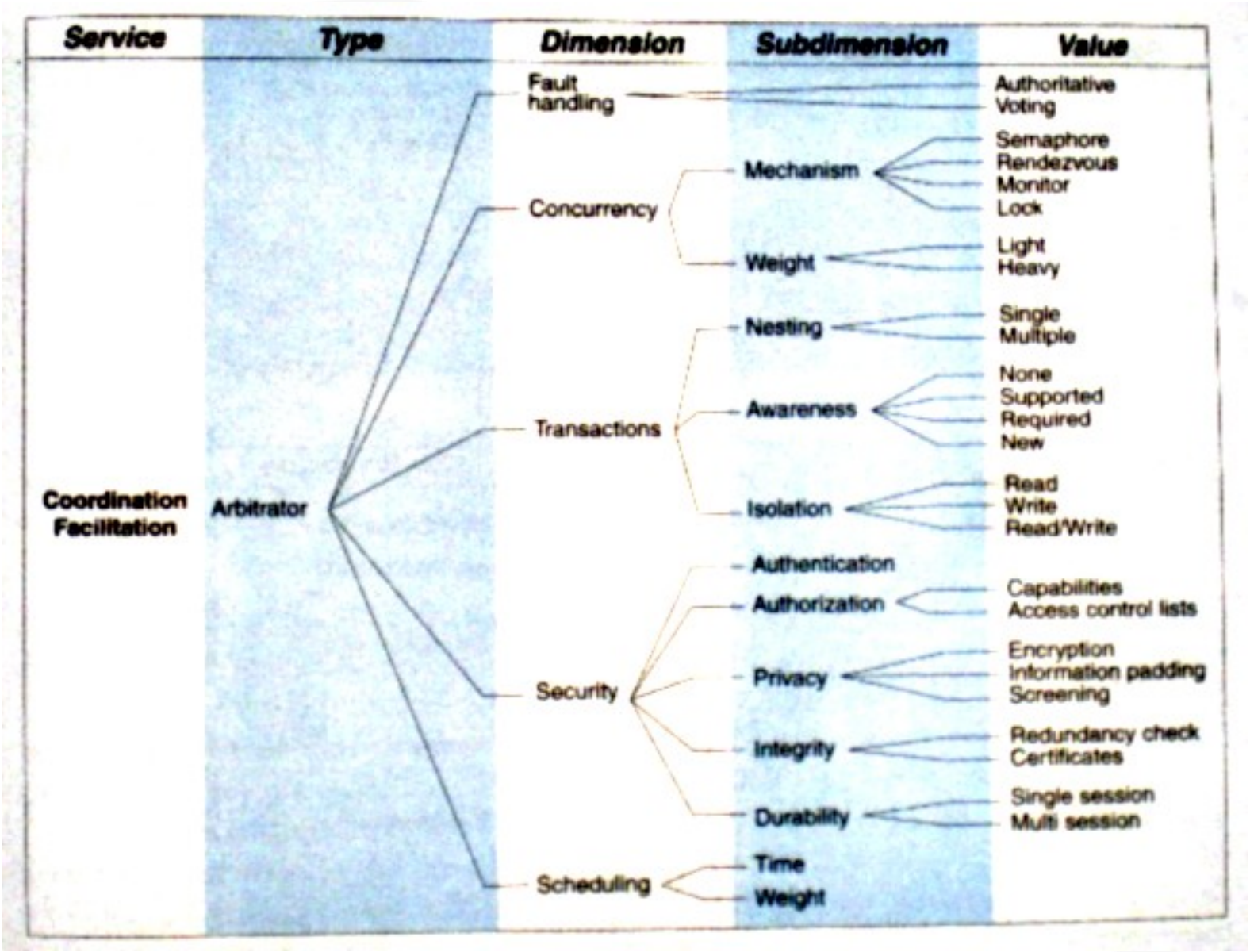
Stream



Tipos de Conectores
Arbitrator

- Facilitam a operação do sistema, resolvem eventuais conflitos (serviços de **facilitação**) e redirecionam o fluxo de controle (serviços de **coordenação**) naquelas situações onde um componente conhece a presença de outros componentes porém nada pode assumir sobre suas necessidades e seus estados
  - Ex: garantia de consistência e atomicidade de operações, através de sincronização e controle de concorrência, em sistemas *multithreaded* com memória compartilhada
  - Ex: negociação de níveis de serviço e mediação de interações que requerem confiabilidade e atomicidade
  - Ex: serviços de escalonamento e balanceamento de carga
  - Ex: *reliability*, *security* e *safety* para implementação de *dependability* e *trustworthiness*

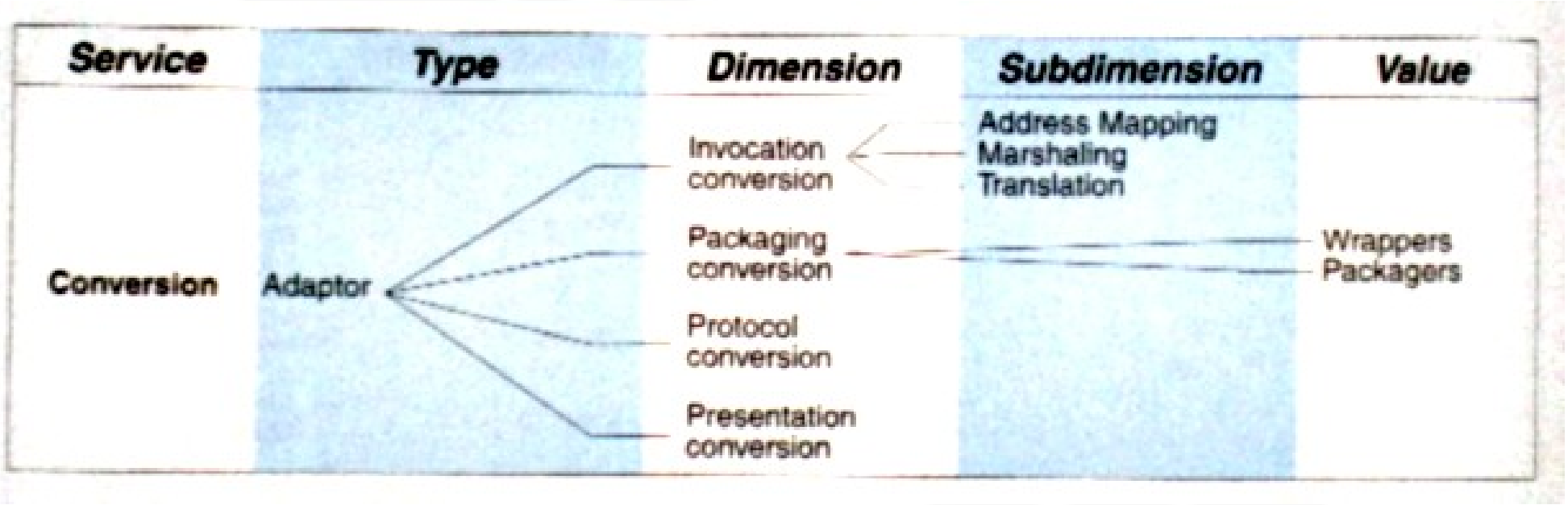
Tipos de Conectores
Arbitrator



Tipos de Conectores
Adaptor

- Suportam a interação entre componentes que não foram originalmente projetados para interoperar
- Compatibilizam políticas de comunicação e protocolos de interação (serviços de **conversão**)
- Necessários em ambientes heterogêneos
- A conversão pode ter a melhoria do desempenho como foco:
    - Um *Remote Procedure Call* pode ser automaticamente convertido para um *Procedure Call* local caso os dois componentes estejam na mesma máquina
- Podem aplicar transformações (ex*: look-ups*) para compatibilizar os serviços requeridos com as facilidades disponíveis

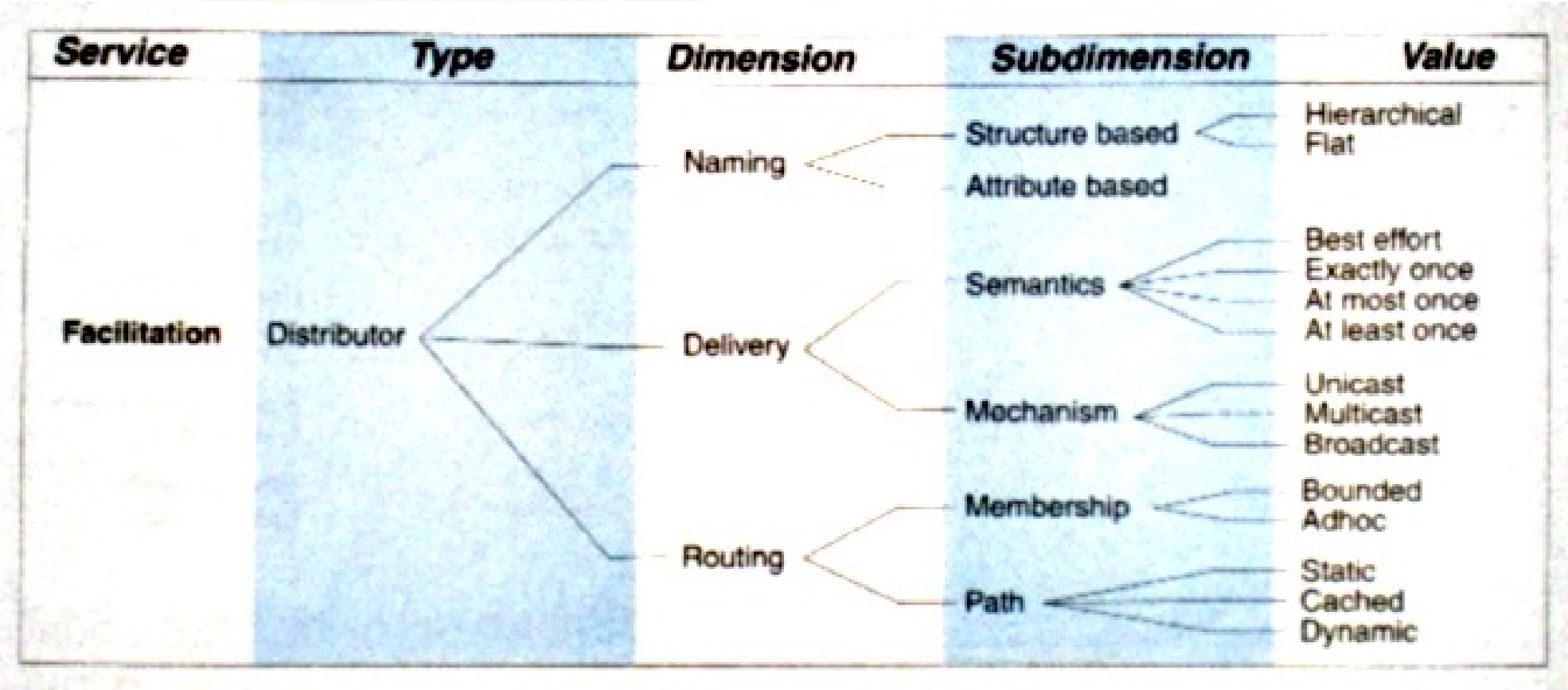
Tipos de Conectores
Adaptor



Tipos de Conectores
Distributor

- Realizam a identificação dos caminhos *(paths)* de interação e o subsequente roteamento, de informações de comunicação e coordenação, através de objetos ao longo deste caminho (serviços de **facilitação**)
- Nunca existem de forma isolada, eles dão assistência a outros conectores como *Stream* ou *Procedure Call*
- Direcionam o fluxo de dados durante troca de informação em sistemas distribuídos
- Ex: serviços de identificação da localização de componentes e de caminhos até eles, a partir de nomes simbólicos (DNS)
- Tem efeito importante na escalabilidade e resiliência do sistema

Tipos de Conectores
Distributor



INF016 – Arquitetura de Software

05 – Conectores

Sandro Santos Andrade
sandroandrade@ifba.edu.br

Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Tecnologia Eletro-Eletrônica
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas