Middleware Reflexivo

Construindo Sistemas Distribuídos Flexíveis, Adaptáveis e Reconfiguráveis

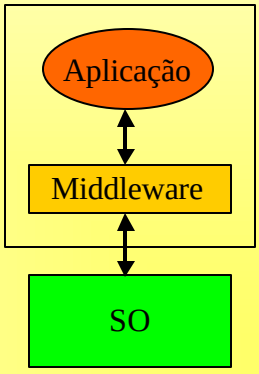
Prof. Dr. Fabio Kon

Departamento de Ciência da Computação
IME / USP

21 de maio de 2003
SBRC'2003 - Natal, RN

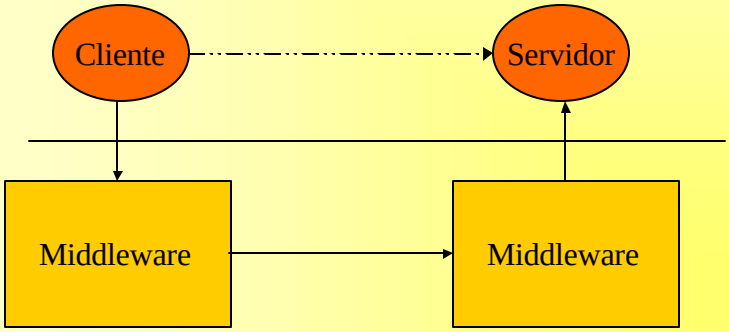
O que é "Middleware"?

- Software que
  - reside entre o sistema operacional e a aplicação
  - a fim de facilitar (simplificar) o desenvolvimento de aplicações.



O que é "Middleware de Comunicação"?

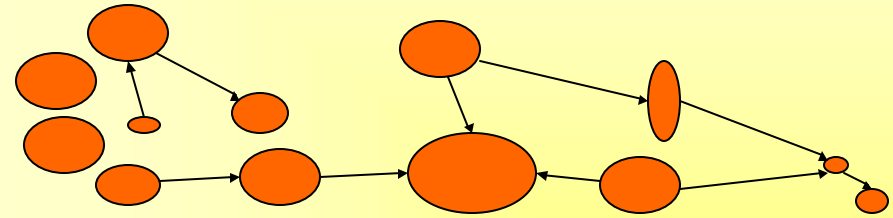
- Middleware cujo objetivo é facilitar o desenvolvimento de aplicações e sistemas distribuídos.



Benefícios do Middleware de Comunicação

- esconde do programador diferenças entre
  - plataformas de hardware
  - sistemas operacionais
  - bibliotecas de comunicação
  - protocolos de comunicação
  - formatação de dados
  - linguagens de programação
  - modelos de programação

Esconde as Heterogeneidades



Middleware	Middleware	Middleware	Middleware	Middleware
Linux	Windows	Solaris	MacOS	PalmOS
PC	PC	Sparc	iMAC	Palm

Outros Benefícios

- Gerenciamento de Nomes e Referências
- Ativação Automática de Serviços
- Seleção do Receptor das Mensagens
- Migração de Serviços
- Controle de Qualidade de Serviço (QoS)
- Gerenciamento de Concorrência
  - único *thread, thread* para cada serviço, para cada cliente, *pool* de *threads* etc.
- Gerenciamento de Conexões

Serviços Padronizados

- Nomes
- Negociação (*Trading*)
- Tolerância a Falhas
- Segurança
- Transações
- Persitência
- etc. etc.

Principais Plataformas de Middleware Existentes

- CORBA da OMG
- J2EE da Sun (também J2ME e J2SE)
- DCOM da Microsoft
- .NET da Microsoft (ainda em formação)

Conclusão Parcial

- Sistemas de middleware facilitam enormemente a criação de aplicações distribuídas complexas.
- Mas o mundo está se tornando mais dinâmico a cada dia.
- O middleware convencional está preparado para os desafios das próximas décadas?

Limitações dos Primeiros Sistemas de Middleware

- Modelo de concorrência fixo
- Protocolo de transporte fixo (e.g., IIOP/TCP/IP)
- Estratégia de segurança fixa
  - (normalmente nenhuma)
- Algoritmo de escalonamento fixo
- Formatação de dados fixa
- Arquitetura monolítica
  - tudo ou nada ⇒ sistemas enormes

Ambientes onde arquiteturas fixas e monolíticas não funcionam

- Sistemas Altamente Dinâmicos
  - Computação Móvel
  - Computação Ubíqua
  - Multimídia
- Sistemas Especializados
  - Sistemas embutidos ou dedicados
  - Aplicações adaptativas

Paradoxo do Middleware

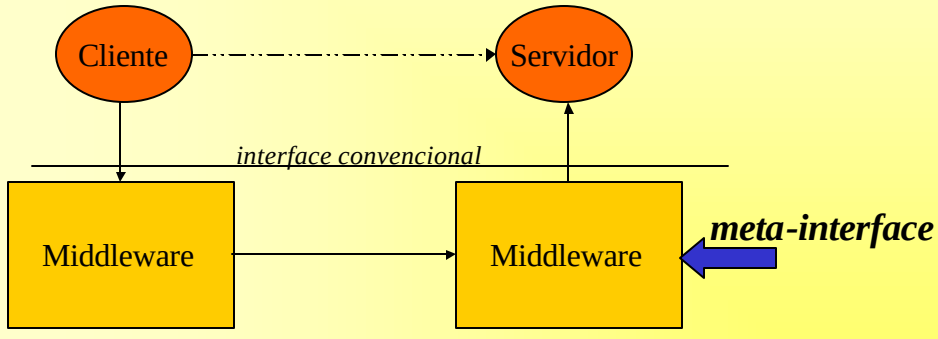
- A maior parte das aplicações necessitam que o middleware esconda os detalhes das camadas inferiores.
- Algumas aplicações podem obter grandes melhorias em desempenho se puderem observar e manipular os detalhes das camadas inferiores.

Solução:
Modelo Reflexivo para Middleware

Middleware Reflexivo:

- Oferece transparência para as aplicações que a desejam.
- Permite a inspeção e manipulação dos detalhes internos do Middleware e do SO para as aplicações que necessitarem.

Middleware Reflexivo



Interfaces

1. Interface Convencional
- permite a execução de aplicações convencionais
2. Meta-Interfaces
- inspeção dos componentes internos do middleware
- alteração da configuração interna do middleware
- inspeção do estado dinâmico do middleware
- inspeção do estado dinâmico do SO
- inspeção da configuração estática do SO e do HW

Reflexão Computacional

- Sistemas Reflexivos [Smith 84]
- Meta-Object Protocol (MOP) [Kiczales 91]
- ORBs Reflexivos [Singhai and Campbell 97]

- Reflexão: "pensar" sobre si mesmo.
  - manter representação de sua estrutura interna
  - inspecionar configuração interna
  - alterar configuração interna

Exemplos de Sistemas de Middleware Reflexivos

- *dynamicTAO*
  - University of Illinois at Urbana-Champaign
- Open ORB
  - Lancaster University
- UIC
  - UIUC / UbiCore
- Outros
  - LuaORB, LORB, mChaRM, COMERA, OpenCOM

dynamicTAO

- Extensão do ORB CORBA *TAO* [Schmidt et al]
- Escrito em C++
- Design modular baseado em padrões de projeto OO.
- TAO já permitia configuração no momento da inicialização:
  - modelo de concorrência (threads)
  - demultiplexação de mensagens
  - escalonamento
  - gerenciamento de conexões

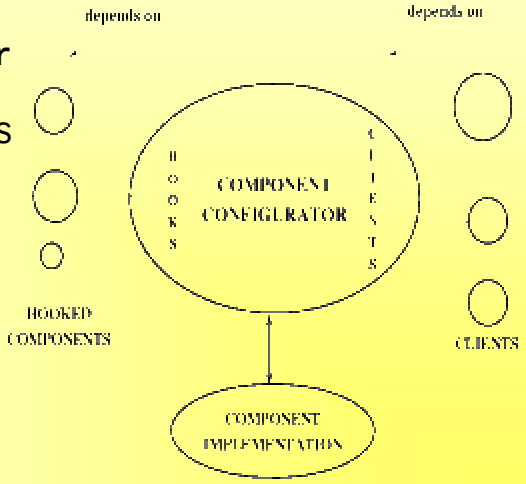
Acrescentando Suporte para Reconfiguração Dinâmica

dynamicTAO exporta a interface **`DynamicConfigurator`** que permite:

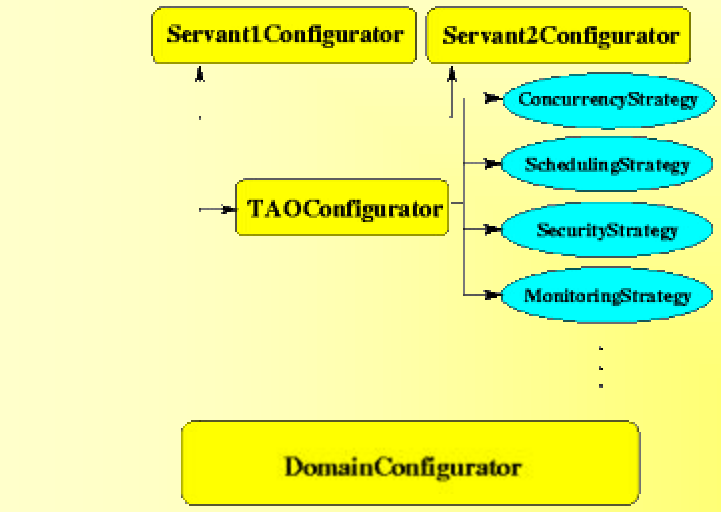
1. Transferência de componentes através da Internet.
2. Carga e descarga dinâmica de componentes.
3. Inspeção e modificação da configuração do ORB (e das aplicações sendo executados sobre ele).

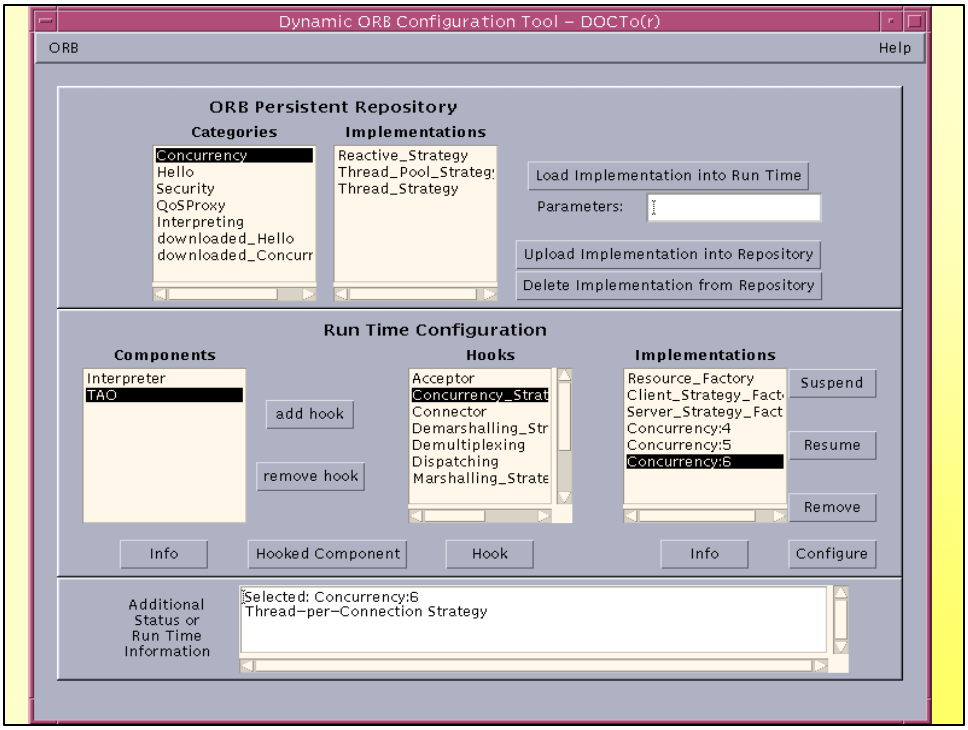
Reificação da Estrutura do ORB

- Arcabouço **`ComponentConfigurator`**
- Armazena dependências entre componentes.
- Permite navegação, inspeção e reconfiguração.
- Pode ser personalizado através de herança OO.



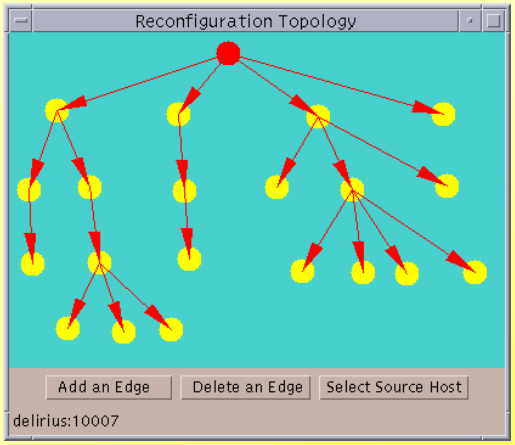
Estrutura do *dynamicTAO*





Reconfiguração com Agentes Móveis

- Para sistemas de grande escala
- Agente carrega
  - grafo
  - script de reconfiguração
  - estado
  - resultados



Aplicações de *dynamicTAO*

- Monitoramento
  - configurável
- Segurança
  - controle de acesso com DAC, MAC, RBAC.
- Distribuição escalável de multimídia
- QoS em multimídia (MONET @ uiuc)
- Base do SO experimental *2K* (SRG @ uiuc)

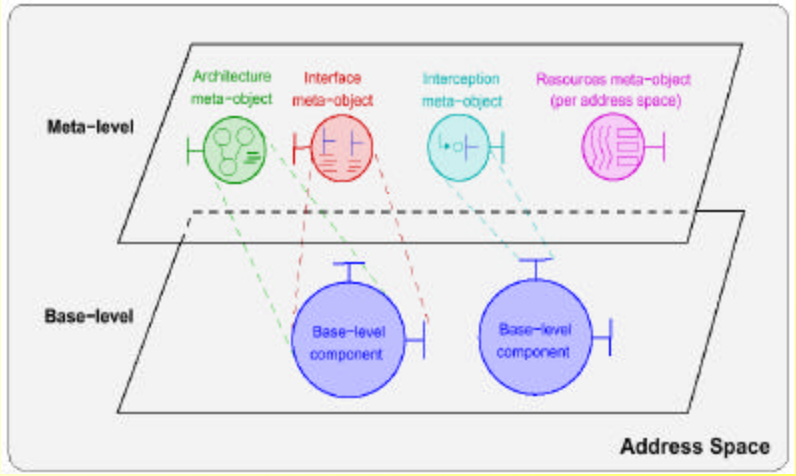
Open ORB

- Construído a partir do zero no modelo reflexivo
- Componentes:
  - implementam os vários elementos funcionais do ORB
- Instâncias personalizadas da plataforma:
  - produzidas através da combinação dos componentes apropriados
- Modelo de componentes
  - composição hierárquica
  - distribuição
  - componentes como entidades de tempo de execução

Open ORB: Reconfiguração Dinâmica

- Reflexão como princípio fundamental
  - Uma clara separação entre
    - *nível base* : componentes que implementam os serviços usuais de middleware
    - *meta-nível* : mecanismos que reificam a implementação da plataforma
      - inspeção e adaptação dinâmica
  - O meta-nível é definido utilizando o mesmo modelo de componentes do nível base
    - reflexão pode também ser aplicada ao meta-nível

Open ORB: Múltiplos Modelos de Meta-espaço



Open ORB: Meta-espaço de *Interfaces*

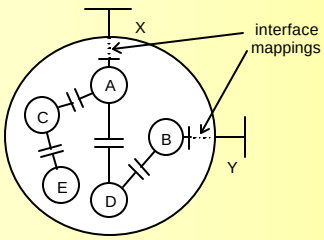
- Representação *externa* de um componente
  - conjunto de interfaces exportadas
  - operações e atributos de uma interface
- MOP: Inspeção
  - busca e enumeração de interfaces
  - enumeração das operações e atributos de uma interface

Open ORB: Meta-espaço de *Arquitetura*

- Representação interna de um componente
- Arquitetura de software do componente
  - grafo de componentes
    - interconexões entre os componentes mais primitivos que compõem o componente reificado
  - restrições arquiteturais
    - regras para validar configurações de componentes
    - facilitam a manutenção da consistência do ORB após reconfigurações

Open ORB: Modelo de Arq*uitetura* – Exemplo

Grafo de componentes



Restrições arquiteturais

- **Componente A depende do componente C**
- **Componente D não pode ser substituído ou removido**
- **Interface 1 do componente B só pode ser ligada a uma interface exatamente equivalente**

Open ORB: Meta-espaço de *Interceptadores*

- Permite adaptar as propriedades não-funcionais associadas às interfaces de componentes, tais como
  - segurança e sincronização em tempo real
- Pré- e pós-processamento das interações emitidas e recebidas em uma interface
- Interceptadores podem ser adicionados ou removidos
  - dinamicamente
  - em pontos arbitrários da implementação da plataforma

Open ORB: Meta-espaço de *Recursos*

- Acesso padronizado
  - aos recursos subjacentes da plataforma
  - aos mecanismos utilizados para o seu gerenciamento
- Inspeção e reconfiguração
  - recursos alocados a tarefas específicas
    - quantidade alocada
  - parâmetros associados aos recursos
    - prioridades, algoritmos de gerenciamento
- Fundamental para o gerenciamento de QoS em ambientes dinâmicos

Open ORB: Protótipos

- OOPP – Open ORB Python Prototype
- OpenCOM

dynamicTAO e Open ORB: Considerações gerais

- Dois projetos independentes, mas com
  - motivações muito parecidas
  - levaram ao mesmo modelo arquitetural: reflexão
- *dynamicTAO*
  - re-engenharia de TAO para adicionar flexibilidade de tempo de execução.
- Open ORB
  - projeto e implementação do middleware em função dos princípios de reflexão computacional.

Middleware (Enxuto) Baseado em Componentes

- Uso da tecnologia de componentes de software para construção da plataforma.
- Serviços e mecanismos do ORB implementados como componentes
  - implementações alternativas com propriedades diferentes.
- Plataforma pode ser configurada com apenas os componentes necessários.

UIC

- Universally Interoperable Core
  - UIUC e ubi-core.com
- A configuração de componentes pode ser alterada em tempo de execução
  - adicionar, substituir, remover componentes
- Implementação de diferentes personalidades de ORB
  - Ex.: CORBA, Java RMI

UIC: configuração abstrata

- Um esqueleto de ORB com componentes abstratos (ganchos) representando os tipos de funcionalidades comuns de middleware:
  - protocolos de comunicação
  - mecanismos de gerenciamento de conexões
  - seriação e de-seriação
  - chamada de métodos
  - escalonamento
  - geração de referências de objetos
  - APIs para clientes e servidores
  - gerenciamento de memória e concorrência

UIC: configuração concreta

- Implementações concretas de cada componente abstrato são carregados dinamicamente.
  - Diferentes personalidades de ORB podem ser criadas.
  - Permite atender aos requisitos de aplicações específicas.

UIC: reconfiguração dinâmica

- Mecanismos internos do ORB podem ser reconfigurados em tempo de execução
  - substituição da implementação de componentes abstratos.
- Por exemplo: variação de conectividade
  - computador móvel em rede com fio: TCP
  - deslocamento para uma rede sem fio: WTCP
  - mecanismos de transporte otimizados para cada ambiente.

UIC: Configurações Enxutas

- Middleware para um cliente minimal adotando personalidade CORBA:
  - 29KB no Windows CE
  - 72KB no Windows 2000
  - 18KB no PalmOS
- Middleware minimal para cliente/servidor
  - 48.5KB no Windows CE
  - 100KB no Windows 2000
  - 31KB no PalmOS
- Conclusão: ideal para sistemas embutidos

Outros Sistemas de Middleware Reflexivo

- mChaRM (U. Genova, Ítália)
- COMERA (Microsoft)
- AspectIX (Friedrich-Alexander U., Alemanha)
- LuaORB (PUC-Rio)
- Meu foco atual:
  - LORB (PUC-Rio) (Projeto InteGrade)

O Futuro do Middleware

- Arquiteturas convencionais de middleware tem adotado os resultados de pesquisas em middleware reflexivo
  - Ex.: interceptadores em CORBA, carga dinâmica de componentes em Orbix, arquitetura do JBoss.
  - Middleware existente está se tornando mais flexível e reconfigurável: abordagem *ad hoc.*
- Entretanto, a natureza de "caixa preta monolítica" de middleware convencional limita o quanto da implementação da plataforma pode ser exposta.

O Futuro que Esperamos

- Adoção de reflexão como princípio básico para o design de middleware.
- Quais seriam os benefícios:
  - solução genérica,
  - abertura abrangente dos aspectos internos da plataforma,
  - manipulação e adaptação da implementação da plataforma de maneiras não previstas *a priori.*

O que ainda falta fazer?

- Definição de um padrão internacional para middleware reflexivo
  - de forma a garantir interoperabilidade.
- Pesquisa teórica e prática em ferramentas para garantia de consistência em reconfigurações dinâmicas.
- Uso de middleware reflexivo por aplicações variadas.
- Bibliotecas de mecanismos automatizados para adaptar o middleware em tempo de execução
  - com o objetivo de facilitar a construção de aplicações adaptativas.

International Workshop on Reflective Middleware

- 1st Workshop on Reflective Middleware
  - junto com Middleware'2000 em NY
- 2nd Workshop on Adaptive and Reflective Middleware
  - junto com Middleware'2003 no Rio em junho
- Em ambos os casos os artigos estão disponíveis na Web.

Conclusões

- Middleware Reflexivo já é um conceito bem conhecido e aceito na comunidade acadêmica de middleware.
- A penetração em sistemas e padrões comerciais está avançando (mas mais lentamente do que gostaríamos).
- É uma área de pesquisa ativa.

Maiores Informações

Visão geral sobre Middleware Reflexivo:

www.ime.usp.br/~kon/papers/cacm02.pdf

Grupo de Sistemas Distribuídos do IME/USP:

http://gsd.ime.usp.br