

MAC-5715 – Tópicos de P.O.O.

POSA II – *Interceptor e Leader/Followers*

Germano Capistrano Bezerra
germanob@ime.usp.br

Roteiro

- **Apresentação**
- **Motivação do POSA II**
- **Organização do POSA II**
- **Padrão “Interceptor”**
- **Padrão “Leader/Followers”**
- **Comentários Finais**

Objetivos

**Apresentar a estrutura básica do livro
POSA II e discutir padrões em particular:
Interceptor e Leader/Followers**

Motivação para o POSA II

- **POSA I** – Padrões de uso geral
- **POSA II** – Padrões com foco específico em concorrência e aplicações em rede
- **Padrões apresentam forte sinergia** – Representam a base para uma linguagem de padrões para software concorrente e em rede
- **Desafio de ambientes distribuídos e com concorrência**

Organização do POSA II

- **Há quatro tipos de padrões:**
 - **Configuração e acesso a serviços** – para o desenvolvimento de APIs para estabelecer acesso e configuração de serviços
 - **Gerenciamento de Eventos** – para controlar a captura, demultiplexação, disparo e processamento de eventos
 - **Sincronização** – possibilitar que aplicações absorvam os benefícios da concorrência
 - **Concorrência** – políticas e mecanismos para possibilitar que vários processos ou fluxos de execução ocorram de forma simultânea

Organização do POSA II

- **Formato dos padrões é baseado no estabelecido para o primeiro volume:**
 - Exemplo
 - Contexto
 - Problema
 - Solução
 - Estrutura
 - Dinâmica
 - Implementação
 - Exemplo Resolvido
 - Variações
 - Usos Conhecidos
 - Conseqüências
 - Ver Também

Interceptor

O padrão arquitetural *Interceptor* possibilita que serviços sejam adicionados de forma transparente a um arcabouço e disparados automaticamente quando certos eventos ocorrem

Interceptor – Exemplo

- **MiddleORB** – Provê serviços de comunicação que simplificam o desenvolvimento de aplicações distribuídas
- **Além dos serviços de comunicação, pode-se necessitar de serviços como:**
 - Transações
 - Segurança
 - Balanceamento de carga
 - Tolerância a falhas
 - Auditoria

Interceptor – Exemplo

- **MiddleORB precisa possibilitar a integração desses serviços estendidos**
 - Poderia integrar o maior número possível de serviços na configuração padrão, mas nem todos os serviços podem ser antecipados
 - Poderia deixar ser o mais simples e conciso possível. Os novos serviços devem ser adicionados diretamente no código do cliente e do servidor

Interceptor – Contexto

- **Contexto** – Desenvolvimento de arcabouços que podem ser estendidos de forma transparente
- **Problema**
 - Arcabouços não podem antecipar todos os serviços que precisam oferecer
 - Alguns arcabouços podem ser difíceis de se estender
 - Para não prejudicar reutilização, a tarefa de extensão não pode ficar a cargo das aplicações

Interceptor – Problema

- **Requisitos gerais para um arcabouço**
 - Deve permitir a integração de serviços adicionais sem necessitar de alterações em sua estrutura central
 - A integração de serviços específicos a determinadas aplicações não deve afetar os componentes existentes
 - Aplicações podem precisar monitorar e controlar o comportamento de um arcabouço

Interceptor – Solução

- **Registro de serviços através de interfaces pré-definidas**
- **Exposição da implementação do arcabouço**
 - Interface define um interceptador para um conjunto de eventos – aplicações especificam interceptadores concretos
 - Despachante definido para cada Interceptador
 - Objetos de Contexto permitem introspecção

Interceptor – Estrutura

<i>Classe</i> Concrete Framework	<i>Colaboradores</i> -Dispatcher
<i>Responsabilidades</i> -Define os serviços de aplicação -Integra os despachantes que permitem a interceptação de eventos -Delega eventos aos despachantes associados	

Interceptor – Estrutura

<i>Classe</i> Interceptor	<i>Colaboradores</i>
<i>Responsabilidades</i> -Define uma interface para a integração de novos serviços	

Interceptor – Estrutura

<i>Classe</i>	<i>Colaboradores</i>
<p>Concrete Interceptor</p> <p>Responsabilidades</p> <ul style="list-style-type: none">-Implementa um serviço específico-Utiliza os objetos de contexto para controlar o <i>Concrete Framework</i>	<p>-Context Object</p>

Interceptor – Estrutura

Classe Dispatcher	Colaboradores -Interceptor -Application
Responsabilidades -Possibilita que aplicações registrem e removam instâncias de <i>Concrete Interceptor</i> -Despacha chamadas aos interceptadores quando ocorrem eventos	

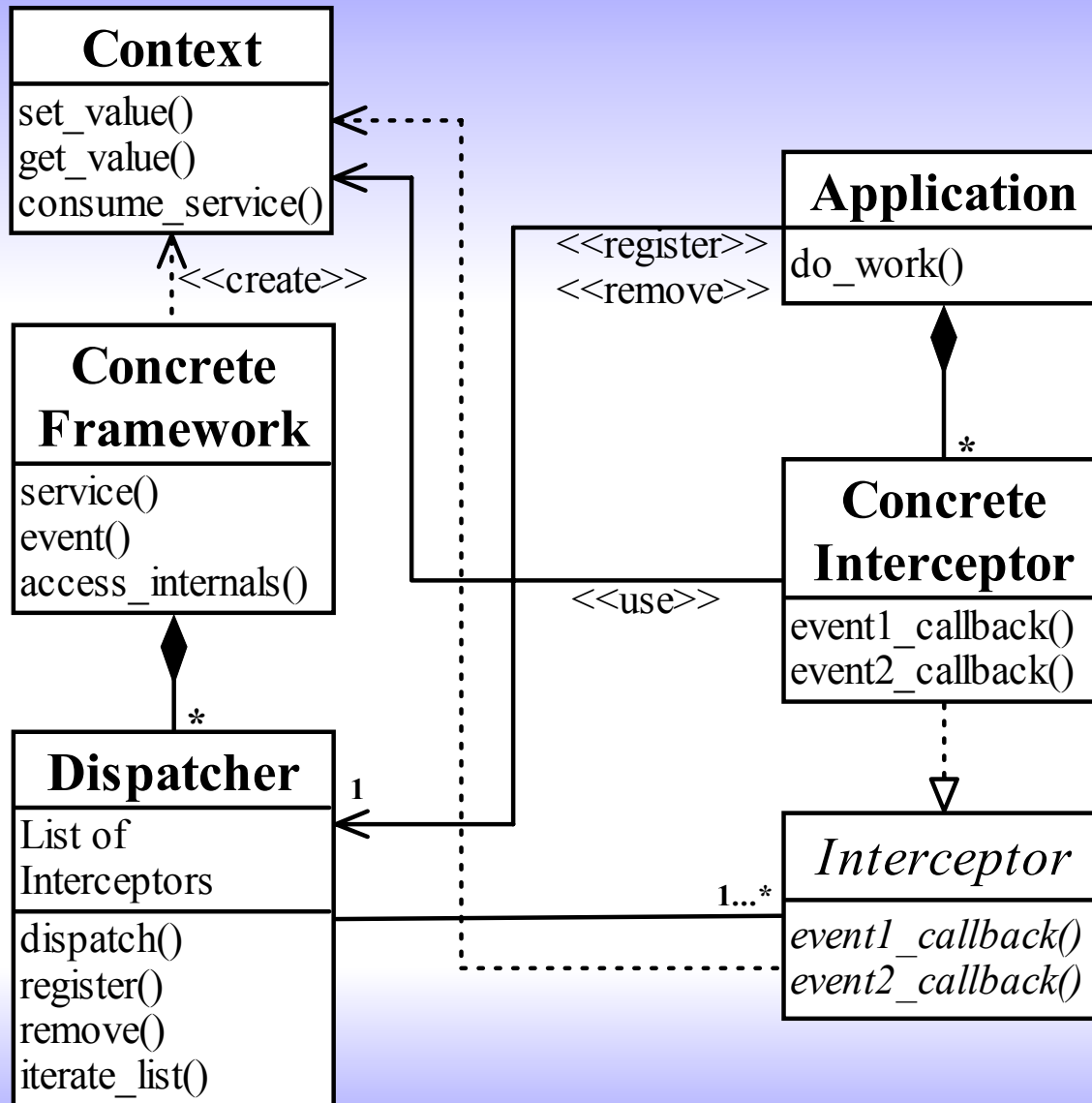
Interceptor – Estrutura

Classe	Colaboradores
<p>Context Object</p> <hr/> <p>Responsabilidades</p> <ul style="list-style-type: none">-Possibilita que serviços obtenham informações sobre o <i>Concrete Framework</i>-Possibilita que serviços controlem certos comportamentos do <i>Concrete Framework</i>	<ul style="list-style-type: none">-Concrete Framework

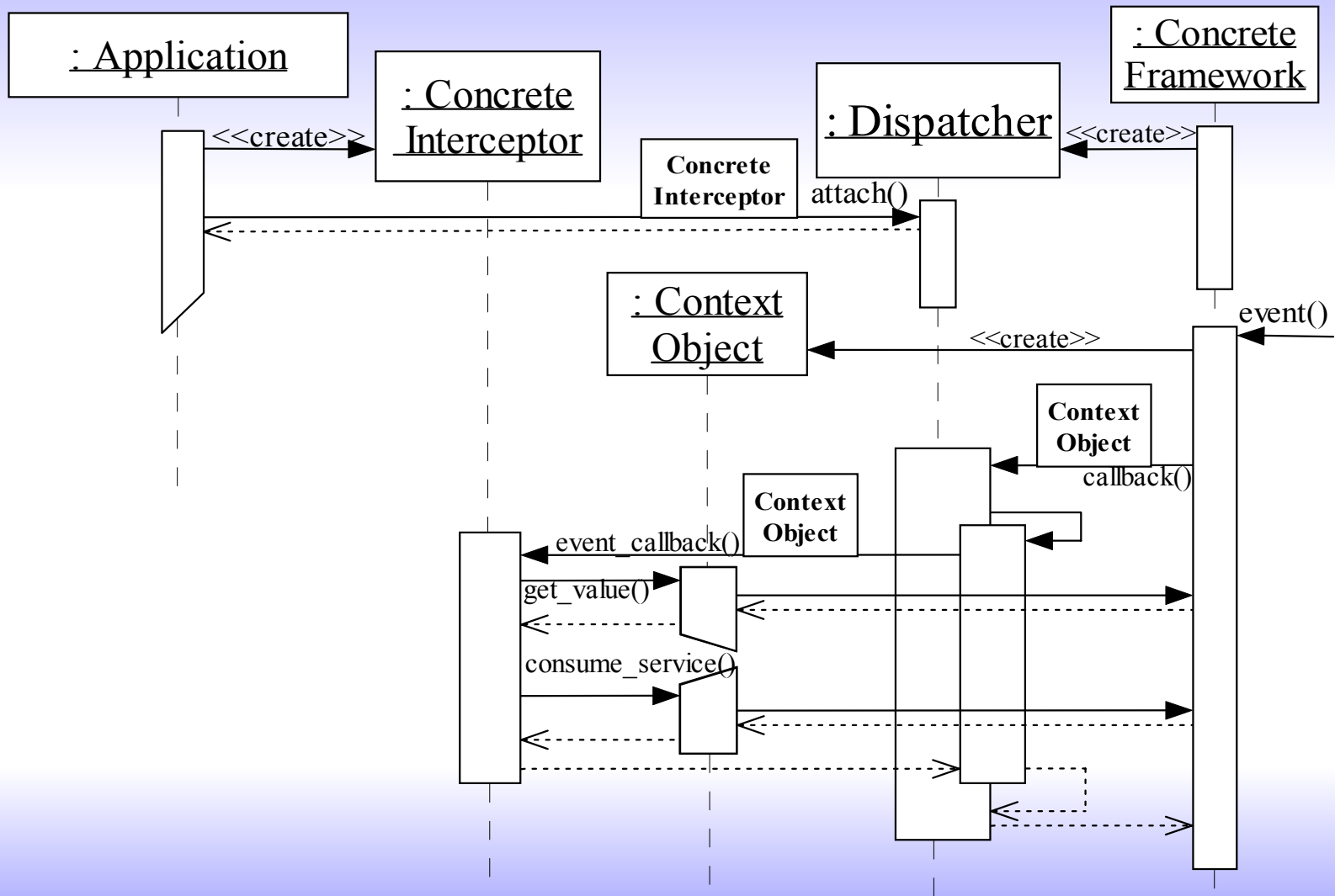
Interceptor – Estrutura

<i>Classe</i>	<i>Colaboradores</i>
<p><i>Application</i></p> <p><i>Responsabilidades</i></p> <ul style="list-style-type: none">-Executa sobre o <i>Concrete Framework</i>-Implementa instâncias de <i>Concrete Interceptor</i> e os registra com despachantes	<ul style="list-style-type: none">-Dispatcher-Concrete Interceptor

Interceptor – Estrutura



Interceptor – Dinâmica



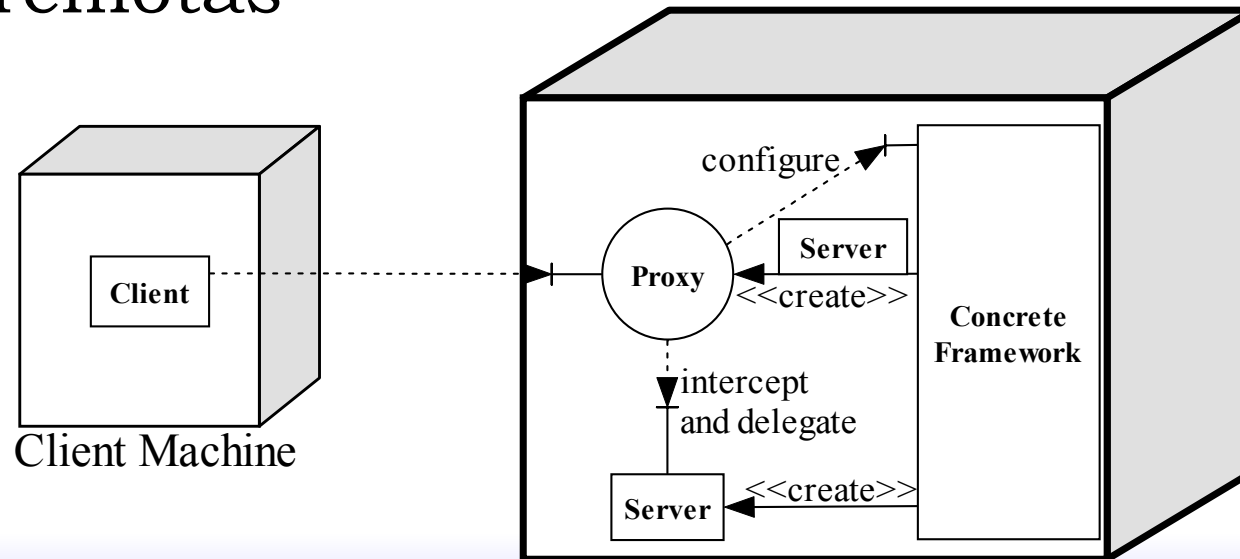
Interceptor – Implementação

1. Modelar o comportamento interno do arcabouço
2. Identificar e modelar os pontos de interceptação
3. Especificar os objetos de contexto
4. Especificar os interceptadores
5. Especificar os despachantes
6. Implementar os mecanismos de *callback*
7. Implementar os interceptadores concretos

Interceptor – Variação

- **Interceptor Proxy**

- utilizado por servidores de aplicações distribuídas para interceptar operações remotas



Interceptor - Usos Conhecidos

- **Servidores de Aplicação baseados em componentes**
 - EJB, CCM de CORBA e COM+ da Microsoft
- **Algumas implementações de CORBA**
 - TAO e Orbix
- **ORB reflexivo dynamicTAO**
 - Usa interceptadores para monitoramento e segurança

Interceptor – Conseqüências

- **Benefícios**

- Mecanismos de extensão e flexibilidade
- Separação de tarefas
- Provê suporte para monitoramento e controle de arcabouços
- Simetria de camadas
- Reutilização

Interceptor – Conseqüências

- **Malefícios:**

- Complexidade de projeto
- Interceptadores falhos
- Uso potencial de processamento em cascata

Interceptor - Ver também

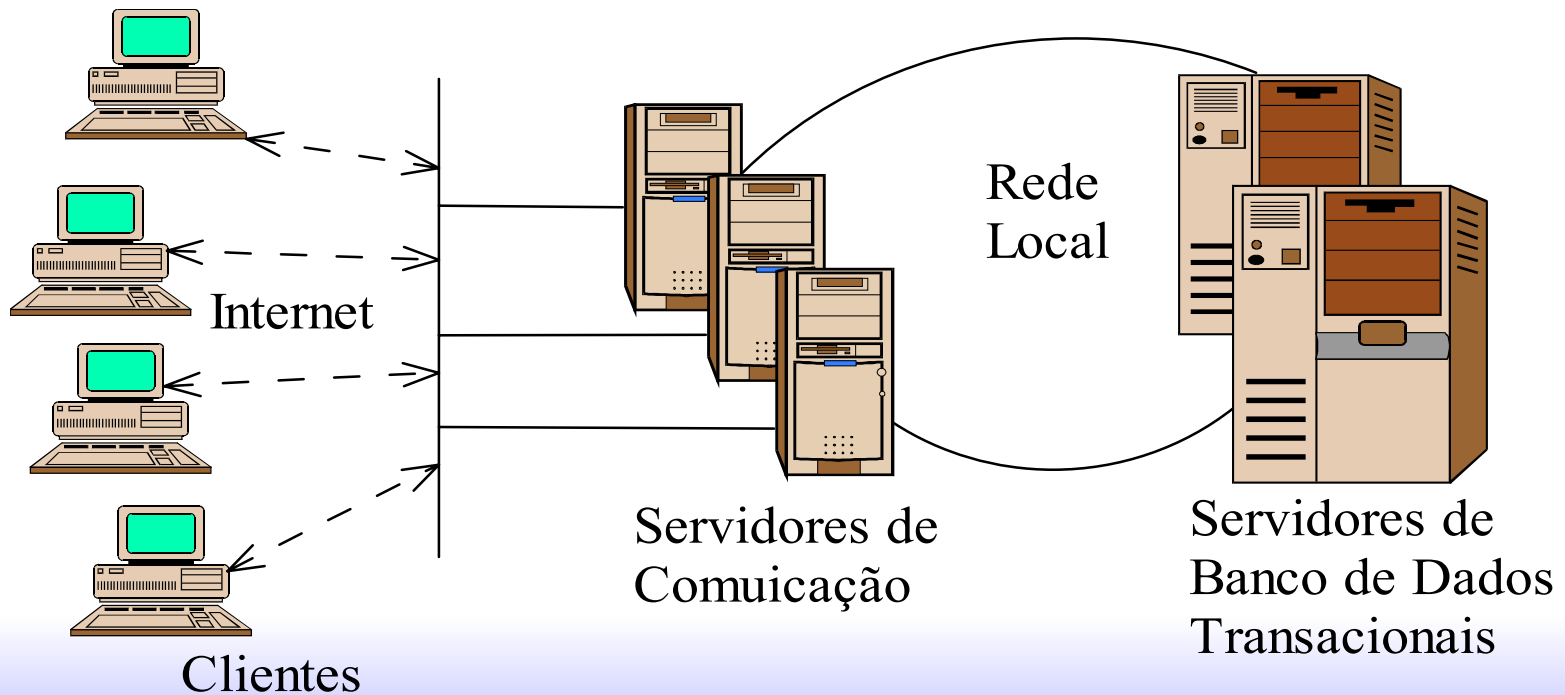
- **GoF**
 - Template
 - Chain-of-Responsibility
 - Proxy
 - Observer
- **POSA I**
 - Pipes and Filters
 - Publisher - Subscriber
 - Reflection
- **POSA II**
 - Reactor

Leader / Followers

O padrão arquitetural *Leader / Followers* provê um modelo de concorrência com múltiplos fluxos de execução (*threads*) compartilhando um conjunto de fontes de eventos, visando detectar, desmembrar, despachar e processar as requisições de serviço que ocorrem nessas fontes de eventos

Leader / Followers – Exemplo

- **Sistema de processamento de transações *on-line* (OLTP)**



Leader / Followers – Exemplo

- **OLTP poderia ser implementado:**
 - Baseado num modelo de processamento de eventos com um único fluxo de execução (*single-threaded*)
 - Usando um modelo de concorrência, com múltiplos fluxos de execução, que processa as requisições simultaneamente

Leader / Followers – Contexto

- Aplicações dirigidas a eventos onde múltiplas requisições de serviços enviadas para um conjunto de fontes de eventos precisam ser processadas eficientemente por múltiplos fluxos de execução que compartilham essas fontes.

Leader / Followers – Problema

- **Processamento de múltiplos fluxos de execução** – técnica comum, mas difícil de ser implementada em alto desempenho
- **Considerações**
 - Demultiplexação eficiente de fluxos e eventos
 - Minimização de sobrecarga relacionada à concorrência
 - Prevenção de condições de corrida

Leader / Followers – Solução

- **Projetar um mecanismo de fluxos de execução coordenados para detectar, demultiplexar, despachar e processar eventos**
 - Um fluxo é o líder e aguarda um evento nas fontes de eventos
 - Quando detecta evento, promove novo líder e faz processamento
 - Vários fluxos de execução podem ser processados enquanto líder aguarda evento

Leader / Followers – Estrutura

<i>Classe</i> Handle e Handle Set	<i>Colaboradores</i>
<i>Responsabilidades</i> -Handle identifica um fonte de eventos no sistema operacional -Handle pode enfileirar eventos -Handle Set é uma coleção de objetos Handle	

Leader / Followers – Estrutura

<i>Classe</i> Event Handler	<i>Colaboradores</i> -Handle
<i>Responsabilidades</i> -Define uma interface para processamento de eventos que ocorram em um objeto Handle	

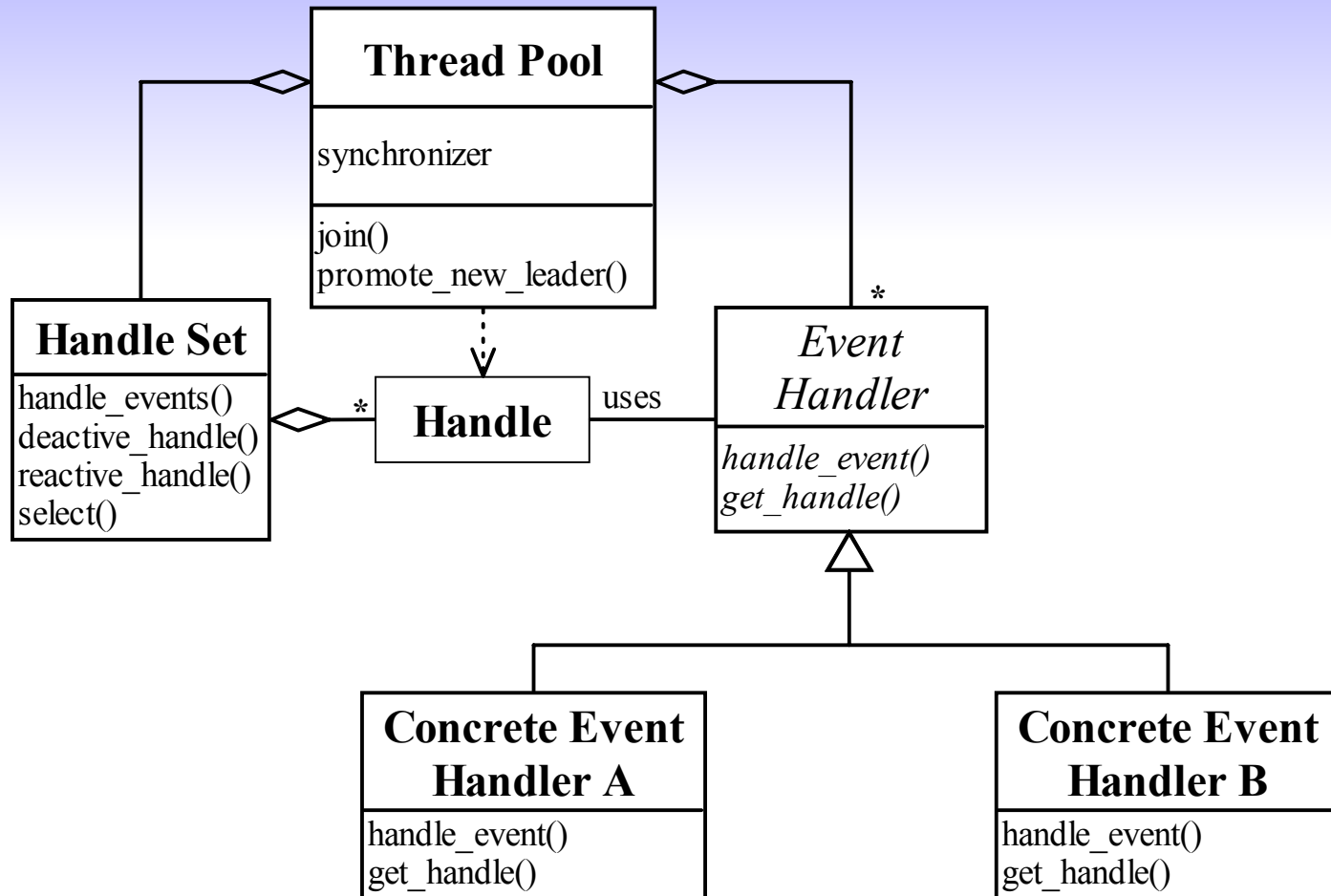
Leader / Followers – Estrutura

<i>Classe</i> Concrete Event Handler	<i>Colaboradores</i> -Handle
<i>Responsabilidades</i> -Define um serviço de aplicação -Processa eventos recebidos em um objeto Handle de maneira específica à aplicação -É executado em um fluxo de execução em processamento	

Leader / Followers – Estrutura

<i>Classe</i>	<i>Colaboradores</i>
<p>Thread Pool</p>	
<p>Responsabilidades</p> <ul style="list-style-type: none">-Contém os fluxos de execução que podem assumir os papéis: de líder, de processamento e de seguidor-Contém um sincronizador	<ul style="list-style-type: none">-Handle Set-Handle-Event Handlers

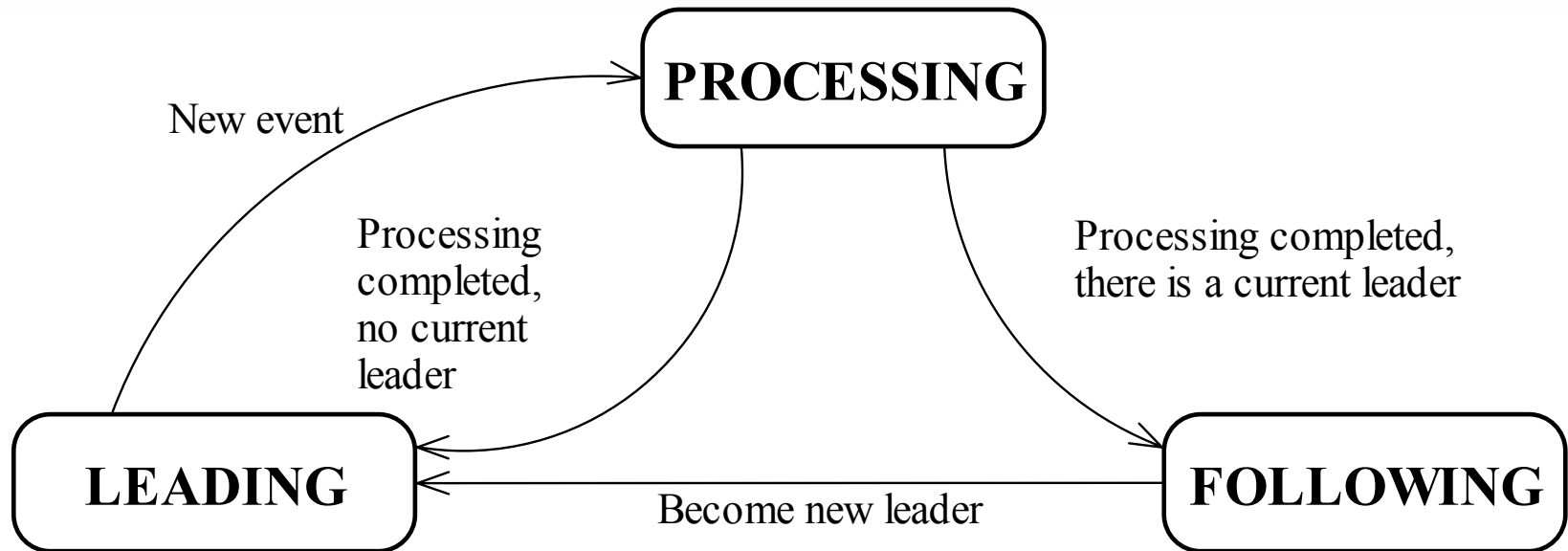
Leader / Followers – Estrutura



Leader / Followers – Dinâmica

- Fluxo líder aguarda evento
- Promoção do fluxo seguidor
- Antigo líder faz, concorrentemente, demultiplexação e despacho do evento recebido
- Acabado o processamento o fluxo de execução volta ao *pool* de fluxos

Leader / Followers – Dinâmica

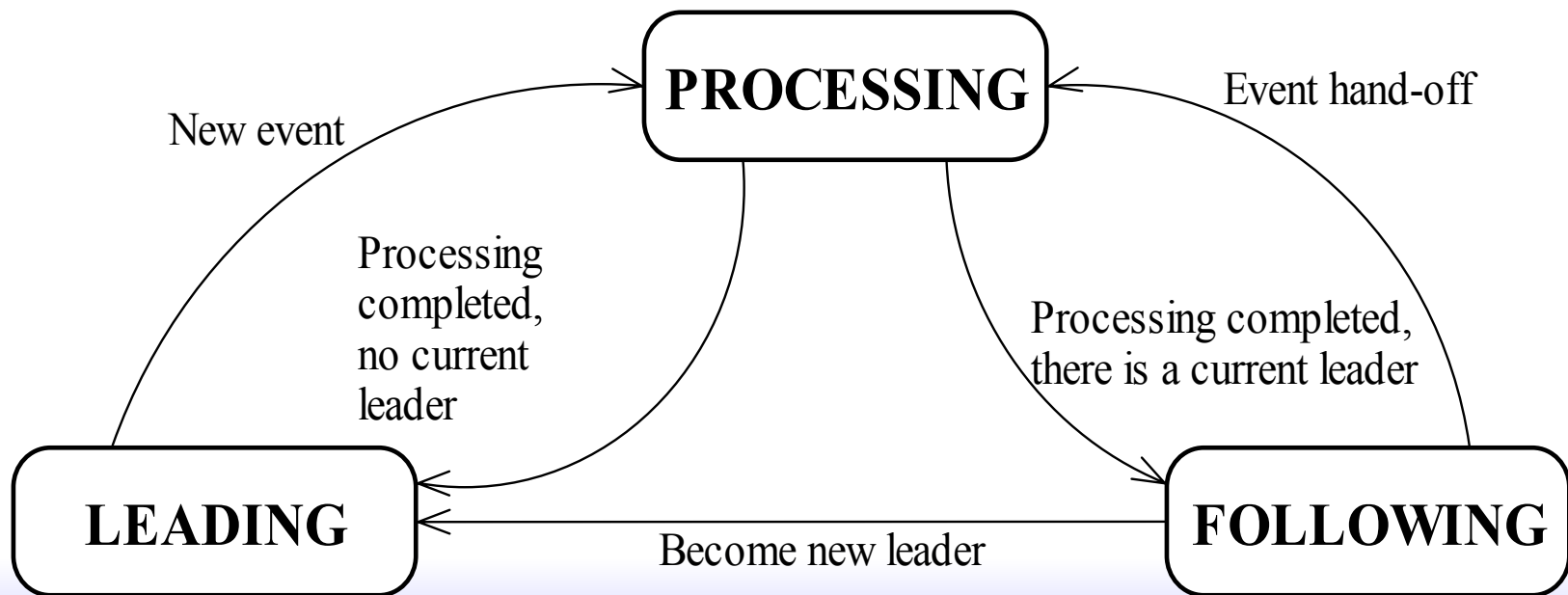


Leader / Followers – Implementação

1. Escolher os mecanismos para Handle e Handle Set.
2. Implementar um protocolo para ativar e desativar objetos Handle em um Handle Set
3. Implementar o *pool* de fluxos de execução
4. Implementar o protocolo de promoção dos seguidores
5. Implementar o tratamento de eventos

Leader / Followers – Variação

- **Bound Handle/Thread Associations**
 - Há associação entre os fluxos e os objetos Handle



Leader / Followers – Usos Conhecidos

- **Arcabouço ACE Thread Pool Reactor**
- **ORB's de CORBA**
 - Chorus COOL ORB e TAO
- **Servidores Web**
 - JAWS
- **Pontos de Taxi**

Leader / Followers – Conseqüências

- **Benefícios:**
 - Melhorias de desempenho
 - Programação simplificada de modelos de concorrência
- **Malefícios**
 - Complexidade de implementação das variações mais avançadas
 - Perda de flexibilidade
 - Gargalos de E/S na rede

Leader / Followers – Ver também

- **POSA II**
 - Reactor
 - Proactor
 - Half-Sync/Half-Async
 - Active Object

Comentários Finais

- **Necessidade de se estabelecer padrões no domínio de aplicações em ambientes distribuídos e com concorrência**
- **Interceptor – extensão de arcabouços**
- **Leader/Followers – gerenciamento de um conjunto de fluxos de execução compartilhando fontes de eventos**