Uma Facilidade de Gerenciamento de Configuração para Aplicações CORBA

Cláudio Márcio da Silveira Edmundo R. M. Madeira
IC - Instituto de Computação
UNICAMP - Universidade Estadual de Campinas
13083-970 Campinas - SP - Brasil
{cms,edmundo}@dcc.unicamp.br

Resumo

Este artigo apresenta o projeto e implementação da Facilidade de Gerenciamento de Configuração que tem sido desenvolvida na UNICAMP para compor a Arquitetura de Gerenciamento Integrado de Sistemas Distribuídos da plataforma Multiware. A Facilidade desenvolve, a partir da especificação XCMF, um modelo para o projeto de aplicações CORBA distribuídas cujas estrutura e interconexões entre os objetos distribuídos possam ser gerenciadas dinamicamente e interativamente a partir de gerentes externos. Os componentes da aplicação serão Instâncias Configuráveis capazes de receberem operações de gerenciamento de configuração.

Palavras chaves: Gerenciamento de Aplicações; Gerenciamento de Configuração; XCMF; Ciclo de Vida, Objetos CORBA.

Abstract

This paper presents the design and implementation of the Configuration Management Facility which has developed at University of Campinas to compose the Integrated Management Architecture for Distributed Systems of the Multiware platform. From the XCMF specification, the Facility develops a model for the design of distributed CORBA applications whose structure and interconnections between distributed objects can be interactively managed at run-time by external managers. Components that make up the application will be "XCMF Instances" that are able to receive configuration management operations.

Keywords: Application management; Configuration management; XCMF; Life-Cycle maintenance; CORBA objects;

1 Introdução

O crescimento e a complexidade dos sistemas distribuídos têm tornado o seu gerenciamento um tema de especial interesse para pesquisadores e projetistas. O Gerenciamento de Configuração no contexto de Sistemas Distribuídos envolve o gerenciamento da estrutura topológica da aplicação distribuída. Em CORBA (Common Object Request Broker Architecture) as interconexões entre os objetos distribuídos (objetos CORBA) que compõem a aplicação distribuída devem participar também deste gerenciamento.

Gerenciamento de Configuração, de acordo com o modelo OSI de gerenciamento de redes, é o *mecanismo geral* através do qual gerentes interagem com entidades gerenciadas diretamente ou indiretamente via adaptadores (software). Configurar tais objetos geralmente significa determinar e atribuir parâmetros e valores ao estado de um objeto que afetam seu comportamento. Este processo de gerenciamento foi desenvolvido no contexto de gerenciamento de redes para suportar hardwares heterogêneos e objetos de software relativamente simples.

Como observado por [Fos97], tais dispositivos e componentes são assumidos de serem parte do sistema por um longo período de tempo e o gerenciamento de configuração é aplicável após a entidade gerenciável ter sido instalada na rede, como por exemplo, a

conexão de uma nova impressora ou a instalação de um novo servidor de nomes. O Gerenciamento de Configuração, nesse sentido, considera estas atividades individualmente e não se preocupa com o problema da criação (ou inclusão) e destruição (ou remoção) de serviços de software.

O Gerenciamento de Configuração no contexto de aplicações distribuídas, no entanto, possui aspectos distintos. Neste caso, uma preocupação fundamental será com as operações de Ciclo de Vida dos componentes da aplicação. Torna-se necessária a definição do processo pelo qual objetos de software são criados, conectados ao sistema e gerenciados. Tais operações devem permitir que as aplicações sejam dinamicamente gerenciadas e incrementalmente evoluídas, devido, muitas vezes, à impossibilidade de tais aplicações serem interrompidas para sua necessária manutenção.

Aplicações distribuídas possuem uma estrutura que é resultado dos relacionamentos entre os componentes da aplicação. Esta estrutura é frequentemente identificada com a Configuração da aplicação. Ela deve estar ao alcance das aplicações de gerenciamento de configuração para uma possível reconfiguração, ou seja, mudança na estrutura da aplicação em tempo de execução. Esta configuração, sendo também persistente, nos fornece mecanismos para a recuperação de uma possível falha nesta estrutura.

Modelos e conceitos para arquiteturas de gerenciamento tem sido propostas e analisadas [Que97] [Slo95]. Visando a padronização dos mecanismos de gerenciamento, a especificação XCMF (*Common Management Facilities*) [Com97], baseada no *"X/Open System Management Reference Model"*, foi aprovada e adotada pelo OMG como o padrão para a Facilidade Comum de Gerenciamento de Sistemas Distribuídos. Os serviços da especificação XCMF formam um *framework* projetado para suportar aplicações que gerenciam um grande número de objetos representando recursos.

Neste artigo apresentamos o projeto e implementação da Facilidade de Gerenciamento de Configuração desenvolvida para compor a Arquitetura de Gerenciamento Integrado de Sistemas Distribuídos da plataforma Multiware [LMM+94] sendo desenvolvida na UNICAMP. A Facilidade desenvolve, a partir da especificação XCMF, um modelo para se projetar uma aplicação distribuída capaz de receber dinamicamente e interativamente operações de gerenciamento de configuração.

Este artigo apresenta na Seção 2 o contexto de desenvolvimento da Facilidade de Gerenciamento de Configuração. A especificação XCMF é comentada na Seção 3. Na Seção 4, o projeto da Facilidade de Gerenciamento de Configuração é analisado e, na Seção 5, é apresentada a modelagem. Na Seção 6, alguns aspectos do processo de implementação são comentados e, na Seção 7, alguns detalhes de como se trabalhar com a Facilidade são descritos. Trabalhos relacionados são apresentados na Seção 8 e, por fim, algumas observações e conclusões são colocadas na Seção 9.

2 Arquitetura de Gerenciamento de Sistemas Distribuídos da Plataforma Multiware

A arquitetura descrita nesta seção foi concebida para compor a plataforma Multiware, ilustrada na Figura 1. A plataforma Multiware [LMM+94], desenvolvida na UNICAMP, adota o padrão "de jure" do Modelo de Referência ODP (Open Distributed Processing) e incorpora os conceitos do padrão "*de facto*" da Arquitetura de Gerenciamento de Objetos do OMG, assimilando idéias de outros padrões existentes como o DCE (Distributed Computing Environment).

A plataforma está organizada em três camadas de suporte a aplicações distribuídas:

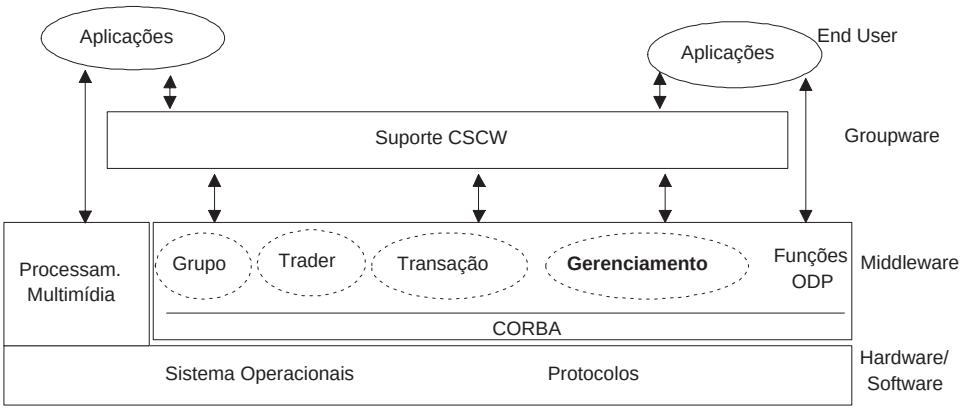

Figura 1: Plataforma Multiware.

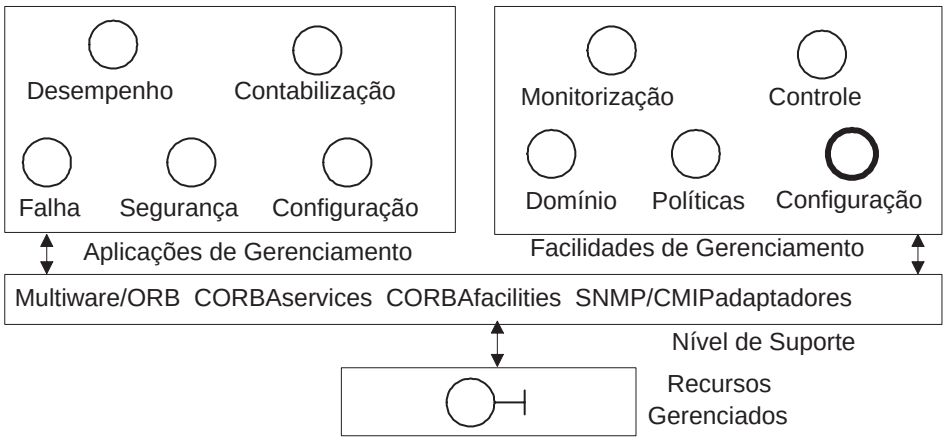

Figura 2: Arquitetura de Gerenciamento.

uma de Hardware/Software básico, a Middleware e a Groupware. A camada Middleware é composta por um ORB e de alguns serviços ODP, como o Trader, Gerenciamento de Sistemas Distribuídos, e suporte a Transação e a Grupo. Sobre esta camada, aplicações CSCW (Computer Supported Cooperative Work) são suportadas pela Groupware.

O modelo para a Arquitetura de Gerenciamento da plataforma Multiware foi proposto por [Que97] (Figura 2) e tem por objetivo oferecer, de forma integrada, um conjunto de serviços úteis para o gerenciamento de aplicações distribuídas. O modelo prevê aplicações de gerenciamento, cobrindo as diversas áreas funcionais do modelo OSI, além de um conjunto de Facilidades de Gerenciamento Integradas, oferecendo serviços para o desenvolvimento destas aplicações. Atualmente os esforços para o desenvolvimento da Arquitetura se concentram no desenvolvimento das Facilidades de Gerenciamento.

3 XCMF - Common Management Facilities

A especificação XCMF [Com97] é baseada no Modelo de Referência para Gerenciamento de Sistemas da X/Open e foi aprovada como Facilidade Comum para Gerenciamento de Sistemas Distribuídos pelo OMG. XCMF foi criada como infra-estrutura para desenvolvimento de aplicações distribuídas gerenciáveis.

Do conjunto de Facilidades de Gerenciamento identificadas pelo OMG em [Gro97a], especificações para os seguintes serviços são apresentadas:

Gerenciamento de Conjuntos (Managed Sets) - Suporta o conceito de conjunto de objetos gerenciados com a finalidade de organizar os objetos em grupos. Este serviço se identifica com a Facilidade de Gerenciamento de Coleções identificada pelo OMG [Gro97a] e com a Facilidade de Domínios encontrada na Figura 2. O agrupamento de objetos em conjuntos (ou domínios) pode refletir a conectividade física da rede ou um organograma administrativo, permitindo a divisão de responsabilidade e autorização entre diferentes gerentes.

Gerenciamento de Instâncias - Fornece a infra-estrutura requerida para o gerenciamento de múltiplas instâncias de um tipo de objeto. O Serviço de Gerenciamento de Instâncias fornece as capacidades básicas para a criação e gerenciamento de todos os tipos de objetos gerenciados (Instâncias). Esta facilidade apresenta uma descrição sintática e semântica para um serviço de criação baseado na especificação do Serviço de Ciclo de Vida [Gro97b]. Esta Facilidade compõe a Facilidade de Gerenciamento de Configuração apresentada na Figura 2, que é o escopo deste trabalho. Três papeis são definidos neste serviço:

- Instância gerenciada;
- Gerenciador de Instâncias - Representa um *factory* e um conjunto gerenciado para um tipo específico de Instâncias;
- A *Library* - Representa um *factory* e um conjunto gerenciado para Gerenciadores de Instâncias.

Gerenciamento de Políticas - Uma política é uma regra que o administrador aplica ao sistema. Elas fornecem ao administrador um meio de adaptar as aplicações às suas necessidades específicas. Este serviço define mecanismos para o estabelecimento de políticas de validação e de inicialização.

Os serviços da especificação XCMF formam um *framework* projetado para suportar aplicações que gerenciam um grande número de objetos representando recursos. Usando os serviços XCMF, os projetistas das aplicações podem definir conjuntos de objetos (modelando *containers*) e associar políticas de inicialização ou validação para estes conjuntos. Estes objetos serão gerenciados pelo tipo e gerados dinamicamente. Em adição, as políticas associadas a um determinado tipo de instâncias poderão ser modificadas dinamicamente pelo gerente.

4 A Facilidade de Gerenciamento de Configuração

Passamos agora a apresentar o projeto da Facilidade, juntamente com suas idéias e conceitos. Pelo que já foi discutido, o projeto visa alcançar, fundamentalmente, os seguintes objetivos:

- acompanhar e controlar a estrutura de uma aplicação distribuída, através das interconexões entre os objetos (objetos CORBA) que compõem a aplicação;
- suportar as operações de ciclo de vida dos componentes da aplicação distribuída, ou seja, cópia, movimentação e remoção dos objetos distribuídos;
- oferecer suporte ao gerenciamento interativo no sentido de capacitar a reconfiguração em tempo de execução (configuração dinâmica) fazendo uso extensivo de notificações de eventos para manter gerentes externos atualizados com a configuração da aplicação;
- oferecer o serviço de gerenciamento de forma independente de qualquer aplicação de gerenciamento.

Figura 3: Serviços de Configuração.

O caminho adotado para o desenvolvimento visa incorporar ao *framework*, desenvolvido na especificação XCMF para modelar objetos gerenciados, a capacidade de estabelecer, controlar e acompanhar a configuração de um conjunto destes objetos. Desta forma, adicionamos um novo objetivo ao projeto:

- tornar os detalhes envolvidos nos mecanismos de gerenciamento da especificação XCMF (mais precisamente, o gerenciamento de Instâncias) transparentes tanto para os gerentes externos quanto para o projetista dos objetos gerenciados. Serviços serão oferecidos tanto para as aplicações de gerenciamento quanto para a própria aplicação gerenciada (Figura 3).

O projeto considera uma aplicação distribuída em CORBA como um conjunto de objetos CORBA, projetados conjuntamente para a solução de um problema e agrupados para o gerenciamento. Os *limites* entre os objetos da aplicação e os demais objetos do ambiente são determinados pelo projetista da aplicação. A *estrutura* nesta aplicação será determinada pelos relacionamentos entre os objetos, inclusive, entre tais objetos pertencentes à aplicação e demais objetos do ambiente de computação distribuída.

Parte dos problemas envolvidos com as operações de ciclo de vida de um objeto é tratada na especificação do Serviço de Ciclo de Vida e na própria especificação XCMF. Portanto, temos as soluções presentes nestas especificações como ponto de partida para o trabalho.

Nosso projeto é baseado em três serviços:

- OMG *Life Cycle Service*;
- XCMF *Managed Sets* - (Especificação para o nosso serviço de Domínios);
- XCMF *Instance Management* - suporte explícito para a criação e gerenciamento de Instâncias.

A utilização do Serviço de Ciclo de Vida é realizada através do Serviço XCMF de Gerenciamento de Instâncias. A partir deste serviço o projeto desenvolve o conceito de uma "Instância Configurável", ou seja, uma Instância, como definida no serviço XCMF de Gerenciamento de Instâncias, que possua a capacidade de sofrer operações de configuração. Este conceito se realiza na interface **ConfigInstance** descrita na próxima seção.

Como já foi dito, XCMF modela seus serviços como a *Library* e os *Gerenciadores de Instâncias* em interfaces derivadas (especializadas) das interfaces de conjuntos. Para organizar e agrupar as "Instâncias Configuráveis" no ambiente de gerenciamento, desenvolvemos um tipo especial de conjunto para representar a aplicação distribuída. Deste modo, teríamos também todas as aplicações distribuídas do ambiente tendo seu representante na árvore de domínios. Este conjunto especial, que se realiza na interface **ConfigApplication**, teria como membros os objetos instanciados da aplicação, os quais, teriam a capacidade de receberem operações de configuração. **ConfigApplication** é a interface central no processo de provisão dos serviços de configuração.

Para operações de configuração, é necessário, basicamente, que:

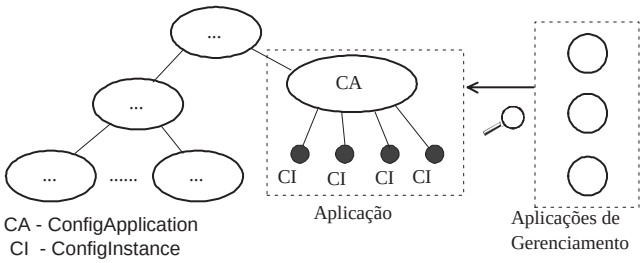

Figura 4: Ambiente de Gerenciamento.

- Os objetos tenham a capacidade de sofrer operações de ciclo de vida;
- Os objetos tenham a capacidade de fornecer informações a respeito de seus relacionamentos, permitindo ainda que tais relacionamentos sejam administrados;
- Os objetos tenham a capacidade de sofrer operações específicas de configuração tais como a substituição de implementação e o *rebind* de referências;

ConfigApplication tem ainda a capacidade de conhecer todos os tipos Instâncias Configuráveis pertencentes à aplicação, permitindo que estes objetos sejam criados interativamente, movidos e copiados.

Cabe aqui mencionarmos que nem todo componente de uma aplicação distribuída é um objeto servidor CORBA. Temos também os componentes que funcionam apenas como processos clientes destes objetos. Como estes processos participam da aplicação e armazenam referências de objetos, também devem participar do mecanismo de gerenciamento de configuração.

A Figura 4 fornece uma pequena ilustração destes conceitos. Ela ilustra a árvore de domínios (conjuntos) de um ambiente de gerenciamento acomodando uma aplicação distribuída desenvolvida e instalada segundo os moldes da Facilidade de Configuração. Ilustra também aplicações de gerenciamento que, localizando o objeto representante da aplicação, passam a gerenciá-la.

5 Modelagem da Facilidade

Passamos agora a analisar os elementos presentes no projeto de nossa Facilidade de Configuração. A Figura 5 apresenta a modelagem da Facilidade. Nesta Figura, as caixas com as bordas mais largas representam as interfaces definidas para a nossa Facilidade de Configuração enquanto que as demais representam interfaces definidas na especificação XCMF e no serviço de Ciclo de Vida (OMG).

5.1 ApplicationContainer

Esta interface representa um *factory* para **ConfigApplication**. Nosso interesse é colocar este serviço de criação, no ambiente de gerenciamento, ao lado da Library e não como um serviço projetado segundo o *framework* XCMF, por exemplo, como uma interface Instance especializada. Podemos citar duas razões para isto:

- idealizamos tornar a tarefa de gerenciamento de aplicações transparente com relação aos serviços XCMF;
- esta representação deste serviço torna direto a localização do objeto que representa a aplicação (**ConfigApplication**) por parte de seus componentes. A pesquisa de localização é feita neste serviço.

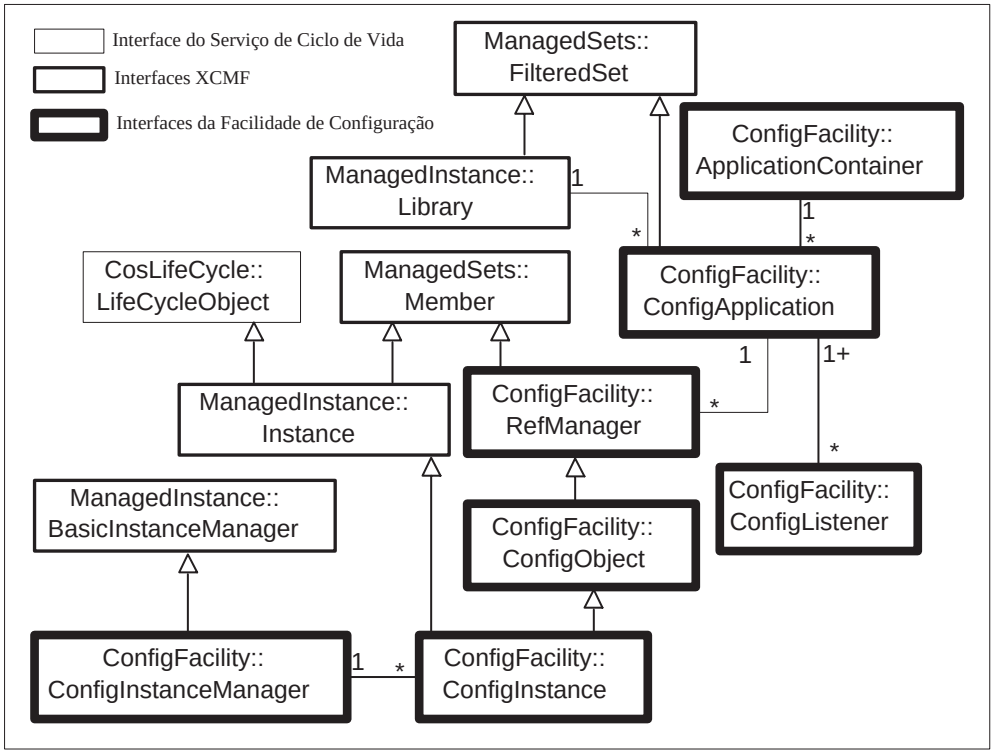

Figura 5: Modelagem da Facilidade de Configuração.

5.2 ConfigApplication

Esta é a interface central da Facilidade. Ela representa, no ambiente de gerenciamento, a própria aplicação distribuída fornecendo uma interface para o seu gerenciamento. **ConfigApplication** oferecerá serviços tanto para os componentes da aplicação distribuída quanto para as aplicações de gerenciamento. Esta interface deverá oferecer, principalmente, serviços para:

- registrar todos os tipos dos componentes da aplicação (**ConfigInstance**);
- registrar "listeners" (**ConfigListener**) para eventos de atualização da configuração da aplicação;
- enviar eventos notificando mudanças na estrutura da aplicação ou no estado dos componentes da aplicação (**ConfigInstance** e **ConfigObject**);
- localizar os demais componentes da aplicação baseando-se em critérios de pesquisa (nome, localização, identificação de interface, identificação de implementação);
- criar instâncias (**ConfigInstance**) para a aplicação baseando-se em critérios de criação, inclusive interativamente;
- copiar, mover e destruir instâncias (**ConfigInstance**) da aplicação;
- transferir referências de um objeto para outro (*rebind*);
- substituir implementações de instâncias (**ConfigInstance**);

ConfigApplication conhecerá toda a estrutura da aplicação, ou seja, todos os objetos componentes da aplicação e todas as referências mantidas por estes componentes. **ConfigApplication** manterá ainda uma "Lista de Dependências" para cada objeto da aplicação com informações dos demais **RefManager** que guardam uma referência registrada para aquele objeto (ver seção seguinte).

Todas estas operações deverão ser efetuadas dinamicamente, em tempo de execução. Após o processo de instalação, isto é, após a criação do objeto **ConfigApplication** e

registro dos tipos dos componentes, a aplicação está pronta para trabalhar. **ConfigApplication** será responsável pelo preparo do ambiente de gerenciamento, principalmente dos recursos XCMF necessários.

Isto significa que **ConfigApplication** usará a "XCMF *Library*" para criar um Gerenciador de Instâncias (**ConfigInstanceManager**) para cada tipo de componente (**ConfigInstance**) registrado.

No Anexo A, é apresentada a descrição IDL da interface **ConfigApplication**.

5.3 RefManager

Esta interface é responsável pelo gerenciamento das referências em um componente da aplicação. Toda referência que necessitar ser gerenciada ou que compõe a estrutura da aplicação deverá ser registrada. **RefManager** sempre estará associado a um único objeto **ConfigApplication**.

A interface **RefManager** exercerá o gerenciamento das referências em especializações das interfaces **ConfigObject** e **ConfigInstance** ou em "processos clientes" que instanciarão um objeto implementando esta interface para exercer o gerenciamento de suas referências. Neste aspecto de sua atividade, **RefManager**, oferecerá serviços para:

- registrar (e remover) referências do gerenciamento;
- bloquear momentaneamente qualquer operação de gerenciamento em uma referência;
- obter, com base em sua identificação, a referência desejada atualizada (válida);
- verificar se uma referência registrada está disponível para uso.

RefManager também estará oferecendo serviços ao próprio ambiente da aplicação distribuída e para as aplicações externas de gerenciamento. Nesta atividade, esta interface oferece serviços para:

- atualizar referências registradas como resultado de alguma operação de ciclo de vida no objeto referenciado ou de uma operação de *rebind*;
- obter uma listagem de todas as referências registradas juntamente com informações sobre as mesmas;

É importante notar que não existirá um componente "puramente cliente" em nossa aplicação distribuída configurável, já que os processos componentes da aplicação deverão possuir no mínimo um objeto CORBA (**RefManager**).

Ao se registrar uma referência, uma classificação para a mesma é adicionada, permitindo um controle adicional sobre sua manipulação no ambiente de gerenciamento. Esta classificação também será oportuna para auxiliar uma possível construção gráfica da estrutura da aplicação enriquecendo as informações topológicas. Esta classificação para as referências se apresenta como:

- *ManageableRef* → Determina referências sujeitas a atualizações e operações de gerenciamento;
- *FixedRef* → Determina que a referência está sujeita apenas às atualizações, provenientes, principalmente, de operações de ciclo de vida dos objetos referenciados. Não permite operações de gerenciamento como o *rebind*;
- *OpenRef* → São referências a objetos externos à aplicação distribuída. Estas referências não sofrem qualquer tipo de atualização ou operação de gerenciamento.
- *CompositeRef* → São referências para objetos instanciados (**ConfigObject**) diretamente pelo componente associado ao **RefManager**. Estas referências não sofrem operações de gerenciamento.

As referências ainda possuem um estado associado. Este estado pode ser:

- *Normal* → A referência está desbloqueada e pronta para uso;
- *Blocked* → Aplica-se somente às referências *ManageableRef*. Indica que a referência atual não pode ser atualizada por um processo de *rebind*;
- *Blocked_waiting* → Aplica-se somente às referências *ManageableRef*. Se uma referência bloqueada sofrer uma operação de *rebind* ela passará a este estado indicando que assim que for desbloqueada, a referência será atualizada;
- *Suspended* → Indica que o objeto referenciado está sofrendo uma operação de ciclo de vida e, portanto, está indisponível;
- *Removed* → Indica que o objeto referenciado foi removido do ambiente (destruído).

No Anexo A, a especificação IDL desta interface é apresentada.

5.4 ConfigInstance

Esta interface representa a realização do conceito de "Instância Configurável". Isto significa que um objeto **ConfigInstance** pode ser criado interativamente além de ser copiado, movido e removido do ambiente da aplicação distribuída. Esta interface estende a funcionalidade de **ConfigObject** que também representa um objeto componente da aplicação distribuída, mas que, não sendo uma Instância XCMF, não poderá ser criado interativamente ou sofrer operações de ciclo de vida.

ConfigInstance deverá ter dois modos de criação (*Normal* e *Requesting*). No modo *Requesting*, um objeto **ConfigInstance** irá requerer um conjunto de referências de **ConfigApplication**. Este mecanismo favorece a criação interativa dos componentes da aplicação.

Um objeto **ConfigInstance** sempre terá um **ConfigInstanceManager** associado. **ConfigInstanceManager** cobre a necessidade de definirmos um tipo especial de gerenciador de instâncias para **ConfigInstance**.

5.5 ConfigObject

Esta interface deverá ser suportada por todo objeto componente da aplicação distribuída. O objeto será então capaz de sofrer as operações básicas de configuração, operações estas relacionadas principalmente com o gerenciamento de referências.

Um objeto **ConfigObject**, não sendo uma Instância XCMF, não poderá ser criado interativamente por **ConfigApplication** ou sofrer operações de ciclo de vida. Estas operações serão definidas na interface **ConfigInstance**.

Esta capacidade de **ConfigObject** de gerenciar suas referências deve-se à sua especialização da interface **RefManager**.

Um objeto **ConfigObject**, se existir, terá sua existência e seu gerenciamento de ciclo de vida a cargo de um outro componente da aplicação, provavelmente um objeto **ConfigInstance**. Esta interface é requerida para situações onde um objeto componente da aplicação atua como *factory* de outros objetos que também deverão compor a aplicação e participar de seu gerenciamento. Estes objetos terão a capacidade de gerenciar suas referências mas serão dependentes de seu *factory* no tocante ao seu processo de ciclo vida.

Quando um destes objetos *factory* instancia um **ConfigObject** ele deverá registrar uma referência para o mesmo do tipo CompositeRef. Estas referências não sofrem operações de gerenciamento, mas servem para indicar a situação.

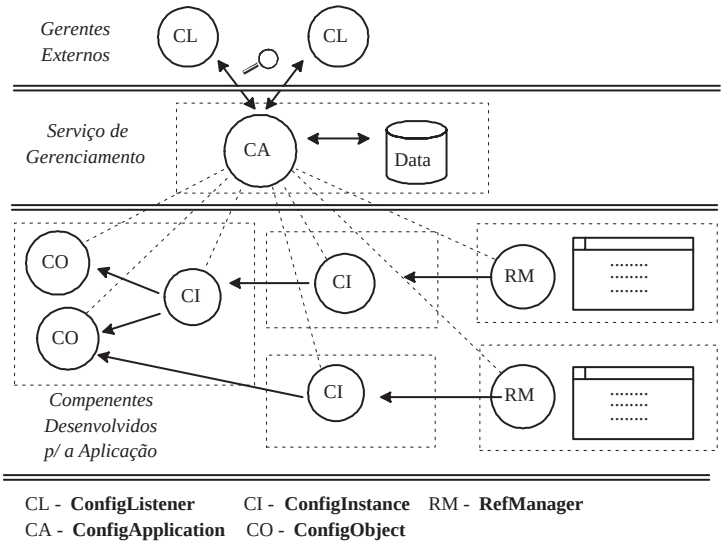

Figura 6: Ilustração dos conceitos.

5.6 ConfigListener

Esta interface deverá ser implementada por qualquer objeto externo à aplicação (gerentes externos) que desejar receber notificações sobre a ocorrência de mudanças na configuração da aplicação distribuída.

A Figura 6 ilustra os conceitos apresentados nesta seção. Ela apresenta uma aplicação composta por três objetos **ConfigInstance**, sendo que um deles atua como *factory* de dois objetos **ConfigObject** componentes da aplicação. Ela também apresenta dois processos clientes que instanciam, cada um deles, um objeto **RefManager** para o gerenciamento de suas referências. O controle da estrutura formada por todos estes componentes e a representação da aplicação no ambiente de gerenciamento fica por conta de um objeto **ConfigApplication**. Este ainda transmite notificações para dois gerentes externos que implementam a interface **ConfigListener**.

6 A Implementação

Um protótipo para a Facilidade de Gerenciamento de Configuração foi desenvolvido em Java sobre a plataforma OrbixWeb [ION97] da IONA. Também faz parte do trabalho de desenvolvimento a implementação dos serviços XCMF dos quais depende a Facilidade de Configuração ("*Managed Sets*" e "*Instance Management*").

A implementação está produzindo serviços para serem introduzidos no ambiente de gerenciamento (interfaces **ConfigApplication** e **ApplicationContainer**), além de um conjunto de bibliotecas de programação para as interfaces **ConfigObject**, **ConfigInstance** e **ConfigListener**, que deverão ser especializadas pelo projetista da aplicação distribuída, e para a interface **RefManager**. Considerando que o trabalho deste projetista será basicamente o de especializar e implementar estas interfaces, é necessário que um conjunto de "*ganchos" (hooks)* seja bem definido no desenvolvimento desta biblioteca.

Um ambiente de gerenciamento gráfico, composto por dois *browsers*, está sendo desenvolvido. O primeiro permite ao gerente navegar no ambiente de gerenciamento constituído por uma árvore de domínios. O segundo *browser*, que é ativado pelo primeiro, trabalha sobre um objeto **ConfigApplication** permitindo ao usuário visualizar a configuração da

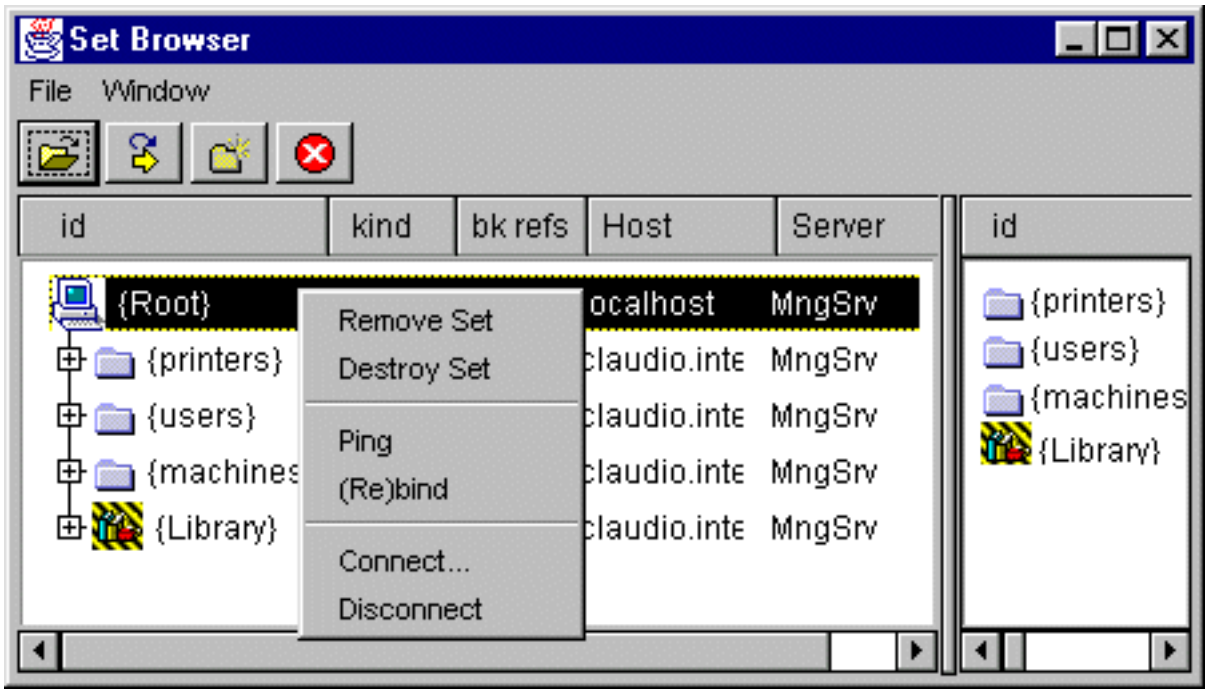

Figura 7: Set (Domain) Browser.

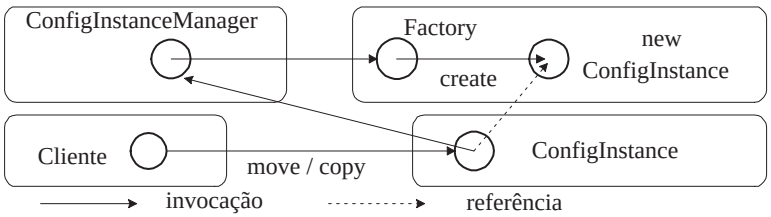

Figura 8: Operações de Ciclo de Vida.

aplicação distribuída e executar, interativamente, operações de gerenciamento de configuração. A Figura 7 apresenta a interface do primeiro *browser*.

O processo desenvolvido no trabalho para operações de ciclo de vida de um objeto **ConfigInstance** é ilustrado na Figura 8. Nestas operações, o primeiro argumento (o **FactoryFinder**), como definido na interface **CosLifeCycle::LifeCycleObject**, nunca será usado porque, de acordo com a especificação do serviço XCMF de Gerenciamento de Instâncias, toda Instância, conhecendo o seu Gerenciador, já dispõe do *factory* correto e necessário ao processo.

Um problema importante, perseguido durante a implementação, foi a manutenção da consistência das informações de configuração da aplicação. Podemos ilustrar esta preocupação com uma visão geral dos passos envolvidos na movimentação de objetos **ConfigInstance**:

void ConfigInstsnce::move(FactoryFinder finder, Criteria the_criteria)

{

(i) "Verificar se o objeto permite ser movido"

(ii)"Através de seu Gerenciador de Instâncias, criar objeto temporário. Este objeto é mantido a parte, a espera de sua confirmação no lugar deste sendo movido"

(iii)"Notificar **ConfigApplication** do início da operação. **ConfigApplication** automaticamente irá suspender todas as referências para este componente."

(iv) "Chamar a função gancho apropriada para a transferência do estado para o objeto temporário"

(v) "Transferir os relacionamentos para o objeto temporário"

(vi) "Fazer o objeto temporário tomar o lugar do objeto sendo movido (No Gerenciador)"

(vii) "Notificar a **ConfigApplication** do fim da operação. **ConfigApplication** automaticamente atualizará todas as referências para este objeto."

(viii) "remover este objeto"

}

Todo este processo de movimentação deve ser implementado como uma ação atômica. Se algum destes passos falhar, todo o processo deverá ser cancelado sem corromper a consistência da configuração da aplicação.

7 Trabalhando com a Facilidade de Configuração

O processo de desenvolvimento de uma aplicação distribuída, segundo os moldes da Facilidade, apresenta os seguintes passos:

- projeto dos componentes da aplicação. Estes serão especializações das interfaces **ConfigInstance** ou **ConfigObject**;
- para cada especialização de **ConfigInstance** desenvolvida, desenvolver seu *factory* e um servidor para instanciá-lo. Um Gerenciador de Instâncias XCMF procurará pelo *factory* daquele tipo de instância que gerencia sempre que precisar criar uma destas instâncias.

Para o processo de instalação, será necessário:

- criar um objeto **ConfigApplication** para a aplicação;
- registrar as especializações de **ConfigInstance** no objeto **ConfigApplication**;
- registrar no Repositório de Implementação todos os servidores desenvolvidos para os *factories*.

O objeto **ConfigApplication** prepara o ambiente de gerenciamento criando um objeto **ConfigInstanceManager** para cada tipo de **ConfigInstance** registrado. Este Gerenciador de Instâncias será responsável, de acordo com o *framework* XCMF, pela criação das instâncias do tipo que gerencia. Este trabalho, como já foi dito, é feito por intermédio de um *factory* específico desenvolvido e instalado pelo projetista da aplicação.

Agora, através da interface do objeto **ConfigApplication**, objetos **ConfigInstance** poderão ser instanciados dinamicamente. Objetos **ConfigInstance** poderão ser instanciados em modo *Normal* ou em modo *Requesting*. No modo *Normal* os objetos são responsáveis pela obtenção de suas referências, estabelecendo seus relacionamentos. No modo *Requesting* o objeto **ConfigInstance** poderá requerer do objeto **ConfigApplication** um conjunto de referências preestabelecidas. Deste modo, um gerente externo será capaz de participar, através da interface de **ConfigApplication**, do estabelecimento dos relacionamentos destes objetos.

Como já foi dito, processos clientes modelados como componentes da aplicação terão seus relacionamentos gerenciados por um objeto **RefManager**. A instanciação de tais processos, ou mesmo, a instanciação de objetos **ConfigObject** deverá ser notificada ao objeto **ConfigApplication**.

A estrutura da aplicação poderá ser construída graficamente por um gerente externo obtendo do objeto **ConfigApplication** todas as informações necessárias para tal. Poderá, ainda, manter estas informações atualizadas através de notificações de atualização recebidas do objeto **ConfigApplication**.

Embora as operações de ciclo de vida possam ser invocadas diretamente sobre objetos **ConfigInstance**, é recomendado que estas sejam aplicadas através da interface do objeto **ConfigApplication** que elimina muitos detalhes envolvidos na sintaxe de tais operações.

8 Trabalhos Relacionados

[BC95] apresenta o ambiente DisCo com sua abordagem para gerenciamento de configuração utilizando a linguagem de configuração CL, um Gerenciador de Configuração que recebe comandos de configuração já validados pelo ambiente de compilação da linguagem CL, um Servidor de Nomes que mantém informações sobre os módulos da aplicação e Gerenciadores Auxiliares cujo papel é executar comandos de configuração nas máquinas onde residem. O ambiente de comunicação adotado é baseado nos protocolos TCP/IP. Fazendo um paralelo, podemos relacionar o objeto **ConfigApplication** com o Gerenciador de Configuração e com o Servidor de Nomes. Os Gerenciadores Auxiliares podem ser relacionados aos demais componentes da aplicação que, por serem especializações de interfaces definidas, possuem funções auxiliares de gerenciamento. Nosso projeto não pretende prover a especificação de uma linguagem de configuração, visando, principalmente, estender o *framework* XCMF para cobrir as necessidades relacionadas com o gerenciamento de configuração. O oferecimento de um serviço baseado em uma linguagem de configuração poderá ser desenvolvido, no nível das aplicações de gerenciamento, absorvendo os serviços oferecidos pela Facilidade.

[Fos97] apresenta um ambiente completo para gerenciamento de configuração interativo, integrado com a linguagem de configuração Darwin. O sistema é associado com um ambiente de gerenciamento gráfico. Toda a configuração da aplicação é mapeada para domínios de configuração, permitindo manter toda a configuração da aplicação persistente e independente da configuração original. Implementações para este ambiente de configuração foram desenvolvidas para plataformas distintas como ANSAware e CORBA. Em nosso trabalho, visamos determinar mecanismos para o suporte ao gerenciamento interativo, principalmente nas operações de criação de componentes e *rebind* de referências. Também foram desenvolvidas ferramentas gráficas para possibilitar o gerenciamento interativo.

Nossa Facilidade foi projetada para ser encaixada no contexto da Arquitetura de Gerenciamento apresentada na Seção 2, que se encontra em fase de desenvolvimento. Como a Facilidade de Configuração é fundamental na determinação das estruturas das aplicações Gerenciáveis e da própria estrutura de uma arquitetura integrada de gerenciamento, espera-se que, com a sua definição, o desenvolvimento da arquitetura possa ser completado em breve.

9 Conclusões

A especificação XCMF apresenta um *framework* para desenvolvimento e modelagem de recursos do ambiente de computação capazes de serem gerenciados. Com relação às operações de ciclo de vida destes objetos modelados (Instâncias), a especificação XCMF

trata quase que exclusivamente dos problemas envolvidos com o processo de criação. Nesta especificação, as instâncias destes objetos são gerenciadas individualmente.

Este artigo apresenta o projeto de uma Facilidade para desenvolvimento de aplicações de gerenciamento que estende o *framework* XCMF desenvolvendo um mecanismo que possibilita o gerenciamento conjunto de um grupo destas instâncias. Desta forma foi possível completar o suporte a operações de ciclo de vida que afetam o ambiente de gerenciamento, como a movimentação, e providenciar o suporte para operações necessárias ao controle da configuração deste grupo de objetos que passam agora a compor uma aplicação distribuída.

O padrão CORBA fornece um modelo de objetos rico para a especificação de interfaces de objetos, mas não suporta adequadamente a configuração estrutural dos objetos. CORBA não facilita a estruturação do sistema. Neste contexto, o processo desenvolvido para gerenciar a configuração de um conjunto de objetos inter-relacionados é adequado e eficiente. Entretanto, como resultado do modelo de criação de objetos e da característica dinâmica da ativação de objetos em CORBA, atrasos podem ocorrer nas operações de ciclo de vida dos componentes da aplicação.

Agradecimentos: Os autores agradecem à FAPESP, CAPES e ao CNPq pelo suporte providenciado ao trabalho.

Referências

[BC95] S. Bandeira and P. Cunha. Suporte à Execução do Ambiente DisCo. *XIII Simpósio Brasileiro de Redes de Computadores*, 1995. pp. 619-636.

[Com97] X/Open Company. System Management: Common Management Facilities. 1997. (ISBN: 1-85912-174-8, C423).

[Fos97] H. Fossa. Interative Configuration Management for Distributed Systems. 1997. PhD Thesis. Dept. of Computing, Imperial College, London. ftp://dse.doc.ic.ac.uk/dse-papers/management/FOSSA-THESIS.PS.GZ.

[Gro97a] Object Management Group. CORBAfacilities: Common Facilities Architecture, rev. 4.0. Novembro 1997.

[Gro97b] Object Management Group. CORBAservices: Common Object Services Specification. Março 1997.

[ION97] IONA Technologies. *Orbix Web Reference Guide*, November 1997.

[LMM+94] W.P.C. Loyolla, E.R.M. Madeira, M.J. Mendes, E. Cardozo, and M.F. Magalhães. Multiware Platform: An Open Distributed Environment for Multmedia Cooperative Applications. *IEEE Computer Software and Applications Conference. COMPSAC'94*, Novembro 1994. Taipei, Taiwan.

[Que97] J.A.G. Queiroz. Gerência de Sistemas Distribuídos Heterogêneos: Facilidade de Monitorização em um Ambiente CORBA. 1997. Tese de Mestrado. Instituto de Computação - UNICAMP.

[Slo95] M. Sloman. Management Issues for Distributed Services. *EEE Second International Workshop on Services in Distributed and Networked Environments (SDNE 95)*, 1995. ftp://dse.doc.ic.ac.uk/dse-papers/management/sdne95.ps.Z.

[SM99] C. M. Silveira and E. R. M. Madeira. A Configuration Facility for CORBA Applications. *Second IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems - DAIS'99*, Junho 1999. Helsinki, Finland (aceito para publicação).

A Principais Descrições IDL

```
interface ConfigApplication : ManagedSets::FilteredSet {
    //Administracao de Labels
    boolean is_valid_label(in string label);
    string get_valid_label(in string base_label);
    string reserve_label(in string base_label);
    //Registro de informacoes
    void register_ConfigInstanceType(in string id_interface, in string id_implementation)
        raises (CFGInvalidType);
    void add_client(in Component comp) raises(CFGInvalidLabel, CFGInvalidType);
    void add_config_object(in Compenet comp) raises(CFGInvalidLabel, CFGInvalidType);
    void add_listener(in ConfigListener listener);
    void remove_client(in Component comp) raises (CFGNotFound);
    void remove_config_object(in Component comp) raises(CFGNotFound);
    void remove_listener(in ConfigListener listener) raises(CFGNotFound);
    //Recuperacao de informacoes
    ManagedInstances::Library get_library();
    InterfaceImplList get_registred_types() raises(CFGNotFound);
    Component get_uptodate_comp(in string label) raises(CFGNotFound, CFGCanNotOperate);
    Component localize_component(in string id_label) raises (CFGNotFound);
    ComponentList localize_components (in string id_interface, in string id_implementation,
        in string host_name) raises(CFGNotFound);
    ComponentList get_all_components() raises (CFGNotFound);
    ComponentList get_components(in ComponentType type) raises(CFGNotFound);
    DependenceList get_dependence_list(in Component comp) raises(CFGNotFound);
    ReferenceList get_reference_list(in Component comp) raises(CFGNotFound);
    Reference get_reference(in Component comp, in string local_id) raises(CFGNotFound);
    //Operacoes de Configuracao
    Component create_instance(in string id_label, in string host_name, in string id_interface,
       in string id_implementation) raises(CFGCanNotCreate,CFGInvalidLabel, CFGInvalidType);
    Component create_instance_ex( in string id_label, in string host_name, in string id_interface,
       in string id_implementation, in CosLifeCycle::Criteria criteria)
        raises(CFGCanNotCreate,CFGInvalidLabel,CFGInvalidType,CFGInvalidCriteria);
    CriationalInformation interative_create_instance(in string id_label, in string host_name,
        in string id_interface, in string id_implementation, in ConfigListener listener)
        raises (CFGCanNotCreate,CFGInvalidLabel,CFGInvalidType);
    CriationalInformation interative_create_instance_ex(in string id_label, in string host_name,
        in string id_interface, in string id_implementation, in ConfigListener listener,
        in CosLifeCicle::Criteria criteria) raises (CFGCanNotCreate,CFGInvalidLabel,
        CFGInvalidType,CFGInvalidCriteria);
    void interative_bind(in ConfigInstance, in BindInformation);
    void move (in ConfigInstance ci, in string host_name) raises(CFGCanNotOperate);
    ConfigInstance copy (in ConfigInstance ci, in string host_name) raises(CFGCanNotOperate);
    void rebind (in RefManager rm, in Reference reference, in Component newcomp)
        raises(CFGCanNotOperate);
    void replace_implementation(in ConfigInstance obj, in string implementation)
        raises(CFGCanNotOperate);
    //Operacoes Auxiliares às operacoes de Configuracao
```

```
    boolean suspend_all_dependences(in Component comp);
    boolean restore_all_dependences(in Component comp);

    //Operacoes para notificacao de eventos ocorridos nos componentes
    void rm_notify_register_ref(in Component comp, in Reference ref) raises (CFGNotification);
    void rm_notify_remove_ref(in Component comp, in Reference ref)raises (CFGNotification);
    void rm_notify_remove_all(in Component comp)raises (CFGNotification);
    void rm_notify_lock_ManageableRef(in Component comp, in Reference ref) raises (CFGNotification);
    void rm_notify_unlock_ManageableRef(in Component comp, in Reference ref)
        raises(CFGNotification);
    void ci_notify_initmove(in Component comp)raises (CFGNotification);
    void ci_notify_cancelmove(in Component comp)raises (CFGNotification);
    void ci_notify_endmove(in Component comp, in ConfigInstance newobj)raises (CFGNotification);
    void ci_notify_endcopy(in Component newcomp, in ReferenceList reflist)raises (CFGNotification);
    void ci_notify_remove(in Component comp);
};

interface RefManager : ManagedSets::Member {
    //Administracao de referencias. Operacoes voltadas para o proprio objeto
    void register_ref(in Reference reference, in boolean notify_app) raises (CFGInvalidLabel);
    void remove_ref(in string name, in boolean notify_app) raises(CFGNotFound);
    void remove_all(in boolean notify_app);
    void lock_ManageableRef(in string name)raises(CFGNotFound,CFGCanNotOperate);
    void unlock_ManageableRef(in string name)raises(CFGNotFound,CFGCanNotOperate);
    boolean ready_to_call(in string name);
    long set_timeout(in long timeout);
    Object get_uptodate_ref(in string name) raises(CFGNotFound,CFGTimeout);

    //Recuperacao de informacao
    ReferenceList get_all_references();
    ReferenceList get_references(in RefClass refclass);
    Reference get_reference(in string name) raises(CFGNotFound);
    ConfigApplication get_config_app();
    Component describe_component();

    //Operacoes de Configuracao
    void rebind(in Reference oldreference, in Component newcomp, in boolean notify_app)
            raises (CFGNotFound, CFGInvalidType, CFGCanNotOperate);

    //Operacoes de notificacao
    void suspend_ref(in Reference reference);
    void restore_ref(in Reference reference); //retira do estado suspendo
    void update_ref (in Reference oldreference, in Component newcomp);
    void notify_remove(in Reference reference);

    //funcoes de recuperacao
    void restore_all();
    void update_all (); //informacao buscada em ConfigApplication

    //operacoes de notificacao
    oneway void ap_notify_suspend_ref(in Reference reference);
    oneway void ap_notify_restore_ref(in Reference reference);
    oneway void ap_notify_update_ref (in Reference oldreference, in Component newcomp);
    oneway void ap_notify_remove(in Reference reference, in Component removed_comp);
    void ap_notify_rebind (in Reference oldreference, in Component newcomp) raises (CFGNotification);
};
```