# Uma metodologia para o desenvolvimento de aplicações na plataforma OSIMIS baseada no paradigma de configuração

*Roberto Wagner da Silva Rodrigues*
Departamento de Informática
UFPE/ETFCe
50732-970 Recife-PE
E-mail: rwsr@di.ufpe.br

*Paulo Roberto Freire Cunha*
Departamento de Informática
Universidade Federal de Pernambuco
50732-970 Recife-PE
E-mail: prfc@di.ufpe.br

*Mauro Oliveira*
LAR - ETFCe/UFC/UECE
60430-531 Fortaleza-CE
E-mail: mauro@benfica.etfce.br

**Resumo**

Herança não se apresenta como o meio mais adequado para expressar relacionamento em aplicações de gerenciamento de redes, que são de natureza essencialmente distribuída. Este trabalho propõe uma metodologia de desenvolvimento baseada no paradigma de configuração, utilizando a plataforma OSIMIS(*OSI Management Information Services*) como suporte. Mecanismos de configuração são implementados para apoiar o desenvolvimento de tais aplicações, fornecendo uma alternativa mais flexível do que a herança de classes presente na plataforma, privilegiando sistemas de gerenciamento de grande porte, como em ambientes de redes corporativas. Um estudo de caso que faz uso da metodologia proposta é apresentado e sua implementação no OSIMIS é descrita.

**Abstract**

Inheritance is not the most adequate means of conveying relationship in management applications that have essentially distributed nature. This work proposes a configuration-based methodology of development, using the OSIMIS (OSI Management Services) platform as a back-end. Configuration mechanisms are implemented.

providing a more flexible alternative then inheritance of classes, supplied by the platform. This proposed methodology enhances the applications for large systems, specially in corporative networks. A case study is presented and its implementation in OSIMIS is described.

# 1 Introdução

Conforme a ISO [1], a finalidade do gerenciamento de redes é a de prover mecanismos de controle, monitoração e coordenação de recursos num ambiente de redes. As atividades de monitoramento dizem respeito à leitura de dados para posterior manipulação e geração de informações. Atividades de controle correspondem a modificações no estado dos recursos gerenciados, alterando os atributos que os representam, possibilitando realizar mudanças no comportamento da rede de modo a controlar o seu funcionamento.

Um dos fatores que levaram à necessidade de desenvolvimento de aplicações de gerenciamento de redes foi a rápida disseminação do seu uso, aumentando substancialmente o volume de dispositivos a serem gerenciados, principalmente nas redes corporativas. Por outro lado, as atividades de monitoramento e controle, mesmo automatizadas, tiveram sua complexidade incrementada pelo alto grau de heterogeneidade, devido a oferta de soluções de diferentes fornecedores, direcionadas para atender necessidades específicas de gerenciamento, combinadas ou não às ferramentas já existentes sem nenhum compromisso com padrões. Como consequência, o volume e complexidade das informações de gerenciamento a serem tratados tornou a tarefa dos administradores difícil de serem executadas.

Para administrar uma rede heterogênea, há a necessidade de um gerenciamento global que permita a cooperação entre diferentes sistemas, requerendo ferramentas automatizadas capazes de manter uma elevado nível de abstração dos recursos gerenciados[10]. O desafio de desenvolver tais aplicações exige ferramentas e técnicas adequadas, principalmente quando consideramos o gerenciamento distribuído para redes de grande porte que possuem centenas de dispositivos remotos e/ ou locais a serem gerenciados.

A plataforma OSIMIS propõe-se a resolver essas questões, fornecendo um conjunto de APIs(*Application Program Interfaces*) numa metodologia totalmente orientada a objetos, incorporando uma implementação do protocolo CMIP (*Common Management Information Protocol*)[20] em todos os seus serviços de modo a facilitar o desenvolvimento de aplicações de gerenciamento distribuído de redes, dentro do modelo OSI.

Tais aplicações distribuídas representam uma instância de Sistemas Distribuídos, área em que o Departamento de Informática da UFPE tem realizado vários trabalhos, utilizando o paradigma de configuração[3, 2, 8]. A proposta deste trabalho consiste então, em propor uma metodologia baseada nesse paradigma, como uma alternativa a ser usada na plataforma OSIMIS, tratando tanto aspectos de modelagem quanto de implementação das aplicações de gerenciamento distribuídos nos moldes do padrão OSI. Um conjunto de classes está sendo implementada, a fim de permitir uma alternativa de relacionamento de objetos, que complemente o mecanismo de herança presente na plataforma.

Este artigo está estruturado da seguinte forma: a seção 2 descreve o modelo OSI de gerenciamento de redes, no qual os mecanismos de configuração propostos estarão baseados. A seção 3 descreve as características das arquiteturas básicas, usando os conceitos desse

[1] International Standards Organization

modelo. A seção 4 descreve o OSIMIS e os principais recursos existentes. Na seção 5 os princípios que fundamentam o paradigma de configuração são comentados e é mostrado o seu uso no OSIMIS. A metodologia é apresentada na seção 6 e um exemplo aplicado a um cenário típico de distribuição é descrito na seção 7 e na seção final é apresentada uma conclusão do trabalho.

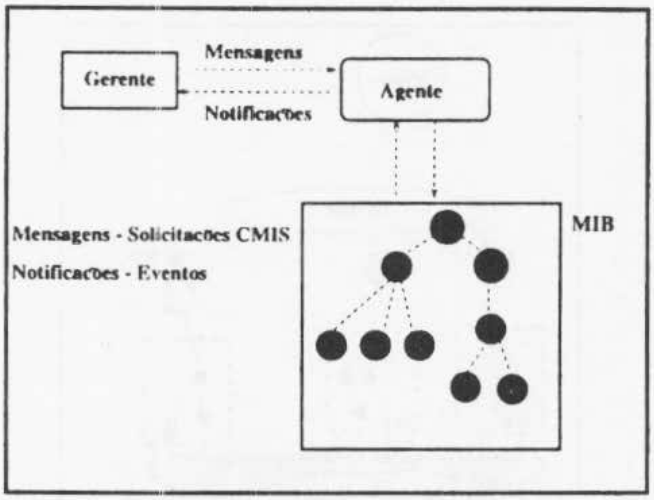


Figura 1: Modelo de gerenciamento OSI.

## 2 O modelo de gerência OSI

A ISO propõe um *framework* de gerenciamento bastante genérico, nos moldes do modelo cliente/servidor para computação distribuída , baseado na tríade **gerente**, **agente** e **MIB**(figura 1).

O objeto **gerente** é do tipo actor[17], podendo cooperar com outros objetos mas nunca pode ser operado por eles, podendo no entanto, receber notificações. Por outro lado, uma vez que mantém seu próprio *thread* de controle, o gerente pode ser visto como um objeto ativo que modela um processo de gerência. O gerente solicita informações para fins de monitoração ou realiza alterações para fins de controle, enviando operações por meio do objeto agente[21].

O objeto **agente** pode operar sobre outros objetos ou ser operado por eles. O objetivo do agente é atender as solicitações emitidas pelo processo gerente, acessando a base de informações capaz de prover as informações requisitadas. O agente é então responsável por executar as operações de gerenciamento atuando sobre os objetos que têm acesso aos recursos que num determinado instante irão ser submetidos às ações de gerenciamento. As notificações retornadas pelos objetos gerenciados, são repassadas ao gerente. As informações de gerenciamento são trocadas entre os objetos, utilizando-se o protocolo CMIP.

A **MIB** é um repositório de objetos que representam os recursos gerenciados de forma abstrata [5]. Esses recursos podem ser qualquer dispositivo físico da redes, tais como *modens, hubs, gateways*, ou entidades lógicas tais como processos, outros objetos, tabela de roteamento, etc.

*Templates* foram padronizados de modo a tornar claro a forma como uma MIB precisa ser definida dentro dos conceitos OSI de orientação a objetos. Por meio deles, pode-se especificar implementações de objeto gerenciados. A ISO propôs um guia para desenvolver MIBs usando uma linguagem formal para especificação chamada de GDMO(*Guide for Development of Managed Object*)[5].

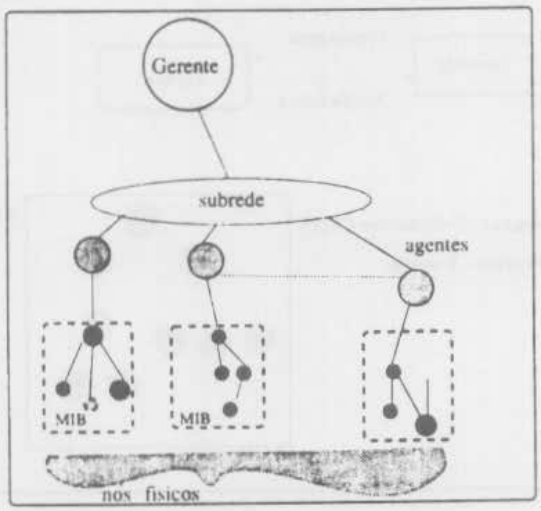


Figura 2: Arquitetura centralizada

# 3 Arquiteturas de gerenciamento

Uma análise das arquiteturas de gerenciamento mais comuns, nos permite avaliar as principais questões envolvidas no que diz respeito às restrições impostas pelo OSIMIS. Existem três diferentes arquiteturas básicas que podem ser utilizadas: *arquitetura centralizada, distribuída e hierárquica.*

A *arquitetura centralizada* mostrada (figura 2) é a mais comum. Tem-se um objeto gerente responsável por coordenar toda a rede. Esse controle consiste na manipulação das informações de gerenciamento, bem como da comunicação a ser estabelecida com os agentes que atuam sobre os objetos gerenciados. Essa arquitetura apresenta várias desvantagens, principalmente no que diz respeito a falhas. Entretanto tem sido a mais empregada, por ser mais fácil de se implementar, já que todo o controle é feito num único nó de processamento implementado por um único objeto ou processo.

A arquitetura de *plataformas* (figura 3) é uma variação do modelo centralizado, sendo implementado pelo OSIMIS e consiste em se ter pelo menos dois objetos. Um que gerencia a plataforma e outro que gerencia as aplicações. O objetivo do primeiro é esconder do programador detalhes de comunicação a ser provido pela plataforma. Problemas de heterogeneidade são resolvidos nesse nível. Esses serviços são acessados pelas aplicações por meio de APIs. A plataforma é responsável pelo monitoramento e controle, implementando funções padronizadas de gerenciamento, tais como de contabilização[7] e relatório de eventos[13]. O segundo objetiva implementar a aplicação propriamente dita, produzindo informações de gerenciamento alto nível. Neste esquema entretanto, temos

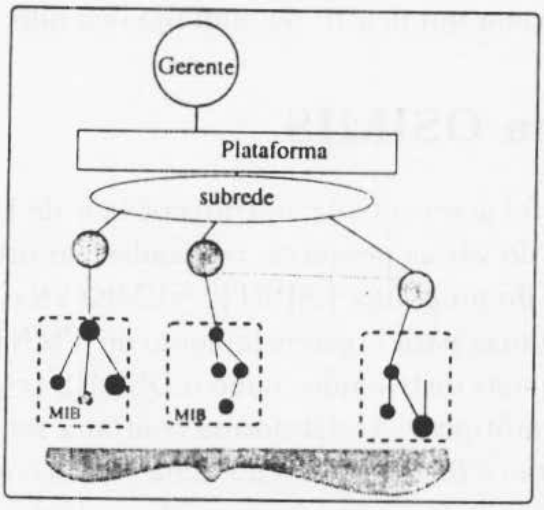


Figura 3: Arquitetura usando plataforma

restrições em termos de escalabilidade, próprios das arquiteturas centralizadas e principalmente vulnerabilidade a falhas.

À *arquitetura distribuída* (figura 4) está associado o conceito de domínios, onde vários

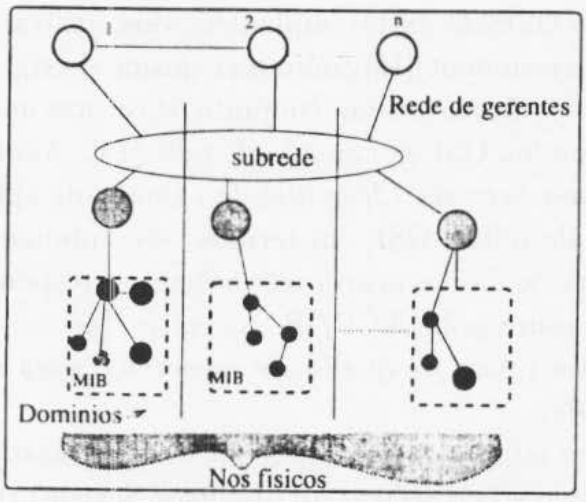


Figura 4: Arquitetura distribuída

gerentes se associam de forma a compor uma aplicação de gerenciamento. Cada gerente atua de acordo com a definição da natureza desse domínio, podendo ser baseado em critérios de distribuição geográfica dos equipamentos, administrativos, podendo ser também um reflexo da natureza da organização onde a rede está inserida, etc.

Gerentes utilizam-se dos serviços uns dos outros de acordo com as informações que são necessárias para cada domínio. A distribuição favorece principalmente a escalabilidade, uma vez que o sistema pode crescer apenas adicionando novos domínios da aplicação.

A *arquitetura hierárquica*, caracteriza-se pela existência de um objeto gerente coordenador e abaixo dele, objetos gerentes subordinados até chegarmos no nível de objetos gerenciados. Nesse caso, temos um esquema de gerente de gerentes, onde cada parte da

aplicação é realizada por cada um dentro do domínio definido.

# 4 A plataforma OSIMIS

A plataforma OSIMIS foi desenvolvida na Universidade de Londres (*University College London*), sendo resultado de várias pesquisas realizadas em projetos europeus, tais como INCA, PROOF e MIDAS do programa ESPRIT, NEMESYS e ICM do programa RACE. Estas pesquisas estão voltadas para o gerenciamento de TMNs(*Telecommunications Management Network*) e sistemas distribuídos onde o OSIMIS tem sido usado como suporte ao desenvolvimento de protótipos. A plataforma continua sendo desenvolvida, sendo de domínio público até a versão 4.0. Recentemente uma versão comercial foi disponibilizada.

OSIMIS é uma plataforma de gerência orientada a objetos, implementada principalmente em C++. Ela fornece um ambiente para o desenvolvimento de aplicações, escondendo todos os detalhes de serviços de gerenciamento que lhes dá suporte, através de uma interface de programação orientada a objeto, permitindo que os implementadores se dediquem a construção somentes das aplicações, deixando de lado detalhes de comunicação, como é característico das arquiteturas que se utilizam do modelo de plataforma. O OSIMIS otimiza o modelo de gerência OSI tal qual descrito anteriormente, possibilitando que objetos agentes, num dado instante façam o papel de gerente de acordo com a organização hierárquica do sistema.

Muitos dos recursos do OSIMIS foram implementados, utilizando as primitivas do ISODE(ISO Development Environment)[16] como mecanismo de suporte a comunicação OSI. O ISODE é um ambiente composto por um conjunto de rotinas que permitem implementar as camadas mais altas da pilha OSI da camada de rede [15]. Assim, o ISODE implementa todos os ASES (*Application Services Elements*) da camada de aplicação. Adicionalmente, permite diferentes perfis de pilhas OSI, em termos de combinação de protocolos das camadas de rede e transporte a fim de prover os serviços correspondentes, sendo disponível tanto sobre X.25 quanto sobre pilha TCP/IP. As cinco classes da camada de transporte, que representam diferentes níveis de qualidade de serviço para diferentes tipos de redes são também implementadas.

O OSIMIS fornece uma biblioteca de classes que pode ser usada para apoiar o desenvolvimento de aplicações tendo a GMS(*Generic Managed System*) como modelo arquitetural para implementação. O uso das classes depende do enfoque dado a aplicação e da forma como se quer que elas trabalhem.

Um primeiro enfoque suporta o desenvolvimento de aplicações de gerenciamento de redes orientado a eventos. Cada evento é ativado, usando ou a estratégia de *pooling* ou usando *triggers*. Se a aplicação se adequa a uma dessas estratégias, então ela pode ser implementada a partir da especialização das classes *Coordinator* e *KS*, respectivamente para gerente e agente. Outro enfoque é relacionado à programação. Um compilador traduz de especificações altamente abstratas em ASN.1 para estruturas C e vice-versa. Dessa forma, o implementador manipula um código real, ao invés de estruturas abstratas.

As principais classes do OSIMIS são mostradas na figura 5 juntamente com suas operações, utilizando a notação de objetos de OMT(*Object Modelling Technique*) [17]. A classe *attr* fornece uma interface para gerar e manipular novos atributos. Qualquer outro atributo além dos já predefinidos na plataforma, precisam herdar a classe *attr*, que

não tem nenhum relacionamento com as outras classes. As classes *MO* e *Top* fornecem métodos para desenvolver a base de informação de gerenciamento. Outras interfaces para desenvolver gerentes e agentes foram adicionadas, usando o conceito de MIBs remotas[12].

O OSIMIS oferece vários serviços que já estão disponibilizados[11], tais como:

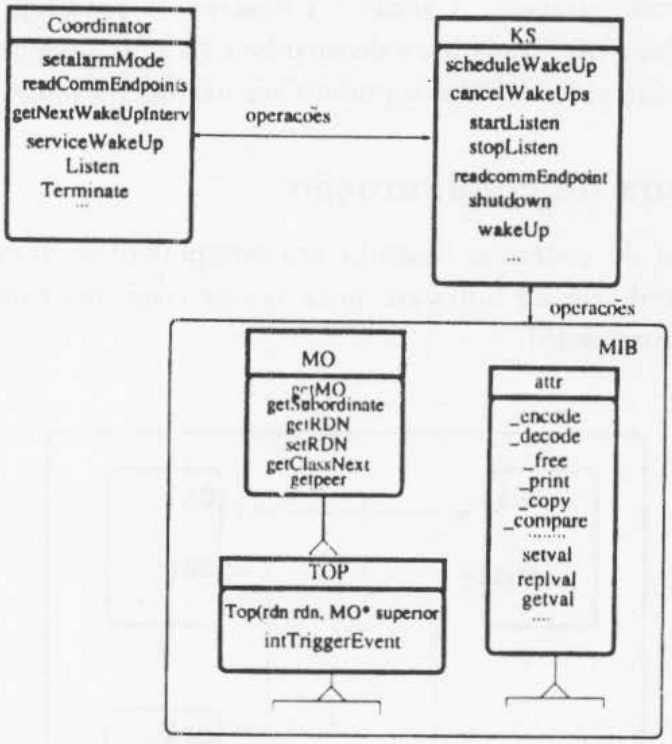


Figura 5: classes do OSIMIS

- várias primitivas de comunicação que implementam o protoco padrão OSI( CMIP) de gerenciamento de redes;
- um agente OSI que realiza todas às funções especificadas no modelo de gerência, tais como *scoping* e *filtering*;
- um exemplo trivial de MIB UNIX que fornece um objeto gerenciado o qual informa o número de usuários que estão utilizando determinada estação de trabalho;
- uma aplicação genérica que prover um *gateway* entre sistemas que usam o protocolo OSI/CMIP e Internet/SNMP [4];
- um mecanismo de coordenação através do objeto *coordinator* que gerencia todo o processo de comunicação do sistema;
- um compilador para gerar código C a partir de especificações feitas em GDMO;
- um mecanismo de transparência à localização de agentes utilizando a implementação do ISODE do serviço de diretório OSI, além de outros.

# 5 Configuração no OSIMIS

Um componente é uma unidade básica de programação em que grandes sistemas podem ser decompostos, sendo compilados e testados separadamente. O sistema portanto, é construído através da interconexão entre os componentes. Às diferentes formas e regras pelas quais os componentes podem ser interligados caracterizam a construção de um sistema por meio de configuração. A seção 5.1 descreve o paradigma de configuração e aponta os principais elementos úteis para desenvolver sistema de gerenciamento de redes. A seção 5.2 detalha como esse elementos podem ser usados na plataforma.

## 5.1 O paradigma de configuração

O desenvolvimento de sistemas baseado em componentes, tem mostrado ser uma técnica eficaz de Engenharia de Software para tratar sistemas complexos, em particular dos sistemas distribuídos[19].

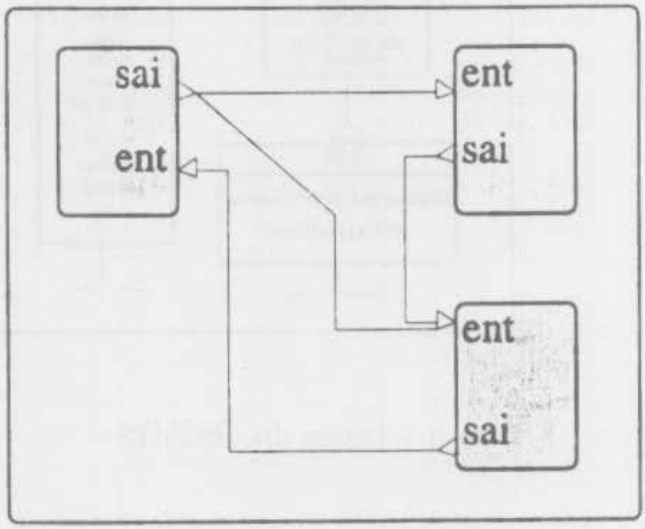


Figura 6: configuração de componentes

Uma mensagem trocada entre os componentes corresponde à menor unidade de execução, e a comunicação entre eles só é possível após instruções de configuração entre os módulos terem sido estática ou dinamicamente aplicadas. Essa mensagem pode gerar efeitos internos, modificando o estado do componente ou efeitos externos provocando mudanças no ambiente no qual estão inseridos. A figura 6 mostra estes conceitos, onde se observa, um processo *bottom-up* de desenvolvimento de sistemas.

Importantes características fazem da configuração, um paradigma aplicável no desenvolvimento de sistemas de gerenciamento. Existem muitas similaridades com orientação a objeto. A mais importante delas, no contexto deste trabalho, é a possibilidade de permitir um projeto modular de construção de sistemas, onde a configuração usa o conceito de componentes e a orientação a objetos, o conceito de classes. A tabela 1 mostra as similaridades entre ambos os modelos.

O paradigma de configuração está fundamentado sobre alguns princípios que demonstram as sua reais potencialidades. Primeiro, a linguagem de configuração permite

| CARACTERÍSTICAS | ORIENTAÇÃO A OBJETOS | CONFIGURAÇÃO |
|---|---|---|
| Relacionamento | herança/polimorfismo | configuração |
| Acesso a componentes | interface | portas |
| Modularidade | classes | componentes |
| Hierarquia | herança | agrupamento |
| Tipagem | objetos de uma classe | objetos do tipo componente |
| concorrência | objetos comunicantes | processos comunicantes |

Tabela 1: orientação a objetos x configuração

fazer uma descrição estrutural do sistema sem se preocupar como os componentes foram implementados. Além disso, os componentes possuem interfaces bem definidas e são totalmente independentes uns dos outros. Dessa forma, tendo um componente fonte enviado uma mensagem, o componente de destino não precisa ser conhecido. Isso leva ao que se chama de independência de contexto. Outro ponto, é que o desenvolvimento de componentes mais complexos é feito através da composição hierárquica de componentes básicos por meio de agrupamentos. O componente mais externo pode ser visto como se fosse um componente básico. Finalmente, todas as modificações são feitas a nível estrutural, onde a preocupação com as mudanças reside na manipulação de componentes e conexões.

Os dois paradigmas são ortogonais no sentido de que um não interfere no outro, pois no paradigma de configuração não importa em qual linguagem o componente foi desenvolvido. Dessa forma, podemos propor a coexistência no OSIMIS dos dois modelos, uma vez que observados os princípios acima, a plataforma pode ser até remodelada sem causar nenhuma interferência. A introdução de serviços de configuração no OSIMIS é facilitada pela própria característica modular da orientação a objetos, utilizando a biblioteca de classes embutida na plataforma. Dependendo da aplicação, o programador poderá usar conceito de componentes configuráveis, tornando o trabalho de desenvolver aplicações mais flexível, e portanto, mais fácil e seguro.

Além dos fatores relacionados a facilidades de desenvolvimento, existe a questão do particionamento dos módulos nas diversas máquinas da rede. Todos os módulos são alocados aos processadores antes dos programas serem executados. Entretanto, é útil poder reconfigurar o programa ou evoluir de alguma forma essa configuração enquanto ele estiver sendo executado. É um caso a ser considerado já que muitos sistemas não podem parar para fazer modificações, sendo essa capacidade conhecida como reconfiguração dinâmica.

## 5.2 Adicionando mecanismos de configuração no OSIMIS

A grande contribuição da visão de componentes no OSIMIS, utilizando o paradigma de configuração, é o fato de permitir maior flexibilidade e independência para o desenvolvimento de aplicações de gerenciamento. O conceito de herança, forte na plataforma, não é tão flexível de modo a facilitar esse desenvolvimento.

Conforme descrito em [9], a herança favorece a coesão interna de elementos dos componentes, fortalecendo a idéia de localidade, e o relacionamento entre os componentes implementados por classes é implícito. A configuração de componentes favorece à idéia de

distribuição, tornando esse relacionamento explícito e configurável independente da forma como o componente foi desenvolvido.

O OSIMIS oferece um conjunto de classes genéricas que podem ser especializadas quando da implementação de novas aplicações. Na nossa abordagem, acrescentamos novas classes à plataforma, o qual permitem que os conceitos de configuração sejam aplicados, onde o uso da herança não fornece a flexibilidade necessária que as aplicações distribuídas requerem. Essas classes podem ser especificadas a partir de *templates* de configuração propostos, tendo uma função similar àquelas definidas na linguagem GDMO.

As principais classes a serem criadas são: *Port*, incluídas nos componentes para que os mesmos possam usar o conceito de portas. Qualquer componente de gerenciamento criado precisa utilizar essa classe, para que por meio dela possa haver a interconexão entre os componentes. A classe *System*, por meio da qual as configurações da aplicação como um todo possam ser gerenciada.

Em C++, linguagem em que o OSIMIS foi escrito, existe o conceito de programação de componentes. Esses componentes nada mais são que arquivos compilados separadamente, sendo sufixados pela letra *h*, e são conhecidos como arquivos *header*. Os componentes propriamente ditos são sufixados por *c*. A dependência entre os componentes é feita através da macro *#include* e a linguagem faz a distinção entre a interface e sua implementação necessárias nesse contexto.

Isto favorece tecnicamente a construção de componentes usando C++, nos moldes

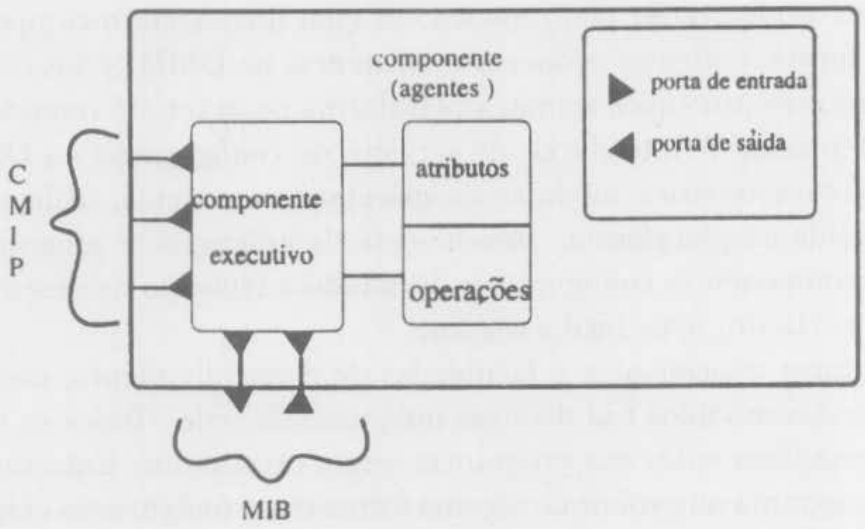


Figura 7: arquitetura de componentes

propostos como na linguagem CONIC[18] e adotado pela linguagem CL[6], cuja estrutura é mostrada na figura 7. Em CONIC, um componente é denominado **nó**, sendo a menor unidade de configuração em cima da qual as mudanças a nível de configuração acontecem. Caso seja necessário uma modificação, todos as conexões daquele nó serão afetadas. Em CL[2], essa visão foi remodelada, de modo a diminuir o impacto da reconfiguração, focando a configuração também nas conexões entre os nós. Esses nós possuem um conjunto de portas que formam a interface por meio da qual ocorre a comunicação com os outros componentes. Por representar um avanço nos conceitos usados em CONIC, *templates* baseados na linguagem CL serão utilizados neste trabalho.

Em termos de gerenciamento de redes, para usar esses conceitos, um nó ou componente corresponde a um agente ou um gerente. A MIB também pode ser considerada dessa forma, desde que possua um objeto ou processo, intermediando a interconexão entre o agente e os objetos gerenciados por meio de portas. Dentro dos conceitos de configuração, a terminologia adequada seria chamar esses elementos de componentes de gerenciamento.

# 6 Metodologia de implementação

Aplicações de gerenciamento de redes já vêm sendo desenvolvidas a algum tempo. Sendo a idéia de configuração nova nessa área, consideramos que uma metodologia que discipline o uso do paradigma é o primeiro passo para estimular o seu uso no OSIMIS para desenvolver sistemas de gerenciamento distribuídos. A seção 6.1 apresenta as limitações do OSIMIS que justificam o uso do paradigma de configuração. A seção 6.2 apresenta os passos da metodologia a serem seguidos e a seção 6.3 mostra um cenário de como o OSIMIS funciona usando esses conceitos e um exemplo é dado para ilustrar o seu uso.

## 6.1 Limitações do OSIMIS

Por usar uma arquitetura de plataforma, a idéia de localidade embutida no OSIMIS tende a ser restritiva em situações em que o comportamento em função de herança pode ser sensível à cooperação entre métodos. Este trabalho adota uma visão mais composicional como uma alternativa a ser usada na plataforma, complementar à herança, tornando explícita a dependência entre componentes.

Apesar das inerentes vantagens oferecidas pelo OSIMIS por causa da orientação a objetos, certas aplicações exigem uma forma mais flexível de relacionamento que não somente por meio de herança. Em aplicações que são de natureza essencialmente distribuída, herança não se apresenta como um mecanismo mais adequado, pois tende a favorecer mais o acoplamento entre componentes, enquanto que a distribuição estimula seu particionamento pela rede, exigindo maior flexibilidade. A título de ilustração dessas limitações destacamos dois casos:

1. O OSIMIS não prover uma forma flexível o suficiente para configurar gerentes pertencentes a diferentes domínios de gerenciamento numa arquitetura distribuída. Esta é uma situação em que a cooperação entre gerentes é necessária para se obter uma informação de gerenciamento. Os gerentes em geral podem estar remotos. As classes existentes não são suficientes para estabelecer relacionamento entre esses gerentes. O conceito de MIBs remotas são usadas neste caso, porém não fornecendo a mesma flexibilidade que o paradigma de configuração oferece.

2. A plataforma não fornece mecanismos para reconfiguração dinâmica, em que é preciso restringir ou ampliar a comunicação entre gerentes ou entre gerente e agentes distribuídos em tempo de execução. Neste caso, o uso de herança é inadequado para se estabelecer relações entre gerentes e agentes. Além do mais, nesta situação a aplicação necessita maior robustez, onde mecanismos de tolerância a falhas é fundamental. A plataforma não explora mecanismos dinâmicos, para aplicações que

requerem mudanças constantes, seja devido a alta volatilidade da aplicação, seja porque isso é uma característica dos recursos gerenciados.

## 6.2 Definição dos templates

A proposta para a implementação da metodologia define novos *templates* baseados em CL, como um complemento à linguagem de especificação GDMO. Por ser a linguagem de especificação padrão do modelo OSI de gerenciamento o seu uso combinado com os *templates* de configuração propostas nos dá o ferramental necessário para especificar uma aplicação de gerenciamento usando a abordagem de configuração. Esses *templates* serão usados para definir componentes na plataforma que possam ser relacionados, usando o conceito de interconexão. A partir daí, a configuração de componentes de modo a formar um sistema de gerenciamento pode ser especificada mais claramente.

Os objetos gerenciados são definidos em termos dos seus atributos, operações que ele realiza, notificações que ele pode emitir e seu relacionamento com outros objetos, implementado por meio de herança. Para cada uma dessas necessidades existe um *template* em GDMO para especificar esses itens. O OSIMIS possui um compilador de GDMO para C, de modo a reduzir partes estruturas repetitivas do código necessários para criar os objetos gerenciados. No anexo são mostrados os dois *templates* básicos propostos para expressar o relacionamento por meio de configuração.

GDMO possui a noção de componentes que é referenciado nos *templates* como *package*. A especialização da classe é indicada pela clásula CHARACTERIZED BY que incorpora um ou mais *packages* necessários. A cláusula *CONDITIONAL PACKAGE* é o que mais se aproxima da idéia de configuração. Um determinado componente poderá estar presente ou não a fim de realizar uma determinada função, desde que a condição estabelecida seja verdadeira. Assim, o componente está presente ou ausente naquela classe.

Os dois *templates* de configuração usam o conceito de portas que serão usadas para realizar a interconexão entre os componentes. Elas podem ser de saída ou de entrada. Outros componentes(pacotes) que possam fazer parte são declarados pela cláusula *USE*. A cláusula BEHAVIOR serve apenas para descrever o que faz o componente.

O *template* de sistema possui a cláusula *USE*. Neste caso, é usada para compor o sistema de gerenciamento. O restante são operações que podem ser realizadas nos componentes. *LINK* e *UNLINK* conectam e desconectam componentes gerentes a agentes e vice-versa, estabelecendo uma ligação entre eles. Destaca-se aqui a grande flexibilidade em situações em que tivermos numa arquitetura distribuída, onde um gerente de outro domínio quer se conectar a um agente de cujos serviços ele não dispunha anteriormente. As operação CREATE permite instanciar um componente e associar um nó processador a ele.

Os *templates* em GDMO foram criados para especificar relações de objetos de uma MIB. O seu relacionamento com os gerentes e agentes é realizado por meio da troca de mensagens, representada pelas operações *get*,*set*, etc., implementadas por meio do CMIP. Os *templates* de configuração permitem especificar os componentes do sistema de gerenciamento e o seu relacionamento por meio de conexões. As ações de configuração podem ser realizadas de forma dinâmica. Numa situação em que tivermos um gerente conectado a vários agentes, a introdução ou remoção das conexões pode ser realizada, mantendo o agente em execução se ele tiver sendo usado por um outro gerente ou fazendo o papel de

um para determinado domínio de gerenciamento.

Uma vez que o CMIP prover a comunicação entre objetos comunicantes, a conexão será implementada por meio do sobrecarregamento das operações padronizadas *M_create*, *M_delete*, *M_action*, *M_get* e *M_set*, acrescidas do conceito de portas, a fim de manter a conformidade com o *framework* de gerenciamento. Assim, será possível ao implementador usar uma abordagem que vê um sistema de gerenciamento como um conjunto de compontes *gerentes*, *agentes* e *MIBs* interconectados para formar uma aplicações específica de gerenciamento dentro de um determinado domínio. De outro modo, os componentes de gerenciamento seriam conectados por meio de *sockets*.

## 6.3 Um exemplo: computação distribuída no OSIMIS

No ambiente OSI, agentes e gerentes são tidos como processos. Essa abstração é necessária, enquanto os sistemas operacionais continuarem trabalhando com esse conceito. Aqui utilizamos o termo objetos ativos[1], e a capacidade se comunicarem os tornam objetos comunicantes. Uma classe para que represente essa abstração precisa ser criada de modo a prover a parte de execução dos componentes.

A figura 8 apresenta um exemplo, onde se tem um cenário de execução em que um gerente(cliente) e vários agentes(Servidores) trocam algumas mensagens. Um gerente pode estar conectado a um ou vários agentes. Um gerente não interfere nas ações dos agentes que são totalmente independentes, conforme a visão de configuração. Nas diversas estações de trabalho estão espalhados diversos agentes que têm diferentes funcionalidades. Agentes SMA (*System Management Agent*) do OSIMIS são particionados em várias máquinas, sendo uma situação real de uso da plataforma.

Após os objetos gerenciados serem definidos e criados, as classes de configuração são

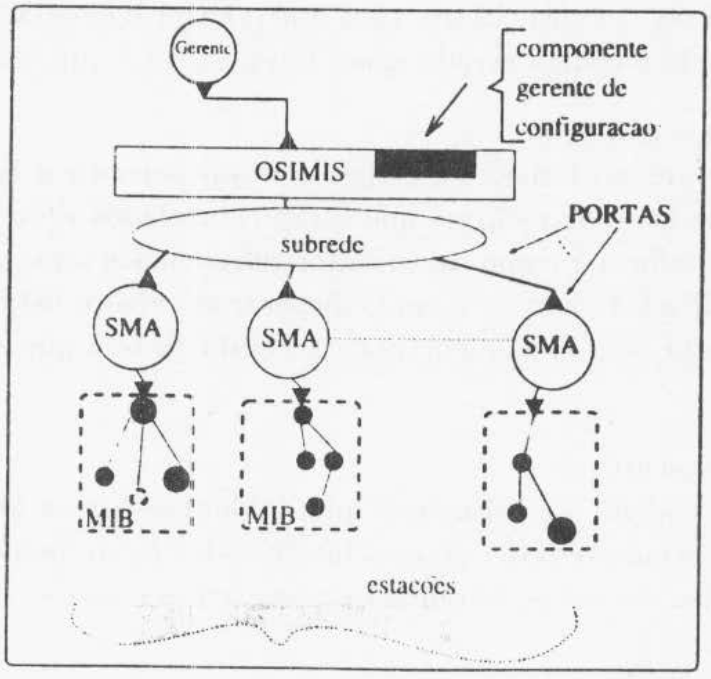


Figura 8: cenário de configuração com OSIMIS

incluídas e os diversos componentes são então configurados. Cada componente de gerencia-

mento pode ser desenvolvido com os recursos do próprio OSIMIS, pois são independentes entre si.

Os *templates* propostos não são padronizados e, portanto, não podem ser reconhecidos como entrada do compilador GDMO. Por isso, um mapeamento para C++ é mostrado. Os *templates* servem para especificar de forma abstrata a configuração do ambiente, antes que os componentes de gerenciamento sejam implementados.

## 6.4 Passos da Metodologia

A seguir são apresentados 6 passos necessários para realizar uma aplicação de gerenciamento no OSIMIS, usando configuração.

1. *Definindo a aplicação*
   Examinar minuciosamente o que é preciso gerenciar. Determinar os atributos, ações que se deseja realizar sobre esses recursos e que tipo de eventos podem ocorrer com eles; esta fase corresponde a uma análise de requisito e neste caso a técnica OMT é recomendada.

2. *Especificando a aplicação definida*
   Após o exame acima, cria-se a especificação com o *template* de configuração para a aplicação como um todo, considerando o cenário mostrado, definindo os componentes que comporão o sistema de gerenciamento. Estes por sua vez, são especificados em GDMO.

3. *Aplicando a especificação*
   Aplicar ao compilador, o texto da especificação definida no passo anterior. Ele produzirá muito do código que representa os objetos gerenciados já especificados. Na realidade, serão geradas classes, para que possam ser reusadas qualquer que seja a definição. Como o código gerado não é tão perfeito, é preciso fazer modificações;

4. *Implementando a aplicação*
   Neste ponto, é preciso fornecer o código que vai permitir a comunicação entre os objetos gerenciados e os recursos que serão controlados e/ou monitorados. Além disso, é preciso informar como os atributos selecionados serão atualizados, como as ações serão realizadas e como e quando disparar os eventos listados. Toda essa parte precisa ser escrita pelo implementador. No OSIMIS terá que ser escrita em código C++;

5. *Construindo o agente*
   Com todo esse código, é preciso criar uma bibiioteca para a MIB gerada, contendo os objetos gerenciados em que se está interessado. Neste ponto, ou se constrói um agente particular, ou usa-se ferramentas para se fazer isso; e

6. *Introdução das portas*
   Todos os componentes(MIBS, agentes e gerentes) são acrescidos das classe *port* de modo a permitir a sua comunicação. Uma vez em execução, o relacionamento entre agentes, gerentes e MIBs podem ser tratadas em mais alto nível, e modificadas de forma dinâmica, desde que um gerente de configuração seja criado.

Até o quinto passo, podemos criar um agente associado a sua respectiva MIB, onde a estrutura do programa permite implementar de modo usual com as interfaces do OSIMIS. O passo 6 envolve a inclusão das classes *Port*, *Component* e a classe *Process* para introduzir os processos nas estações. Esse processo é feito usando a cláusula #include do C. As operações *p_get* e *p_set* são equivalentes às operações *get* e *set* acrescidas das portas de comunicação. O gerente envia uma mensagem a qualquer agente SMA para o qual ele esteja configurado. O agente que irá atender aquela mensagem desconhece quem a solicitou. Os objetos gerenciados são conectados ao SMA por meio de arquivos específicos do OSIMIS. Na próxima seção um estudo de caso é descrito. Detalhes de sua implementação pode ser encontrado em [14]

# 7 Um estudo de caso implementado

Nesta seção apresentamos uma implementação baseada na metodologia proposta a qual denominamos de *dtalk* (*distributed talk*) que se utiliza das facilidades de distribuição existentes no OSIMIS e adicionamos os mecanismos de configuração somente com a classe *Port*, umas vez que as outras classes estão em fase de teste. Esta aplicação visa evitar a necessidade de ter que realizar uma busca visual a procura da máquina por meio da qual seu possível interlocutor poderá ser acessado. Isto normalmente é feito utilizando o comando *who*, levando o usuário a despender tempo quando consideramos uma ambiente de dezenas de estações . A seção 7.1 discute as vantagens e desvantagens do uso do dtalk; a seção 7.2 descreve sua implementação como um estudo de caso de uso da metodologia.

## 7.1 dtalk - um talk distribuído

Examinando minuciosamente os parâmetros do *talk* do UNIX, identificamos aqueles essenciais para realizar o acesso ao recurso gerenciado. Neste caso, desejamos saber se determinado usuário está presente ou ausente nos principais servidores da rede do DI da UFPE. A presença de pelo menos uma instância de um processo do usuário interlocutor dá início ao talk abrindo a janela no UNIX. Caso o usuário não esteja na rede ou ocorra um *timeout*, uma notificação é gerada informando uma das duas situações(figura 9).

O *talk* exige que o usuário use o comando *who* para encontrar o usuário e a respectiva máquina em que está executando alguma serviço. Este comando não suporta o uso de *pipe*(ambiente SUN) do UNIX por meio do qual poderíamos recuperar essa informação. Após a busca visual uma linha com talk@usuário seria dado. Num cenário onde tivéssemos centenas de *hosts*, essa busca visual seria no mínimo tediosa.

A vantagem do dtalk, é que o usuário não precisa saber previamente onde está o seu possível interlocutor. O talk do UNIX é chamado somente quando a estação é encontrada, evitando ter que usar o comando *who*. Por termos nos preocupado apenas com a aplicação, o uso do protocolo *finger* para acessar as estações precisaria ser personalizado para ter melhor desempenho.

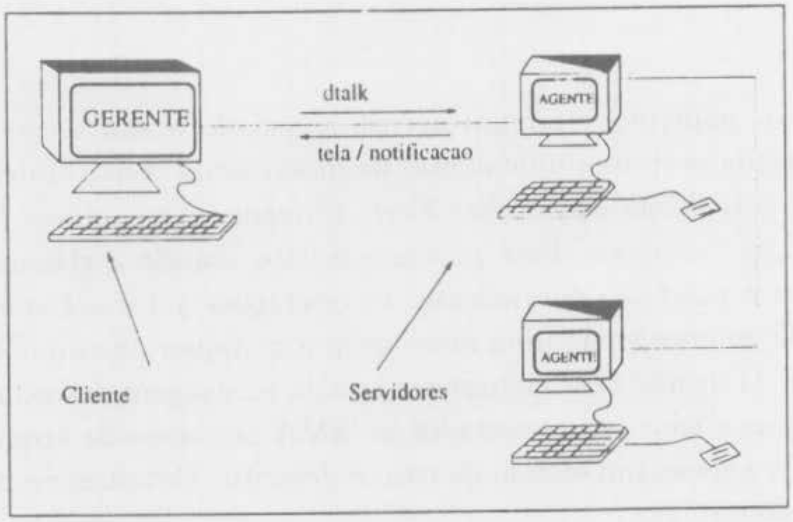


Figura 9: Funcionamento do dtalk

## 7.2 Implementando a metodologia

Para implementar o *dtalk*, de conformidade com o modelo, um gerente, um agente e uma MIB precisam ser desenvolvidos. No caso do OSIMIS, uma grande parte das funções necessárias para implementar o dtalk já estão disponíveis. O agente SMA é carregado nas estações e o programa *dtalk* que faz o papel de gerente faz todas as consultas através dele. Abaixo do SMA os objetos gerenciados são conectados.

O modelo de objetos (passo 1) é trivial neste exemplo, por ser uma aplicação eminentemente interativa. Uma classe chamada de dtalk é especializada a partir da classe *Attr* descrita na seção 4. Como o dtalk se utiliza do comando *finger* do UNIX, os atributos da classe é facilmente extraído da saída desse comando mapeada para tabela 2.

| *unxTalkTable* | | | | | |
|---|---|---|---|---|---|
| unxLoginUser | unxLoginname | unxLogTTY | unxTimeAccess | unxMachine | unxUsrStatu |
| rwsr | Roberto Wagner | p0 | Mon 08:58 | floresta | |
| cagf | Carlos Andre Guimara | p1 | Mon 08:35 | pedra | |
| neuman | Jose Neuman | p1 | Tue 10:40 | lia.ufc.br | |

Tabela 2: Instância da saída do *finger*

A especificação usando a template de configuração(passo 2) e a sua transformação em código C++ é mostrada abaixo. Por razões de espaço os templates em GDMO e a sintaxe dos atributos em ASN.1 não são mostradas, podendo ser encontrada também em [14]. Somente as especificações em GDMO foram submetidas ao compilador (passo 3). A de configuração foi transformada em código C++.

A programação que acessa as estações foi anexada ao código gerado pelo compilador (passo 4). O mecanismo de portas é adicionado por meio do módulo *porta.h*. No caso do dtalk, os passos 4 e 6 são realizados simultaneamente, uma vez que até este estágio da implementação, portas foram adicionados apenas entre a MIB e o SMA, um agente não

precisa ser criado (passo 5), dado que o SMA é suficientemente genérico para acessar os objetos gerenciados.

*< templates e código C++ equivalente>*

```
---######### componentes_OSIMIS        ############## -----

   dtalk   COMPONENT
   PORT      p_get IN
             p_Set OUT
   USE       gerente1
   BEHAVIOR  " Componente gerente       "

   SMA       COMPONENT
   PORT      p_get  IN
             p_set OUT
   BEHAVIOR  "agente OSIMIS que acessa os objetos gerenciados"

---#########    OSIMIS_sistema     ############# -----

   OSIMIS_sistema   SYSTEM
   USE   SMA, gerente;
   CREATE SMA_mib1   AT   ulysses   ## instancias de SMA
          SMA_mib2   AT   garanhuns
          SMA_mib3   AT   athena;
   LINK   dtalk.p_get TO SMA_mib1.p_set
          dtalk.p_get TO SMA_mib2.p_set
          dtalk.p_get TO SMA_mib3.p_set;


--- CODIGO C++  -------

 #include    "porta.h"
 main()
   {
      Port   SMA;
      while (FOREVER)
          {
              ....
              sma_mib1.p_get(nome-oper);
              .... // linha de acesso a MIB
              sma_mib1.p_set(nome-oper,dado);
          }
   }
#include    "porta.h"
```

```
main()
  {
     Port   dtalk;
     while (FOREVER)
        {
            .........
            sma.p_set(nome-oper);
            sma.p_get(nome-oper,dado);
            printf(%s,dado);
        }
  }
```

Para uma aplicação bem mais complexa, temos que considerar a construção de um gerente de configuração. Ele será responsável por direcionar as mensagens dos gerentes aos respectivos agentes que estão aptos a fornecer aquele serviço e vice-versa. A sua principal função entretanto, será de permitir a reconfiguração dinâmica, conforme já mencionado, em situações em que o sistema de gerenciamento não pode ser interrompido.

# 8 Conclusão

A utilização do OSIMIS como plataforma em diversas situações de gerenciamento, notadamente em sistemas TMNs, tem tornado a mesma um atraente ambiente de desenvolvimento. Tendo sua concepção orientada a objeto, o OSIMIS apresenta limitações devido ao mecanismo de herança no tratamento de aplicações de natureza distribuída.

Este trabalho, ao inserir mecanismos baseados em conexão de componentes, procura combinar o grande poder de modelagem da orientação a objetos com a alta flexibilidade do paradigma de configuração. Os conceitos de herança e agregação favorecem requisitos como de escalabilidade, mas não fornecem flexibilidade suficiente para o desenvolvimento de grandes sistemas de gerenciamento. A grande vantagem do paradigma para tratar estas questões é que a independência dos componentes permite continuar utilizando orientação a objeto em que a plataforma foi construída, não interferindo nos recursos disponíveis.

A distribuição das atividades de gerenciamento é facilitada, quando se tem uma arquitetura de rede distribuída associada a técnicas de engenharia de software adequada a essa arquitetura. Considera-se que o desenvolvimento de aplicações de gerenciamento distribuídos para grandes redes, com características de heterogeneidade ou não, requer o uso dessas técnicas. Dentro desse contexto o paradigma de configuração associado à orientação a objeto funcionam como visões complementares. A tarefa de construir sistemas de gerenciamento é substancialmente facilitada, principalmente em redes distribuídas e dinâmicas. O sistema de comunicação baseado em portas evita a dependência entre componentes, possibilitando que um ou mais gerentes se associem a um ou mais agentes. O resultado final é um ambiente que dá suporte não só a modelagem de objetos, mas também fornece ao programador alternativas para construir aplicações distribuídas de gerenciamento. A metodologia proposta permite então, disciplinar o desenvolvimento de acordo com o paradigma de configuração, possibilitando implementar a metodologia na plataforma conforme os padrões OSI de gerenciamento.

# Anexo - *templates* de configuração

```
 -- ############  Classe Componente ###############

<componente> COMPONENT
    PORT <porta>[,<porta>]* IN
         <porta>[,<porta>]* OUT
    USE  <pacote>[,<pacote>]*;
    [BEHAVIOR <define-comportamento>[,<define-comportamento>]*;]

 -- ############  Classe Sistema    #############

<sistema>  SYSTEM
    USE <componente>[,<componente>]*
    CONFIGURATION ACTION
    CREATE        <componente>   AT <processador>
    [CREATE       <componente>  AT <processador>]*;
    [LINK         <componente>.[<porta> TO <componente>.[<porta>]]
    [,LINK        <componente>.[<porta> TO <componente>.[<porta>;]*;]
    [UNLINK       <componente>.[<porta> TO <componente>.[<porta>]]
    [,UNLINK      <componente>.[<porta> TO <componente>.[<porta>]*;]
    [ACTIVATE     <porta>||<componente>[,<porta>||<componente>]*;
    [,ACTIVATE    <porta>||<componente>[,<porta>||<componente>]*;
    [UNACTIVATE   <porta>||<componente>[,<porta>||<componente>]*;
    [,UNACTIVATE <porta>||<componente>[,<porta>||<componente>]*;
```

# Referências


[1] Grady Booch. *Object Oriented Design with Applications.* The Benjamin/Cummings Series in ADA and Software Engineering. The Benjamin/Cummings Publishing with applications. 1991.

[2] Jorge de Araújo Lima Filho. Desenvolvimento de um Modelo Baseado em Conexões para Configuração Dinâmica de Sistemas Distribuídos. Master's thesis, DI/CCEN/UFPE. Recife-PE, 1991.

[3] Virgínia Carneiro C. dePaula. Linguagem de Configuração CL: Visão Geral e Sugestões de Extensão. January 1993.

[4] J.N. deSouza, N. Agoulmine, K. Mccarthy, and G. Pavlou. CMIP/SNMPv1 Translation Through Applications Level Gateway Using The OSIMIS/ISODE Platform. RACE IS&N Conference, 1993.

[5] ISO/IEC10164-4. Information technology - open systems interconnection structure of management information - part 4 : Guidelines for Definitions of Managed Objects. Technical report, ISO, March 1991.

[6] George Justo and Paulo Cunha. Programming Distributed Systems with Configuration Languages. International Workshop on Configurable Distributed Systems, London, March 1992.

[7] L.F. Korman, A. Coser, E.M. Vieira, and C.B. Westphall. Extensão da plataforma de gerência OSIMIS pela implementação da Função de Medida de Contabilização. *Simpósio Brasileiro de Redes de Computadores*, pages 371–388, maio 1995.

[8] J. A. Lima Filho and P. R. F. Cunha. Um Ambiente Distribuído para Suporte à Configuração Dinâmica. *VI Simpósio Brasileiro de Engenharia de Software*, pages 325–342, Novembro 1992.

[9] Satoshi matsuoka and Akinori Yonezawa. *Analysis of Inheritance Anomaly in Object-oriented Languages*, volume 1, chapter 4, pages 107–150. Gul Agha, Peter Wegner and Akinori Yonezawa, 1993. Research directions in concurrent object-oriented programming.

[10] A.M Oliveira and N. Agoulmine. Interoperability of open Management Systems. pages 209–224. Elsevier Science Publishers B.N. North-Holland, 1994.

[11] G. Pavlou, G. Knight, M. Kevin, and S. Bhatti. The OSIMIS platform: Making OSI Management Simple. August 1994.

[12] G. Pavlou, G. Knight, and S. Walton. *Experience of Implementing OSI Mangament Facilities*, pages 259–270. Integrated network Managemnent II ,W. Zimmer editors, 1991.

[13] G. Pavlou, Bhatti S.N., and Knight G. Automating the OSI Internet Management Conversion Through The Use of an Object-Oriented platform .

[14] Roberto W.S. Rodrigues. OSIMIS - a tutorial by example. Technical report, Universidade Federal de Pernambuco, November 1995.

[15] T.M ROSE. *The ISO Development Environment:User's Manual , V.1 Application Services. Version 7.0.* Performance Systems International, Inc., July 1991.

[16] T.M ROSE. *The ISO Development Environment:User's Manual , V.4, Version 7.0.* Performance Systems International, Inc., July 1991.

[17] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall International Editions, 1991.

[18] M. Sloman, J. Magee, and J. Kramer. Building Flexible Distributed Computing Systems in Conic. 1981.

[19] Morris Sloman and Jeff Kramer. *Distributed Systems and Computer Networks.* Prentice/Hall International. 1987.

[20] International Standards Organization / International Electrotechnical Commission. Information Technology - Open Systems Interconnection - Common Management Information Protocol Specification. *ISO/IEC 9596*, may, 1990.

[21] International Standards Organization / International Electrotechnical Commission. Information Technology - Open Systems Interconnection - Systems Management Overview. *ISO/IEC DIS 10040*, may, 1991.