CONCEPÇÃO E IMPLEMENTAÇÃO DE UMA APLICAÇÃO DE GERÊNCIA DE BILHETES DE ANORMALIDADE EM AMBIENTE DISTRIBUÍDO

Marco Antonio de Carvalho Guapo
CEFET-PR
e-mail: guapo@dainf.cefetpr.br

Manoel Camillo de Oliveira Penna Neto
CEFET-PR
e-mail: penna@dainf.cefetpr.br

Resumo

Este artigo descreve uma aplicação de gerência de bilhetes de anormalidade em um ambiente distribuído e heterogêneo, baseada na arquitetura CORBA como ambiente de desenvolvimento. A aplicação, denominada sistema gerenciador de bilhetes de anormalidade (SGBA), destina-se principalmente a auxiliar administradores de ambientes distribuídos a resolverem problemas de modo mais sistemático e rápido. Enfoca-se neste artigo, o aspecto da utilização de objetos distribuídos no desenvolvimento de uma aplicação de gerência, mais especificamente através da arquitetura CORBA. É apresentado o projeto da aplicação com um conjunto de objetos distribuídos, avaliando-se assim as vantagens e dificuldades que a utilização desta tecnologia traz para a construção de sistemas de gerência distribuídos e heterogêneos.

Abstract

This paper describes a trouble ticketing management application in a distributed and heterogeneous environment, based on CORBA architecture as the development environment. The application, named trouble ticketing management system (SGBA), aims to support distributed systems administrators to solve system problems in a more systematically and quickly way. This paper focus on using distributed objects for the development of management applications, in particular, using CORBA architecture. We present the application design as a collection of distributed objects, making assessments concerning advantages and difficulties of the use of this technology for the construction of distributed and heterogeneous management systems.

1.INTRODUÇÃO

A gerência de ambientes distribuídos é conhecida como um problema complexo. A construção de aplicações de gerência é uma atividade que visa tornar disponível aos gerentes destes sistemas, ferramentas para otimizar a realização desta tarefa. A gerência do ambiente distribuído, é por sua natureza também distribuída, e nada mais natural que utilizar ferramentas distribuídas na sua execução.

Tomemos como exemplo, um cenário simplificado do fluxo de ações na solução de uma anormalidade ocorrida durante o desempenho de um sistema distribuído. A anormalidade é levadas ao conhecimento de um operador, através de um contato por iniciativa do usuário que identificou o problema, através da emissão de um *e-mail*, ou conversa telefônica, por exemplo. O operador fará um registro do problema, o qual será encaminhado para um especialista. O registro do problema e seu acompanhamento durante a sua solução é denominado bilhete de anormalidade (*trouble ticket*). Um sistema gerenciador de bilhetes de anormalidade (SGBA) é uma ferramenta que suporta o processo de solução de problemas, informando aos operadores as anormalidades existentes e o estágio de solução em que se encontram, coordenando múltiplos especialistas que atuem em conjunto na solução de um problema [TERP87].

Um SGBA pode tornar o trabalho de gerenciamento de problemas de uma rede eficiente em muitos aspectos:

- eficiência na solução dos problemas: com os bilhetes de anormalidade os responsáveis pela solução dos problemas terão todas as informações disponíveis sobre as anormalidades existentes, e o estágio em que se encontra a solução das mesmas, podendo gerenciar a estratégia de solução, evitando a execução de tarefas redundantes que acarretariam em maior tempo na sua solução;

- obtenção de informações estatísticas sobre o processo de solução de bilhetes de anormalidade: com estes dados estatísticos, o coordenador dos operadores e especialistas poderá analisar o tempo médio de solução de problemas, dispondo ainda de dados para avaliar a eficiência de operadores e especialistas;

- geração de relatórios para análise dos diversos componentes do sistema: estes relatórios fornecem os dados necessários para avaliação dos componentes do sistema, permitindo a identificação daqueles mais afetados, e provendo uma base de referência para como resolver novos problemas, através da análise das atividades de reparo de problemas semelhantes;

- integração com o sistema de gerência de falhas: o sistema gerenciador de bilhetes de anormalidade pode ser integrado ao sistema de gerência de falhas, onde certos alarmes provocariam a abertura automática de bilhetes de anormalidade.

Um bilhete de anormalidade pode ser criado a partir de problemas em qualquer componente do sistema distribuído, independentemente da localização dos operadores e especialistas, e dos ambientes em que foi gerado onde será resolvido. Esta características nos levam a uma solução baseada em uma arquitetura distribuída e heterogênea. De modo a atender os requisitos de distribuição e heterogeneidade do sistema, selecionamos a arquitetura CORBA (*Common Object Request Broker Architecture*) [OMG91], como base para o desenvolvimento. Os dois motivos principais desta escolha são: a arquitetura é suficientemente completa para os objetivos aos quais sistema se propõe, e é um padrão aberto com grande aceitação industrial.

2. MODELO PARA BILHETE DE ANORMALIDADE

A Figura 1 ilustra o processo de solução de problema usando o SGBA, representado como um fluxo de ações de gerência.

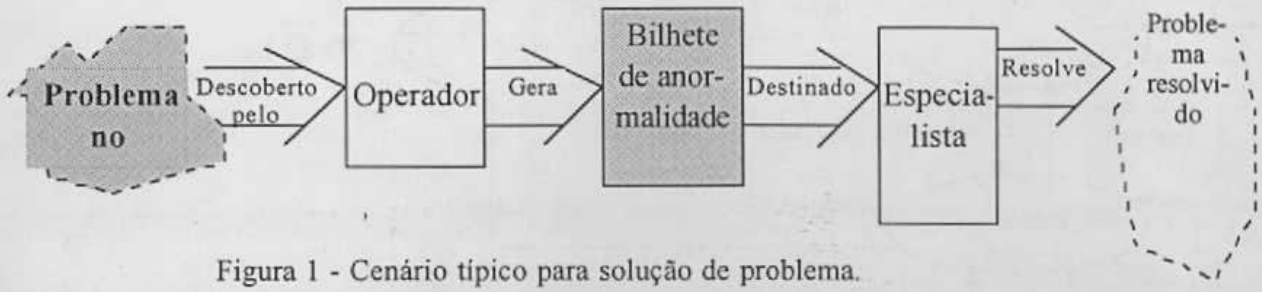

Figura 1 - Cenário típico para solução de problema.

Um problema no sistema é detectado por um operador, através de uma mensagem ou contato com o usuário, ou outros procedimentos operacionais. Este problema pode ser de hardware e/ou de software, ou de qualquer outra natureza. A partir daí o operador cria um bilhete de anormalidade (BA), tornando-o público e oficial. No ato da criação do BA, várias informações devem ser fornecidas, tais como, a descrição do problema, a prioridade de solução, o(s) componentes(s) afetados(s), a(s) rede(s) atingida(s), e o operador responsável. Após a criação, o gerente de bilhetes de anormalidade atribui o bilhete ao especialista em melhores condições para solucionar o problema, levando em consideração o perfil dos especialistas, o volume de trabalho que cada um já tem atribuído, e outras informações pertinentes.

A medida que atividades relativas à solução do problema vão sendo executadas, informações relativas a estas ações são acrescentadas ao BA, para que se possa saber que medidas já foram e quais ainda não foram tomadas. Estas atividades serão chamadas aqui de atividades de reparo. Durante a solução do problema, o bilhete de anormalidade pode ser modificado de várias maneiras, como por exemplo, atualização das informações contidas no bilhete de anormalidade, inclusão e remoção atividades de reparo.

Outra alteração importante que pode ocorrer durante a existência de um bilhete de anormalidade: a sua atribuição a outro especialista. Isso pode ocorrer devido a inadequação do perfil do especialista corrente, ou por outra razão de cunho gerencial. A partir do momento em que o problema foi resolvido, o BA é encerrado, sendo arquivado para que possa ser fonte de informações para relatórios e estatísticas.

A partir do cenário apresentado, concebemos um BA como um objeto, cuja máquina de estados é apresentada na Figura 2. Os estados que o BA pode assumir são:

- QUEUED: neste estado o bilhete foi criado mas não foi atribuído a um operador.

- OPEN: neste estado o bilhete já foi atribuído a um operador porém não está sendo consultado. Isto não impede que o bilhete seja atribuído a outro operador.

- CLEARED: neste estado o bilhete está sendo modificado por um operador, ou seja, o bilhete está sendo analisado por alguém que poderá realizar ações no intuito de solucionar a anormalidade descrita pelo bilhete. Neste estado, nenhum operador, a não ser o que colocou o BA neste estado, poderá realizar operações neste bilhete.

- CLOSED: neste estado o BA foi encerrado, não significando que foi a anormalidade foi solucionada. Este estado pode ser atingido a partir de todos os outros estados, nas seguintes condições:

• se o bilhete estava no estado QUEUED e veio para este estado, quer dizer que o bilhete foi interpretado por um dos operadores como uma duplicação;

• se o bilhete estava no estado OPEN e veio para este estado, quer dizer que o BA, por uma razão bem determinada, não poderá ser solucionado;

• se o BA estava no estado CLEARED e veio para este estado, quer dizer que a anormalidade foi solucionada.

•

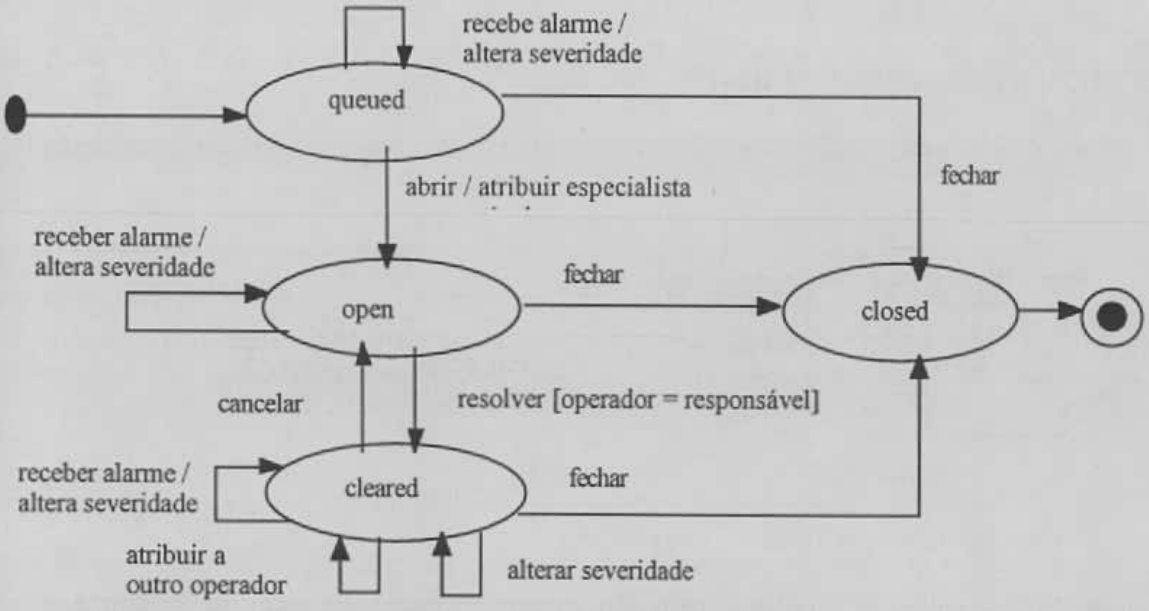

Figura 2 - Diagrama de estados do bilhete de anormalidade

3. ESPECIFICAÇÃO

3.1 Arquitetura

Nesta seção é apresentada a arquitetura do SGBA, isto é, os diversos objetos que compõem o sistema e o inter-ralacionamento entre eles. A Figura 3 representa a arquitetura.

- GerenteBA: objeto que realiza a atribuição de um bilhete de anormalidade a um determinado operador. A atribuição se dará baseando-se nos atributos que indicam o operador mais apropriado a um determinado tipo de problema. Está relacionado a vários bilhetes de anormalidade.

- Alarme: objeto que verifica se o tempo de solução de um determinado bilhete de anormalidade expirou, gerando uma exceção quando isto ocorre. O monitoramento por alarmes permite que ações gerenciais sejam tomadas decorrido um certo intervalo de tempo, como por exemplo, aumentar a prioridade de solução do BA. Está sempre relacionado a um bilhete de anormalidade.

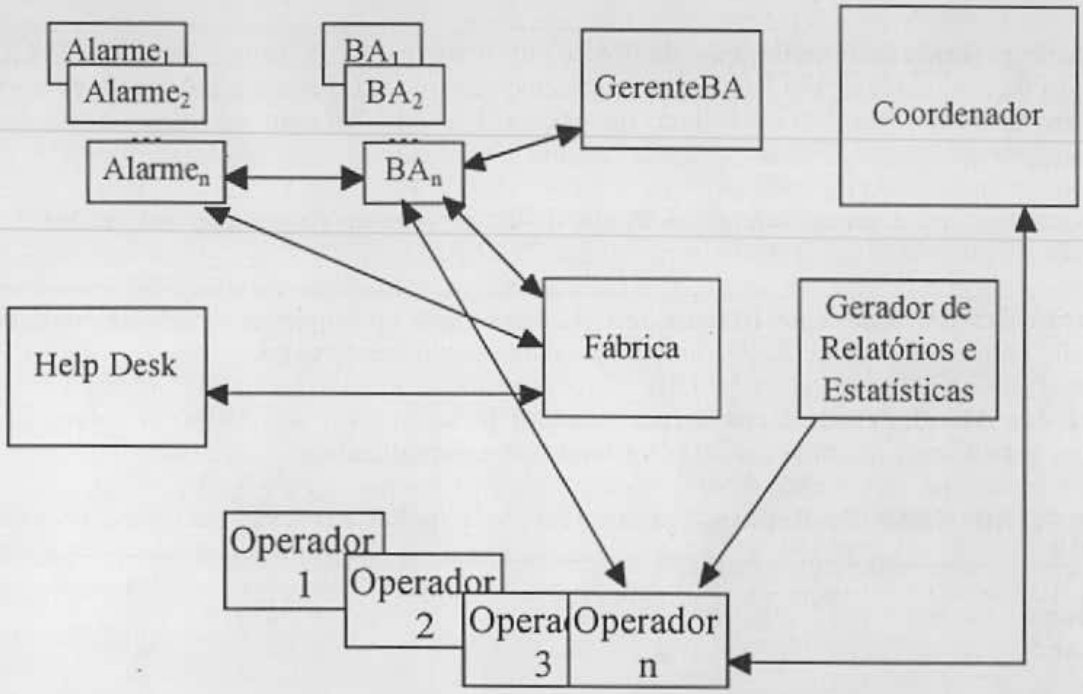

Figura 3 - Arquitetura do sistema

- Coordenador: objeto responsável pela coordenação dos operadores no sistema. Realiza verificação de senhas e operações de cadastro dos operadores. Está relacionado a vários operadores.

- Fábrica: objeto que realiza a criação dos objetos bilhete de anormalidade e alarme. Está relacionado a vários objetos *help desk* e a vários bilhetes de anormalidade e alarmes.

- Operador: objeto responsável pela solução de bilhetes de anormalidade. Interage diretamente com os operadores humanos. Está relacionado a um gerador de relatório e estatísticas, a um coordenador, e a zero ou mais bilhetes de anormalidade.

- *Help Desk*: objeto responsável pela criação de bilhetes de anormalidade. É através deste objeto que os bilhetes de anormalidade são inseridos no quando um problema é identificado. Está relacionado a uma fábrica.

- Gerador de relatórios e estatísticas: objetos que realizam respectivamente a criação de relatórios e estatísticas, a partir de dados fornecidos pelos bilhetes de anormalidade. Estão relacionados a vários operadores.

3.2 Modelo de Objetos

Para construção do modelo de objetos do SGBA, utilizou-se a metodologia OMT [RUMB94]. Esta metodologia prevê a construção de três modelos: o modelo de objetos, o modelo funcional, e o modelo dinâmico. Através destes três modelos tem-se uma especificação completa do sistema, a nível de estruturas de informação estáticas, comportamento dinâmico e fluxo de informação. Não é possível descrever os detalhes dos três modelos neste artigo, mas as mesmas encontram-se disponíveis em forma de relatório técnico [CEFET-TELEBR96].

3.2.1 Bilhetes de Anormalidade

Os bilhetes são os principais objetos com que o sistema irá lidar. Eles possuem atributos para armazenar as informações sobre as anormalidades detectadas e implementam um conjunto de métodos que controlam sua própria evolução, de acordo com o modelo dinâmico (ver Figura 2). São os seguintes os métodos de um BA:

- **Remoção de um BA**: a remoção de bilhetes de anormalidade será efetuada automaticamente, no momento em que o BA atingir o estado CLOSED. Entretanto, esta remoção será lógica, pois este BA continuará armazenado em um arquivo de *Log*, durante um certo período de tempo, para servir de informação para estatísticas e relatórios.

- **Mudança de prioridade e atribuição de BA a outro operador**: a atribuição de um BA a outro operador e a mudança de prioridade de um BA, somente poderão ser realizadas quando o BA estiver no estado OPEN, por qualquer operador, ou quando estiver no estado CLEARED, pelo operador correntemente responsável pelo BA.

As operações descritas a seguir, referentes às atividades de reparo, só poderão ser realizadas pelos operadores responsáveis, quando os BA estiverem no estado CLEARED.

- **Modificação das Atividades de Reparo**: um operador pode modificar as atividades de reparo que foram sugeridas a ele, ou que foram descritas de uma forma mas executadas de outra.

- **Remoção das Atividades de Reparo**: um operador pode remover atividades de reparo incluídas em um BA, e que no entanto não foram necessárias à solução da anormalidade.

- **Inclusão de Atividades de Reparo**: um operador pode incluir atividades de reparo necessárias à solução do problema.

3.2.2 Coordenador

Este objeto é o responsável pela coordenação dos operadores, tendo a função de cadastrar alterar e excluir os operadores. Suas operações são:

- **Cadastrar Operador**: permite cadastrar os operadores, gravando as suas informações: nome, senha, e tipos de equipamentos, ambientes e componentes conhecidos.

- **Alterar Operador:** permite ao coordenador alterar as informações dos operadores, como senhas e especialidades do operador.

- **Excluir Operador:** permite a exclusão de operadores. Esta operação dever ser seguida por uma confirmação de exclusão, sendo esta uma operação importante e que deve ser realizada com cuidado.

- **Verificar Senha:** realiza a verificação da senha do operador que estiver entrando em operação, enviando uma resposta de permissão para uso do sistema ou não.

3.2.3 Gerenciador de Estatísticas e Relatórios

Uma das principais funcionalidades de um SGBA é poder analisar as anormalidades ocorridas em um determinado intervalo de tempo. Esta análise pode ser, por exemplo, a nível de eficiência dos operadores, analisando-se por exemplo o tempo médio de solução dos problemas, ou ainda a nível de deterioração de componentes, identificando aqueles mais afetados. Diversas conclusões podem ser tiradas a partir da análise dos relatórios e estatísticas de bilhetes de anormalidade. Atualmente, as seguintes estatísticas e relatórios estão implementadas :

- tipo de problemas mais recentes (problemas de hardware ou problemas de software, e quais os elementos mais afetados);
- tipo de problemas mais freqüentes (problemas de hardware ou problemas de software);
- tempo médio de solução dos problemas;
- componentes mais afetados (tanto de hardware como de software).

Alguns relatórios podem ser gerados, baseados nos seguintes critérios:

- hora e data de solução de problemas;
- duração dos problemas;
- solução dos problemas (uma descrição rápida de como foram solucionados);
- componentes afetados;
- estado dos BAs;
- geral.

3.2.4 Temporizadores

Quando um BA é criado, pode ser necessário um controle automático, relativo ao cumprimento de prazo de solução do problema. Para isto existe a opção de criar-se um temporizador associado ao BA, com o objetivo de alertar o operador responsável de que o tempo estimado para solução do problema expirou. O controle de temporização é realizado pelos objetos temporizadores, que monitoram continuamente o BA. Quando um temporizador expira, automaticamente a prioridade do BA associado é incrementada. Caso o BA seja solucionado antes da expiração do temporizador, este é automaticamente destruído.

3.2.5 Operadores

Os objetos operadores implementam a interface entre os operadores humanos e os bilhetes de anormalidade. É através deste objeto que um operador humano interage com os bilhetes de anormalidade atribuídos a ele.

3.2.6 *Help desk*

O *help desk* é o objeto que implementa a interface entre usuários do sistema distribuído que pretendam relatar um problema e os bilhetes de anormalidade. É através dele que os bilhetes de anormalidade são inseridos no sistema. A partir destes dados é realizada a criação do bilhete de anormalidade correspondente.

3.2.7 Fábrica de Bilhetes de Anormalidade e Temporizadores

A fábrica de BA é o objeto responsável pela criação dinâmica dos objetos BA e temporizador. Para cada evento de criação de BA disparado pelo objeto *help desk*, o objeto fábrica é ativado para criar o objeto correspondente. Apesar da sua funcionalidade ser diferente dos demais, este objeto é definido e implementado como os demais. Os objetos fábrica (*factory*) fazem parte do serviço de gerência de ciclo de vida da arquitetura CORBA e são especificados no documento *CORBA services: Common Object Services Specification* no capítulo *Life Cycles Services Specification* [OMG95].

3.2.8 Gerente BA

O Gerente BA é o objeto responsável em atribuir operadores aos bilhetes de anormalidade. Toda vez que um bilhete for criado, o gerente BA atribuirá automaticamente um operador ao bilhete, levando em consideração o problema definido no BA e o operador que poderá mais facilmente solucioná-lo. Existe a opção da atribuição ser realizada por um gerente humano, caso se queira sobrepor ao procedimento automático.

4. IMPLEMENTAÇÃO

4.1 Definição das Interfaces

Concluída a fase de projeto inicia-se a fase de implementação, quando são usados os recursos da arquitetura CORBA. A principal característica desta arquitetura é a definição de uma plataforma para objetos distribuídos, que se acoplam à ela através de uma interface de programação (*API - Aplicattion Programming Interface*) orientada a objetos e padronizada. Objetos não padronizados, acoplam-se à plataforma de duas maneiras distintas: interface de invocação estática, implementada como nos sistemas de invocação de procedimentos remotos tradicionais, e interface de invocação dinâmica, onde as interações com os objetos são construídas em tempo de execução.

Na construção do SGBA utilizamos a interface de invocação estática, que se baseia na definição de uma linguagem de definição de interface (*IDL - Interface Definition Language*) que possui os recursos para a definição de interfaces de objetos, isto é, seus atributos, métodos e suas relações de herança. Para efeito de ilustração, apresentamos a seguir a interface em IDL do objeto bilhete de anormalidade:

```
interface bilhete_anormalidade {
        attribute string esp_responsável;
        attribute sequence <string> atividades_reparo;
        attribute char severidade;
        attribute string gerador_BA;
        attribute sequence <string> máquinas_envolvidas;
        attribute sequence <string> redes_envolvidas;
        attribute char equipamento;
        attribute char ambiente;
        attribute char protocolo;
        attribute string descrição_suscinta;
        attribute string desrição;
        attribute string end_mail_gerador;
        readonly attribute data data_abertura;
        readonly attribute horário horário_abertura;
        readonly attribute data data_fechamento;
        readonly attribute horário horário_fechamento;
        attribute unsigned long ID;

        short Alterar_Severidade( );
        void Enviar_Mail_para_Gerador( );
        string Atribuir_Especialista_Responsável( );
        short Incluir_Atividade_Reparo( in string descrição_atividade);
        short Excluir_Atividade_Reparo( in unsigned short número_atividade);
        short Modificar_Aivídade_Reparo( in unsigned short, in  número_atividade,
                                in string descrição_atividade);
        void Desativar_Temporizador( );
        void Gravar_Informações( );
}
```

Os seguintes pontos podem ser observados:

- A palavra chave ***interface*** define uma classe, neste caso *bilhete_anormalidade*.

- a palavra chave ***attribute*** indica a declaração de um atributo do objeto. A declaração de um atributo é equivalente à declaração de um par de funções de acesso ao atributo, uma para atribuir valores ao atributo e a outra para ler o valor corrente do mesmo.

- a palavra chave ***readonly*** indica que o valor do atributo está disponível apenas para leitura, sendo equivalente a declarar apenas uma função de acesso ao atributo, aquela que lê o seu valor.

- a definição de métodos em IDL guarda grande semelhança com a definição de protótipos de função em linguagens de programação. Uma característica importante está na definição das categorias dos parâmetros de cada método, que podem ser três: ***in***, ***out*** e ***inout***. Os parâmetros definidos como *in* são passados do cliente para o servidor, os parâmetros *out* são passados do servidor para o cliente, e os parâmetros *inout* são passados em ambas as direções.

Como pode-se notar através deste exemplo, a linguagem IDL contém os elementos necessários para a definição de tipos de dados dos argumentos, para a definição das assinaturas dos métodos, organizando-os na definição da interface do objeto em questão.

4.2 Compilando a IDL

Toda implementação CORBA deve incluir compiladores IDL, que geram automaticamente o código de acoplamento dos objetos à plataforma através da interface de invocação estática. Este código de acoplamento inclui as rotinas formatação dos argumentos (ou resultados) das invocações em mensagens e sua respectiva recuperação no sistema remoto, conhecidas como rotinas de *marshalling* e *unmarshalling*, e as rotinas de comunicação.

Para que o código gerado pelos compiladores IDL possam ser usados de forma adequada em linguagens de programação, a arquitetura CORBA define de que modo objetos especificados em IDL devem ser manipulados em uma linguagem particular. Estas regras de programação são denominadas de *language*

binding. Em CORBA já foram definidos *languages binding* par diversas linguagens, entre outras, C, C++, SmallTalk e JAVA. Particularmente, o sistema descrito neste artigo foi implementado em C++.

Para compilar uma especificação IDL, as definições devem estar separadas em arquivos fontes de acordo com necessidade e dependência ou não das interfaces. Geralmente estes arquivos possuem o sufixo ".idl", porém isto pode variar de compilador para compilador. Cada arquivo IDL deve ser compilado, tanto para verificação de sintaxe quanto para o mapeamento para uma linguagem específica, antes que possa ser utilizado por clientes e servidores.

Supondo que a interface do objeto bilhete de anormalidade apresentada acima tenha sido colocada no arquivo "bilhete_anormalidade.idl". Uma possível linha de comando para executar o compilador IDL seria a seguinte:

idl -B bilhete_anormalidade.idl

Um compilador IDL típico geraria ao menos três arquivos para cada arquivo IDL compilado. Por exemplo, para o arquivo bilhete_anormalidade.idl, seriam produzidos os seguintes arquivos:
- bilhete_anormalidade.hh: arquivo que será utilizado pelo cliente e pelo servidor como arquivo de cabeçalhos;
- bilhete_anormalidadeC.cc : código *stub* que deve ser é compilado e ligado à parte cliente;
- bilhete_anormalidadeS.cc : código *skeleton* que deve ser compilado e ligado à implementação do objeto (parte servidora).

Estes arquivos contém códigos gerados automaticamente e não devem ser editados pelo programador. Eles contêm todo o código necessário para as funções de *marshalling*, *unmarshalling* e comunicação.

4.3. Divisão dos clientes e servidores do SGBA

Cada objeto que utiliza os recursos de distribuição da arquitetura CORBA, terá uma parte cliente e uma parte servidora. Na parte cliente são incluídos os recursos para uso do objeto, e na parte servidora são incluídos os recursos para a implementação propriamente dita do objeto. Na fase de projeto já se deve fazer a separação e identificação das partes cliente e servidora no sistema. No SGBA os módulos e os respectivos objetos estão organizados da seguinte forma:

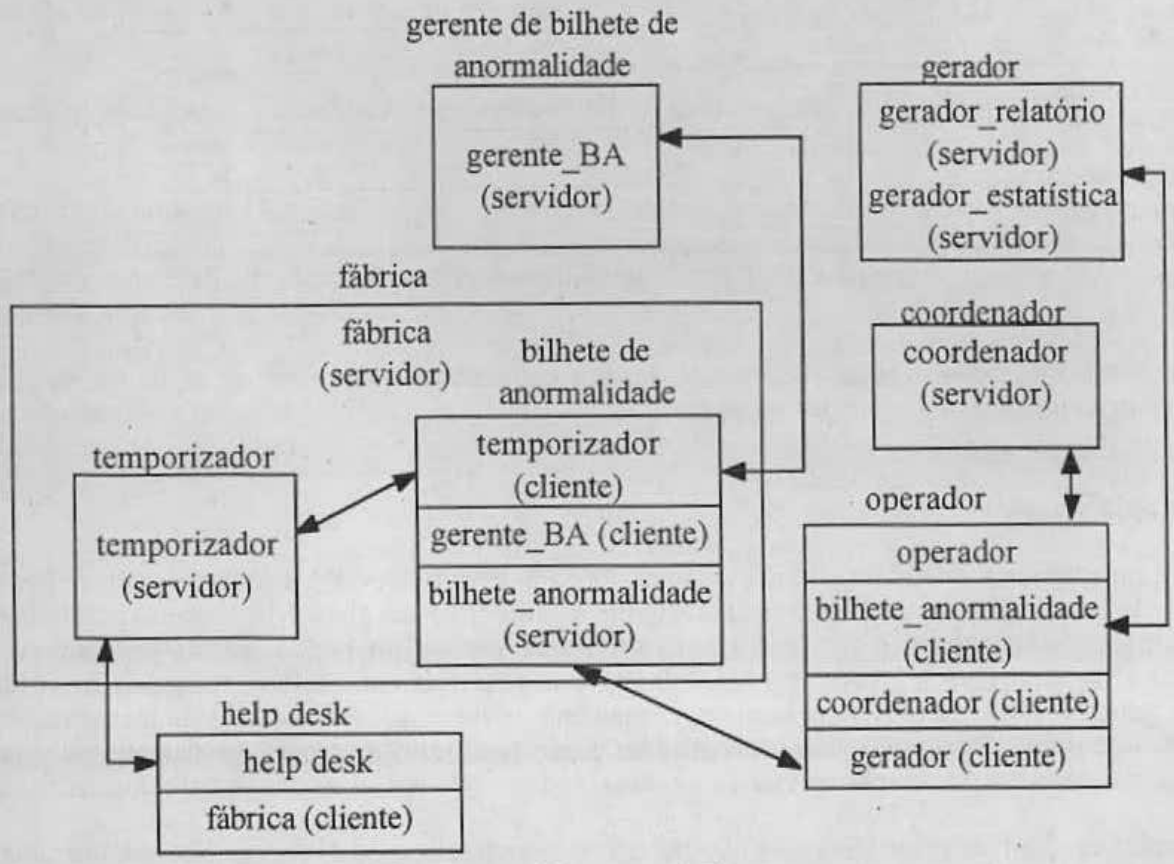

Figura 4 - Clientes e servidores no SGBA

Nota-se que clientes e servidores estão em módulos separados, o que caracteriza a distribuição do sistema. Observa-se ainda, que a nivel de projeto, foram apenas indicados os as partes cliente e servidora, e não a sua distribuição fisica no ambiente, conseguindo-se uma maior independência com relação às plataformas e ao ambiente de execução.

Somente os objetos que possuem uma parte servidora e outra cliente é que necessitam de uma definição em linguagem IDL, que serão posteriormente compiladas produzindo-se os *stubs* (cliente) e os *skeletons* (servidor), permitindo a comunicação seja efetuada entre as partes do objeto distribuído, como explanado na seção anterior.

4.4. Objeto Bilhete de Anormalidade

O principal objeto do SGBA é o bilhete de anormalidade, já que é para seu controle que o sistema existe. Um bilhete de anormalidade tem a função de guardar as informações referentes a um problema qualquer no sistema distribuído. Ele armazena desde hora e data de início, até as atividades de reparo que foram executadas para a solução do problema, além de executar tarefas importantes como alterar sua própria severidade e atribuir-se um especialista. A interface homem máquina que permite o acesso a um objeto bilhete de anormalidade é a seguinte:

hora de início:	data de início:	nome do responsável:	
hora de término:	data de término:	equipamento:	ambiente:
severidade:	descrição breve:		
máquina(s) envolvida(s):		rede(s) envolvida(s):	
protocolo:		tipo do problema:	
Descrição detalhada:			
Atividades de reparo:	Atividade de reparo 1		
Atividade de reparo 2			
Atividade de reparo n			

Figura 5 - Interface de um bilhete de anormalidade

Observamos que todas as informações referentes a um problema ocorrido ou existente na rede estão na interface para análise dos especialistas ou operadores.

4.5. Persistência do objeto BA

Uma característica importante da implementação do SGBA refere-se à persistência do objeto bilhete de anormalidade, isto é, a capacidade de armazenamento deste objeto em memória secundária. O objeto BA, por guardar muitas informações e por poder existir durante longos intervalos de tempo, poderia causar sobrecarga no sistema em que executa, permanecendo ocioso apenas consumindo recursos. Além disso, as informações armazenadas em um bilhete de anormalidade devem sobreviver a paradas voluntárias ou involuntárias do sistema. Decidiu-se então transformar este objeto em um objeto persistente, salvando-se o seu estado em memória secundária, depois de se ter atingido um certo limite de tempo de ociosidade.

A implementação da persistência de objetos não é uma tarefa trivial em um sistema distribuído. Em termos de implementação, a transformação do objeto bilhete de anormalidade em um objeto persistente, exigiu um grande esforço, apesar do suporte adequado do software Orbix da IONA Inc [IONA95a, IONA95b],

usado na implementação. O Orbix disponibiliza uma função denominada *Object Loader*, para suporte à persistência de objetos, que funciona da seguinte forma:

- Existe um objeto chamado *loader* que implementa duas funções: *save*() e *load*(). Estas funções são utilizadas para salvar os dados de um objeto persistente, colocando-o em estado inativo, e para restaurar os seus dados, retornando-o ao estado ativo.

- Deve-se escrever um objeto que especializa por herança o objeto *loader*, por exemplo, *meu_loader*, onde são implementadas duas funções virtuais: *load*() e *save*() que implementas os procedimentos de armazenamento e restauração das informações do objeto persistente. Estas funções podem ser implementadas de acordo com as necessidades específicas do sistema, não havendo portanto um procedimento rígido esta implementação.

- A função *save*() é ativada quando o objeto *meu_loader* identificar que os dados do objeto persistente devem ser armazenados em memória secundária. A função *save*() recebe como parâmetro uma referência ao objeto em questão, através do qual ativa a função de gravação para o objeto correspondente. No SGBA, cada objeto bilhete de anormalidade tem necessariamente um objeto *meu_loader* associado.

- A função *load*() do objeto *meu_loader* realiza o inverso da função *save*(), isto é, carrega da memória secundária os dados relacionados ao objeto persistente, para que este retorne ao estado ativo. A cada objeto persistente é associado um identificador único, que a função *load*() utiliza para localizar o objeto inativo antes de colocá-lo em estado ativo. A função *load*() é ativada quando um objeto no estado inativo recebe uma invocação.

- Após a conclusão dos passos anteriores, deve-se instanciar o objeto *meu_loader* no servidor que realiza a criação dos objetos persistentes. No SGBA o objeto *meu_loader* é criado pelo objeto fábrica, sendo associado por herança aos objetos bilhete de anormalidade. Esta especialização por herança os transforma automaticamente em objetos persistente.

Todo o controle de passagem ao estado passivo e restauração é realizado pelo Orbix. Por exemplo, se os dados de um objeto estão em memória secundária e o objeto encontra-se em estado inativo, e então uma invocação é endereçada a ele, se o objeto não é persistente, isto provoca a ocorrência de uma exceção. Caso o objeto seja persistente, ele possui um *object loader* associado, então a função *load*() é ativada, fazendo com que os dados do objeto sejam automaticamente restaurados, retornando-o para o estado ativo. A invocação é portanto honrada, de forma transparente ao objeto cliente.

5. CONCLUSÃO

Neste artigo descrevemos algumas fases da modelagem e implementação de uma aplicação de gerência distribuída para gerência de bilhetes de anormalidade. Foram apresentados os requisitos do sistema, e o seu modelo de objetos. Apresentou-se ainda de que maneira o sistema foi implementado utilizando-se a arquitetura CORBA.

O SGBA foi implementado no sistema operacional Solaris, tendo se utilizado a linguagem de programação C++. As interfaces homem máquina foram desenvolvidas em Motif. A plataforma de distribuição CORBA utilizada no desenvolvimento foi o Orbix da IONA. A performance do sistema é satisfatória, não havendo consumo de recursos excessivos nas máquinas envolvidas, e apresentando bom desempenho nas interações.

A implementação do SGBA foi considerada um sucesso, existindo planos de integrá-lo a uma aplicação de gerência de falhas em ambiente TMN (*Telecommunication Management Network*), de modo que certos alarmes emitidos pelos equipamentos gerenciados provoquem a criação automática de bilhetes de anormalidade.

6. REFERÊNCIA BIBLIOGRÁFICA

1. [TERP87] Terplan, Kornel. Communication Networks Management. --New Jersey, Prentice-Hall. 1987. 595p.

2. [OMG91] OMG Document Number 91.12.1. The Common Object Request Broker: Architecture and Specification. Revision 1.1. 1991. 174p

3. [RUMB94] Rumbaugh, James & Blaha, Michael & Premerlani, William & Eddy, Frederick & Lorensen, William. Modelagem e Projetos Baseados em Objetos.
Tradução de Dalton Conde de Alencar. -- Rio de Janeiro, Editora Campus, 1994. 652p.

4. [CEFET-TELEBR96] Guapo, Marco Antônio e Penna, Manoel Camillo. Concepção e Implementação de uma Aplicação de Gerência de Bilhetes de Anormalidade em Ambiente CORBA. Relatório Técnico 003/96. Termo de Cooperação CEFET-PR/CPqD-TELEBRÁS, Novembro 1996.

5. [OMG95] OMG Document Number 95-3-31. CORBAservices: Common Object Services Specification. Revision Edition, March 31, 1995.

6. [IONA95a] IONA Technologies Ltd. Orbix 2 Programming Guide. Release 2.0 November 1995.

7. [IONA95b] IONA Technologies Ltd. Orbix2 Reference Guide. Release 2.0 Movember 1995.