RecDin: Uma Infra-estrutura para Reconfiguração Dinâmica de Replicação no FT-CORBA[1]

Lau Cheuk Lung[1], Fábio Favarim[2], Giuliana Teixeira Santos[1]

[1]Programa de Pós-Graduação em Informática Aplicada - PPGIA
Pontifícia Universidade Católica do Paraná - PUCPR

[2]Departamento de Automação e Sistemas – DAS
Universidade Federal de Santa Catarina – UFSC

{lau, giuliana}@ppgia.pucpr.br, fabio@das.ufsc.br

Abstract. *The fault-tolerance provided by the FT-CORBA is basically static, that is, once defined the fault-tolerance properties of a replicated group, these cannot be modified in runtime. A support for dynamic reconfiguration of replication would be highly advantageous in the FT-CORBA architecture to fulfill these requirements. This support makes possible the implementation of mechanisms for adaptive fault-tolerance enabling the FT-CORBA to adapt to the changes that can occur in the execution environment. In this paper, we propose a set of extensions, in the form of interfaces and service object implementations in the FT-CORBA architecture enabling it to support dynamic reconfiguration of replication.*

Resumo. *A tolerância a faltas oferecida pelo FT-CORBA é basicamente estática, isto é, uma vez definida as propriedades de tolerância a faltas de um grupo replicado, estas não mais podem ser modificadas em tempo de execução. Um suporte para reconfiguração dinâmica de replicação seria altamente vantajoso na arquitetura FT-CORBA para cobrir estas necessidades. Este suporte possibilita a implementação de mecanismos para tolerância a faltas adaptativa como forma do FT-CORBA de se adaptar as mudanças que podem ocorrer no ambiente de execução. Neste artigo, propomos como principal contribuição um conjunto de extensões, na forma de interfaces e implementações de objetos de serviço na arquitetura FT-CORBA para que este suporte reconfiguração dinâmica de replicação.*

Palavras chave: *tolerância a faltas, reconfiguração dinâmica, sistemas adaptativos, QoS, CORBA*

1. Introdução

A tolerância a faltas (TF) em sistemas distribuídos é vista hoje como uma qualidade de serviço (QoS) de uma aplicação [Zinky, 1997]. O papel da tolerância a faltas é garantir o contínuo funcionamento de um serviço mesmo que ocorram falhas parciais no sistema distribuído. Dentro deste contexto, existe um grande número de aplicações que podem usufruir da tolerância da faltas para melhorar a confiabilidade dos seus serviços. Por exemplo, serviço de governo eletrônico e/ou gestão pública, computação em grade, comércio eletrônico, serviços *web*, entre outros.

[1] Parcialmente financiado pelo CNPq Projeto Universal 019/2004 Nro. 481523/2004-9 e Fundação Araucária FA-6651/04.

O suporte de tolerância a faltas para aplicações desenvolvidas segundo o modelo de objetos distribuídos CORBA (*Common Object Request Broker Archtecture* [OMG, 2002]) é especificado segundo o padrão OMG FT-CORBA (*Faut-Tolerant CORBA* [OMG, 2000]). Nessa especificação é definido um conjunto de objetos de serviço essenciais para o desenvolvimento de aplicações confiáveis em sistemas abertos. A tolerância a faltas no CORBA é obtida através de técnicas de replicação de objetos, técnicas de detecção de falha e recuperação de falhas. A especificação apresenta um conjunto de interfaces e protocolos que podem ser separados em três módulos: *Gerenciamento de Replicação* (SGR); *Gerenciamento de Falhas* (SGF) e *Gerenciamento de Recuperação* e *Logging* (SLR); além das definições para interoperabilidade próprias da arquitetura CORBA.

O uso de técnicas adaptativas em sistemas tolerantes a faltas é recente [Kim, 1990, Cukier, 1998, Chen, 2001] e suas aplicações têm trazido benefícios no gerenciamento de recursos dos sistemas computacionais. O princípio da abordagem adaptativa é fornecer mecanismos ao sistema tolerante a faltas para obter informações do ambiente em que está executando, e assim se adaptar a suas variações. Adaptações do sistema tolerante a faltas através de diferentes configurações de replicação, visam permitir que se mantenham os níveis de confiabilidade e disponibilidade do sistema, alocando somente os recursos necessários para a obtenção dos requisitos desejados. A forma usual para dar suporte à adaptação é a implementação de mecanismos que possibilitem a reconfiguração dinâmica de sistema.

A tolerância a faltas oferecida pelo FT-CORBA é basicamente estática, isto é, uma vez definida as propriedades de tolerância a faltas de um grupo replicado, estas não mais podem ser modificadas em tempo de execução. Além disso, o sistema não pode ser reconfigurado, por exemplo, para trocar uma técnica de replicação por outra. Um suporte para reconfiguração dinâmica de replicação seria altamente vantajoso na arquitetura FT-CORBA para cobrir estas necessidades. Este suporte possibilita a implementação de mecanismos para tolerância a faltas adaptativa como forma do FT-CORBA de se adaptar às mudanças que podem ocorrer no ambiente de execução. Essas mudanças têm relação, por exemplo, com a freqüência de ocorrência de falhas, tipo de falta ou a carga de tarefas dos componentes replicado com intuito de balanceamento de carga.

O LCMI/DAS/UFSC e o LabSiD/PPGIA/PUCPR têm trabalhado ao longo dos últimos anos na pesquisa e desenvolvimento de soluções para aplicações com requisitos críticos, em ambientes abertos [Fraga 2001]. Um dos projetos, é a ferramenta *GroupPac* (http://grouppac.sourceforge.net), uma implementação completa das especificações FT-CORBA. O GroupPac [Bessani, 2004, Borusch, 2005, Bessani, 2005] é a única implementação em Java dessas especificações disponível como *software* livre.

Neste artigo, apresentamos como principal contribuição um conjunto de extensões, na forma de interfaces e implementações de objetos de serviço [OMG, 1997], na arquitetura FT-CORBA para que este suporte reconfiguração dinâmica de replicação. O uso desse suporte é totalmente transparente à aplicação. Além disso, o modelo apresenta soluções práticas para integrar requisitos de Qualidade de Serviço (QoS – *Quality of Service*) [Zinky, 1997] que devem nortear a seleção da reconfiguração de técnicas de replicação. Desta forma, diferentes níveis de QoS podem ser especificados, visando atender diferentes requisitos de tolerância a faltas. Incluímos no *RecDin* mecanismos que identificam a

necessidade de reconfigurar e de efetivar mudanças no sistema visando atender os requisitos de QoS sem que aspectos como desempenho e estabilidade sejam duramente comprometidos.

Este artigo está organizado da seguinte forma: a seção 2 faz uma introdução sobre as especificações FT-CORBA; a seção 3 apresenta alguns conceitos sobre tolerância a faltas adaptativa. A seção 4 apresenta a proposta de um arquitetura RecDin implementada no *GroupPac*. Na seção 5 são apresentados alguns aspectos de implementação, e na seção 6 são apresentados alguns resultados, na seção 7 apresentamos alguns trabalhos relacionados. Finalmente, na seção 7 o artigo é concluído.

2. As Especificações FT-CORBA

A arquitetura FT-CORBA (implementada no GroupPac [Bessani, 2004, Borusch, 2005]) possui três módulos básicos: *Gerenciamento de Replicação* (SGR); *Gerenciamento de Falhas* (SGF) e *Gerenciamento de Recuperação* e *Logging* (SLR); além das definições para Interoperabilidade próprias da arquitetura CORBA. Cada módulo é composto por uma coleção de serviços (Figura 1). O SGR é responsável pelo gerenciamento de grupo, exercendo um controle dinâmico, nas entradas e saídas (normal ou por falha) de objetos replicados. Para isso, o SGR tem o auxílio do *Serviço de Gerenciamento de Falha*s. No SGF são definidas as interfaces dos serviços de detecção de falhas, notificação de falhas e análise de falhas. Estes três serviços são responsáveis pelo monitoramento e notificação de falhas de objetos de grupo. Por fim, no *Serviço de Gerenciamento de Recuperação e Logging* são definidos os mecanismos para transferência de estado de objeto e recuperação de réplicas faltosas.

O Serviço de Gerenciamento de Replicação é o responsável pelo gerenciamento de grupo. Esta tarefa é delegada ao seu Serviço de Gerenciamento de Grupo de Objetos, exercendo um controle dinâmico, nas entradas e saídas (normal ou por falha) de objetos replicados, de um grupo através da manutenção de uma lista atualizada de seus membros (*membership*). O objeto SGR utiliza o objeto Fábrica Genérica no processo de criação ou remoção de réplicas de um grupo. Nesse processo, o objeto Fábrica Genérica interage com os objetos fábrica locais (representante local do primeiro) para a criação ou remoção de réplicas nas diferentes estações de um sistema distribuído. Além disso, temos o Serviço de Gerenciamento de Propriedades onde são definidas e manipuladas as propriedades de TF de cada grupo de objetos sob o controle do Serviço de Gerenciamento de Replicação. As propriedades definem, basicamente, como cada grupo deve ser gerenciado e controlado pelos serviços do FT-CORBA. Por exemplo, uma das informações mantidas no gerenciamento de propriedades é a definição da técnica de replicação implementada em um grupo, que registra se o mesmo pode funcionar como [OMG, 2000]: Replicação Passiva Fria, Replicação Passiva Quente e Replicação Ativa [Schneider, 1990].

No Serviço de Gerenciamento de Falhas, a detecção (ou monitoramento) de falhas é dividida em três níveis: objeto, processo e *host*. Os detectores de falhas nos três níveis citados são baseados em mecanismos de *timeout*. O detector de falhas em nível de *host* é mantido replicado no sentido de garantir a continuidade do serviço mesmo na presença de falhas parciais. O Serviço Notificador de Falhas é também replicado e tem a função de enviar mensagens de notificação ao gerente de replicação, a partir dos registros de falhas

enviados pelos detectores de falhas dos três níveis. Esses registros são necessários ao Serviço de Gerenciamento de Replicação para atualização das listas de membros.

O Serviço de *Logging* e Recuperação inclui, basicamente, três funções: registrar (fazer uma cópia) as requisições recebidas pelo primário (Objeto Servidor 1), atualizar o estado das réplicas que se juntam ao grupo e recuperar réplicas faltosas. Este serviço atua nos eventos de falha do objeto primário e na inclusão de uma nova réplica no grupo. Por exemplo, em um evento de falha do primário o Serviço de Recuperação envia ao novo primário as requisições enviadas ao primário anterior a partir do último *checkpoint*.

Todas as comunicações entre os objetos são através do ORB e, caso necessário, com o auxílio do sistema de comunicação de grupo. O sistema de comunicação de grupo fornece ao middleware CORBA os mecanismos de comunicação de grupo confiável para o suporte às técnicas de replicação – oferecendo alguma garantia de atomicidade e ordem na entrega das mensagens. Nos itens subseqüentes, é apresentado em mais detalhes, cada um desses objetos de serviço da arquitetura de tolerância a faltas CORBA.

3. Tolerância a Faltas Adaptativa

A tolerância a faltas convencional em sistemas distribuídos tem sido provida através da combinação de redundâncias de *software* e *hardware*, que na maioria das vezes são configuradas estaticamente. Considerando-se as características dinâmicas dos sistemas distribuídos atuais que aumentam em escala dia a dia, a noção fixa e pré-estabelecida na coordenação destes modelos de redundância torna os mesmos, ao menos em parte, ineficientes. Mecanismos de redundância estáticos incorrem também em um alto custo, pois os recursos precisam ser pré-alocados de tal forma a atender ao nível mais crítico de falha (pior caso) do sistema.

Um *software* adaptativo é usualmente visto como aquele que pode ter a sua configuração modificada no sentido de responder às mudanças no seu ambiente de execução. As alterações de configuração baseadas em tolerância a faltas adaptativa podem ser conduzidas, por exemplo, por mudanças nos padrões de comunicação de um sistema distribuído, na freqüência de ocorrências de falhas parciais, tipos de faltas, ou mesmo por novas necessidades da aplicação [Hiltunem e Schlichting, 1996]. Segundo [Kim, 1990], tolerância a faltas adaptativa é conseguida com mecanismos que satisfaçam requisitos de tolerância a faltas variáveis, dinamicamente, através da utilização eficiente (e adaptativa) de uma quantidade limitada e variável de recursos de processamento redundantes. Um *software* adaptativo pode envolver a troca de algoritmos, em tempo de execução, para atender as mudanças do ambiente [Chen et al., 2001].

O objetivo da tolerância a faltas adaptativa é permitir que um sistema mantenha seu nível de confiabilidade, através de procedimentos de adaptação e reconfiguração, face a variações do seu ambiente de execução (por exemplo, de amistoso para hostil ou vice-versa) ou a mudanças nas políticas de confiabilidade. Com o uso de políticas estáticas, o nível de confiabilidade fornecido pelo sistema será sempre limitado aos recursos redundantes alocados. Com um nível de confiabilidade configurável os recursos disponíveis podem ser melhor utilizados, de maneira a alocar somente os recursos necessários para

obter a confiabilidade desejada. Isso reduz também o custo sem que o desempenho do sistema seja afetado.

4. RecDin: Reconfiguração Dinâmica no FT-CORBA

No modelo adaptativo, a idéia principal é que o programador possa especificar requisitos de qualidade de um serviço, definindo os níveis desejados de disponibilidade e de desempenho deste. Para isso, foi necessário estender as especificações FT-CORBA (seção 2) da OMG incluindo novas interfaces IDL e mecanismos para reconfiguração dinâmica de replicação. Na arquitetura FT-CORBA, mostrada na figura 1, foram adicionados os serviços de *Gerenciamento de Adaptatação* e *Gerenciamento de QoS* que selecionam as configurações necessárias para atender os requisitos especificados. Esses serviços podem também ser replicados, tendo em vista a possibilidade de replicação do Gerenciador de Replicação. O sistema adaptativo é sensível ao número e freqüência de falhas nos grupos que levam o serviço a se distanciar das expectativas de QoS do usuário da aplicação.

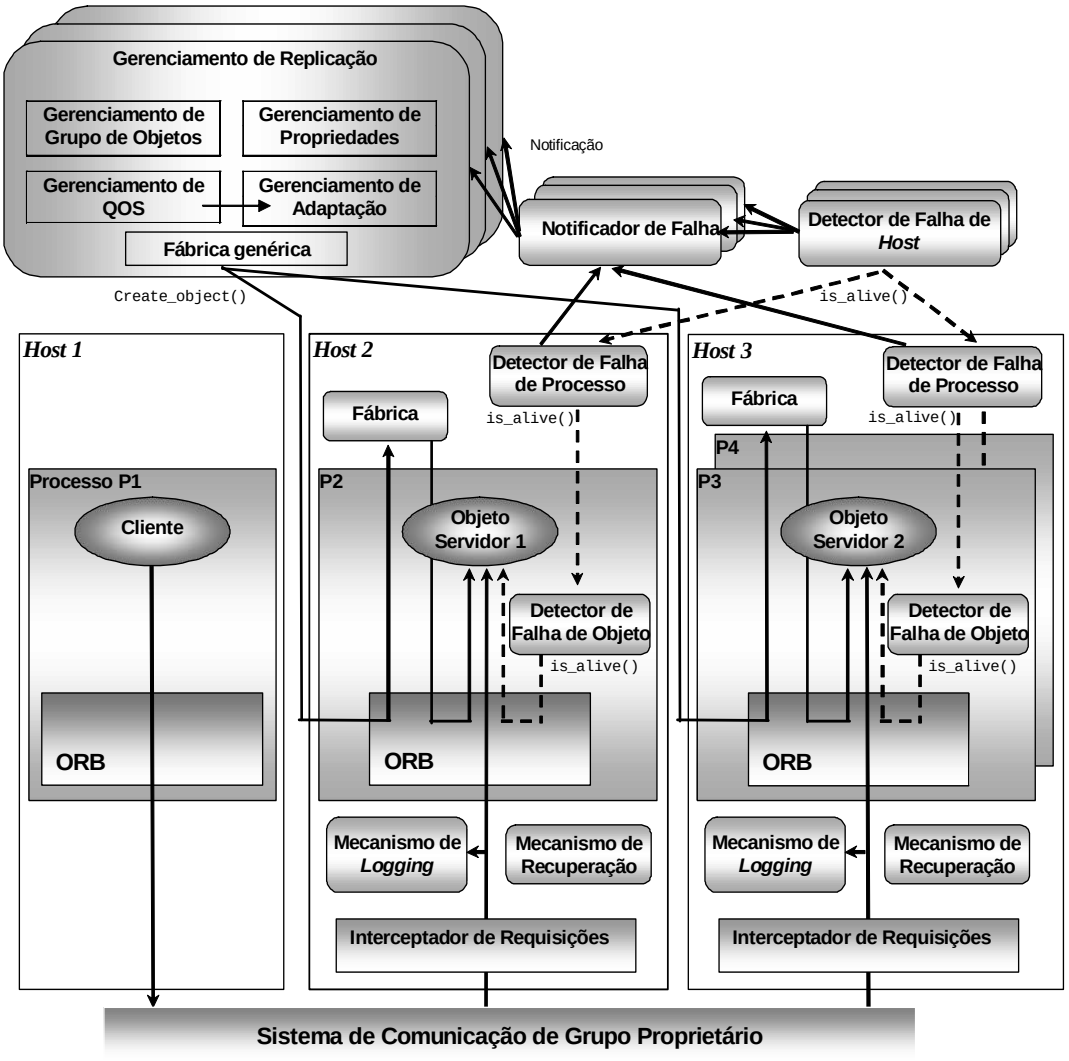

Figura 1. Arquitetura de tolerância a faltas Adaptativa CORBA.

No momento em que o sistema está sendo reconfigurado, as réplicas ficam em um estado suspenso. Neste estado, os Interceptadores de Requisições, implementados no *Grouppac*, são responsáveis por adiar o atendimento das requisições efetuadas pelos clientes e mantê-las em espera no sistema até que este esteja apto a voltar a executar requisições, garantindo desta forma a consistência do sistema [Kramer e Magee, 1988]. Com isto, nenhuma requisição é perdida, aprimorando assim a tolerância a faltas do sistema.

A seguir apresentamos em detalhes as extensões efetuadas na arquitetura FT-CORBA implementadas na arquitetura *RecDin*.

GR – *Gerenciamento de Replicação*

As mudanças efetuadas no Gerenciador de replicação foram feitas de tal forma que todas as funcionalidades referentes às especificações FT-CORBA que já haviam sido seguidas fossem mantidas. A inserção dos serviços *Gerenciamento de Adaptação* e *Gerenciamento de QoS* no módulo *Gerenciamento de Replicação* permite que a replicação desses objetos de serviço esteja implícita na replicação desse módulo, fazendo com que o modelo do sistema não fuja dos padrões adotados pelas especificações FT-CORBA.

GA – *Gerenciador de Adaptação*

A função do Gerenciador de Adaptação é manter a configuração do sistema de acordo com o nível de QoS estabelecido pela aplicação. Alguns aspectos como o número de réplicas e a técnica de replicação a ser utilizada pelo grupo fazem parte dessa configuração, que está diretamente relacionada a um estado pertencente a uma máquina de estados definida pelo programador através do serviço de Gerenciamento de QoS.

Muitas vezes, manter o nível de QoS desejado pela aplicação implicará na reconfiguração do sistema. O Gerenciador de Adaptação é o responsável por efetuar estas mudanças e estas podem implicar na implantação de novas réplicas, troca da técnica de replicação, definição do membro primário (caso a técnica de replicação necessite) e mudança no intervalo de monitoramento de faltas (dos objetos, processos e/ou sítios) e de *checkpoint* das réplicas. Também, existe a possibilidade de troca da réplica primária (nas replicações passiva e semi-ativa [Powell, 1991]) caso a atual esteja apresentando desempenho abaixo do esperado (balanceamento de carga). Por exemplo, a máquina em que está executando o primário está sobrecarregada com outras tarefas ou mesmo por negação de serviço (*Denial of Service*).

Para trabalhar de forma dinâmica, o Gerenciador de Adaptação monitora o SGR (Serviço Gerenciador de Replicação) para detectar eventuais falhas que podem fazer com que o sistema não atenda mais ao nível de QoS correntemente especificado, exigindo assim a reconfiguração do sistema. O Gerenciador de Adaptação também mantém o Gerenciador de QoS informado em relação ao status atual do sistema. Estas informações dizem respeito à quantidade de réplicas, qual a técnica de replicação está sendo utilizada no momento, os intervalos de monitoramento, o intervalo *checkpoint* utilizado e as estatísticas da ocorrência de falhas e desempenho do servidor primário (este dado pode ser fornecido pelos clientes).

GQS – *Gerenciador de Qualidade de Serviço*

O Gerenciador de QoS é uma interface na qual o programador/administrador especifica os requisitos mínimos de QoS desejados (Figura 4). Estes requisitos podem ser

especificados dinamicamente através de uma máquina de estados composta de estados e transições (Figura 2). Cada estado representa um nível de QoS com uma configuração desejada para o sistema, tal como: número de réplicas; técnica de replicação; e os intervalos de monitoramento de falha e *checkpoint* (no caso da replicação passiva).

Uma transição entre dois estados é composta por várias condições de transição, cada uma com um grau de prioridade diferente, garantido que não mais de uma condição seja satisfeita em um mesmo instante de tempo. Uma transição entre dois níveis de QoS é acionada quando o nível de QoS atual não atende mais aos requisitos de confiabilidade desejados. Esta mudança de nível de QoS pode ocorrer com a detecção de falha de uma réplica, detecção de falha no Gerenciador de Replicação ou até mesmo pela ausência de falhas em um certo intervalo de tempo. A figura 2, mostra uma possível configuração da máquina de estados com três níveis de transição, outras combinações podem ser tentadas através de inserção de novos estados.

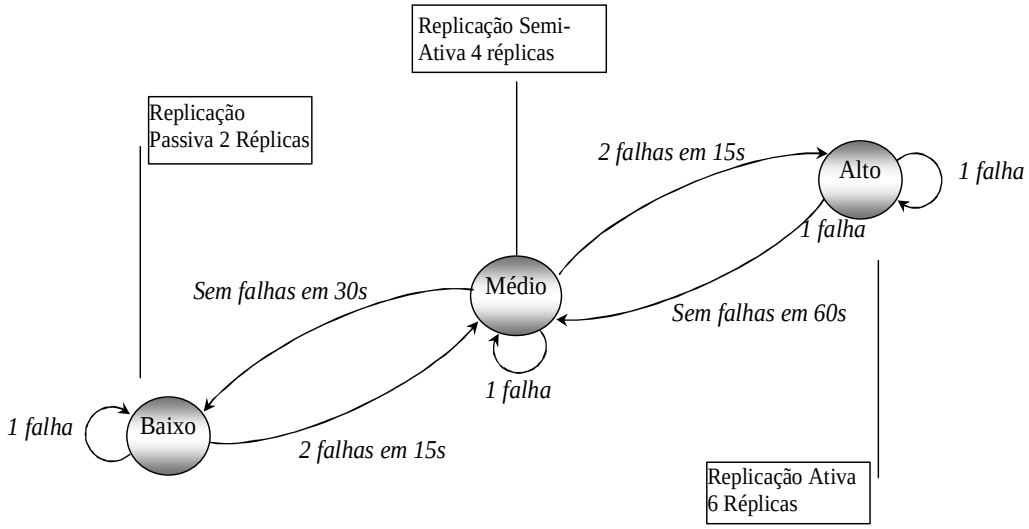

Figura 2. Máquina de estados do Gerenciador de QoS.

Na ocorrência de uma transição, a reconfiguração dinâmica do sistema será feita pelo Gerenciador de Adaptação de acordo com os novos requisitos de QoS. No caso de necessidade de troca do membro primário (na replicação passiva [Budhiraja, 1993]) ou mudança da técnica de replicação, as invocações que forem feitas, enquanto o sistema estiver em transição, ficarão em *loop* dentro do sistema até que este esteja apto a recebê-la e enviar ao servidor replicado. Uma vez definida a nova configuração a ser instalada, o Gerenciador de Adaptação acessa o objeto de serviço Gerenciador de Propriedades (Figura 1) para adicionar as novas propriedades de tolerância a falta do grupo replicado em questão. Todas as propriedades de TF passam a poder ser modificadas dinamicamente, fugindo um pouco do que está definido na especificação. Por fim, quando as novas propriedades de TF tiverem sido atualizadas o Gerenciador de Grupo de Objetos atua no sistema no sentido de implementar a nova configuração de replicação. Neste estado transitório, nenhuma requisição cliente é atendida pelo grupo servidor.

O Gerenciador de QoS também é utilizado pelo Gerenciador de Adaptação para informar o programador do status atual do sistema em um determinado momento.

5. Implementação do RecDin

A implementação do sistema RecDin foi feita usando a linguagem de programação Java, o GroupPac e o JacORB versão 1.4, o qual segue a especificação CORBA [OMG, 2000a]. O diagrama UML das classes representado pela Figura 3 exibe as classes que compõem o Gerenciamento de QoS e Gerenciamento de Adaptação.

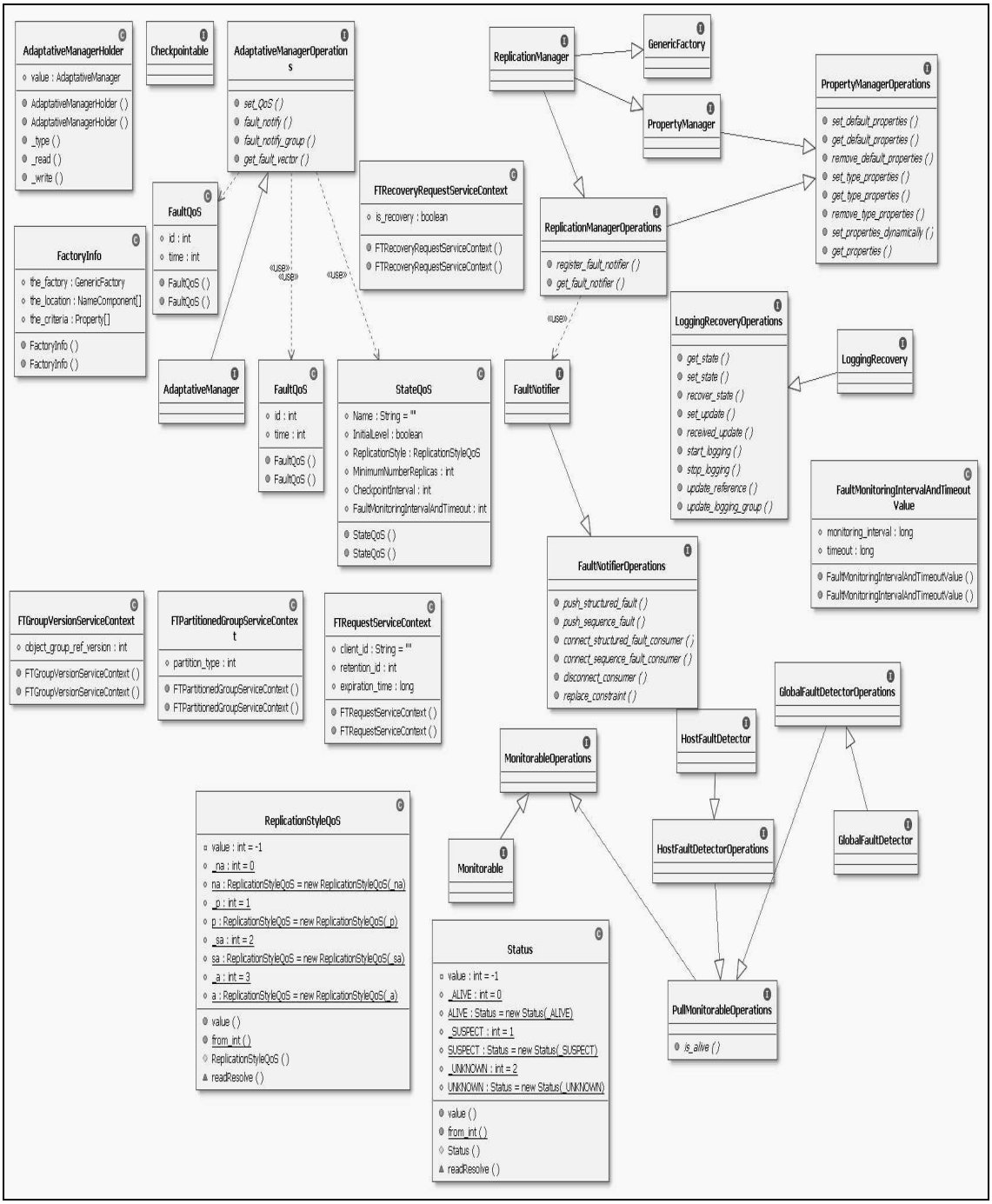

Figura 3. Diagrama UML do RecDin.

O Gerenciador de Adaptação mantém toda a estrutura adaptativa do sistema. Esta estrutura também contém o número atual de réplicas, a técnica de replicação que está sendo utilizada, entre outras informações (Figuras 3 e 4). Estes dados permitem que o sistema seja reconfigurado, possibilitando a alteração de várias propriedades de TF como número de réplicas, técnica de replicação, intervalos de *checkpoint*, monitoramento e *timeout*, estilo de *membership*, estilo de consistência, entre outras [OMG, 2000, Fraga, 2001].

```
struct Transition {
        long TransitionName;
        long Priority;
        long Faults;
        long Seconds;
        string NextState;
};
typedef sequence < Transition > TransitionVector;
enum ReplicationTypeQoS{none, passive, semi-active, active};
struct StateQoS {
         string Name;
         boolean InitialLevel;
         replicationStyleQoS ReplicationStyle;
         long MinimumNumberReplicas;
         long CheckpointInterval;
         long FaultMonitoringIntervalAndTimeout;
};
typedef sequence < StateQoS > StateQoSVector;
interface AdaptativeManager {
             void set_QoS( in StateQoSVector states);
};
```

Figura 4. Descrição em IDL da estrutura adaptativa do RecDin.

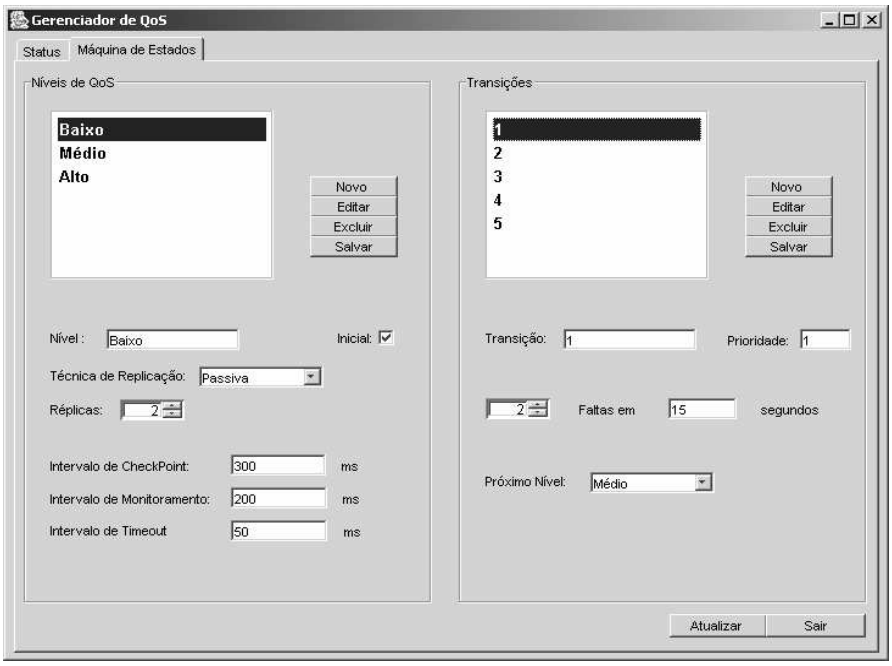

Figura 5. Interface gráfica do gerenciador de QoS.

Na figura 5 é apresentada a interface gráfica do Gerenciador de QoS. Esta interface é utilizada para especificar os requisitos de QoS do sistema, que irão compor a máquina de estados da figura 2. Informações sobre o status atual do sistema também podem ser mostradas ao usuário através da interface gráfica do gerenciador.

Fábricas Genéricas

A fabrica genérica (Figura 6) da arquitetura FT-CORBA é usada no RecDin para criar ou remover réplicas de acordo com a configuração especificada no gerenciador de QoS. Um objeto replicado é criado, usando o método `create_object`, a partir do `TypeId` que identifica o objeto a ser criado pela fábrica. Esse método retorna um identificador `FactoryCreationId` que pode ser utilizado depois caso se queira remover esse mesmo objeto, usando o método `delete_object`.

Interceptadores de requisições

Durante o processo de reconfiguração, nenhuma requisição do cliente pode ser atendida, caso contrário a aplicação poderia ficar inconsistente. Por exemplo, até mesmo a operação de leitura durante a troca do tipo de replicação, da passiva para a semi-ativa, poderia acarretar a leitura de um dado incorreto se todas as réplicas não estiverem sido sincronizadas. Além disso, é necessário garantir que todas as requisições do cliente foram respondidas antes do início da reconfiguração, deixando o sistema em um estado consistente [Kim, 1990]. Para isso, usamos os interceptadores do CORBA para fazer esse controle de acesso. O interceptador funciona como mecanismo de *plug-in* que pode ser atrelado tanto no lado cliente como no servidor. Existem várias funções possíveis para esse mecanismo [Fraga, 2001], no nosso caso, ele é utilizado para impedir a execução das requisições clientes durante os períodos de reconfiguração.

```
module FT {
   interface GenericFactory {
       typedef any FactoryCreationId;

       Object create_object(in TypeId type_id,
                            in Criteria    the_criteria,
                    out FactoryCreationId factory_creation_id)
              raises (NoFactory, ObjectNotCreated,
                      InvalidCriteria, InvalidProperty,
                      CannotMeetCriteria);

       void delete_object(in FactoryCreationId
                          factory_creation_id)
            raises (ObjectNotFound);
    };
 };
```

Figura 6. Descrição em IDL da fábrica de genérica.

A figura 7 apresenta uma reconfiguração de replicação (de replicação passiva para replicação ativa) em que o cliente envia uma requisição M2. Na primeira tentativa, a requisição M2 é enviada para o Servidor 1 (réplica primária), o interceptador deste verifica se o sistema está em estado de reconfiguração, caso afirmativo, uma exceção é retornada para o Cliente. Para garantir a transparência no lado cliente (ele não precisa receber essa exceção), o interceptador do lado cliente captura essa exceção e faz novas tentativas, até

que perceba que um novo primário foi promovido (caso a técnica de replicação seja mantida) ou que a técnica de replicação foi trocada. Neste caso, o interceptador Cliente obtém a nova referência de grupo (IOGR – *Interoperable Object Group Reference* [OMG, 2000, Fraga, 2001]), e é avisado pelo Gerenciador de Replicação que a técnica de replicação atual é a abordagem ativa.

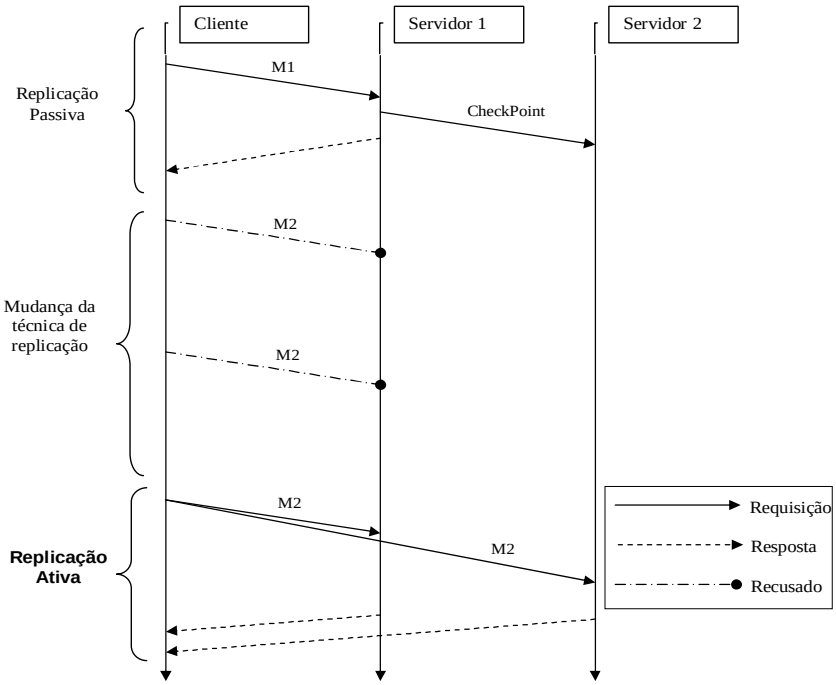

Figura 7. Ações do interceptador durante uma reconfiguração de replicação.

6. Resultados e medidas de desempenho

Para verificar o desempenho da implementação do RecDin, executamos testes[2] em uma rede local Ethernet de 100Mbps composta por 5 estações Intel Pentium IV 2.6Ghz com 256Mb de memória RAM, GNU/Linux Mandrake versão 10 e JDK 1.4. Todas as réplicas estavam executando o serviço de "Fábrica Genérica", e em uma delas foi instanciado o sistema completo (Gerente de Replicação, Gerente Adaptativo, Detector de Faltas, Fábrica Genérica, Servidor de Nomes e Servidor HTTP).

Os testes foram efetuados com a intenção de medir a sobrecarga gerada pela adição do módulo adaptativo ao FT-CORBA implementando pelo *GroupPac*. O teste efetuado foi executado no sentido de medir o tempo necessário para que o sistema se recuperasse de uma falta de *crash*, esta tendo sido gerada intencionalmente. A recuperação do sistema após uma falta consiste da soma das seguintes tarefas: (1) Detecção da falha e notificação desta ao Gerente de Replicação; (2) Dependendo da técnica de replicação, no caso da replicação passiva, se a falha foi no membro primário será necessário a eleição de um novo primário para o grupo; (3) Se o número mínimo de réplicas for maior que o número atual de réplicas

[2] Os resultados foram obtidos a partir da média de 1000 execuções.

após a falha, será necessário “lançar” uma nova réplica para o grupo faltoso, isto é, fazer a adaptação no sistema. A figura 8 ilustra o tempo gasto pelo sistema para a recuperação de uma eventual falha de *crash*. FTA-CORBA significa FT-CORBA convencional com adaptação e os valores em parêntese o número de réplicas para cada caso.

Pode-se observar no gráfico que com a adição do módulo adaptativo (RecDin), o sistema não teve nenhuma alteração em seu tempo de recuperação em relação ao FT-CORBA padrão. Isto é devido a tarefa de reconfiguração dinâmica, que consiste em alterar as propriedades do grupo de acordo com as transições entre os estados que definem o nível de QoS e em inserir ou remover novas réplicas no grupo faltoso, é completada antes mesmo que a tarefa executada pelo Gerente de Replicação no tratamento das Faltas seja finalizada. O próprio FT-CORBA já causa a perda de desempenho na criação, manutenção e remoção de réplicas. Como o RecDin usa toda infra-estrutura do FT-CORBA implica que, por exemplo, o tempo para a criação de uma réplica no RecDin é o mesmo que no FT-CORBA. Porém, a vantagem é que o RecDin toma decisões dinâmicas de maneira a usar melhor os recursos do sistema.

Além disso, pode-se observar que para um grupo com apenas 2 réplicas, o tempo gasto para a recuperação de uma falha em relação a grupos com mais réplicas (3, 4 e 5 réplicas) o tempo gasto também não aumentou muito.

Reconfigurações efetuadas pelo Gerenciador de Adaptação nos grupos por motivos de ausência de faltas em um determinado intervalo de tempo não necessitam citação, pois tratam apenas de remoções de réplicas sobressalentes e mudanças nas propriedades do grupo como: intervalo de monitoramento e *checkpoint*. Estas mudanças, mesmo que efetuadas em tempo de execução, não causam instabilidade na execução do sistema pelo grupo.

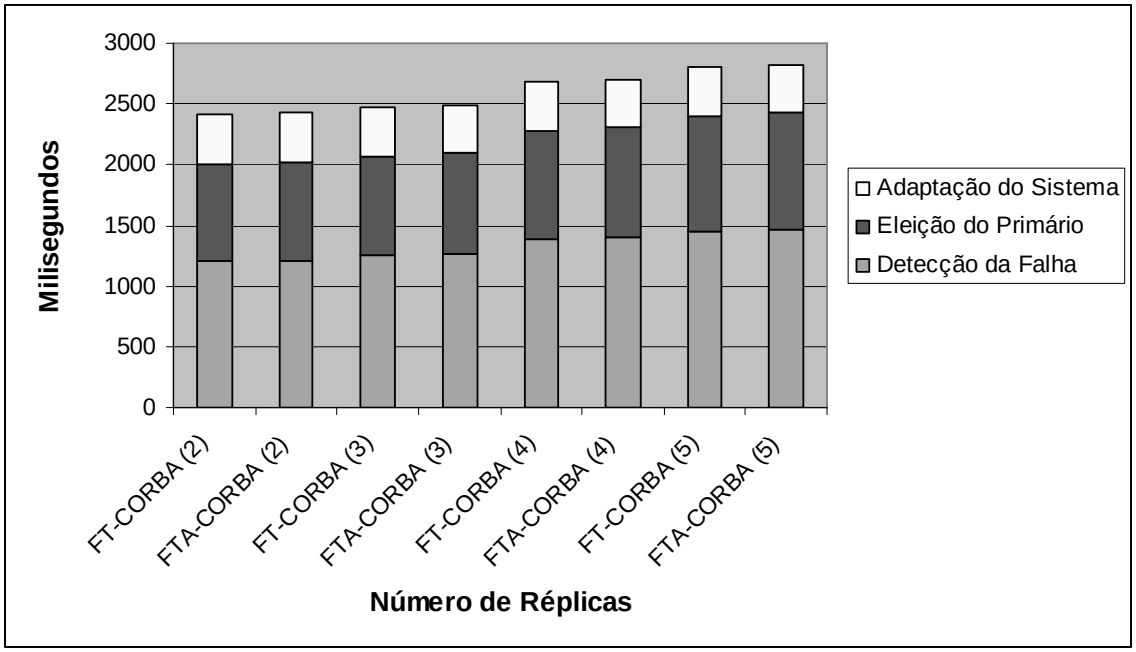

Figura 8. Gráfico de Medidas.

Também foi medido o tempo médio gasto para trocar a técnica de replicação[3]. Neste experimento foi observado o tempo médio para a estabilização entre replicação passiva e semi-ativa, foi de aproximadamente **360ms** e o tempo médio de estabilização da replicação semi-ativa para ativa foi de aproximadamente **390ms**.

7. Trabalhos Relacionados

Na literatura, existe uma quantidade considerável de trabalhos sobre a introdução de técnicas de tolerância a faltas no CORBA. Estas experiências podem ser classificadas em três abordagens: integração [Maffeis, 1995, Isis, 1995], serviço [Felber, 1998] e interceptação [Fraga, 1997, Moser, 1998, Lung, 1999]. Esses trabalhos se preocupam basicamente em manter características de portabilidade e de desempenho. Considerações sobre interoperabilidade eram limitadas, uma vez que estes trabalhos foram realizados antes da publicação das especificações FT-CORBA pela OMG.

Os trabalhos publicados em [Bessani, 2004, Borsuch, 2005, Bessani, 2005] são resultados de pesquisa das ultimas versões do GroupPac, sem ainda com as extensões apresentadas neste artigo para suportar reconfiguração dinâmica de replicações.

Em termos de introdução de mecanismos para adaptação ou reconfiguração dinâmica de replicação no modelo de objeto CORBA, existem poucos trabalhos – alguns relacionados ao modelo de componentes do CORBA [Favarim, 2003]. Dentre esses trabalhos, o mais conhecido é o AQuA (*Adaptive Quality of Service Availability*) [Cukier et al., 1998] que visa fornecer tolerância a faltas adaptativa para aplicações distribuídas. AQuA permite que os programadores de aplicações especifiquem os níveis de confiabilidade desejados, que são alcançados através da configuração do sistema em função da disponibilidade de recursos e de acordo com as falhas ocorridas. AQuA utiliza o QuO [Zinky, 1997] para especificar os requisitos de QoS em nível de aplicação e o gerenciador de confiabilidade do Proteus [Sabnis et al., 1998] para configurar o sistema em resposta a falhas e aos requisitos de disponibilidade. O Ensemble [Hayden, 1998] é usado no AQuA para fornecer serviços de comunicação de grupo. O QuO e o AQuA exigem que os requisitos de QoS sejam definidos em tempo de compilação. Como o AQuA foi desenvolvido antes da padronização do FT-CORBA, sua arquitetura não é compatível com essa especificação.

O *Chameleon* [Bagchi et al., 1998] é uma infra-estrutura adaptativa que fornece diferentes níveis de requisitos de confiabilidade para aplicações distribuídas. O *Chameleon* faz o uso de ARMORs (*Adaptive, Reconfigurable, Mobile Objects for Realiability*), objetos móveis adaptativos e reconfiguráveis para prover a confiabilidade exigida. O *Chameleon* pode, seletivamente, usar combinações de ARMORs a fim de prover diferentes níveis de confiabilidade. Além de fornecer um ambiente de tolerância a faltas altamente especializado, ARMORs também são independentes de localização – eles podem executar suas ações em qualquer nó de uma rede heterogênea. Os ARMORs também formam o mecanismo através do qual novas funcionalidades podem ser introduzidas no sistema de maneira incremental.

[3] Os resultados foram obtidos a partir da média de 100 trocas entre as técnicas de replicação e três réplicas do objeto da aplicação.

Um trabalho mais recente, é o Juggler/OGS+ [Moura, 2003], neste artigo os autores apresentam também uma proposta de integração de mecanismos de adaptação na arquitetura CORBA onde as propriedades de acordo são asseguradas pelo OGS [Felber, 1998], que é uma plataforma proprietária e não completamente conforme com o padrão FT-CORBA.

8. Conclusão

Apresentamos neste artigo um conjunto de extensões na arquitetura FT-CORBA para que esta possa ser reconfigurável dinamicamente de acordo com as condições do ambiente de execução. Esta reconfiguração é implementada fazendo uso dos padrões disponíveis no CORBA [OMG, 2002] e FT-CORBA [OMG, 2000] da OMG.

Apesar de técnicas adaptativas também serem empregadas em vários outros trabalhos para prover requisitos de tolerância a faltas, estes trabalhos não são baseados totalmente no padrão FT-CORBA. Testes realizados com o protótipo do modelo mostraram que não houve aumento nos custos de desempenho quando adicionado os mecanismos de adaptação.

Dentro da perspectiva de fornecer adaptação não somente em nível de falhas de *crash*, pretendemos adicionar suporte a outras classes da faltas.

Referências Bibliográficas

Bagchi, S. et al. (1998). The Chameleon Infrastructure for Adaptive, Software Implemented Fault Tolerance. In: *17th IEEE Symposium on Reliable Distributed Systems*. p. 261–267, West Lafayette, Indiana.

Bessani, A. N. ; Fraga, J. S. ; Lung, L. C.; Alchieri, E. A. B. *Active Replication in CORBA: Standards, Protocols and Implementation Framework*. In: VI International Symposium on Distributed Objects and Applications, 2004, Larnaca, Cyprus. Proceedings of DOA'04. Heidelberg : Springer Verlag in the LNCS, 2004. v. 3291. p. 1395-1412.

Bessani, A. N. ; Fraga, J. S. ; Lung, L. C.. Extending the UMIOP Specification for Relible Multicast in CORBA. In: 7th International Symposium on Distributed Objects and Applications, 2005, Agia Napa, Cyprus. Proceedings of DOA'05. Heidelberg : Springer Verlag in the LNCS, 2005. v. 3760. p. 662-679.

Borusch, D. ; Lung, L. C. ; Bessani, A. N. ; Fraga, J. S. Integrating the ROMIOP and ETF Specifications for Atomic Multicast in CORBA. In: 7th International Symposium on Distributed Objects and Applications, 2005, Agia Napa, Cyprus. Proceedings of DOA'05. Heidelberg: Springer Verlag in the LNCS, 2005. v. 3760. p. 680-697.

Budhiraja, N., Marzulo, K., Schneider, F. B. e Toueg, S. (1993). The Primary-Backup Approach. In: *Distributed Systems*, capítulo 4. Addison Wesley. Segunda edição.

Chen, W. K., Hiltunen, M. A., e Schlichting, R. D. (2001). Constructing Adaptive Software in Distributed Systems. In: *21st International Conference on Distributed Computing Systems*.

Cukier, M. et al. (1998). AQuA: An Adaptive Architecture that Provides Dependable Distributed Objects. In *17th IEEE Symposium on Reliable Distributed Systems*.

Favarim, Fábio. (2003). *Componentes em um Esquema de Tolerância a Faltas Adaptativa*. Dissertação (Mestrado em Engenharia Elétrica) – Centro Tecnológico, UFSC, Florianópolis. Março.

Felber, P. (1998) *The CORBA Object Group Service – A Service Approach to Object Groups in CORBA*, PhD. Thesis, École Polytechnique Fédérale de Lausanne, Lausanne.

Fraga, J. S., Maziero, C., Lung, L. C. e Loques, O. (1997). Implementing Replicated Services in Open Systems Using a Reflective Approach, In: *Proceedings of the 3th IEEE International Symposium on Autonomous Distributed Systems*.

Fraga, J. S., Lung, L. C, Westphall, C. e Montez, C. B. (2001), Suporte para Aplicações Críticas nas Especificações CORBA: Tolerância a Faltas, Segurança e Tempo Real, Mini-curso SBRC'01- SBC, Florianópolis – SC. Maio.

Hayden, M. G. (1998). *The Ensemble System*. PhD thesis, Cornell University.

Hiltunen, M. e Schlichting, R. (1996). Adaptive Distributed and Fault-Tolerant Systems. In: *International Journal of Computer Systems Science e Engineering*, 11(5):125–133.

Isis Distributed Systems Inc, IONA Technologies, Ltd. *Orbix+Isis Programmer's Guide*, Document D070-00, 1995.

Kramer, J. e Magee, J. A Model for Change Management (1988). In: *Proceddings of* IEEE International Workshop on Future Trends in Distributed Computing Systems in the '90s, p.296-300. Hong Kong.

Kim, K.H. e Lawrence, T. (1990). Adaptive Fault Tolerance: Issues and Approaches. In *Proceedings of Second IEEE Workshop on Future Trends of Distributed Computing Systems*. p. 38-46, Egypt.

Lung, L.C., Fraga, J. S., Farines, J. M., Ogg, M., Ricciardi**,** A. (1999), *CosNamingFT – A Fault-Tolerant CORBA Naming Service*, Proceeding of the 18th IEEE Symp. on Reliable Distributed Systems - SRDS'99, Lausanne, Suiça.

Maffeis, S. (1995), *Run-Time Support for Object-Oriented Distributed Programming*, Ph.D. Thesis University of Zurich. Zurich.

Moser, L. E., Melliar-Smith, P. M. P., Narasimhan, P., *Consistent Object Replication in the Eternal System*, Theory and Practice of Object Systems, 4(2): 81-92, 1998.

Moura, M. A. M., Endler, M. (2003), *Managing Adaptive Fault Tolerant CORBA Applications*. I Latin-American Dependable Computing Symposium – LADC'03, SBC, Sao Paulo – SP. Outubro de 2003.

OMG (1997). *CORBAservices: Common Object Services Specification*, OMG Document. Março..

OMG (2000). Object Management Group, *Fault-Tolerant CORBA Specification V1.0*, OMG Document: ptc/2000-04-04, Abril.

OMG (2002). *The Common Object Request Broker Architecture* v3.0. OMG Document 02-06-33.

Powell, D. (1991). *Delta-4 Architecture Guide*. Esprit II P2252, Delta-4 Phase 3.

Sabnis, C. et al. (1998). *Proteus: A Flexible Infrastructure to Implement Adaptive Fault Tolerance in AQuA*. In *7th IFIP Int. Working Conference on Dependable Computing for Critical Applications*.

Schneider, F. B. (1990). *Implementing Fault-Tolerant Service Using the State Machine Approach: A Tutorial*, ACM Computing Survey, 22(4):299-319, Dezembro.

Zinky, J. A., Bakken, D. E. e Schantz, R. E. (1997). Architectural Support for Quality of Service for CORBA Objects. *Theory e Practice of Object Systems*, 3(1):53–73.