Desenvolvimento Cooperativo de Aplicações Distribuídas Reconfiguráveis

Cidcley Teixeira de Souza *
cts@cin.ufpe.br

Paulo Roberto Freire Cunha †
prfc@cin.ufpe.br

Universidade Federal de Pernambuco
Centro de Informática
50732-970 Recife, Pernambuco

Resumo

A evolução dos middleware como a arquitetura CORBA, têm fornecido complexos mecanismos para a construção e distribuição de componentes de software. Esse componentes são desenvolvidos e disponibilizados para serem utilizados em diversas aplicações. A necessidade de evolução das aplicações exige mecanismos de controle sob esses componentes compartilhados entre aplicações, de modo que essas alterações evolutivas não afetem outras aplicações. Esse artigo propõe um modelo para coordenar a evolução de aplicações distribuídas de forma cooperativa. Nesse modelo, domínios de reconfiguração delimitam fronteiras de responsabilidade, e politicas de reconfiguração controlam as modificações em componentes compartilhados, minimizando os efeitos de modificações evolutivas entre as aplicações.

Abstract

The middleware evolution, such as CORBA architecture, has provided complex means for components construction and distribution. These components are built and availabled to be used on several applications. The needs of evolution require control mechanisms over these shared components among applications, so that these evolutionary changes do not affect other applications. This paper proposes a cooperative model to coordinate the distributed application evolution. On this model, reconfiguration domains underlying boundaries of responsibility, and reconfiguration policies controls the changes in shared components, minimizing the evolutionary changes affects among applications.

1 Introdução

Atualmente é largamente reconhecido que para se construir grandes aplicações, é necessário dividi-las em componentes que possam ser programados, compilados e testados separadamente. Sendo o sistema construído pela configuração desses componentes.

*Doutorando do Centro de Informática da UFPe (Bolsa CAPES)
†Prof. Titular do Centro de Informática da UFPe

A utilização de mecanismos de reconfiguração dinâmica em aplicações distribuídas tem sido discutido já há vários anos [KMS92, JC92, JCdP93]. Principalmente em virtude das vantagens conseguidas com essa abordagem na construção e, mas notadamente, na evolução de grandes aplicações distribuídas que possuam um ciclo de vida longo e não permanecem estáticas ao longo de sua vida operacional [SKM85].

Através de reconfiguração dinâmica, aplicações podem ser alteradas em tempo de execução, sem a necessidade da paralisação completa da mesma para a realização de modificações como: a troca de componentes software (módulos), o deslocamento de um módulo para uma outra máquina, a adição ou remoção de módulos da aplicação.

As grandes aplicações distribuídas atualmente, em virtude da característica de reuso, requerem o compartilhamento de informações e de componentes de software entre aplicações distintas. Não obstante, essa tarefa de se reusar/compartilhar componentes não é tão trivial [Gog86].

Para exemplificar as dificuldades do compartilhamento de componentes em aplicações distribuídas, seja o exemplo de um sistema simples de monitoramento de pacientes, apresentado em [MMC74] e projetado seguindo-se o modelo de configuração em [KMS89]. Esses sistema (Figura 1) modela uma unidade de tratamento intensivo de um hospital constituído por um conjunto de pacientes, modelados pelo tipo de componente Paciente. Esses pacientes são continuamente monitorados em alguns fatores, como: pressão e pulso. Para cada paciente as leituras dos equipamentos podem ser mostradas tanto diretamente na cama como na unidade de enfermagem, modelada pelo tipo de componente Enfermeira. Se qualquer fator monitorado estiver fora de limites especificados, um alarme é emitido para a central de enfermagem.

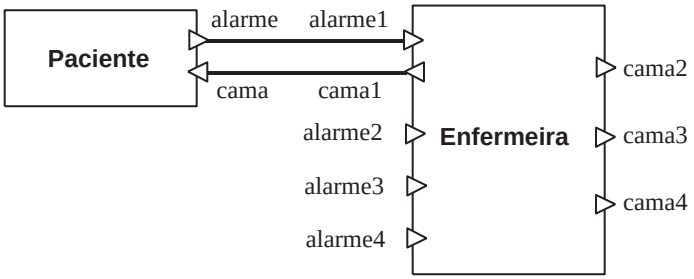

Figura 1: Sistema de Monitoramento de Pacientes.

Supondo agora a existência de uma outra aplicação hospitalar na qual o componente **Paciente** seja utilizado, por exemplo uma aplicação de controle de pagamento de despesas de seguro saúde. Supondo também que essas aplicação tenha sido desenvolvida e é controlada por uma outra equipe de desenvolvimento, Dessa forma, imaginando-se a característica de reuso, o módulo Paciente deve ser compartilhado entre as aplicações da mesma forma em que foi concebido.

Supondo que haja a necessidade de uma das aplicações realizar uma modificação nesse componente, como por exemplo a inclusão de um novo dado sobre o paciente ou a modificação de algum parâmetro de alguma operação do mesmo, isso deve ser realizado através de uma especificação de reconfiguração. Dai surge algumas questões: quem realizará de fato a alteração/reconfiguração ? Como será o impacto dessa alteração sobre a outra

aplicação na qual esse componente também faz parte ? Deve-se quebrar a característica de reuso e criar instâncias diferentes para aplicações diferentes ?

Como pode ser observado, o reuso/compartilhamento de um componente de software dificulta o controle de alterações sob esses códigos compartilhados se for considerado que os grupos de desenvolvedores estão geograficamente distribuídos.

Nesse sentido, esse artigo apresenta um modelo para o desenvolvimento de aplicações distribuídas reconfiguráveis de forma cooperativa. Nesse modelo é proposto um serviço de domínios de reconfiguração que permite a criação e manutenção de aplicações distribuídas reconfiguráveis por grupos de trabalhos distribuídos com o compartilhamento de componentes de aplicação.

Esse artigo está organizado da seguinte forma: na sessão 2 são discutidos aspectos de desenvolvimento de aplicações reconfiguráveis e os problemas decorrentes do compartilhamento de componentes por grupos de desenvolvimento distintos. Na sessão 3, são apresentados os conceitos de reconfiguração cooperativa e os elementos utilizados na construção desse conceito. O conceito de domínio de reconfiguração é tratado na sessão 4. Na sessão 5 é apresentado um protótipo de implementação do modelo proposto. Algumas conclusões são apresentadas na sessão 6.

2 Evolução Cooperativa de Aplicações

2.1 Aplicações Distribuídas Reconfiguráveis

Em grandes sistemas de software, pode não ser possível parar o sistema inteiro para realizar alguma modificação em algumas parte dele. Um grande desafio e fornecer facilidades para a realização de mudanças dinâmicas dos sistemas sem interromper o processamento das outras partes que não são diretamente afetadas. Geralmente, modificações evolucionárias são difíceis de serem acomodadas pelo fato de não serem previstas em tempo de projeto. Nesse sentido, os sistemas deveriam ser suficientemente flexíveis para permitir mudanças incrementeis arbitrárias [EvBD93].

Através de reconfiguração, aplicações podem ser alteradas em tempo de execução, sem a necessidade de se realizar uma paralisação completa da mesma. Exemplos típicos de modificações que podem ser necessárias são a troca de componentes software (módulos), o deslocamento de um módulo para uma outra máquina, a adição ou remoção de módulos da aplicação [PH91].

Seguido-se o conceito de configuração [KMS89, JC92, JCdP93, dSO98], uma aplicação distribuída consiste em um conjunto de processos interoperantes, onde cada processo é implementado por um módulo. Esses módulos são interconectados por canais de comunicação através de portas ou interfaces. Essa interconexão forma a estrutura da aplicação. A geometria da aplicação [PH91] descreve como essa estrutura é mapeada em uma arquitetura distribuída. Nesse sentido, três tipos gerais de modificações podem ocorrer na aplicação e devem ser gerenciadas pelos desenvolvedores:

- **Implementação dos Módulos:** Onde a estrutura da aplicação permanece a mesma, mas existe a necessidade de alteração em um ou mais módulos individuais.
- **Estrutura:** A estrutura lógica do sistema (topologia) é modificada. A conexão entre os componentes pode ser alterada, novos módulos podem ser incluídos e outros retirados da aplicação.

- **Geometria:** Nesse tipo de modificação o mapeamento da aplicação na arquitetura distribuída é alterado. Nesse caso os módulos que compõem a aplicação podem migrar para outros hosts sem a alteração da estrutura lógica da mesma.

Um ambiente de gerenciamento de reconfiguração deve controlar diversas atividades de modo que todas essas possíveis modificações possam ser realizadas de forma consistente. O que nem sempre é garantido, pelo fato de que essas modificações, normalmente, tenham que ser realizadas de forma manual[PH91], através de especificações de reconfiguração, o que é uma tarefa tediosa e sujeita a erros.

2.2 Relacionamento entre Componentes

Na maioria das aplicações reconfiguráveis, a reutilização de componentes (módulos) é controlada pela equipe de desenvolvimento. Ou seja, qualquer alteração necessária na estrutura de um módulo ou mesmo na configuração das aplicações, são previsíveis, pois essas alterações são realizadas pelas próprias pessoas que construíram essas aplicações.

Esse mesmo comportamento não pode ser esperado em grandes aplicações distribuídas que utilizam as novas tecnologias de distribuição de software, como os brokers de objetos distribuídos. Tecnologias como CORBA[COR98], permitem se disponiblizar aplicações e componentes de software que possam ser utilizados em diversas outras aplicações na forma de serviços.

Esses novos mecanismos de fornecimento de serviços trás consigo algumas novas características que devem ser consideradas pelos desenvolvedores de software:

- Existem serviços disponíveis na rede que podem suprir as necessidades de algumas aplicações;
- O controle dos códigos desses serviços são de responsabilidade de quem os desenvolveu;
- As aplicações que utilizem esses serviços estão sujeitas às políticas (acesso, manutenção, etc.) implementadas pelos seus desenvolvedores;

Nesse sentido, uma nova forma de se pensar o reuso de software pode ser imaginado, o reuso comportamental, no qual um serviço e reutilizado em outras aplicações, mas sem o controle direto de sua evolução. Para minimizar os danos causados pela unilateralidade do controle dos códigos distribuídos e, portanto, de sua evolução, deve ser fornecido um serviço de controle de evolução entre quem disponibiliza o software e quem o utilizará. Esse serviço deve controlar de forma bilateral alterações que possam ser danosas a aplicações com componentes de software compartilhados, como a exclusão de uma operação de um objeto.

O objetivo principal desse artigo é apresentar os conceitos necessários para se implementar um serviço de controle cooperativo de evolução de aplicações distribuídas reconfiguráveis. Onde sejam contemplados os conceitos de reconfiguração, de desenvolvimento com objetos distribuídos e de evolução cooperativa de aplicações. Na sessão seguinte os elementos necessários para a construção desse serviço são definidos.

3 Reconfiguração Cooperativa

Pode-se definir reconfiguração cooperativa como um serviço cooperativo de controle de evolução de aplicações distribuídas reconfiguráveis. Esse serviço deve permitir a evolução de aplicações distribuídas reconfiguráveis com componentes compartilhados sem que nenhuma aplicações seja afetada pela mudança/evolução desses componentes.

Para se construir um serviço de suporte reconfiguração cooperativa, alguns elementos e conceitos devem ser definidos:

- **Componente:** Um módulo de software independente de contexto e passível de ser (re)configurado;
- **Sistema:** Componentes interconectados que fazem parte de uma mesma configuração (aplicação distribuída configurável);
- **Domínio de Reconfiguração:** Um conjunto de sistemas controlados por um mesmo grupo de trabalho;
- **Políticas de Reconfiguração:** Um conjunto de regras que regem as relações entre componentes de domínios de reconfiguração diferentes, com a finalidade de garantir o controle dos componentes compartilhados entre aplicações;
- **Gerente de Domínio:** Módulo responsável por coordenar as operações de modificação nos componentes do domínio sob sua responsabilidade, respeitando o contrato de reconfiguração dos componentes compartilhados;
- **Grupo de Trabalho:** Um grupo de trabalho é formado por um conjunto de desenvolvedores que controlam a construção de sistemas dentro de um determinado domínio de reconfiguração.

Essa definição de grupo de trabalho permite se estabelecer responsabilidades sobre determinados sistemas, pois um ou mais sistemas ficam sob o controle de determinado grupo (Figura 2). Esse controle sobre um sistema é indicado pela interação direta entre os participantes do grupo no desenvolvimento e manutenção dos componentes desse sistema.

Todas as fases do projeto devem ser acompanhadas pelos membros do grupo, que tomam decisões relativas aos sistemas sob seu controle, sem a preocupação imediata de que alguns dos componentes desses sistemas possam estar sendo utilizados por outras aplicações, que, eventualmente, podem ser controladas por outros grupos de trabalho.

Fica a cargo do gerente de domínio, através das políticas de reconfiguração, restringir e negociar de forma cooperativa com os outros grupos de trabalho, as alterações necessárias em determinados componentes. Na sessão seguinte, esses mecanismos serão discutidos.

4 Domínios de Reconfiguração

O conceito de domínios de reconfiguração possibilitam se estabelecer fronteiras de gerenciamento de responsabilidade e autoridade sobre componentes. Um domínio de reconfiguração é a representação de um conjunto de aplicações reconfiguráveis sujeitas ao controle de um mesmo grupo de desenvolvimento. Todos os componentes que estão sob o controle de um mesmo grupo de desenvolvedores, e portanto, têm o comportamento previsível com

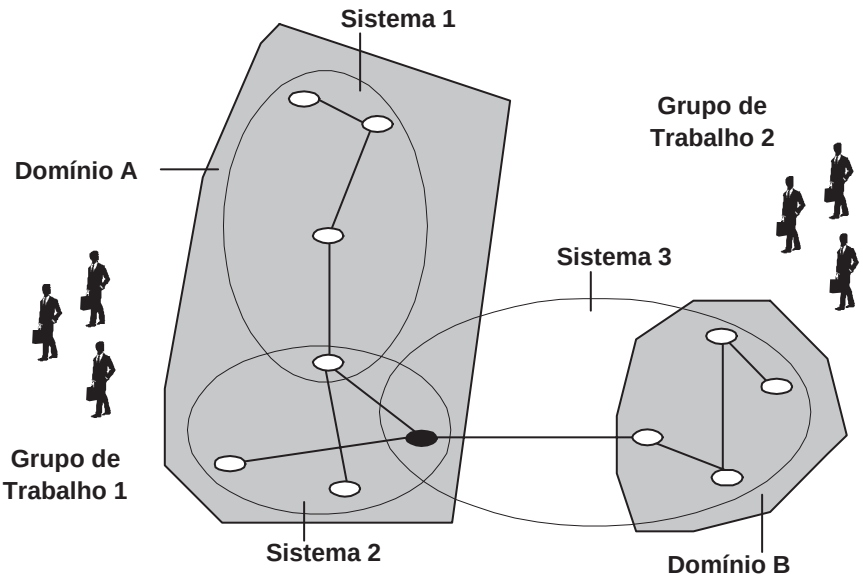

Figura 2: Domínios de Reconfiguração x Grupos de Trabalho.

relação a modificações evolucionárias, sendo essas passíveis de serem checadas, estão em um mesmo domínio.

Domínios de reconfiguração permitem a criação de regras que regem tanto a relação de dependência entre componentes em domínios diferentes como o comportamento de componentes compartilhados entre domínios distintos. O reuso de componentes, impõe um controle sobre esses objetos compartilhados entre os sistemas. A modificação de um determinado componente pode afetar severamente o comportamento de outra aplicação em um outro domínio de reconfiguração. Pode também haver uma relação entre componentes em domínio diferentes baseada na utilização de serviços de determinados componentes. Ou seja, um objeto estar relacionado a outro pelo fato de realizar a invocação de um determinado método.

Para apresentar a relação entre sistemas, domínios e grupos de trabalho, seja o seguinte exemplo: a Figura 2, apresenta um conjunto de três sistemas. Todos os componentes dos sistemas 1 e 2 são controlados pelo mesmo grupo de trabalho (Grupo de Trabalho 1). Portanto, esses dois sistemas formam um domínio de reconfiguração (Dominio1). Já o sistema 3, controlado pelo Grupo de Trabalho 2, tem um dos seus componentes, representado por uma elipse escura na Figura 2, que não está sob o controle da equipe, dessa forma, o domínio de reconfiguração para esse grupo de trabalho exclui o componente marcado, visto que quem de fato controla a implementação desse componente é o Grupo de Trabalho 1.

Dessa forma, deverá haver uma cooperação entre os dois grupos de trabalho para que as eventuais tarefas de evolução do componente compartilhado, requeridas por qualquer um dos grupos, possam ocorrer sem o comprometimento dos outros sistemas. O controle da modificação da implementação (reconfiguração) do componente compartilhado, deverá estar sujeita as restrições e ao relacionamento entre os domínios, que são concebidas considerando-se os tipos de domínios.

4.1 Tipos de Domínios de Reconfiguração

Para se oferecer um conjunto de regras de controle de reconfiguração de componentes compartilhados em aplicações distribuídas, inicialmente deve ser estabelecido um relacionamento entre os domínios que compartilham os componentes. Esse relacionamento, por sua vez, deve considerar as características de interação entre os grupos que controlam esses domínios e a relação de posse do componente compartilhado. Essas características são fornecidas através da definição de tipos de domínio de reconfiguração. Desse modo, três tipos de domínios são definidos:

- **Domínio Mestre:** São os proprietário do componente, ou seja, esse tipo de domínio controla os códigos dos componentes;
- **Domínio Privilegiado:** Esse tipo de domínio pode requisitar a realização de alterações nos componentes compartilhados. Além de que, todas as alterações realizadas pelo domínio mestre devem ser indicadas aos domínios privilegiados e essas só serão de fato executadas com a concordância dos do gerente desse domínio.
- **Domínio Não Privilegiado:** Não pode realizar nem requisitar nenhum tipo de alteração no componente. Os sistemas desse domínios estão sujeitos a aceitar as alteração realizadas nos componentes compartilhados. Sendo essas alterações apenas informadas;

De acordo com a quantidade e tipos de domínios interagindo, várias impasses podem ocorrer, como por exemplo (Figura 3), pode ser que seja necessário a realização de uma reconfiguração por causa da modificação de um método no componente em um domínio, mas um dos domínios privilegiados que utiliza esse componente não concorda com a modificação. Para que essas interações possam ocorrer, algumas políticas de reconfiguração devem ser definidas.

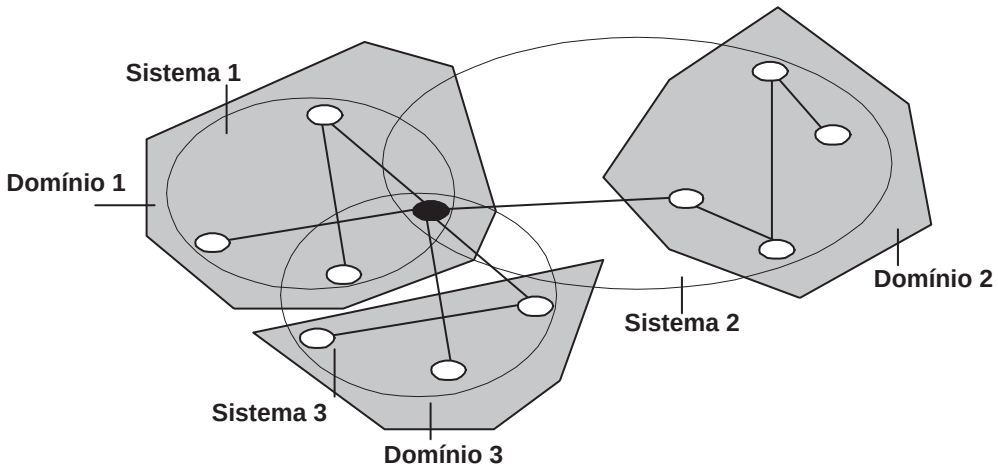

Figura 3: Impasse entre domínios.

4.2 Políticas de Reconfiguração

As interações entre os domínios de reconfiguração devem permitir que os sistemas não parem em caso de impasse entre diversos domínios. As políticas de reconfiguração são as regras que definem o comportamento do Gerente de Domínio no caso de impasses.

Nos ambientes de reconfiguração cooperativos, os desenvolvedores devem considerar que outros grupos podem estar utilizando alguns componentes distribuídos e, dependendo do tipo de alteração a ser realizada e do tipo de relacionamento entre os domínios, essas alterações devem ser controladas. O fluxograma a seguir (Figura 4) apresenta as interações entre os Gerentes de Domínios e as atitudes a serem tomadas para a resolução de impasses, onde os pedidos de reconfiguração são controlados de acordo com o tipo de domínio que a gerou, com o tipo de modificação a ser realizada e com a opinião dos domínios receptores.

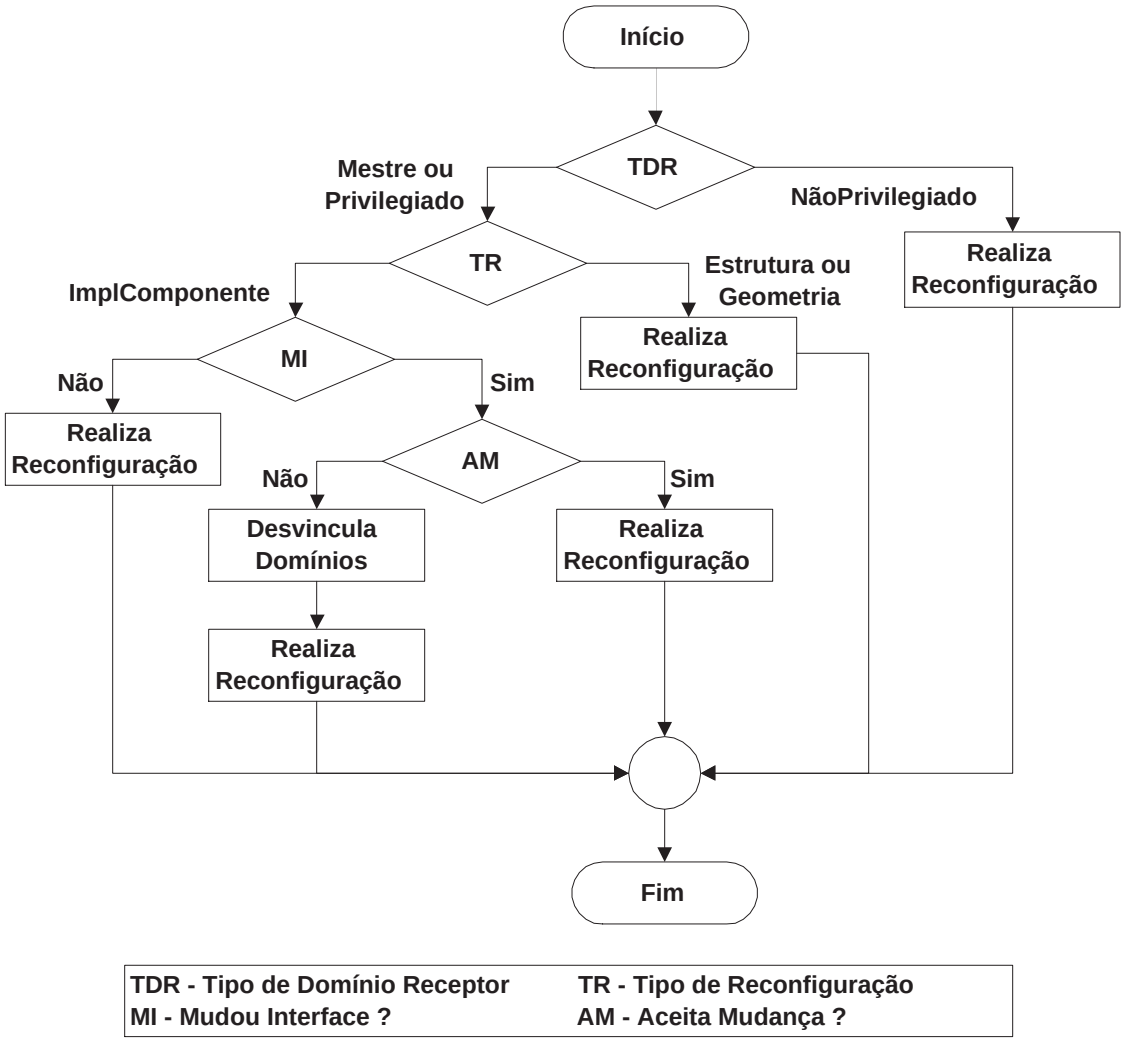

Figura 4: Interação entre domínios de reconfiguração.

Os pedidos de reconfiguração podem ser gerados apenas pelos domínios do tipo mestre ou privilegiado. Quando um pedido de reconfiguração é recebido (Figura 4), deve haver uma verificação no tipo de alteração que está sendo solicitada. Em caso de alteração de estrutura ou de geometria, essa reconfiguração pode ser implementada sem perigo de

afetar outras aplicações, mas no caso de alterações na implementação de um componente compartilhado, deve haver uma verificação se a interface do componente não está sendo modificada. Em caso de ser necessário a modificação de alguma informação na interface, os domínios que utilizam o componente devem ser consultados.

Se o domínio que estiver recebendo o pedido de reconfiguração for um domínio mestre e esse não aceitar a modificação solicitada, essa modificação não é efetivada e o domínio que realizou a solicitação fica com a opção de receber uma cópia do componente e se desligar do esquema de cooperação (não mais compartilhar/reusar o componente). Da mesma forma, se o domínio que estiver recebendo o pedido de reconfiguração for um domínio privilegiado, este poderá se desligar da cooperação não aceitando a modificação proposta.

Esse política de controle de pedidos de reconfiguração garante a autonomia do domínio mestre sobre seus componentes e também garante a cooperação entre domínios com componentes compartilhados.

5 Protótipo de Implementação

5.1 Definição da Arquitetura

A Figura 5, apresenta a arquitetura proposta para implementar o modelo de reconfiguração cooperativa. Essa arquitetura é formada por algumas camadas que encapsulam serviços específicos para o suporte ao controle das aplicações. Sua definição foi na utilização de um middleware com a tecnologia de objetos distribuído, como o CORBA. A arquitetura e formada pelas seguintes camadas:

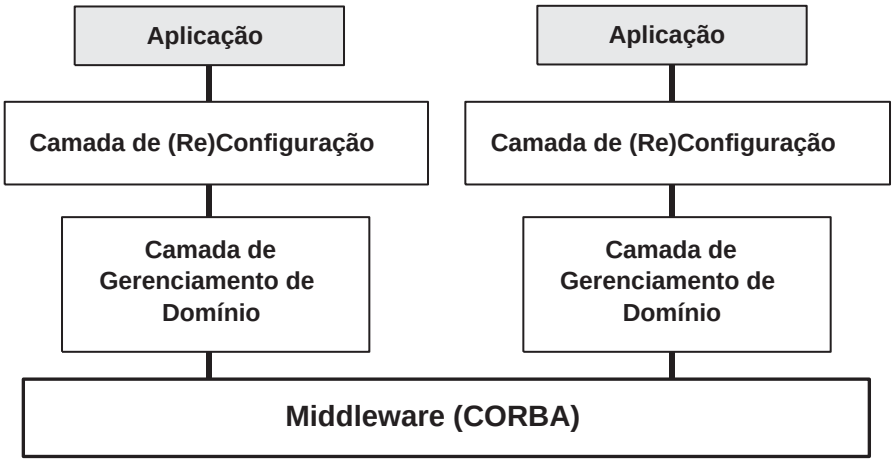

Figura 5: Arquitetura do Protótipo.

- **Camada de Reconfiguração:** Essa camada tem por finalidade tornar reconfigurável as invocações de objetos remotos. Ela é formada pelas seguintes partes:

    1. **Interceptador de Chamadas** - Realiza a interceptação das invocações de objetos remotos que são repassadas ao gerente de configuração para serem tratadas.

2. **Gerente de Configuração** - Recebe comandos de (re)configuração e os executa. Também recebe as invocações interceptadas e, através do serviço de (re)configuração, define o objeto destino da invocação, acionando o gerente de comunicação para que essa invocação seja realizada de fato.
3. **Interpretador Local** - Recebe comandos de (re)configuração a partir do teclado ou através de especificações de reconfiguração (scripts), interpreta-os e invocando o gerente de configuração para realizar as tarefas encontradas.
4. **Gerente de Comunicação** - Gerencia as invocações de objetos. Ele recebe as chamadas do gerente de configuração e cria um pedido para o objeto remoto, baseado na tecnologia do middleware.

- **Camada de Gerenciamento de Domínios:** Essa camada tem por finalidade controlar todos os fatores relativos aos domínios de reconfiguração. Ela é formada pelos seguintes elementos:

  1. **Gerente de Domínio Local** - Realiza o controle dos domínios locais (estado dos objetos, nomes locais, sistemas participantes, etc).
  2. **Controlador de Compartilhamento** - É o responsável pelo controle da evolução de aplicações de acordo com as permissões do grupo de trabalho. Serve de interface para o serviço de configuração, controlando as trocas de pedidos de reconfiguração entre domínios.

5.2 Aspectos de Implementação

Atualmente encontrasse em fase de desenvolvimento um protótipo dessa arquitetura. Para implementá-la, um ORB CORBA, o JacORB 1.1[Bro97], é a ferramenta que está sendo utilizada como middleware. Esse ORB de código aberto implementa parte da especificação CORBA 2.0 e foi totalmente escrito em Java.

Uma das dificuldade iniciais foi prover esse ORB com mecanismos de (re)configuração. Para isso, o mecanismo de interceptadores do JacORB foi utilizado. Através desse mecanismo, as invocações a objetos podem ser interceptadas, tratadas e manipuladas. A linguagem de configuração que está sendo utilizada é baseada na linguagem CL[JC92, JC96].

5.3 Considerações

Um dos problemas encontrados nesse implementação atualmente, está na redução de desempenho causada pela utilização de interceptadores de chamada, além da exigência desses interceptadores da utilização da Interface de Invocação Dinâmica (DII) para a realização das invocações a objetos. Essa mecanismo gera uma sobrecarga de comunicação que, aliada aos mecanismos de controle de domínios, tem influenciado no desempenho das aplicações. Uma forma de minimizar essas influência, foi a criação de um mecanismo que foi denominado de Invocação Dinâmica Cacheada. Esse mecanismo permite que, sempre que uma invocação dinâmica tenha que ser realizada, uma informação de log seja armazenada no cliente, se outras invocações forem realizadas para o mesmo objeto, o gerente de comunicação local faz um pedido de stub do servidor remoto ao gerente de comunicação remoto. A partir daí o cliente utilizará invocações estáticas nas próximas invocações.

6 Conclusão

Reconfiguração dinâmica é um aspecto chave na evolução de aplicações distribuídas em termos da adição de novas funcionalidade, reconfiguração de relacionamentos entre componentes e modificação de sua localização na rede física. Não obstante, a tendência atual de se disponibilizar serviços distribuídos, tem exigido dos desenvolvedores a tarefa de se preocupar com a evolução dos componentes compartilhados pelas aplicações, pois as conseqüências de alterações nesses componentes podem ser imprevisíveis.

Esse artigo apresenta um modelo de suporte ao desenvolvimento cooperativo de aplicações distribuídas reconfiguráveis. Através do modelo proposto, vários grupos de desenvolvimento poderão implementar aplicações distribuídas reconfiguráveis na qual a evolução dos componentes dessas aplicações é controlada de forma cooperativa entre os grupos que compartilham componentes comuns. Como principais contribuições desse trabalho podem ser citadas:

- A proposta de um modelo de suporte à evolução cooperativa de aplicações distribuídas reconfiguráveis;
- A definição do conceito de domínios de reconfiguração e de regras de interação entre domínios distintos;
- A descrição de uma arquitetura que implementa o modelo de reconfiguração cooperativa em ambientes CORBA.

Alguns aspectos do modelo proposta ainda devem ser analisados de forma mais precisa, como o impacto desse modelo no desempenho de grandes aplicações. Alguns outros pontos ainda devem ser estendidos, como a introdução de novas políticas de reconfiguração mais abrangentes e a criação de um contrato distribuído que possa ser renegociado entre os domínios de reconfiguração. Como trabalhos futuros um tratamento formal do conceito de reconfiguração cooperativa deve ser providenciado, além da análise da utilização do mecanismo proposto em sistemas com módulos compostos e do tratamento de questões de segurança.

Referências

[Bro97] G. Brose. Jacorb: Implementation and design of a java orb. In *Distributed Applications and Interoperable Systems*, 1997.

[COR98] *The Common Object Request Broker: Architecture and specification. V3.0*, 1998.

[dSO98] Cidcley T. de Souza and Mauro Oliveira. Abaco : Um ambiente de desenvolvimento baseado em objetos distribuidos. In *XIII Simposio Brasileiro de Engenharia de Software*, 1998.

[EvBD93] Mohammed Erradi, Gregor v. Bochmann, and Rachida Dssouli. A framework for dynamic evolution of distributed systems specifications. Technical report, Universite de Montreal, 1993.

[Gog86] J. A. Goguen. Reusing and interconnecting software components. *IEEE Computer, (Designing for Adaptaility)*, 19, 1986.

[JC92] G. R. R. Justo and P. R. F. Cunha. Programming distributed systems with configuration languages. In *International Workshop on Configurable Distributed Systems*, 1992.

[JC96] G. R. R. Justo and P. R. F. Cunha. Framework for developing extensible and reusable parallel and distributed applications. In *IEEE Conf. on Algorithms and Architectures for Parallel Processing*, 1996.

[JCdP93] G. R. R. Justo, P. R. F. Cunha, and V. C. C. de Paula. Distributed systems programming based on configuration-oriented languages. In *XIX Informatics Latin-American Conference*, 1993.

[KMS89] J. Kramer, J. Magee, and M. Sloman. Configuration support for system description, construction and evalution. In *5th International Workshop on Software Epecification and Design*, 1989.

[KMS92] J. Kramer, J. Magee, and M. Sloman. Configuring distributed systems. In *ACM Workshop on Models and Paradigms for Disttributed Systems Structuring*, 1992.

[MMC74] W.P. Myers, G.F Myers, and L.C. Constantine. Strutured design. *IBM Systems Journal*, 13(2), 1974.

[PH91] J. Purtilo and C. Hofmeinster. Dynamic reconfiguration of distributed programs. In *11th International Conference on Distributed Computing Systems*, 1991.

[SKM85] M. Sloman, J. Kramer, and J. Magee. The conic toolkit for building distributed systems. In *6th IFAC Distributed Computer Control Systems Workshop*, May 1985.