

Tópicos em Ciência da Computação IME USP

Relatório final de estudos

Aluno: Ricardo Koji Ushizaki

(riko@ime.usp.br)

Orientador: Fabio Kon

(kon@ime.usp.br)

Professora Responsável: Yoshiko Wakabayashi

10 de dezembro de 2001

1 Introdução

Este relatório final faz parte da disciplina de *Tópicos em Ciência da Computação (MAC5701)* realizada durante o segundo semestre de 2001.

O objetivo final deste relatório de estudo é apresentar uma extensão para a atual implementação do arcabouço existente de *Reconfiguração Dinâmica* do protótipo do **PSIT** (*Protótipo de Serviço de Informação sobre o Trânsito*), protótipo inicial do projeto SIDAM (*Sistemas de Informações Distribuídas para Agentes Móveis*) que permite a realização de buscas/atualizações de dados distribuídos.

A extensão foi desenvolvida em Java, utilizando Java RMI, e o código fonte e documentação podem ser encontrados na seguinte URL:

<http://www.ime.usp.br/~riko/topicos>

O resultado será apresentado nas próximas seções, mostrando como é possível aumentar as possibilidades de *Reconfiguração Dinâmica dos Componentes* de um sistema, através de substituição de implementações de componentes, estendendo as idéias dos *ComponentConfigurators*, um arcabouço

para gerenciar as interdependências e configurações dinâmicas de componentes.

2 Reconfiguração Dinâmica

A reconfiguração dinâmica de componentes distribuídos ocorre em tempo de execução, favorecendo a continuidade dos serviços oferecidos por esses sistemas. Isso é de extrema importância para sistemas críticos que não podem ser parados e reinicializados sempre que há alguma necessidade de atualização dos componentes ou até mesmo uma manutenção corretiva dos mesmos.

Para esta disciplina de Tópicos foi desenvolvido um arcabouço baseado em objetos reconfiguráveis (**Reconfigurable Objects**), cuja principal propriedade seria dar suporte à substituição de implementação desses objetos. Deste modo, é possível substituir em tempo de execução implementações do sistema por outras versões, melhorando a qualidade de serviço oferecida, ou por implementações de correção de falhas desses sistemas.

Foram desenvolvidas e implementadas interfaces com o objetivo de simular uma *Reconfiguração Dinâmica* durante a execução do **PSIT**. Este protótipo possui basicamente 4 partes principais:

InformationServers – responsáveis por armazenar informações referentes a determinados micro-domínios. Micro-Domínios é um conjunto informações localizadas, em particular, regiões da cidade de São Paulo, armazenando informações sobre o trânsito de determinada região.

LocatorServers – responsáveis por localizar qual InformationServer é responsável por determinado micro-domínios.

Clientes – buscam informações de certos Micro-Domínios, executando buscas nos *InformationServers*.

SIDAM – agente gerenciador do sistema, mantendo informações sobre quais servidores estão disponíveis no momento.

O PSIT hoje contempla uma implementação dos Configuradores de Componentes (*ComponentConfigurator*), desenvolvidos pelo Prof. Fabio Kon. Os *ComponentConfigurators* são objetos responsáveis por lidar com a parte

de reconfiguração dinâmica do ambiente mantendo informações sobre as dependências dinâmicas entre componentes durante a execução e implementar políticas de reconfiguração. Tais idéias foram estendidas pela Profa. Dilma M. da Silva no PSIT, criando o pacote *DistributedComponentConfigurator*, adicionando uso de Java RMI aos *ComponentConfigurators*.

Para que a *Reconfiguração Dinâmica* fosse possível, os *Reconfigurable Objects* fizeram uso dos Configuradores de Componentes para determinar quais dependências deveriam ser atualizadas. Na próxima seção iremos descrever mais em detalhes como isso foi feito.

3 Reconfigurable Objects

A idéia de *Reconfigurable Objects* foi criar uma interface para todo objeto possível de se reconfigurar. No caso específico do PSIT, poderíamos utilizar os servidores como exemplos de objetos reconfiguráveis (*InformationServers*, *LocatorServers* e *SIDAM*), pois são serviços cuja disponibilidade deva ser alta para atender adequadamente os clientes do sistema.

Esta interface foi criada com o nome de *ReconfigurableObject*, e atualmente dá suporte apenas à substituição de implementações desses objetos, sendo que Migração e Replicação de componentes ainda não foram abordados aqui. Possui os seguintes métodos:

```
public ROState prepareReconfiguration()
```

– prepara esse objeto reconfigurável para uma reconfiguração, encapsulando seu estado no objeto **ROState** (*ReconfigurableObject State*), uma interface criada para a exportação e importação do estado de objetos reconfiguráveis.

```
public Object replaceImplementation(String newImplementationLocation)
```

– executa a reconfiguração (substituição) desse objeto reconfigurável. Recebe o local da nova implementação, carrega uma nova instância, insere o estado obtido por *prepareReconfiguration()*, e devolve essa nova instância.

```
public void initFromROState(ROState ro)
```

– atualiza o estado atual desse objeto reconfigurável a partir de um *ROState*.

Abaixo está um trecho de código de `replaceImplementation()` aplicado a um `InformationServer` no PSIT:

```
ROState currentStateServer = prepareReconfiguration();
Class newClass = getClass().forName(newImplementationLocation);
System.out.println("Loaded new class " + newClass.getName());

Object newOb = newClass.newInstance();
if (newOb instanceof ReconfigurableObject) {
    ReconfigurableObject ro = (ReconfigurableObject) newOb;
    ro.initFromROState(currentStateServer);

    System.out.println("Implementation replaced.");
    return ro;
} else {
    throw new NotReconfigurableException("Object from class "
    + newClassName + " is not a ReconfigurableObject!");
}
```

Em particular, o arcabouço de **ReconfigurableObject** foi aplicado a um *InformationServer* no PSIT. Na próxima seção mostraremos um exemplo de como conseguiu-se substituir implementações de um `DynInformationServer`, uma implementação de *InformationServer* para objeto reconfigurável.

4 Implementação

Os *InformationServers* no PSIT atuam como servidores de microdomínios, registrando-se em um *rmiregistry* junto a um serviço *SIDAM*. Percebe-se disso uma interdependência entre esses dois servidores, explorada e gerenciada através dos *ComponentConfigurators*.

Assim, foi implementada uma classe *DynInformationServer*, que na verdade possui toda a implementação de um *InformationServer* mais a característica de um *ReconfigurableObject*.

Após sua inicialização, todo *InformationServer* deixa uma porta aberta para recebimento de comandos via *telnet*. Para esse estudo foi criado mais um comando:

replace *argumento*

onde *argumento* é o nome da nova implementação a ser substituída nesse servidor.

Mostraremos a seguir a substituição de um *DynInformationServer* por uma nova implementação. Para efeitos de simulação, estamos apenas atualizando a versão de cada implementação, ou seja, cada *replaceImplementation* irá incrementar o número da versão.

A seguinte saída é gerada quando inicializa-se o *InformationServer*:

```
Sending MDAddressSet with micro domains...
It did call rebind
InformationServerConfigurator (DynServer-conf) has been created.
InformationServer DynServer ready.
***** Socket connection on port 1078
```

Agora conectamos por *telnet* no host que este servidor está executando, (no caso porta 1078), enviando o seguinte comando:

```
replace informationServer.DynInformationServerImpl
```

Isso irá invocar o método *replaceImplementation()* já visto, que irá:

1. Guardar o estado atual.
2. Carregar a nova implementação.
3. Atualizar o *rmiregistry*.
4. Atualizar as dependências utilizando os *ComponentConfigurators*.
5. Inserir o estado na nova implementação.
6. Substituir a antiga implementação por essa nova.

Abaixo segue a resposta ao comando no terminal:

```
Implementation replaced ok!
New InformationServer description:
Implementation of DynServerImpl - version 1
```

Isso indica que a versão da implementação foi atualizada de zero para 1.

5 Problemas Encontrados

A reconfiguração dinâmica de componentes não é muito trivial, principalmente em sistemas distribuídos. Durante esse trabalho foram encontrados diversos problemas quanto ao desenvolvimento desse arcabouço.

Um deles foi a exportação e importação do estado corrente do *ReconfigurableObject*. Utilizou-se para isso objetos *Mock* que apenas mantinham referência aos atributos dos mesmos, e muitas vezes eram referências remotas a objetos, e sua serialização não daria o resultado esperado.

Outro problema foi adaptar o PSIT a atender essa nova funcionalidade. Foi necessário criar novos *eventos*, que são basicamente os comandos passados por *telnet*, e fazer o tratamento dos mesmos.

Apesar disso, foi possível aperfeiçoar o PSIT para essa reconfiguração, sem ter que modificá-lo drasticamente. Muitas modificações foram feitas apenas estendendo as funcionalidades dos objetos já existentes.

6 Conclusão

Com este trabalho, foi possível a criação de um arcabouço para a reconfiguração dinâmica de objetos.

Lembrando que para isso foi necessário estender os *ComponentConfigurators*, auxiliando na parte de gerenciar as dependências ao se executar a substituição de objetos.

A reconfiguração dinâmica de objetos mostrou ser uma área relativamente complexa, pois devemos tratar desde manter o contrato entre a nova implementação e os objetos já existentes, além de gerenciar suas interdependências e não quebrá-las com a inserção da nova implementação.

7 Referências

- PSIT

- <http://www.ime.usp.br/~sidam/software/psit>

- Site dos *ComponentConfigurators*

- <http://choices.cs.uiuc.edu/2k/ComponentConfigurator/>