Information Router: Plataforma *webservice* de comunicação entre aplicações

Pedro Silva[1], José Castro[1], e Ildemundo Roque[1]

Telbit, Tecnologias de Informação
Rua Banda da Amizade, 38 r/c Dto. 3810-059 Aveiro
{psilva, jcastro, iroque}@telbit.pt
http://www.telbit.pt

Resumo O Information Router consiste numa arquitectura genérica de interligação de aplicações recorrendo a *webservices*.
O sistema permite garantir, de uma forma fácil, a interoperabilidade e a simplificação das tarefas de configuração da comunicação entre aplicações distintas.
A arquitectura é centralizada e cria uma camada de direcções entre as várias aplicações, de forma a que as mesmas não necessitem de conhecer as características dos sistemas com as quais vão comunicar.
As possibilidades desta ferramenta vão desde o simples reencaminhamento de correio electrónico para outras formas de comunicação, como as mensagens instantâneas o *SMS* ou directamente para formato físico, até à possibilidade de dar suporte aos utilizadores de uma determinada aplicação de várias formas distintas (correio electrónico, *SMS*, mensagens instantâneas, telefone,...), permitindo que o sistema escolha de acordo com a configuração o método a utilizar.
A arquitectura foi inspirada nos sistemas do tipo *Message Oriented Middleware (MOM)* e motivada pela disponibilização de um *workflow* altamente flexível entre aplicações, total transparência na comunicação e no minimizar as tarefas de configuração específicas a cada aplicação.

1 Introdução

A interligação de aplicações é uma necessidade recorrente em ambientes informáticos. Essas necessidades têm muitas vezes de ser pensadas de raiz, e para serem implementadas é comum recorrer a algum tipo de programação distribuída muitas vezes dependente da tecnologia subjacente.

Necessidades específicas e menos vulgares originam a modificação das aplicações de forma a que exista essa interligação. Esse tipo de modificação, quando possível, implica a alteração do código fonte das aplicações. Após a alteração com os devidos transtornos de *debugging* e testes, a aplicação volta a entrar em produção.

O problema surge quando uma nova interligação com uma determinada aplicação é necessária, pois tal implica voltar a reproduzir o mesmo ciclo. Se a

interligação tiver de ser aplicada rapidamente, tal não será possível devido ao tempo despendido no desenvolvimento. A configuração das interligações em que é a aplicação a gerir as suas ligações retira flexibilidade ao sistema.

O Information Router (IR) surge da necessidade de responder a esses problemas e de dotar a interligação de aplicações de maior flexibilidade. Esse requisito implica também que a configuração da arquitectura seja feita rapidamente, sem impactos negativos quer nas configurações já existentes, quer no funcionamento corrente da mesma. A forma de configuração deve também permitir implementar de forma intuitiva algum tipo, ainda que simples, de *workflow* entre as aplicações.

É uma arquitectura centralizada e genérica pensada para a interligação de aplicações. A mesma foi inspirada em sistemas *MOM* como o *Java Message Service (JMS)*, cuja ideia base é a existência de uma camada de abstracção que passa a existir entre o cliente e o servidor. Essa camada recebe mensagens de um ou mais *produtores de mensagens* e faz o *broadcast* para um ou mais *consumidores de mensagens*.

No entanto, apesar de ser inspirado em sistemas *MOM* é construído sobre um paradigma de programação distribuída, os *webservices*. Ou seja a arquitectura do IR é um modelo híbrido de programação distribuída, em que existe uma desacopolação entre as aplicações inspirada nos sistemas *MOM*.

A exposição da arquitectura do IR é iniciada com uma visão geral da mesma, onde é feito o paralelo com outras arquitecturas baseadas em modelos de programação distribuída. De seguida é descrito o interface de contacto dos clientes para com o núcleo decisor. A organização interna do núcleo decisor é exposta com algum detalhe na secção seguinte, sendo complementada com alguns exemplos simples de aplicação do IR. As tecnologias usadas são apresentadas de seguida, bem como algumas considerações acerca de decisões tomadas na escolha das mesmas. A exposição termina com algumas conclusões relativas à arquitectura.

2 Arquitectura

A arquitectura do IR foi projectada tendo como base duas características, simplicidade e flexibilidade. São características sempre desejáveis em qualquer sistema, embora muitas vezes difíceis de conseguir e ainda mais difíceis de fazer coabitar.

2.1 Visão geral

De forma a que a compreensão da arquitectura seja clara irá ser feito um paralelo com arquitecturas baseadas em modelos de programação distribuída, como o *Remote Method Invocation (RMI)* do *Java* e com uma arquitectura do tipo *MOM*. A grande diferença entre o *RMI* e o *MOM* consiste no facto de que em sistemas *MOM*, o contacto entre as aplicações não acontece directamente sendo usado um *middleware de mensagens* para o efeito, como podemos ver na Figura 1.

O IR tem, de grosso modo, uma forma de funcionamento similar aos sistemas *MOM*. As principais diferenças residem no facto da invocação sobre o núcleo do

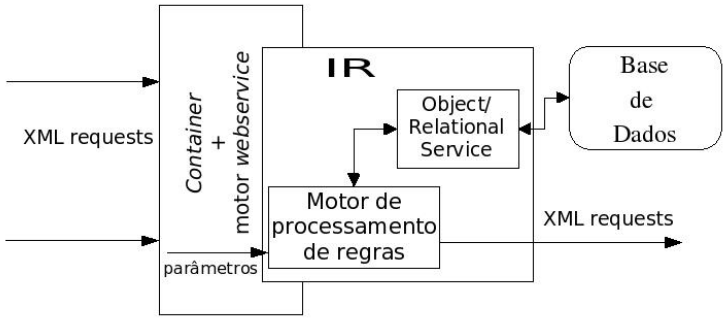

Figura 1. RMI vs MOM

IR ser feita recorrendo a *webservices*, e não usando uma *API* possivelmente dependente da tecnologia. E no facto de o encaminhamento ser *decidido* pelo IR e não por mecanismos de subscrição.

2.2 Os interfaces

O contacto com o núcleo é feito por um interface exposto fixo. Os seus parâmetros são genéricos e assemelham-se (embora não tão complexos) aos cabeçalhos utilizados em protocolos de encaminhamento de dados, onde podemos encontrar estes parâmetros: identificação do cliente, identificação da máquina, dados temporais da invocação, atributos genéricos, acção que desencadeou a invocação, uma mensagem por defeito, etc. É com base nesses parâmetros que o encaminhamento da invocação se dá.

2.3 O núcleo do IR

O IR é uma arquitectura centralizada e como tal, existe um núcleo que controla todos os processos. A única parte visível para o exterior desse núcleo é o interface *webservice*.

Para que uma aplicação possa contactar o IR tem de estar registada no mesmo. Dessa forma o IR passa a ter um nível de consciência adequado a estabelecer a comunicação entre as aplicações. O registo é feito administrativamente. As informações a guardar acerca do cliente são as mais diversas, de entre as quais o endereço *webservice* de destino para um contacto posterior. Desta forma um cliente pode contactar e ser contactado caso exista essa necessidade.

Ainda associado a cada cliente podem existir contactos individuais. Esses contactos individuais definem destinos ainda mais específicos dentro do cliente destino, que o próprio saberá interpretar devidamente. Os exemplos de contactos individuais vão desde endereços de correio electrónico, a contactos de mensagens instantâneas, números de telemóvel, entre outros. Dentro de uma regra é possível

escolher um desses destinos mais específicos quer directamente, quer mediante algum critério (ex. escolha de um sub-conjunto de contactos). A Figura 2 mostra um esquema do núcleo do IR.

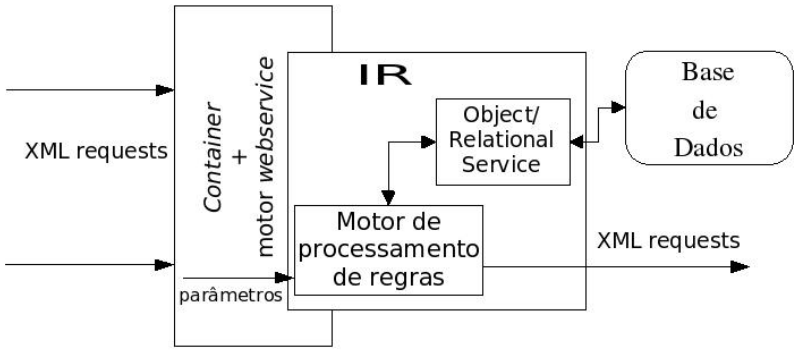

Figura 2. Blocos funcionais do núcleo do IR

2.4 As regras

Associado a um cliente existe um mecanismo que permite decidir que destinos contactar. A decisão é feita mediante um conjunto de regras que quando aplicadas sobre os valores de uma invocação, permitem decidir que clientes contactar. As regras existem no núcleo associadas a um determinado cliente. As mesmas estão organizadas por grupos e dentro de cada grupo as regras não são independentes entre si. Junto com cada regra existem dois valores lógicos, um que decide se o processamento continua para a regra seguinte, e outro que decide se o processamento continua no grupo seguinte. Essa escolha tem em conta a valoração actual da regra. Desta forma para além de poder proporcionar uma organização lógica das regras, podemos flexibilizar e acelerar o processamento das mesmas.

A escolha dos clientes a contactar está presente na própria regra, e os mesmos têm de estar registados no sistema. Ou seja, a validação da regra implica o contacto dos clientes discriminados nas regras. Alguns exemplos dessas regras podem ser vistos na Tabela 1[1].

[1] São exemplos simples de utilização, são também permitidos operadores lógicos encadeados `and, or, not`.

Tabela 1. Exemplos simples da linguagem de regras

Regra	Descrição
`(host match ".*") forward to jabber;`	Qualquer que seja o *host*, encaminha para todos os contactos do cliente *jabber*.
`(client = "vitae") forward to sms match "96.*";`	Se o cliente for o *vitae*, encaminha para o cliente *sms*, cujos contactos se iniciem por 96.

3 Exemplos de aplicação

3.1 Suporte

Um exemplo simples de aplicação, pode ser reconhecido no cenário em que é necessário dar suporte a uma determinada ferramenta a um determinado cliente. O cliente requer suporte por correio electrónico. No entanto a pessoa que lhe dá esse suporte poderá ser contactada de várias formas.

Quando chega uma mensagem de correio electrónico a pedir suporte, a aplicação que gere o endereço de suporte contacta o IR. Mediante as regras associadas à aplicação e mediante os dados que a aplicação envia aquando da invocação, o IR pode decidir contactar o responsável pelo suporte de várias formas, por mensagens instantâneas por correio electrónico, por *SMS* e também pode ser adicionada uma entrada no registo de ocorrências. As decisões podem ser tomadas por exemplo, mediante quem recebe o correio electrónico, consoante a data e hora e pode tomar uma acção diferente consoante seja fim de semana ou dia útil. Uma possível resposta pode também ser encaminhada de forma idêntica. Esses contactos não são feitos directamente pelo IR, o IR contacta aplicações que saibam como o fazer e o que fazer com os mesmos.

3.2 Recepção de *Curriculuns Vitæ*

A recepção de um *Curriculum Vitæ* via um interface *web* mostra um caso simples de utilização. A introdução de um *Curriculum Vitæ* pode fazer despoletar uma invocação sobre o IR. Dentro do mesmo o processamento das regras pode levar ao contacto da pessoa responsável pelos recursos humanos de várias formas, correio electrónico, *SMS*, etc. Pode também ser contactada uma aplicação que trate da impressão directa dos dados mais relevantes de forma a que o *Curriculum Vitæ* possa ser avaliado mais rapidamente. Mais uma vez, esses contactos não são feitos directamente pelo IR, o IR contacta aplicações que saibam como o fazer e o que fazer com os mesmos.

4 Tecnologia

Várias ferramentas foram testadas durante o processo de análise e de decisão acerca da tecnologia a ser utilizada. Algumas hipóteses foram deixadas de parte como é o caso do uso de *Java 2 Enterprise Edition (J2EE)*[2][3]. A complexidade introduzida com o seu uso fazia cair a simplicidade desejada quer para a construção da aplicação, quer depois para o seu ciclo de vida em produção com o respectivo acréscimo de complexidade na manutenção. A complexidade introduzida com o seu uso não era traduzida por quaisquer ganhos.

O núcleo do IR está construído recorrendo à linguagem Java no entanto, como o contacto com o mesmo é feito via *webservices* os clientes podem ser escritos em qualquer linguagem.

Na construção do núcleo foram usadas diversas ferramentas. A combinação `Tomcat + Axis` foi usada para a exposição dos *webservices*. O `Hibernate`[1] foi usado para tratar da persistência dos objectos para o paradigma relacional. A base de dados usada foi o `PostgreSQL` no entanto quase sem qualquer esforço adicional possa ser usada outra qualquer, desde que tenha o suporte por parte do `Hibernate`[1]. Para o processamento das regras a escolha recorreu sobre o `Antlr`, uma ferramenta que permite construir reconhecedores, compiladores e tradutores a partir de uma descrição gramatical.

5 Conclusões

Após a análise da arquitectura e do núcleo podemos verificar que quer a arquitectura, quer o núcleo são simples. O acesso é feito por um interface normalizado suficientemente genérico para acomodar quaisquer tipo de aplicações, esse interface e os parâmetros do mesmo são a única coisa que um cliente necessita de conhecer para contactar o núcleo. Como os interfaces de acesso estão expostos via *webservices* os clientes podem estar escritos em qualquer linguagem.

O funcionamento do núcleo para além de simples é também bastante flexível. A flexibilidade advém da facilidade com que se adiciona um novo cliente e respectivos dados associados, do encaminhamento baseado por regras, da sua organização por grupos e da influência que a valoração de uma regra tem no processamento da regra e do grupo de regras seguinte. Um novo cliente e os dados referentes ao mesmo podem adicionado através do interface de gestão do núcleo do IR. As regras permitem que várias possibilidades de análise sobre as mensagens sejam definidas e que novos destinos sejam adicionados, isto sem que seja necessário alterar quer os clientes, quer o núcleo. A organização por grupos e a influência que a valoração de uma regra tem na avaliação da regra e do grupo de regras seguinte, permite uma organização das mesmas de uma forma inteligente para minimizar a quantidade de regras processadas.

Os dados a ser armazenados acerca das aplicações cliente foram pensados segundo o paradigma da programação orientada aos objectos (a persistência dos mesmos é completamente deixada a cargo do `Hibernate`[1]), e são os suficientes para garantir a flexibilidade necessária. O processamento das regras é feito de

forma célere sem grandes impactos negativos no funcionamento do núcleo com vários pedidos. O mesmo pode ser optimizado com uma organização inteligente das regras.

As limitações da arquitectura prendem-se com o facto da mesma ainda ter sido sujeita a poucos testes. O seu uso para integração de aplicações reais está de momento a decorrer, o que impede uma análise mais profunda das mesmas. No entanto uma das limitações que ainda existe está relacionada com o tratamento de erros, neste ponto do desenvolvimento do IR, se acontecer algum erro no processo a aplicação não tem conhecimento do mesmo não podendo fazer nova invocação.

Sendo uma arquitectura ainda em amadurecimento existe ainda margem de progressão, especialmente no que diz respeito a testes reais de funcionamento da aplicação sobre situações de sobrecarga, definição de *workflow* mais complexo e de situações de interligação de aplicações menos vulgares.

Referências

1. Bauer, C., King, G.: Hibernate in Action. Manning Publications Co. (2004)
2. Johnson, R.: Expert one-on-one J2EE Design and Development. Wrox Press (2002)
3. Johnson, R., Hoeller, J.: Expert one-on-one J2EE Development whithout EJB. Wrox Press (2004)
4. Recomendação W3C: *SOAP Version 1.2 Part 0: Primer*, Junho 2003.
5. Recomendação W3C: *SOAP Version 1.2 Part 1: Messaging Framework*, Junho 2003.
6. Recomendação W3C: *SOAP Version 1.2 Part 2: Adjuncts*, Junho 2003.
7. *W3C Draft: SOAP Version 1.2 Usage Scenarios*, Junho 2002.