FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Avaliação da Plataforma de Integração Oracle Retail

Pedro Miguel Rosário Alves

Relatório de Projecto

Mestrado Integrado em Engenharia Informática e Computação

Orientador: João Correia Lopes (Professor Auxiliar)

7 de Julho de 2008

Avaliação da Plataforma de Integração Oracle Retail

Pedro Miguel Rosário Alves

Relatório de Projecto
Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Teresa Galvão Dias (Professora Auxiliar da FEUP)

Arguente: Paulo Novais (Professor Auxiliar da UM)
Vogal: João Correia Lopes (Professor Auxiliar da FEUP)

16 de Julho de 2008

Resumo

Este projecto baseou-se no estudo das actuais plataformas de integração das aplicações Oracle para a área de negócio da venda a retalho ("Oracle Retail"). Foi estudada a plataforma utilizada à data de início do projecto e a plataforma da nova versão destas aplicações, lançada durante o projecto (Abril de 2008). Foram ainda estudadas as tecnologias "Fusion Middleware", em particular a orquestração de processos baseados em *Web Services* e XML, e o sistema de filas de espera deste *middleware*, compatível com a actual plataforma de integração das aplicações "Oracle Retail".

Após o estudo de todas estas plataformas e tecnologias, entrou em desenvolvimento, como parte integrante do projecto, um protótipo de integração das aplicações "Oracle Retail". Este baseou-se numa junção destas tecnologias, em particular na utilização de *Web Services* para comunicação com bases de dados e filas de espera de mensagens para integração. Este sistema, implementado na nova plataforma, compatível com tecnologias "Fusion Middleware", forneceu o *bus* de integração para o protótipo, que se baseou num caso de integração entre duas aplicações "Oracle Retail", representando um caso generalista, que engloba a maior parte dos cenários de integração. Após a implementação do protótipo, foi avaliada a sua execução. Nessa avaliação, foi ainda comparado este método inovador com a actual plataforma de integração das aplicações "Oracle Retail".

Conclui-se, com a implementação deste protótipo, que este método de integração apresenta diversas vantagens em relação à implementação através de "Java Beans", presente nas plataformas ORIB v12, para aplicações J2EE, e ORIB 13 para todo o conjunto de aplicações. A definição de transformações necessárias entre os diferentes formatos aceites por cada aplicação é efectuada de modo gráfico, através de um mapeamento visual entre os campos (baseado num ficheiro de transformação XSL), excluindo a necessidade de criar "Java Beans" próprios para este efeito. Outras vantagens deste método derivam do uso de *Web Services* para aceder à base de dados e às filas de espera e no uso de XML para descrição destes componentes, assim como no uso de XSD para a verificação do formato das mensagens de integração.

A utilização de um sistema de filas de espera baseado em tabelas e registos de base de dados pode representar uma diminuição de eficiência, devendo esta ser medida futuramente, de forma a comprovar que não existe diminuição de eficiência significativa no sistema global de transacções entre as aplicações. Este é apenas um protótipo inicial, fornecendo uma antevisão de uma futura versão da plataforma de integração de aplicações "Oracle Retail". Devem ser criados meios de permitir uma integração fiável através deste método. Na próxima versão de aplicações "Oracle Retail" está previsto o uso destas tecnologias para integração, pelo que essa versão já deverá possuir uma base estável de integração baseada neste método. Esta integração pode não se resumir apenas à área do retalho, mas possivelmente a outras aplicações Oracle.

Abstract

This project was based on the study of the "Oracle Retail" integration platforms. It was studied the platform used to the date at the beginning of the project and the platform included on the new version of these applications, released during the projecto (April of 2008). There were also studied the "Fusion Middleware" technologies, mainly the processes orchestration based on Web Services and XML, and the queuing system of this middleware, compatible with the actual integration platform of "Oracle Retail" applications .

After the study of all these platforms and technologies, it was developed, as part of the project, an integration prototype of "Oracle Retail" applications. It was based on a junction of these technologies, particularlly in the Web Services for communication with the database and queuing system for integration messages. This system, implemented in the new platform, compatible with "Fusion Middleware" technologies, supplied the necessary integration bus for this prototype, based on an integration case, between two "Oracle Retail" applications, a general case that includes most of integration scenarios. After the prototype deployment, its execution was evaluated. In that evaluation, this innovative method was compared with the actual integration platform of "Oracle Retail" applications.

It was found, with this prototype's implementation, that this integration method presents several advantages comparing to the implementation through "Java Beans", present in the platforms ORIB v12, for J2EE applications, and ORIB v13 for the whole applications group. The definition of the necessary transformations among the different formats accepted by each application is made in a graphic way, through a visual mapping tool that allows to connect the mapped fields (based on a XSL transformation file), excluding the need to create extra "Java Beans" for this effect. Other advantages of this method concern to the use of Web Services for database and queuing system connections and also in the use of XML for these components descriptions, as well as for validation of the content of the integration messages, using XSD.

The use of a queuing system based on database tables and records may represent an efficiency decrease, should this be hereafter measured, in order to prove that the decrease of significant efficiency doesn't exist in the global transactional system among the applications. This is just an initial prototype, supplying an preview of a future version of the integration platform for "Oracle Retail" applications. Means should be created to allow a reliable integration through this method. In the next version of "Oracle Retail" applications the use of these technologies is foreseen for integration purpose, for that it should already possess a stable base of integration based on this method. This integration may not only be summarized to the retail business area, but possibly to other Oracle applications.

Agradecimentos

Expresso aqui os meus agradecimentos a todas as pessoas que me ajudaram de alguma forma ao longo deste projecto.

Um grande obrigado e extremo agradecimento ao meu orientador, o Professor João Correia Lopes, pelo auxílio e apoio prestado durante o projecto e elaboração do relatório.

Fico ainda agradecido ao Eng. António Quaresma pelo auxílio prestado durante a instalação do ambiente necessário ao desenvolvimento do protótipo.

Quero ainda agradecer ao Eng. Cláudio Reis, meu orientador na Wipro Retail pelo apoio durante todo o projecto e pelo conhecimento transmitido.

Por último mas não menos importante, quero ainda agradecer às pessoas que compreenderam a minha ausência física e mental durante a elaboração do relatório, como os meus pais, irmão, namorada e amigos mais próximos.

Um especial agradecimento ao meu amigo Mário Barbosa, licenciado na área de letras pela ajuda com a revisão do documento, de forma a tornar o português mais legível.

Quero ainda agradecer à "Pastelaria Suiça, Lda." pelas várias horas de utilização da corrente eléctrica cedidas durante intermináveis fins de semana e não só, em prol da escrita deste relatório, sempre que a bateria do portátil me deixou ficar mal.

Um muito obrigado a todos pela paciência, pela compreensão, pelo conhecimento transmitido, pelo apoio e pelos empurrões necessários.

Pedro Miguel Alves

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas

AQ	Advanced Queues
Batch	Processamento em grupo de várias mensagens/transacções, normalmente feito durante a noite
BD	Base de Dados
BPEL	Business Process Execution Language
CA	Cadeia de Abastecimento
CMF	Common Message Format, mensagem XML
CPU	Central Processing Unit
DSI	(Direcção de Sistemas de Informação) da Sonae Distribuição
e*Gate	Sistema que implementa o EAI.
e*Way	Canal de comunicação entre o EAI e outro sistema
EAI	Enterprise Application Integration
ESB	Enterprise Service Bus
HDD	Hard Drive Disk
IP	Internet Protocol
JMS	Java Messaging Service
OEMS	Oracle Enterprise Messaging Service
OR	Oracle Retail
ORIB	Oracle Retail Integration Bus
ORMS	Oracle Retail Merchandising System
ORSL	Retail Service Layer
Package	Encapsula um cabeçalho e um corpo contendo código PL/SQL
Queue	Fila de espera, processa as mensagens que chegaram primeiro
RAM	Random Access Memory
SO	Sistema Operativo
Schedule	Tarefas calendarizadas, executadas conforme um horário definido
SIM	Store Inventory Manager
SI	Sistemas de Informação
SOA	Service Oriented Architecture
VM	Virtual Machine (Máquina Virtual)
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSL	eXtensible Stylesheet Language

Capítulo 1

Introdução

O negócio da venda a retalho encontra-se presente em todo o mundo. Como actividade mercadológica abrangente, desde bens de primeira necessidade a bens secundários e de luxo, o mercado do retalho de média e grande dimensão encontra-se presente em todos os países industrializados e em grande parte dos países sub-desenvolvidos.

A Wipro Retail [Tec08] (Ex-Enabler) tem como objectivo fornecer soluções na área dos Sistemas de Informação (SI) e tecnologias de apoio ao negócio do retalho. Com presença marcada na Europa, em especial, e uma expansão crescente no resto do mundo, a Wipro Retail é líder em implementações de SI de apoio ao retalho em médias e grandes empresas.

O avanço das tecnologias desenvolvidas pela Oracle [Cor08f] para o negócio do retalho ("Oracle Retail" [Cor08g]) tem apoiado as empresas que apostam na implementação dos seus produtos e traz cada vez mais benefícios ao apoio do negócio e aumento dos lucros.

Neste projecto foram estudadas as plataformas de integração para aplicações "Oracle Retail" e foram ainda estudadas as tecnologias "Fusion Middleware" [Cor08c] e o suporte tecnológico a esta nova plataforma de integração de aplicações Oracle. Acredita-se que o investimento nesta tecnologia de *middleware* possa trazer grandes vantagens ao nível da integração dos produtos "Oracle Retail". Para tal, é estudado neste projecto o desenvolvimento de um protótipo BPEL (Business Process Execution Language [Cor08d]) para realizar a troca de mensagens de integração entre as aplicações "Oracle Retail" e é avaliada a sua execução.

Figura 1.1: Logotipo da Enabler

1.1 A Wipro no Retalho

A empresa teve origem na Sonae Distribuição, da qual era parte integrante até 1997. Formando-se como nova empresa, tomou o nome de Enabler, como declaração da clara intenção de ser um facilitador (*enabler*) de soluções, auxiliando os clientes na implementação de sistemas de informação que pudessem potenciar os seus negócios.

A empresa tem como objectivo fornecer soluções na área dos SI e tecnologias de apoio ao negócio do retalho. O conhecimento, tanto a nível técnico como do negócio de retalho e a experiência em Tecnologias de Informação (TI) ajudam os retalhistas a reduzir o risco na implementação de projectos críticos ao sucesso do negócio. A visão desta empresa é centrada na liderança especializada no desenvolvimento, implementação e suporte de sistemas de retalho. Estes factores conferem à empresa vantagem no mercado, também devido a uma grande herança do mercado de retalho, juntamente com um balanço entre inovação e pragmatismo.

A sua aposta em Investigação e Desenvolvimento garante ainda a adopção de tecnologias de ponta, adoptando as mais recentes inovações nas soluções desenvolvidas.

Actualmente a empresa atravessa uma fase de transição, após ter sido adquirida pela Wipro Technologies [Tec08], uma empresa sediada na Índia, com mais de 80.000 colaboradores, focalizada no desenvolvimento de soluções de negócio baseadas em TI. Apesar da aquisição, a Enabler manteve a seu cargo o mundo das tecnologias de informação de apoio a empresas do mercado de retalho, facto que se deve à experiência da empresa (como antiga Enabler) neste sector e ao alargado conhecimento na oferta de soluções de SI desta área.

A última alteração na organização, impulsionada pela compra da Wipro, relaciona-se com a mudança do nome Enabler (Enabler-Wipro no estado de transição) para a designação Wipro Retail.

1.2 Motivação e Objectivos do Projecto

Este projecto surge na consequência do lançamento da nova plataforma de integração e do conjunto de aplicações Oracle Retail e na compatibilidade desta com tecnologias "Fusion Middleware". Este conjunto de tecnologias, direccionadas para a comunicação

entre aplicações Oracle, já serve de plataforma de integração de diversas aplicações Oracle. A versão actual dos produtos "Oracle Retail" (aplicações para apoio ao negócio do retalho) possui uma plataforma específica de integração denominada ORIB13 (Oracle Retail Integration Bus, versão 13).

É possível, de imediato, utilizar tecnologias "Fusion Middleware", de forma a integrar as aplicações "Oracle Retail". Esta plataforma de integração inovadora, com recurso a processos, aplicações e tecnologias anteriormente existentes e ao seu aperfeiçoamento e combinação, suporta a comunicação inter-modular de todo o portefólio de aplicações Retail da Oracle, na versão 13.

Nesta nova plataforma são introduzidas tecnologias da Oracle para efeitos de gestão de filas de espera de mensagens de integração. Todo o processo de troca de mensagens entre os módulos de Retail passa a ser suportado por produtos exclusivos da Oracle, descontinuando-se assim a plataforma da Sun Microsystems (Seebeyond [See00]) utilizada até à data. Numa fase posterior, fazendo parte das tecnologias da "Fusion Middleware", entrará em vigor o uso do BPEL (Business Process Execution Language) e ESB (Enterprise Service Bus). Este projecto será centrado na tecnologia BPEL da "Fusion Middleware". A orquestração de processos BPEL possibilita definir e implementar todo o fluxo de troca de mensagens entre as aplicações "Oracle Retail".

Com este projecto, pretende-se realizar um estudo da nova plataforma de integração, particularmente em relação à orquestração de processos BPEL, como tecnologia integrante da plataforma "Fusion Middleware". Para além da compreensão dos seus componentes, deverão ser definidas melhores práticas, assim como devem ser avaliadas as potencialidades oferecidas. Desta forma será possível aos projectos futuros possuírem já integração através desta tecnologia inovadora.

Neste sentido, e de forma a manter a qualidade de serviço da empresa, será necessário compreender o funcionamento destas tecnologias. Como parte integrante do projecto, encontra-se subjacente o desenho e especificação de um protótipo de utilização da nova plataforma de integração (ORIB v13), com recurso à tecnologia BPEL, seguido da implementação do mesmo.

De forma a poder ser colocado em execução este protótipo, é ainda necessário criar um ambiente de execução apropriado. Este ambiente é baseado em *middleware* de suporte Oracle e a versão 13 das aplicações de "Oracle Retail", incluindo o módulo de integração ORIB (Oracle Retail Integration Bus) v.13.

Por fim, deve ser efectuada uma avaliação de todo o fluxo de integração, identificando e detalhando os componentes, bem como a responsabilidade de cada um deles no cenário de integração.

1.3 Estrutura do Relatório

O presente relatório divide-se em cinco capítulos. O capítulo actual é composto pela introdução, dando a conhecer o contexto deste projecto, desde a área de negócio da empresa em que foi desenvolvido até ao problema existente em concreto. O capítulo seguinte (Capítulo dois), apresenta uma revisão do panorama geral a nível da integração de aplicações em empresas (EAI — *Enterprise Application Integration*) e das tecnologias de suporte à solução desenvolvida. No Capítulo três é feita uma antevisão das ferramentas e ambientes de apoio ao desenvolvimento do projecto. A arquitectura e a especificação da solução desenvolvida são apresentadas no Capítulo quatro, enquanto a implementação da mesma é descrita no Capítulo cinco. No Capítulo seis é feita a avaliação do protótipo implementado e no Capítulo sete são retiradas as conclusões dessa mesma avaliação, assim como do projecto de estágio, sendo ainda apresentadas as perspectivas de desenvolvimento futuro.

Capítulo 2

A Integração de Aplicações nas Organizações

Neste capítulo é dada a perspectiva actual da integração de aplicações, no âmbito das organizações. São ainda apresentadas as tecnologias que permitem essa integração de forma completa, particularmente na integração das aplicações Oracle Retail [Cor08g]. É aqui efectuada uma descrição geral do estado da integração de aplicações nas empresas a mais alto nível, de modo a contextualizar o projecto no panorama geral da integração empresarial.

2.1 Introdução

Hoje em dia, as organizações apostam cada vez mais no uso de TI e SI, como forma de melhorar processos de negócio, automatizar tarefas rotineiras, ou simplesmente possuir sistemas de apoio ao negócio. Estes sistemas possuem como objectivo principal a poupança de tempo e o aumento dos lucros da empresa.

No entanto, a existência de sistemas isolados implica muitas vezes a duplicação de dados nas diversas aplicações, entre outras desvantagens. A integração dessas aplicações num sistema unificado permite a existência de uma plataforma que possibilite a comunicação entre as diversas aplicações existentes nas empresas. A sincronização das aplicações existentes na organização permite uma gestão facilitada de todo o processo de negócio, entre diversas vantagens provenientes de uma integração completa [Ena08].

A implementação de aplicações para o negócio do retalho, em particular da Oracle [Cor08f] (Oracle Retail), com base na plataforma de integração fornecida, permite que todas as aplicações Oracle Retail se encontrem em constante comunicação. Desta forma é possível retirar todo o potencial deste conjunto de aplicações.

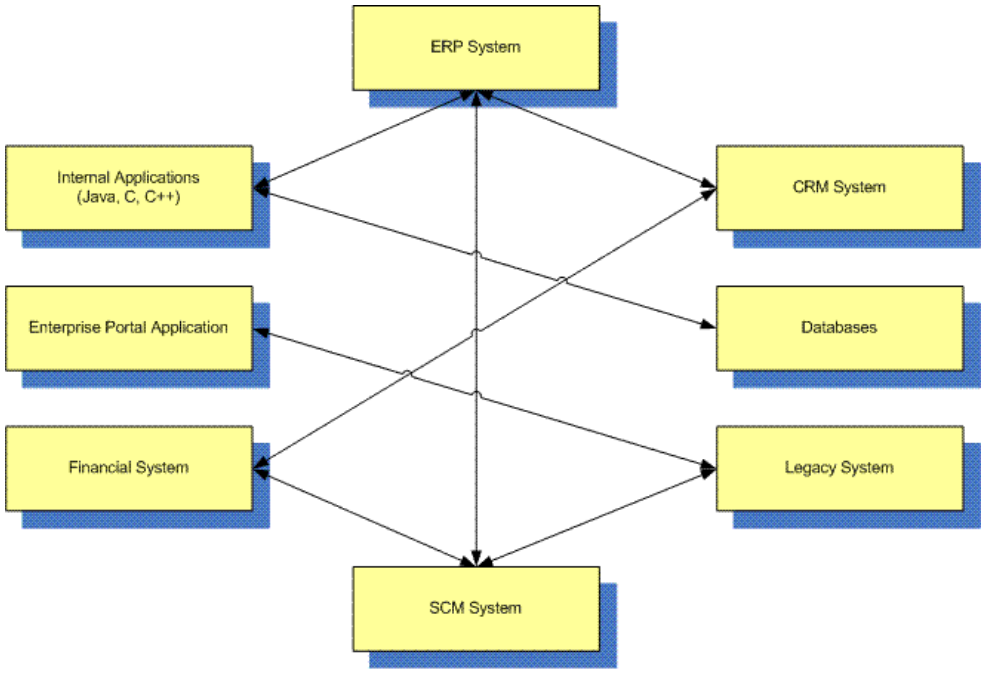

Figura 2.1: Realidade de Negócio sem EAI [Con08]

2.2 EAI

A Integração de Aplicações Empresariais (EAI — *Enterprise Application Integration* [FBVRR05]) como área tecnológica estuda a integração de SI dentro de empresas e organizações. É apoiada por um conjunto de ferramentas e processos que permitem o funcionamento conjunto de diferentes aplicações, indiferentemente da sua plataforma tecnológica, formatos de dados ou localização física. A EAI permite o estabelecimento de uma infraestrutura tecnológica de interligação indiferenciada de aplicações heterogéneas internamente desenvolvidas, como um sistema unificado [Bar05].

Processos e dados podem ser partilhados através da organização e até além das suas fronteiras, de forma a incluir parceiros, clientes e *stakeholders*, o que permite que as organizações se mantenham actualizadas e respondam a alterações, necessidades e expectativas de mercado em tempo real. Essencialmente, o objectivo principal da integração de aplicações consiste em alterar a perspectiva de comunicações e arquitecturas ponto-a-ponto, complexas e redundantes, para uma abordagem mais flexível, centrada num ponto de integração comum [Bar05].

A EAI identifica e interliga *workflows* de utilizadores e de funções aplicacionais através de filas de mensagens e tecnologias Web. As ferramentas de suporte à integração identificam, capturam, integram e fornecem dados e funcionalidades de sistema a utilizadores, através de uma série de interfaces multi-plataforma e inter-funcionais. As tecnologias de

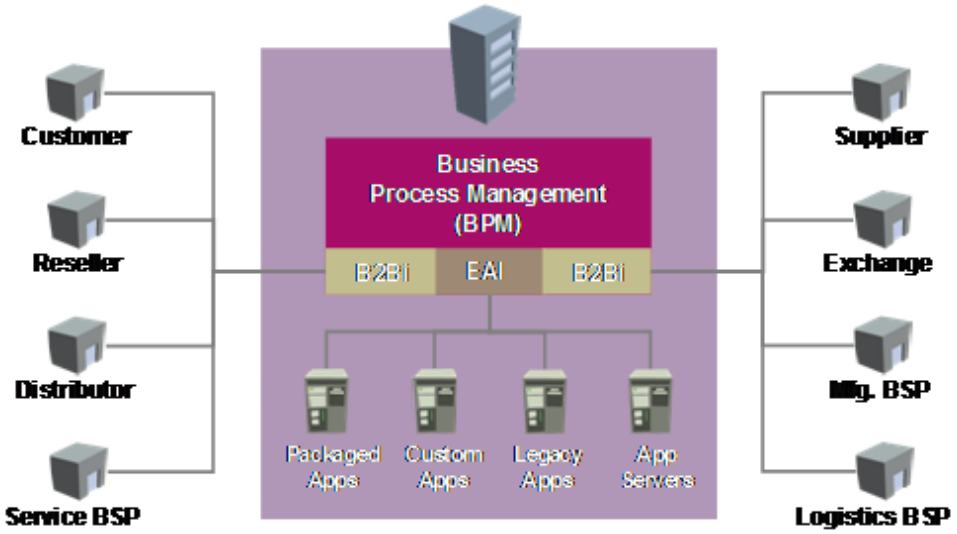

Figura 2.2: Solução de Integração Empresarial [FBVRR05]

message queuing amadureceram ao ponto de poderem suportar a integração destas funcionalidades, sem exigir uma mudança significativa de ambientes *legacy* complexos.

2.2.1 Abordagens de Integração Empresarial

Tradicionalmente, todos os projectos intra-organizacionais eram desenvolvidos isoladamente. A gestão e desenho do projecto eram baseados em interfaces ponto-a-ponto, para cada um dos sistemas, com os quais a aplicação devia interagir, como se pode observar no Diagrama 2.1. Esta abordagem de integração acarretava desafios logísticos e de gestão, aquando do desenvolvimento e implementação do sistema desenvolvido [Fen03]. Tal abordagem levaria frequentemente a trabalho redundante por parte das equipas individuais, forçando a modificação de funções centrais, de forma a permitir a implementação de novos requisitos. Essas modificações resultavam frequentemente numa solução comprometida que, embora preenchesse todos os requisitos, não seria completa, uma vez que possuiria sempre limitações inerentes aos sistemas centrais.

Em abordagens de integração baseadas em EAI, como se pode observar no Diagrama 2.2, cada sistema encontra-se ligado a uma *framework* central que controla as interacções entre eles (*middleware*). Cada nova tecnologia introduzida é imediatamente ligada a essa *framework*, permitindo que essa tecnologia se junte automaticamente aos processos de negócio. Todos os sistemas comunicam e trocam informação e dados através da *framework*.

A Figura 2.3 demonstra e compara as estratégias EAI e ponto-a-ponto, quanto à evolução do custo por transacção. Como facilmente se conclui, uma estratégia baseada em EAI permite uma redução de custos significativa, através da diminuição do número

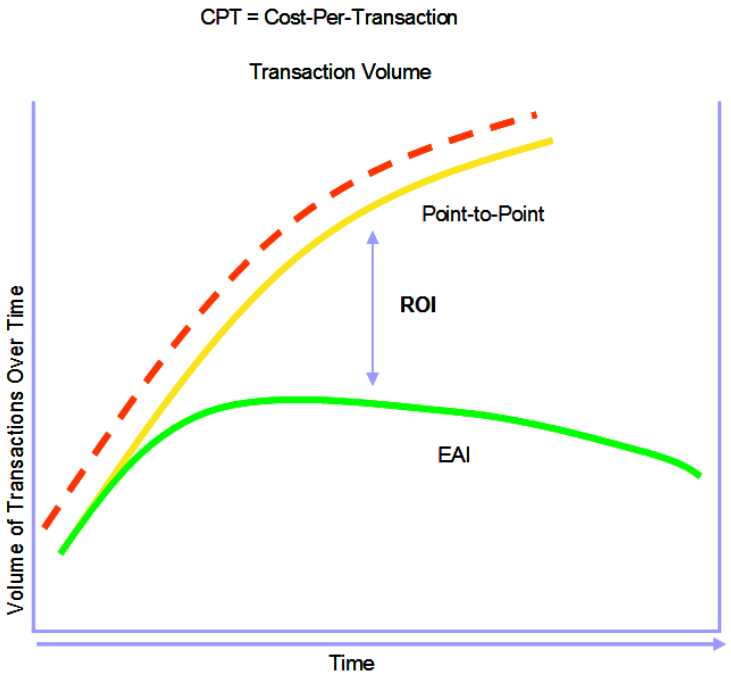

Figura 2.3: Evolução do Custo de Transacção para as Diferentes Estratégias [FBVRR05]

de transacções ao longo do tempo, devido ao método de implementação baseado em *middleware*. Isso representa um retorno de investimento (ROI) grande, relativamente a abordagens ponto-a-ponto.

2.2.2 Middleware

No mundo da EAI, o *middleware* [Fen03] é utilizado como mecanismo simples de transferência de informação e partilha da lógica de negócio entre aplicações. O *middleware* pode ser visto como a tecnologia de base da EAI.

O *middleware* esconde a complexidade inerente ao sistema operativo e à rede de comunicação, de forma a facilitar a integração entre sistemas na empresa. Em termos gerais, os programadores apenas lidam com uma Interface de Programação de Aplicações (API — *Application Programming Interface* [Wik08a]) em cada sistema e o *middleware* trata da transferência de informação entre os diferentes sistemas por parte das aplicações. Essas API são direccionadas para transferência de dados ou possuem mecanismos de invocação de processos. Tipicamente, as API não conhecem as aplicações que estão a interligar. Os programadores necessitam, no entanto, de criar as ligações no *middleware*. Contudo, hoje em dia, as soluções de *middleware* encontram-se orientadas para possibilitar a integração de aplicações com pouca ou até nenhuma programação.

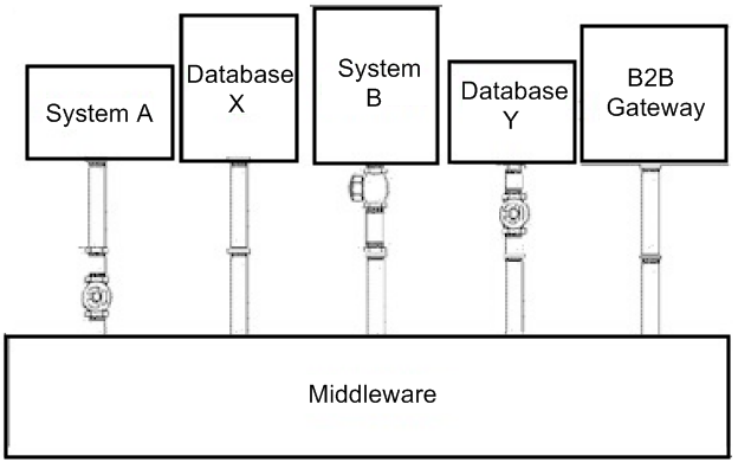

Figura 2.4: Integração Baseada em *Middleware* [Fen03]

O uso de *middleware* fornece interfaces genéricas que permitem a troca de mensagens entre as aplicações. Cada uma dessas interfaces define um processo fornecido pela aplicação.

Desta forma, tal como demonstra a Figura 2.4, é possível integrar diversas aplicações apenas com um ponto de integração para cada uma, reduzindo o custo da integração. Isto torna ainda possível adicionar ou substituir aplicações, de uma forma que não afecte as outras já integradas. O próprio *middleware* encontra-se preparado para realizar operações de redireccionamento, transformação, agregação, separação e conversão de dados que lhe são passados. No entanto, é adicionada complexidade em termos de preparação e estabelecimento efectivo do *middleware*, de forma a ser utilizado, e ainda na conversão das aplicações de forma a poderem utilizar a API do *middleware*. Neste projecto, as aplicações "Oracle Retail" serão integradas com recurso ao ORIB v13 e às tecnologias "Fusion Middleware", que representam o *middleware*.

2.3 *Web Services*

Os *Web Services* (WS [Kis07]) são aplicações auto-descritivas que podem ser invocadas via rede usando protocolos como o SOAP [Kis07] (*Simple Object Access Protocol*).

Os WS são publicados em registos universais denominados UDDI [Sea08] (*Universal Description, Discovery and Integration*), de forma a que qualquer programa que os queira utilizar os encontre facilmente, juntamente com a descrição do serviço em questão.

O que diferencia os WS de outros métodos de computação distribuída como DCOM (*Distributed Component Object Model* [Ruo08]) ou CORBA (*Common Object Request Broker Architecture* [MG08]), que são mais rápidos e mais leves que os WS? A resposta encontra-se no elemento mais simples, a aceitação universal do XML. O XML é dos

elementos mais importantes no mundo da troca electrónica de dados (EDI — *Electronic Data Interchange* [Sys08a]). Todos os sistemas aceitam. Assim sendo, o uso de XML para enviar dados torna-se muito mais simples e muito mais aceitável que qualquer outra tecnologia jamais utilizada para trocas de dados. Em WS, tudo é trocado e descrito através de XML. Os WS podem ser criados para expor funcionalidades de uma aplicação, para aceder a dados de uma aplicação, ou para simples troca de dados através da rede de comunicação. O XML é, actualmente, a linguagem universal para transmitir estruturas de dados, independentemente do ambiente ou suporte.

Ao contrário dos métodos de integração tradicionais, os WS requerem que ambas as partes suportem XML/SOAP ou XML/REST(Representational State Transfer [He04]), alternativamente. Isso obriga a que toda a indústria se encontre em sincronia com os WS.

2.3.1 *Web Services* e EAI

A EAI permite que as organizações se adaptem a um ambiente de negócio em constante mudança. Os WS oferecem uma abordagem de EAI que ultrapassa as limitações das soluções tradicionais. Fornece todos os benefícios da integração, mas sem os riscos e a complexidade de outras estratégias de EAI.

Os WS fornecem uma melhor solução de integração porque são baseados em standards abertos, fáceis de usar e suportados de forma abrangente. Fornecem ainda os benefícios de EAI, tal como o acesso em tempo real à informação que dirige o negócio, redução da duplicação de dados em sistemas isolados e processos de negócio mais eficientes. No caso de sistemas de apoio ao negócio de retalho, esses benefícios transmitem-se através da integração dos diferentes módulos aplicacionais (ORMS — *Oracle Retail Merchandising System*, ORWMS — *Oracle Warehouse Merchandising System*, SIM — *Store Inventory Manager* — e as outras aplicações "Oracle Retail"). A integração dos diferentes módulos Oracle permite a centralização de dados e a troca de informação em tempo quase real.

A utilização de WS como abordagem de EAI expande esses benefícios com vantagens únicas provenientes das suas características, que provêem de uma arquitectura orientada a serviços, baseada em standards [Kis07].

As soluções de WS normalmente fazem uso das seguintes tecnologias:

- XML e SOAP para fornecer um formato de dados para decomposição (*parsing*) da informação entre processos (*consumer*/*provider*);
- WSDL (*Web Services Description Language*), que fornecem o mecanismo de descrição das características dos serviços publicados, como é o caso do BPEL utilizado no projecto [Cor08d];

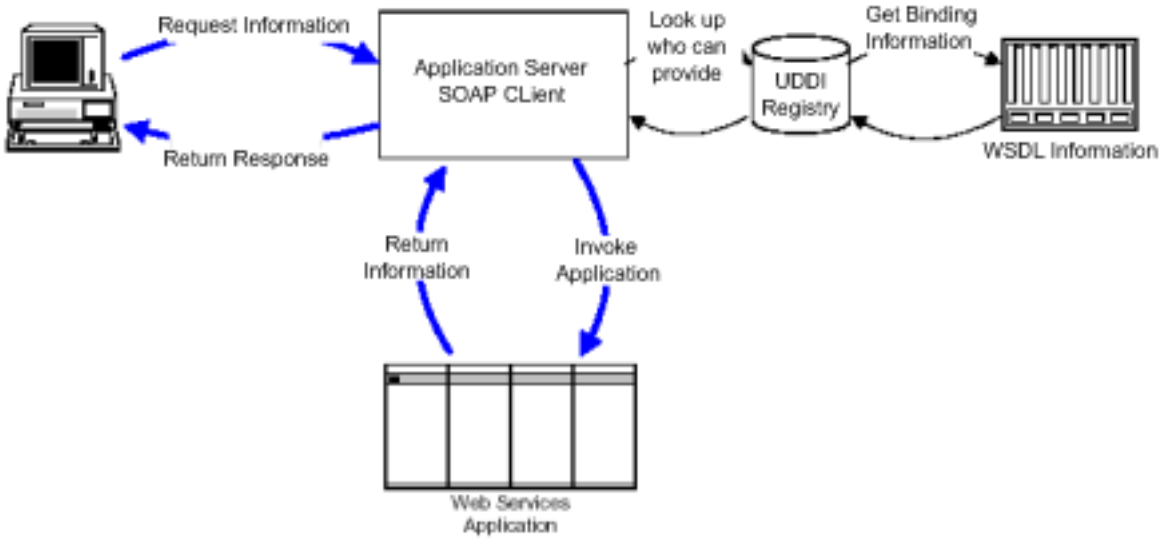

Figura 2.5: Processo de Integração de Serviços [FBVRR05]

- UDDI (*Universal, Discovery, Description and Inventory*), caso se pretenda possuir algo semelhante a um sistema de "páginas amarelas" de WS na rede, de forma a ser possível procurar os serviços necessários;
- HTTP e TCP/IP como mecanismo simples e fiável de transporte de informação e dados entre as diferentes máquinas e plataformas envolvidas na interacção.

A combinação destas tecnologias formam a arquitectura presente na Figura 2.5, com base na qual os WS podem ser utilizados de forma a fornecer um mecanismo robusto para reutilização de processos e sistemas através da organização. Isto permite reduzir custos de desenvolvimento, aumentar a funcionalidade e facilitar a integração, tanto para os sistemas existentes como de novas aplicações, independentemente das plataformas tecnológicas que os suportam [Kis07].

2.3.2 Ferramentas EAI de Suporte

Os WS não fornecem soluções de suporte à transformação, redireccionamento, *workflows*, gestão de processo e integridade transaccional. Isto deve ser previsto separadamente por uma solução EAI.

Os WS foram desenvolvidos assumindo que a segurança depende apenas do transporte. Logo, são apenas capazes de utilizar as soluções existentes, baseadas na camada de transporte, tal como SSL (*Secure Sockets Layer* [Sys08b]) e SMIME (*Secure/Multipurpose Internet Mail Extensions* [Wik08c]). A autenticação e autorização não são suportadas e

vão, como resultado, afectar o crescente uso da colaboração e integração dinâmica, particularmente através de uma rede aberta.

Uma verdadeira solução de integração permite alcançar benefícios como o aumento da flexibilidade, a diminuição do tempo de desenvolvimento do projecto e melhor exploração dos componentes nucleares existentes. A transformação, redireccionamento, capacidades de gestão de processos e erros são também outros benefícios que advêm de uma integração completa. Os WS não conseguem substituir produtos e soluções de EAI, mas com certeza alargam as possibilidades [FBVRR05].

2.4 JMS

O *Java Messaging Service* (JMS [Pra08]) é a resposta da Sun Microsystems [Mic08b] para criação de software utilizando troca de mensagens assíncrona ponto-a-ponto. O JMS consiste numa API para sistemas de filas de espera de mensagens. Os sistemas de filas de espera de mensagens estabelecem um interface entre aplicações de negócio, servindo como interface de EAI. Assim, estes sistemas são normalmente designados por *middleware* [Fen03], uma vez que operam na camada intermédia — entre outros sistemas e entre empresas.

As mensagens envolvidas trocam dados cruciais entre máquinas e contêm informação relacionada com notificações de eventos das aplicações e requisição de serviços. A troca de mensagens é frequentemente utilizada para coordenar aplicações em sistemas diferentes ou com linguagens de programação base diferentes. A utilização da interface JMS permite ao programador invocar serviços de mensagens como "Oracle JMS Advanced Queues", como é o caso deste projecto. Adicionalmente, o JMS suporta mensagens que possuem objectos Java serializáveis e mensagens com conteúdo XML.

O JMS não diferencia o hardware da rede de comunicação (com fios/sem fios) nem o protocolo associado e fornece diferentes níveis de garantia de entrega de mensagens.

Numa vista de alto nível, os sistemas de troca de mensagens são conceptualmente simples. O produtor envia a mensagem para um destino e o consumidor recebe a mensagem. O sistema de troca de mensagens cuida dos detalhes de gestão dos destinos da mensagem, assim como a forma como as mensagens são enviadas e recebidas.

A arquitectura JMS, presente na Figura 2.6, fornece uma comunicação:

Fracamente ligada: O produtor da mensagem não precisa de saber a localização e parâmetros de chamada dos serviços disponibilizados pelo consumidor

Fiável: As mensagens são armazenadas até serem entregues

Assíncrona: O produtor da mensagem é bloqueado até que o servidor JMS receba a mensagem. O consumidor não necessita de se encontrar *online* para receber a mensagem. A mensagem será entregue assim que este se ligar.

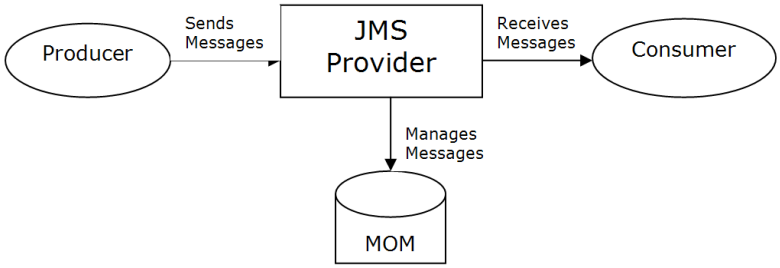

Figura 2.6: Arquitectura JMS [Pra08]

Para além da implementação básica de JMS [Pra08], podem ainda estar implementados outros serviços como QoS (*Quality of Service*), *load balancing* [GRSDB04], escalabilidade, monitorização, métricas, auditoria, entre outros. De facto, a qualidade do produto de *middleware* orientado a mensagens (MOM — *Message-Oriented Middleware* [Wik08b]), como é o caso do JMS, depende do suporte das capacidades do produto que sejam implementadas, entre as anteriormente mencionadas.

O JMS suporta vários tipos de transporte como TCP, HTTP e HTTPS. Suporta ainda persistência de mensagens em armazenamento baseado em ficheiros e Sistemas de Gestão de Bases de Dados Relacionais via JDBC (*Java Database Connectivity* [Cor05a]).

Na versão anterior (versão 12) da plataforma de integração ORIB, é utilizado o sistema JMS Queues como sistema de filas de espera para as mensagens de integração. Este é um sistema baseado em ficheiros, como irá ser descrito de seguida.

2.5 ORIB12

ORIB é a sigla para "Oracle Retail Integration Bus". Faz parte do conjunto de produtos "Oracle Retail" desde a versão 10 e é responsável pela sincronização de dados entre as aplicações "Oracle Retail", tais como ORMS, ORWMS, SIM e ainda aplicações *legacy* e *third-party*. Funciona como plataforma integradora da versão 12 das aplicações "Oracle Retail". No ORIB, a comunicação é realizada de forma assíncrona, através do uso de vários componentes, como pode ser observado na Figura 2.7.

Existem duas camadas de implementação do ORIB [AS06]. A primeira consiste num conjunto de funcionalidades de integração horizontal (APIs, *Rib Helpers*, gestão de erros e recuperação), enquanto a segunda corresponde aos fluxos de dados entre as aplicações "Oracle Retail".

O ORIB utiliza o paradigma *Publish/Subscribe* [TEFGK02] para fluxos de trocas de dados, mas também existe a possibilidade de utilizar transferência por ficheiros.

O paradigma *Publish/Subscribe* consiste num método de comunicação assíncrona de dados muito próxima de tempo real. Não exige que existam conexões entre o publicador

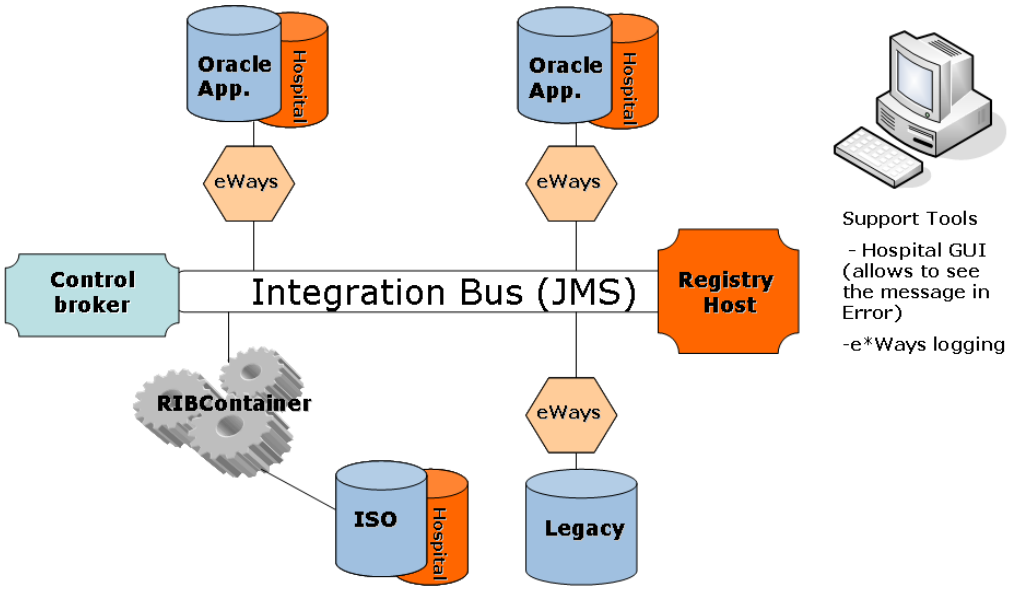

Figura 2.7: Arquitectura Detalhada do "Oracle Retail Integration Bus 12" [Rei07]

e subscritor (quem envia e quem receba mensagem, respectivamente). Desta forma, não existem dependências físicas entre as aplicações, sendo a comunicação feita através de um *bus* central de integração [Rei07]. A utilização deste paradigma como meio de comunicação, permite: consistência de dados; performance; escalabilidade; fiabilidade e ordenação de mensagens.

A troca de mensagens no ORIB v12 [Rei07] é feita através de tabelas temporárias que armazenam a mensagem a ser publicada para de seguida ser retirada pelo subscritor. Cada mensagem pertence a uma família distinta de mensagens. Cada Família de mensagens contêm informação específica de uma entidade de negócio. Exemplo: *Items*, *Orders*, *Transfers*.
Existe ainda, dentro de cada família de mensagens, vários tipos de mensagens. Cada família pode ter mais do que um tipo de mensagem, o que normalmente acontece.

Existem duas vertentes de uso do ORIB: J2EE (Java 2 Enterprise Edition) e não-J2EE. Estas abordagens encontram-se detalhadas em seguida.

2.5.1 Conectividade não J2EE

A primeira, e mais usada, é referente à sincronização de aplicações não J2EE, através do uso da ferramenta de integração da Sun, "SeeBeyond" [See00]. O "SeeBeyond" possui, por sua vez, uma camada baseada em tecnologia Java ("SeeBeyond eGate 5.0.5"). Os dados de cada mensagem são guardados em mensagens XML e estas são envolvidas num envelope XML (envelope ORIB). O envelope ORIB possui ainda informação de roteamento e atributos relativos à mensagem. O armazenamento dessas mensagens é feito

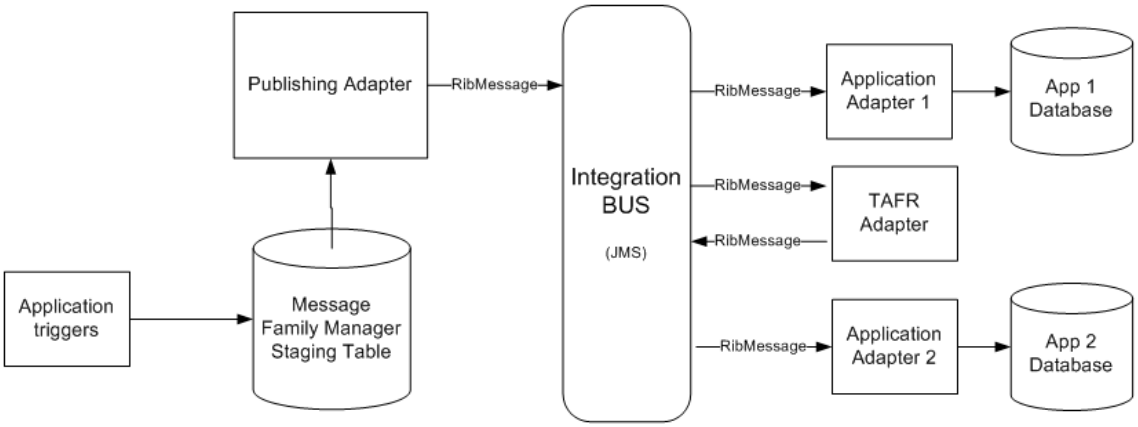

Figura 2.8: Ciclo de Vida de Mensagens ORIB (não-J2EE) [Rei07]

em JMS Queues e os dados publicados ficam disponíveis para novos destinos (vários subscritores). Desta forma, torna-se fácil a implementação de novos fluxos de dados.

Seguindo a arquitectura detalhada na Figura 2.8, é possível demonstrar o processamento das mensagens na plataforma ORIB v12 [Rei07]. Inicialmente, os gatilhos (*application triggers*) retiram do evento Oracle o tipo de operação a efectuar (criação, modificação, eliminação). Retiram ainda o identificador do objecto e fazem uma chamada ao gestor da família de mensagens (MFM — *Message Family Manager*)[1]. Esse MFM, utiliza o identificador do objecto para adicionar um registo à tabela temporária (*staging table*).

O Adaptador de publicação (eWay) realiza uma chamada ao seu MFM[2] de forma a retirar a mensagem a publicar. Esse mesmo MFM retorna o objecto Oracle construído com base no identificador do objecto presente na tabela temporária (*staging table*). O eWay converte o objecto Oracle em XML e publica-o como mensagem ORIB, com o conteúdo da mensagem.

O TAFR (*Transform Address Filter/Router*) realiza as transformações necessárias entre diferentes formatos de mensagens. Realiza ainda o redireccionamento de mensagens, dependendo do conteúdo das mesmas ou de condições existentes na secção de informação de redireccionamento da mensagem ORIB.

Finalmente, o adaptador de subscrição retira o conteúdo da mensagem presente na mensagem ORIB e converte-o num objecto Oracle. Realiza uma chamada ao pacote de consumo PL/SQL apropriado, convertendo o objecto Oracle em variáveis PL/SQL. Os registos são validados e só então inseridos nas tabelas finais ou novas tabelas temporárias, se necessário.

[1] Mais específicamente ao *stored procedure* — `addtoq`
[2] *Stored procedure* — `getnxt()`

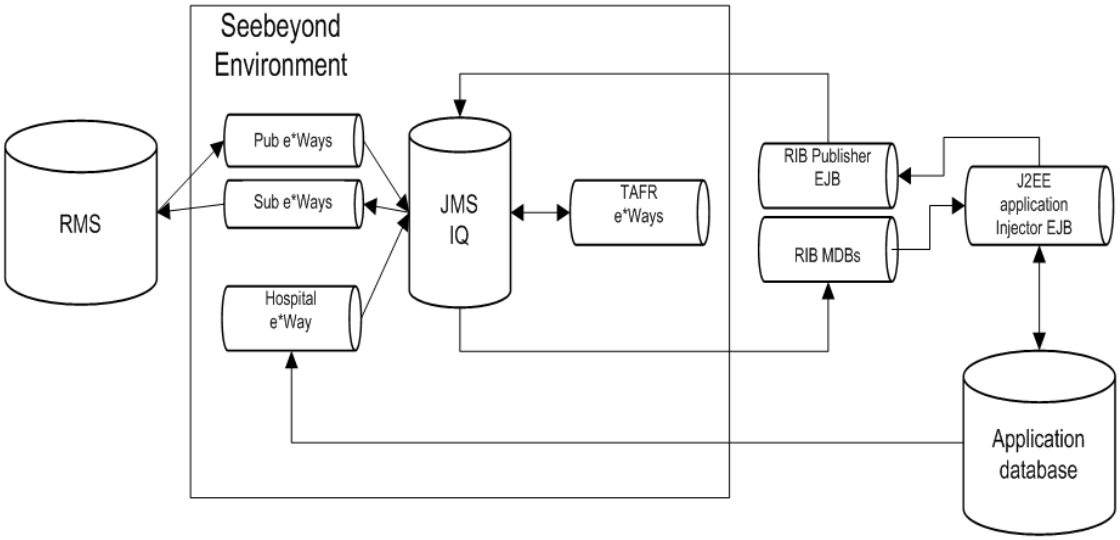

Figura 2.9: Ciclo de Vida de Mensagens ORIB (J2EE) em Integração com ORMS (não-J2EE) [AS06]

2.5.2 Conectividade J2EE

Na sua outra vertente, o ORIB pode ser usado para sincronizar aplicações J2EE [AS06], através *Enterprise Java Beans* (EJB) e *Message Driven Beans* (MDB). O armazenamento de mensagens é feito igualmente em JMS Queues.

A plataforma J2EE consiste numa arquitectura cliente/servidor multi-camada que permite que as aplicações sejam implementadas como conjuntos de componentes reutilizáveis num ambiente de processamento distribuído. Os componentes da camada de cliente executam numa máquina cliente, enquanto componentes de camada de negócio executam num servidor J2EE e os componentes de base de dados correm num servidor de base de dados.

Uma aplicação J2EE [Rei07] interage com o ORIB através de objectos de carga Java, que consistem em Java Beans simples que armazenam os dados de eventos da aplicação. Os objectos de carga ORIB asseguram a mesma informação que se encontra definida nos DTD (*Document Type Definition*) ou XML *Schemas* que definem as mensagens.

Num ambiente J2EE, a publicação para o ORIB é efectuada através de implementações de EJB (*Enterprise Java Beans*). A subscrição do ORIB é efectuada através da implementação dos MDB. A Figura 2.9 demonstra a configuração necessária para integrar uma aplicação não-J2EE (ORMS), com uma aplicação J2EE, utilizando ORIB. O ORMS liga-se ao ORIB utilizando linguagem PL/SQL através de pontos de conexão (e*Ways). A aplicação J2EE liga-se ao JMS utilizando MDB e EJB [Mic08a]. É efectuada uma chamada aos MDB pelo recipiente J2EE com dados da mensagem de um tópico JMS previamente configurado. Os EJB podem ser chamados pelos MDB de forma a processar a

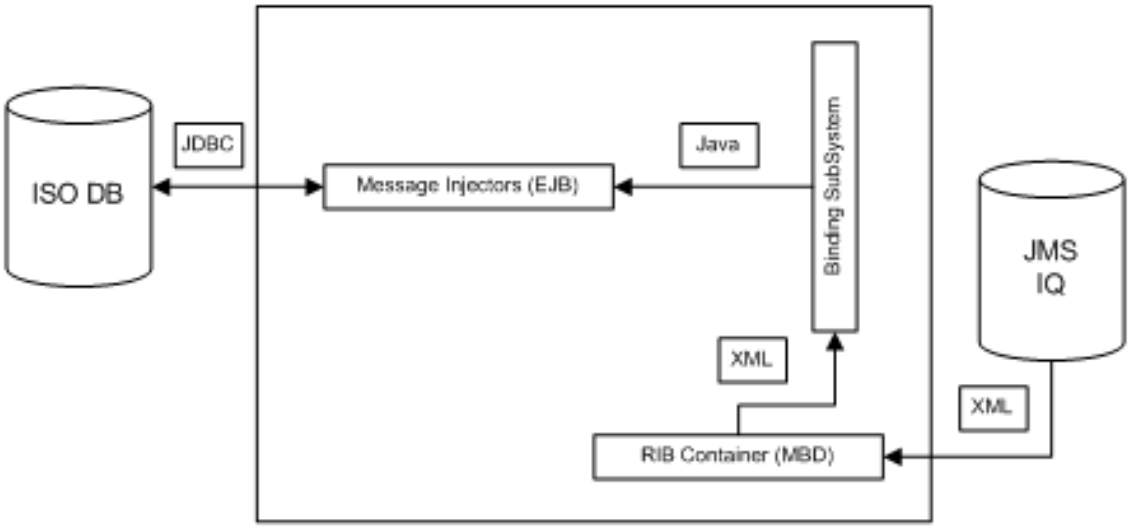

Figura 2.10: Esquema Representativo da Subscrição em J2EE no ORIB [AS06]

informação da carga da mensagem. O ORIB também implementa EJB como interface de publicação de mensagens para o servidor JMS.

Processo de Subscrição

Os MDBs são utilizados para processar mensagens de um ou mais servidores JMS. O servidor J2EE é responsável pela leitura da mensagem do servidor JMS e pela entrega desta ao procedimento `onMessage()` do MDB. O programador da aplicação deve criar o método `onMessage()` para implementar a lógica específica da aplicação. A implementação de cada MDB é diferente para cada subscrição de família de mensagens, uma vez que cada MDB escuta um diferente tópico no JMS.

Como se pode observar na Figura 2.10, um MDB é responsável pela chamada ao código obrigatório do ORIB para o processamento de cada mensagem. O código obrigatório do ORIB é responsável por realizar a chamada ao InjectorEJB [Rei07] da aplicação J2EE. Este componente (InjectorEJB) aplica a lógica de negócio para determinar a forma como os dados serão inseridos na base de dados da aplicação. Caso uma excepção seja retornada da aplicação J2EE, a transacção será desfeita e a mensagem XML será enviada para o Hospital de erros do ORIB.

Processo de Publicação

Os EJB são um meio de implementar componentes, sem que o programador tenha que se preocupar com implementações de baixo nível tais como *threading*, protocolos de transacção e *load balancing* [GRSDB04].

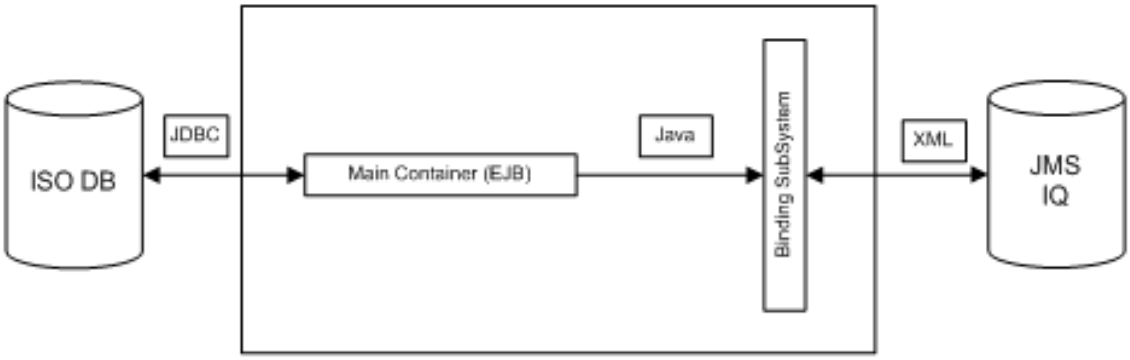

Figura 2.11: Esquema Representativo da Publicação em J2EE no ORIB [AS06]

Existem duas características importantes nos EJB: Sessão *versus* entidade e sem estado *versus* com estado. Todos os EJB de ORIB são *Stateless Session Beans*. Isso significa que esses *Beans* não se encontram associados a uma entidade de base de dados específica, mas mantêm uma sessão com o cliente associado. Significa ainda que o estado entre activações do *Bean* não é preservado.

O RIBMessagePublisherEJB [Rei07] fornece ao ORIB uma interface para conversão de carga em XML e publicação de mensagens no JMS, como se pode observar na Figura 2.11. Os *stubs* e ficheiros de referência necessários para chamar esse método encontram-se no `rib-client.jar` fornecido pelo ORIB para a aplicação J2EE. O método `publish()` possui o seguinte protótipo para publicação de mensagens ORIB com uma única *tag* `ribMessage`:

```
public void publish(String family, String type, Payload payload,
ArrayList ids, ArrayList ris) throws PublishException{}
```

Os campos `family` e `type` da mensagem são passados como *Strings*, junto com o objecto de carga (`Payload`) que contém os dados do negócio. O parâmetro `ids` é uma *ArrayList* de *Strings* que contem o identificador do objecto de negócio, usado para sequenciamento no ORIB. O parâmetro `ris` é uma *ArrayList* de objectos *RoutingInfo* que possuem a informação para roteamento das mensagens no ORIB.

2.6 ORIB13

A nova solução de integração da Oracle [Rei08] foi disponibilizada ainda durante o período de desenvolvimento deste projecto e trouxe consigo a substituição de algumas tecnologias e melhoria de outras.

Nesta nova versão, a ferramenta de integração "SeeBeyond" foi substituida por JEE5 (*Java Enterprise Edition* 5 [SRC08]), nomeadamente os EJB e MDB, como se verifica na

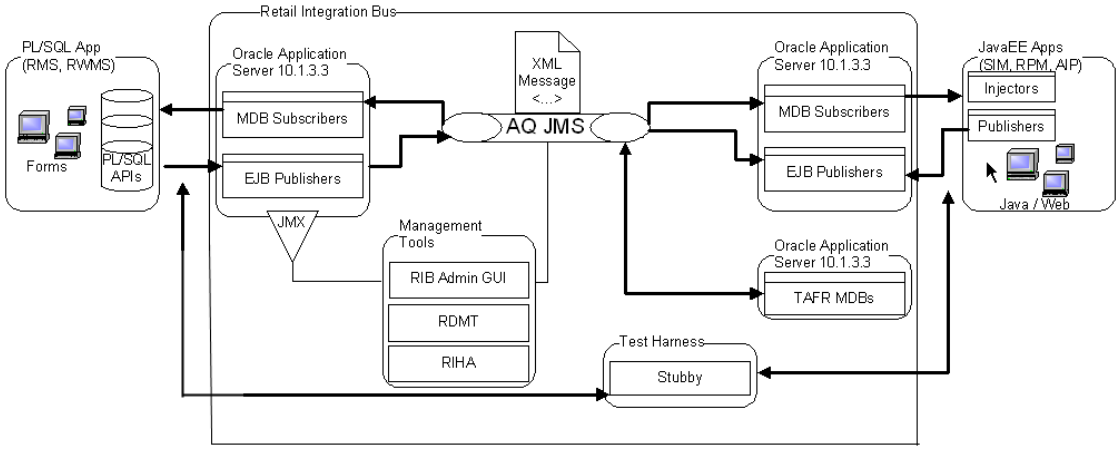

Figura 2.12: Arquitectura ORIB versão 13 [Rei08]

arquitectura demonstrada na Figura 2.12. O conteúdo das mensagens torna-se agora baseado no formato descrito pelo *schema* de cada tipo de mensagem, o que torna possível a compatibilidade com tecnologias "Fusion Middleware" [Cor08c], como BPEL [OJG07] e ESB [OJG07]. Mantêm-se o suporte a standards abertos de JMS, nomeadamente as *Oracle Advanced Queues* (AQ), vindo estas substituir as JMS Queues da versão anterior. A gestão dos componentes é feita agora por JMX (*Java Management Extensions*). Esta tecnologia assenta no conceito de agentes e tem como propósito a monitorização de elementos da JVM (*Java Virtual Machine*) e de redes baseadas em serviços, como é o caso deste projecto.

A versão 13 do ORIB [Rei08] tornou-se mais simples de utilizar, integrando agora um GUI (*Graphic User Interface*) para administração da aplicação, baseado na Web. A instalação da própria aplicação também foi simplificada, tendo os *scripts* de instalação sido substituídos por um instalador gráfico de uso bastante acessível.

2.6.1 Oracle AQ

A utilização de Oracle AQ [Cor05b] (*Advanced Queues*) baseia-se num processo simples de armazenamento temporário de mensagens de integração para permitir a comunicação entre aplicações Oracle. O processo compreende apenas dois passos. Uma aplicação coloca mensagens (*enqueuing*) na AQ [Cor02a], enquanto outra aplicação, com a qual a primeira pretende comunicar, retira essa mensagem da AQ (*dequeuing*) e efectua as acções necessárias com essa nova informação. É possível que a mesma mensagem possua mais que uma aplicação subscritora.

Com a utilização de Oracle AQ, deixa de ser necessária a existência de *middleware* específico para comunicação entre processos. Uma vez que as Oracle AQ são baseadas

em tabelas e outros elementos de base de dados, é ainda possível usufruir de ferramentas da base de dados para monitorização e processamento de transacções.

Num servidor de base de dados Oracle, torna-se bastante eficiente e menos custoso utilizar a interface JMS para Oracle AQ [Cor02b]. Uma vez que as Oracle AQ são parte integrante da base de dados, é desnecessário utilizar um sistema de troca de mensagens independente e uma vez que o utilizador se encontra já a trabalhar com Oracle, o esforço de aprendizagem torna-se automaticamente menor.

Existem três componentes base nas Oracle AQ:

A mensagem: Este é o elemento unitário das filas de espera (*queue*); possui informação de controlo (metadados) e os dados específicos da mensagem (*payload*).

A *Queue* (fila de espera): Esta é a estrutura de dados para as mensagens; podem ser criadas *queues* de utilzadores para processamento normal de mensagens ou *queues* de excepção para mensagens mal retiradas ou expiradas.

A *Queue Table*: Uma tabela de base de dados que contêm uma ou mais *queues*; estas tabelas possuem por omissão uma *queue* de excepção.

Existem muitas outras funcionalidades relativas às mensagens como prioritização, agrupamento de mensagens, subscrição baseada em regras, entre outras.

2.7 Resumo e Conclusões

Como podemos observar após a leitura deste capítulo, existem diversas tecnologias e plataformas associadas ao projecto em questão que, após devidamente estudadas e experimentadas, permitiram a compreensão de diversos mecanismos de integração.

Desde as tecnologias de *middleware* genéricas, às plataformas de integração Oracle mais recentes, a base de integração assenta no método *Publish/Subscribe*. Este mecanismo é de compreensão essencial à implementação do protótipo de forma a compreender a base de integração de aplicações Oracle. As diversas vantagens deste método permitem uma integração completa e fiável.

No entanto, para o desenvolvimento célere de um protótipo de integração de aplicações "Oracle Retail" é necessário um leque de ferramentas específicas, descritas no próximo capítulo.

Capítulo 3

Ambientes e Ferramentas

Este capítulo descreve com pormenor cada um dos ambientes e ferramentas que possibilitaram o desenvolvimento e implementação do projecto. É efectuada uma análise geral e são descritas as potencialidades, destas ferramentas e ambientes de execução mais importantes para o projecto. Este capítulo aparece na sequência da revisão geral da integração de aplicações do capítulo anterior. Estas tecnologias são centradas no projecto e directamente relacionadas com a sua implementação.

3.1 Introdução

Para o desenvolvimento deste projecto foi necessário montar um ambiente propício à execução dos processos necessários para a integração. A restrição relativa ao sistema operativo, exigiu a criação e configuração de uma máquina virtual (VM — *Virtual Machine* [Smi05]) baseada num sistema operativo baseado em Unix.

Foi ainda necessário criar uma plataforma constituída por diversos componentes. Entre esses componentes, encontra-se uma base de dados Oracle e uma plataforma de *middleware* baseada nas ferramentas Oracle. O *Application Server* [GRSDB04] e o *BPEL Server* [Cor08d] criam a plataforma de execução, enquanto o ORIB v13, juntamente com as Oracle AQ, formam o *middleware* necessário. Para o desenvolvimento do projecto, foi ainda necessária a utilização do ambiente de desenvolvimento "Oracle JDeveloper" [Sch08] para a orquestração de processos BPEL. Para acesso à base de dados optou-se pela utilização da ferramenta utilizada na empresas, o "PL/SQL Developer" [Aut08b], devido à multitude de funcionalidades e à perfeita manipulação de bases de dados Oracle que a aplicação permite.

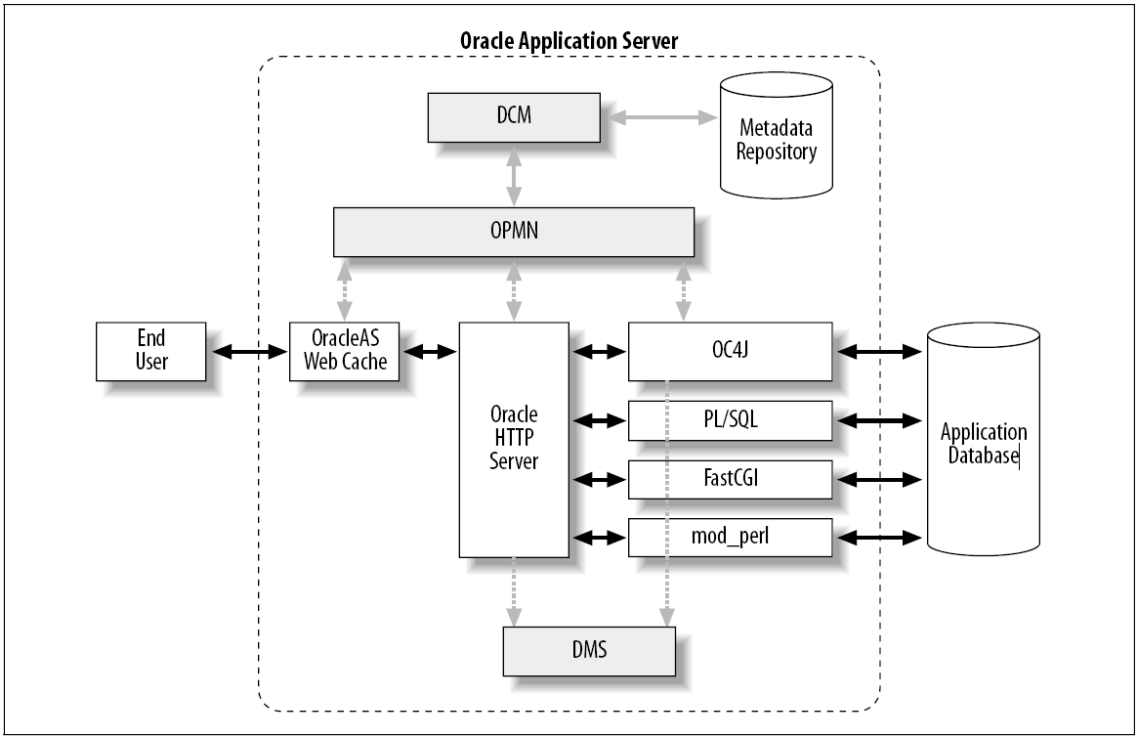

Figura 3.1: Arquitectura do Oracle Application Server [GRSDB04]

3.2 Plataformas de Suporte/Middleware

3.2.1 Oracle Application Server 10.1.3.3

O OAS (Oracle Application Server) [SDB04] consiste numa plataforma de software de camada intermédia baseada em standards, fornecendo serviços e ambientes de execução. Com a instalação do OAS torna-se possível executar procedimentos PL/SQL, aplicações J2EE, CGIs, *scripts* de Perl e ainda aplicações desenvolvidas em C, C++, Java, entre outras.

Nuclearmente, o OAS possui dois grandes componentes, como se pode observar na arquitectura OAS demonstrada na Figura 3.1: O **Servidor HTTP** e o **OC4J**.

O **Servidor HTTP** (baseado num servidor Apache), é um servidor Web especificamente dedicado ao OAS [GDKB04]. Cada pedido é efectuado na forma de um URI (*Uniform Resource Identifier* [GRSDB04]). No caso de ser um pedido de contexto estático, o servidor envia imediatamente o ficheiro HTML. Caso contrário, o pedido de conteúdo dinâmico é redireccionado para o recurso adequado[1] para execução da aplicação que dará a resposta ao pedido.

[1] No nosso caso o mais usado será o mod_oc4j, que consiste num ambiente de execução J2EE.

O **OC4J** (OracleAS Containers for J2EE) consiste num conjunto de recipientes e um tradutor de JSP (*JavaServer Pages*), disponibilizando um ambiente J2EE. Aqui é feito o *deployment* de instâncias de aplicações baseadas em J2EE.

Existe ainda um componente que é importante referir: o OPMN [GRSDB04] (*Oracle Process Manager and Notification Server*). O OPMN controla e monitoriza os componentes do OAS. Sempre que um dos componentes fica indisponível, seja devido a falha ou qualquer outra razão, o OPMN reinicia-o. Informa ainda outros componentes dependentes desse componente, da sua indisponibilidade. A gestão do OAS torna-se assim mais simples, ao mesmo tempo que o conjunto dos seus componentes se encontra sempre com alta disponibilidade.

A versão 10.1.3.3 do *Oracle Application Server* foi utilizada neste projecto, assente na máquina virtual mencionada anteriormente, de forma a fornecer os serviços necessários à execução do ORIB13.

3.2.2 Oracle Database 10g (10.1.0.3)

O Oracle Database 10g é um sistema de gestão de base de dados relacional [RJG02] de suporte aos *schemas* das aplicações ORMS (Oracle Retail Merchandising System) e ORIB da versão 13. Tal como em qualquer base de dados relacional, os dados são armazenados em tabelas bi-dimensionais (linhas e colunas).

A base de dados da Oracle é composta por estruturas lógicas e físicas nas quais a informação de controlo, de utilizadores e de sistema são armazenadas. O software de gestão da base de dados denomina-se "Oracle Database Server". O conjunto da base de dados e do software de gestão é denominado de "Oracle Database System".

Existe um conjunto de ferramentas da Oracle com as quais é possível gerir a base de dados:

Oracle Universal Installer (OUI): Este é o instalador de software da Oracle. Lança automaticamente o "Database Configuration Assistant" (DBCA) para instalar a base de dados

Database Configuration Assistant (DBCA): Este assistente cria a base de dados a partir de templates fornecidos pela Oracle, permitindo ainda a utilização de *templates* criados pelo utilizador.

Oracle Enterprise Manager: Outra ferramenta importante é o "Oracle Enterprise Manager" (OEM). O OEM tem o simples propósito de gerir, através de uma interface Web, uma base de dados previamente criada. Possui ainda outras utilidades como ferramentas de recuperação e "SQL*Loader", que permite carregar dados de ficheiros externos para a base de dados.

Neste projecto, devido às dependências do ORMS 13 e RIB 13, é utilizada a Oracle Database 10g (10.2.0.3.0).

3.2.3 Oracle BPEL

A especificação WS-BPEL 2.0 (*Web Services Business Process Execution Language*) define uma linguagem para orquestração de processos de negócio baseados em WS. Um processo BPEL é utilizado para declarar relações para parceiros de serviços externos, *handlers* e actividades a executar. Faz parte da camada intermédia onde se encontra o OAS, integrando-se com este, de forma a fornecer um ambiente para executar processos BPEL. Este ambiente exige a instalação prévia do OAS 10g J2EE na camada intermédia e inclui serviços de *runtime* e adaptadores e ainda dois componentes importantes:

***Oracle BPEL Server*:** O servidor no qual é efectuado o *deployment* de processos BPEL desenhados com auxílio do "Oracle JDeveloper" [Sch08], contendo adaptadores de WS, serviços de notificação e *workflow* com interacção humana.

***Oracle BPEL Control*:** A consola a partir da qual se executa, gere e testa os processos BPEL previamente alocados. Disponibiliza ainda um interface baseado em *browser* Web para gestão e *debugging* dos processos BPEL.

O "Oracle BPEL Process Manager" é carregado no "Oracle JDeveloper" e facilita o desenvolvimento de aplicações baseadas em arquitecturas SOA (*Services Oriented Architecture*). Tal é conseguido através da composição, orquestração e coordenação de WS de forma síncrona ou assíncrona, integrados no fluxo do processo BPEL. Desta forma, é estendida a funcionalidade do "Oracle JDeveloper" de forma a tornar possível a modelação, edição e desenho de processos de negócio usando BPEL.

Como referido, o BPEL possui uma abordagem SOA para automação eficiente de processos de negócio, baseada em três atributos importantes:

- Consiste numa forma standard de acesso e exposição de funcionalidades de aplicações como serviços;
- Os processos de negócio, do ponto de vista do BPEL, consistem numa colecção de invocações coordenadas de serviços e outras actividades relacionadas que, por sua vez, produzem um resultado. Esse resultado pode restringir-se a uma só organização ou ser proveniente de várias organizações;
- Permite a convergência de duas linguagens de modelação de *workflows*, WSFL (*Web Services Flow Language*) e XLANG.

Actualmente, as empresas começam a notar a importância dos WS como método de integração na construção de aplicações interligadas. Os standards BPEL e WS ajudam a

alcançar esse nível de integração de uma forma aberta, portável e standard, fornecendo diversas vantagens, de entre as quais se salientam:

- A redução do tempo de lançamento para mercado de processos de negócio que requerem colaboração mais estreita entre o negócio e TI;
- A facilidade e aceleração da mudança de processos de negócio através da sincronização de modelos de negócio com os modelos de execução;
- A fácil optimização de processos, baseado nas estatísticas actuais de processos actuais em execução.

Em casos específicos, tem mais sentido aplicar BPEL em ambientes que já possuem uma quantidade considerável de WS expostos. Quanto maior o numero de WS que é possível de utilizar, maior o valor do BPEL nesses casos.

3.3 Ferramentas/Ambientes Utilizados

3.3.1 VM Server

Popek e Goldberg [Smi05] definiram originalmente um máquina virtual como sendo uma réplica eficiente de uma máquina real.

A virtualização é um conceito que remonta aos anos 60 para particionar hardware de grandes *mainframes*. Hoje, os computadores baseados em arquitecturas x86 confrontam os mesmos problemas de rigidez e subutilização que as grandes *mainframes* dos anos 60. A utilização de máquinas virtuais permite reduzir, de alguma forma, esses problemas.

Uma máquina virtual pode ser vista como um bloco isolado de software, dentro de uma máquina física, que permite correr o seu próprio sistema operativo e aplicações, como se de uma máquina real se tratasse. Uma máquina virtual comporta-se exactamente como uma máquina real e possui o seu próprio hardware virtual (CPU, RAM, HDD, interfaces de rede), baseado no hardware físico da máquina real. Os recursos utilizáveis pela máquina virtual são previamente definidos (podendo ser modificados em caso de necessidade). Qualquer peça de software a correr na máquina virtual não pode utilizar recursos que não estejam atribuídos a essa mesma máquina virtual, estando limitado ao mundo virtual em que reside. Existem dois tipos de máquina virtuais:

Máquinas virtuais de sistemas: Estas comportam uma plataforma capaz de suportar a execução de um sistema operativo completo. As vantagens mais significativas deste tipo de máquinas virtuais consiste na possibilidade de co-existência de vários SO (Sistemas Operativos) em execução simultânea na mesma máquina física, perfeitamente isolados entre eles. Ainda existe a vantagem de ser possível a disponibilidade de ISA (*Instruction Set Architecture*) diferentes da existente na máquina física.

Máquinas virtuais de processos: Por contraste com a anterior, este tipo de máquina virtual apenas suporta a execução de um programa, correspondente a um único processo.

Um sistema operativo residente numa máquina virtual não distingue uma máquina virtual de uma física, tal como as aplicações ou máquinas de uma rede. No entanto, uma máquina virtual é composta apenas por software e não possui componentes de hardware físico. A máquina virtual emula os discos rígidos e a memória volátil em software.

Neste projecto foi utilizada a máquina virtual da VMware [VMW08]. A VMware permite a virtualização através da inserção de uma fina camada de software no hardware da máquina real ou no sistema operativo que a suporta. Essa camada de software cria máquinas virtuais e possui um sistema de monitorização que aloca recursos de hardware dinamicamente e de forma transparente, permitindo a execução de múltiplos SO concorrentes na mesma máquina física, sem que qualquer um deles se aperceba da presença dos outros.

3.3.2 Oracle JDeveloper

O "Oracle JDeveloper" [Sch08] (JDev) consiste num ambiente integrado de desenvolvimento (IDE — *Integrated Development Environment*) para construção de aplicações orientadas a serviços, com uso de standards reconhecidos como XML, Java, WS e SQL. Desde modelação, codificação até *debugging* e *deployment*, este ambiente possui funcionalidades de suporte ao ciclo de vida do desenvolvimento da aplicações, através de todas as suas fases. A sua capacidade de abranger diversas tecnologias e a integração da ADF (*Oracle Application Development Framework*), permitem o aumento de produtividade, diminuindo tarefas repetitivas e banais.

A principal vantagem do uso deste ambiente no caso específico do projecto relaciona-se com o desenvolvimento de WS. A integração entre aplicações Oracle e as barreiras presentes entre diferentes linguagens de programação, requerem o uso de WS de modo a integrar essas mesmas aplicações. O JDev permite o uso de standards reconhecidos baseados em XML, como WSDL, SOAP e UDDI, permitindo o uso de componentes independentemente da linguagem de programação utilizada no seu desenvolvimento ou localização física/virtual.

Existem duas formas possíveis de criação de WS com o JDev, consistindo a primeira numa abordagem *Bottom-Up*, ou seja, do código para o WSDL, assim como uma possível abordagem *Top-Down*, sendo criadas classes Java baseadas nas especificações de um WSDL.

Para além dos WS, o JDev possui ainda uma ferramenta importante para o desenvolvimento do projecto: o "BPEL Designer". Este ambiente gráfico integrado no "JDeveloper

Studio Edition" permite uma abordagem de programação *drag-and-drop* de objectos representativos de actividades BPEL e adaptadores de WS. O BPEL Designer fornece assim uma infraestrutura intuitiva e fácil de utilizar para criação, *deployment* e gestão de fluxos de processos baseados em BPEL e orquestração de WS.

O desenvolvimento de XML também é simplificado com o uso do JDev. Este possui um editor gráfico de XML *Schemas*, que permite uma navegação simplificada. A construção de *Schemas* pode ser feita através do ambiente gráfico, através do mecanismo *drag-and-drop* dos componentes, ou através do código do próprio *Schema*. Existe ainda suporte a pesquisa XPath em documentos XML e à criação de documentos XQuery.

Outra ferramenta também significativamente importante no desenvolvimento do projecto consiste na presença de uma ferramenta visual de transformação para criação de ficheiros XSL no JDev. Esta ferramenta incorporada no ambiente de desenvolvimento permitiu o mapeamento visual e simplificado de transformações entre elementos de documentos XML.

O *deployment* dos processos BPEL desenvolvidos é simplificado através da criação de conexões no JDev com o OAS, o Servidor de Integração e a Base de Dados Oracle presentes no nosso servidor. Desta forma, o *deployment* de processos desenvolvidos com o auxílio deste ambiente são automaticamente criados no servidor, na camada de controlo BPEL, facilitando o seu *deployment*.

Existe ainda, na versão Studio do JDev, um ambiente de gráfico de desenho de fluxos de mensagens e serviços ESB.

É importante ainda mencionar que este IDE é gratuito e disponibilizado pela própria Oracle. Tanto esta ferramenta como as tecnologias enunciadas fazem parte da nova geração de *middleware* — "Fusion Middleware" [Cor08c].

3.3.3 PL/SQL Developer

Este IDE [Aut08a] permite aceder remotamente a bases de dados Oracle existentes em qualquer servidor. Mantendo a ligação ao servidor, esta aplicação comporta e estende as funcionalidades do "SQL*Plus", "Oracle Tools", edição de utilizadores e outras tarefas de manipulação de bases de dados Oracle. Possui todas as ferramentas necessárias para desenvolver código PL/SQL e o seu ambiente é *user-friendly*, facilitando as tarefas rotineiras de manipulação de bases de dados [Cou01].

Esta ferramenta permite o fácil acesso e edição de funções, procedimentos, *triggers* e outras componentes a desenvolver perante um ambiente Oracle. A acessibilidade é bastante grande uma vez que possui um *browser* de objectos re-configurável e permite qualquer acção que seja possível efectuar através de um clique do botão direito do rato. A facilidade de operar esta aplicação e a pouca experiência com bases de dados Oracle da parte do autor torna esta uma ferramenta de eleição [Swa04].

Algumas das características mais diferenciadores e importantes para o projecto:

- Integração com documentação e mensagens de erro Oracle (ORA-XXXXX);
- Possui completação de sintaxe automática sensível ao contexto;
- Re-compilação automática de objectos invalidamente compilados;
- Robustez: ausência de *bugs* ou *crashes*;
- É bastante fácil de usar para utilizadores principiantes.

3.4 Resumo

Neste capítulo foram apresentadas as tecnologias e plataformas de *middleware* necessárias à implementação de um ambiente propício ao desenvolvimento do projecto.

Foram ainda descritas as ferramentas a utilizar durante a sua implementação e enunciadas as vantagens que possuem especificamente para este caso. A compreensão antecipada e o estudo pormenorizado destas tecnologias e ferramentas permitiu um desenvolvimento contínuo. Permitiu ainda a existência do menor número possível de obstáculos inerentes ao esforço de aprendizagem requerido, no que toca ao desenvolvimento assente em tecnologias Oracle.

A utilização destas ferramentas irá possibilitar a implementação do protótipo e a compreensão do fluxo de mensagens de integração na plataforma de integração.

Capítulo 4

Especificação e Desenho do Protótipo

Neste capítulo é apresentada a necessidade deste projecto para a empresa em que se insere e é descrita a implementação do protótipo a desenvolver. É descrita a arquitectura de implementação de uma integração baseada em processos BPEL. Esta arquitectura representa a implementação de uma integração completa. A especificação descrita neste capítulo refere-se a uma integração total de aplicações "Oracle Retail", pormenorizando para cada processo a especificação para o protótipo desenvolvido.

4.1 Introdução

O desenho e especificação do protótipo baseou-se no estudo de mecanismos de integração e comunicação entre as aplicações, existente nas plataformas de integração ORIB v12 e ORIB v13 (apresentadas nas Secções 2.5 e 2.6, respectivamente), em confronto com a utilização da tecnologia BPEL, integrante da plataforma "Fusion Middleware". A utilização de Oracle AQ (*Advanced Queues*), tecnologia integrante da "Fusion Middleware" e da plataforma ORIB v13 permite, nesta nova abordagem de integração, a utilização da base de dados para armazenamento das mensagens de comunicação entre as aplicações. Uma vez que esta tecnologia se encontra fortemente integrada no ambiente de desenvolvimento "Oracle JDeveloper" mencionado no capítulo anterior, foi facilitada a sua utilização e integração nos processos BPEL.

Os processos BPEL de fluxo de mensagens entre as aplicações e as Oracle AQ, foram sendo orquestrados à medida que foi efectuada a configuração das Oracle AQ. A implementação contínua de novas versões desses processos no *middleware* de suporte (OAS + BPEL Server), permitiu a realização de testes contínuos de execução à medida que uma nova versão dos processos era implementada e executada. Nesta fase de

especificação não foram encontradas dificuldades, mas foi exigido um extenso estudo das tecnologias e das ferramentas de desenvolvimento a utilizar na implementação.

4.2 Descrição do Problema e Necessidade deste Projecto

O lançamento da plataforma de integração ORIB 13 e as tecnologias "Fusion Middleware" vieram permitir uma integração baseada em WS e Oracle AQ entre as aplicações "Oracle Retail" na versão 13.

Como havia sido referido na Secção 2.5, na versão 12 do ORIB existe a distinção entre a integração de aplicações J2EE ou não J2EE, recorrendo a implementações diferentes para cada caso. As aplicações J2EE recorrem a uma implementação baseada em *Java Beans*, enquanto a integração de aplicações não-J2EE deve recorrer a implementações de "e*Ways" e procedimentos PL/SQL. Ambas as implementações possuem como *bus* de integração o sistema de filas de espera JMS Queues para as mensagens de integração na versão 12 do ORIB, descrito na Secção 2.4. Na plataforma de integração ORIB v13, todas as aplicações são integradas através da implementação de EJB, descontinuando-se o uso de e*Ways e procedimentos PL/SQL. Nesta plataforma, o *bus* de integração passa a ser baseado em Oracle AQ, uma tecnologia Oracle que veio substituir as JMS Queues.

A plataforma "Fusion Middleware" integra actualmente outras aplicações Oracle como "Oracle E-Business Suite" [Cor08e] e outras aplicações adquiridas pela Oracle como "PeopleSoft" [Cor08h], "JD Edwards" [Cor08a], "Siebel" [Cor08j] e "Stellent" [Cor08b]. Esta plataforma consiste num conjunto de tecnologias que formam um *middleware* de integração das aplicações Oracle, baseada em WS, processos BPEL e ESB (*Enterprise Service Bus*), das quais fazem ainda parte as Oracle AQ, que representam o ponto comum entre esta plataforma e a versão 13 do ORIB. Prevê-se que esta plataforma de *middleware* venha também a integrar as aplicações "Oracle Retail", sendo a utilização das Oracle AQ um ponto importante neste processo de transição, representando a compatibilidade destas aplicações com tecnologias "Fusion Middleware". De forma a compreender a possibilidade da utilização desta plataforma, este projecto pretende utilizar a tecnologia BPEL da "Fusion Middleware" para orquestração de processos baseados em WS e as Oracle AQ como sistema de filas de espera, também uma tecnologia integrante da "Fusion Middleware", mas igualmente presente na plataforma ORIB v13.

A junção de tecnologias de ambas as plataformas de integração (ORIB v13 e "Fusion Middleware") permitirá integrar as aplicações "Oracle Retail" através de um novo método, antecipando a visão de integração do lançamento da próxima versão de aplicações "Oracle Retail". Este método de integração poderá vir ainda a permitir a integração, não apenas entre aplicações "Oracle Retail", mas também com outras aplicações Oracle que façam uso das tecnologias da plataforma "Fusion Middleware".

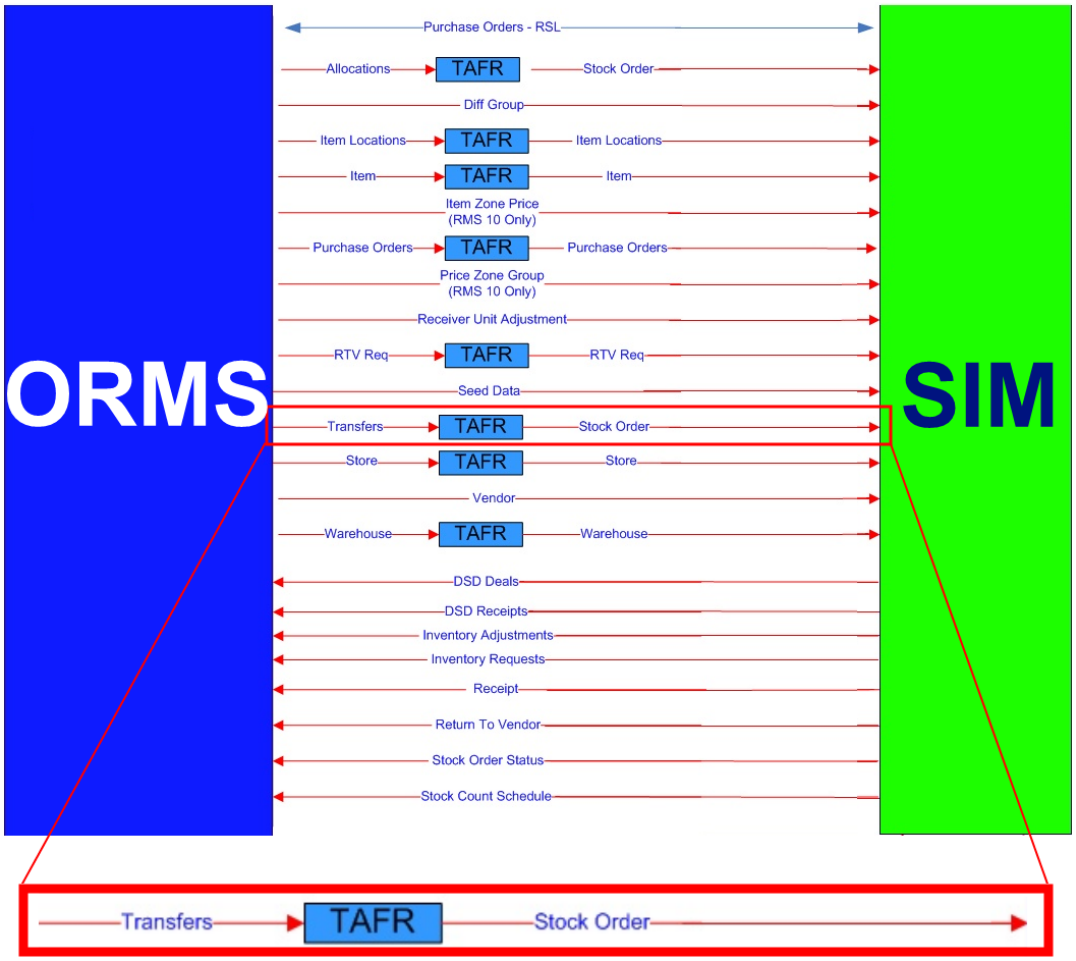

Figura 4.1: Troca de Mensagens entre ORMS e SIM

De forma a compreender as possibilidades de integração que a junção destas tecnologias proporciona, foi necessário desenvolver um pequeno protótipo representativo de um cenário generalista que englobe as várias hipóteses. Para este protótipo foram escolhidas as aplicações ORMS e SIM. Estes são os dois módulos mais utilizados das aplicações "Oracle Retail" e a integração entre estas duas aplicações é bastante comum na Wipro Retail.

O "Oracle Retail Merchandising System" (ORMS) permite um ambiente com processamento centralizado ou distribuído para a gestão de múltiplas lojas e entrepostos.

Este sistema adapta-se muito bem à realidade actual dos grandes retalhistas, na medida em que está bem preparado para a internacionalização. Nomeadamente, suporta uma moeda primária e múltiplas moedas secundárias. Isto permite operar localmente com a respectiva moeda, consolidando, no entanto, os movimentos na moeda primária para fins de relatórios corporativos. Para além da moeda, suporta ainda vários idiomas numa única instalação, o que permite melhor compreensão aos vários utilizadores de vários países.

Existe um idioma primário do sistema e podem definir-se vários idiomas secundários ao nível dos utilizadores. As diferentes traduções têm que ser introduzidas no ORMS, podendo em qualquer altura ser invocada a função de tradução.

Para além destas, possui ainda mais um conjunto de funcionalidades que lhe garantem uma elevada robustez e flexibilidade para responder às necessidades das grandes empresas de distribuição no que se refere à gestão de informação:

- Hierarquias

  Hierarquia organizacional e mercadológica. A primeira traduz a estrutura operacional da empresa e a segunda reflecte o modo como os artigos são geridos na empresa.

- Gestão de Artigos e Fornecedores

  O RMS permite fazer uma gestão completa dos artigos e fornecedores, estabelecendo uma relação entre estas duas entidades: para cada artigo existe um fornecedor prioritário e para comparar preços de custo é utilizado o valor deste fornecedor. O RMS gera um número de sistema (`SKU` — *Stock Keeping Unit*) para cada artigo, sendo este utilizado para aceder directamente aos dados desse artigo.

- Gestão de Mercadorias

  Engloba as funções relacionadas com as actividades de compra e venda de artigos efectuadas diariamente por um retalhista. Desempenha, entre outras, as seguintes tarefas: gestão de encomendas, informação sobre as recepções, análise de vendas, distribuição, transferências, informação de devoluções aos fornecedores.

- Gestão de Preços e Promoções

  A função de gestão de preços e promoções controla a criação de preços e a sua manutenção. Apresenta diferentes estratégias de preços tais como preços por zonas, preço a níveis de mercado, estratégias de promoções e/ou saldos, entre outros.

- Gestão de Inventário

  Fornece um mecanismo para controlo automático dos inventários. Calcula as necessidades para cada loja e consolida tal informação com a informação existente no respectivo entreposto. Ordens de compra e requisições são geradas automaticamente.

- Reaprovisionamento e Distribuição

  O ORMS suporta vários métodos de reaprovisionamento (min/max, constante, *floating point*, dinâmicos, sazonais, entre outros) que incluem a entrega directa na loja e *Cross Docking* através de um entreposto.

O "Oracle Retail Store Inventory Management" (SIM) permite realizar uma série de operações de loja de forma a receber mercadoria, gerir inventários, realizar contagens de *stock*, encomendar ou transferir artigos, assim como atribuição de preços e impressão de etiquetas. Este sistema permite que os dados se encontrem sincronizados com sistemas externos, incluindo sistemas da organização, como por exemplo o ORMS ou outros sistemas mercadológicos ou de armazém.

As funcionalidades mais importantes deste sistema de inventário de loja consistem na sua integração com os dados presentes no ORMS, na possibilidade de receber, monitorizar e transferir mercadorias forma fácil e eficiente e de aceder a dados de inventário, incluindo *stock* em loja ou em trânsito. Este é um sistema de gestão de lojas que normalmente se encontra integrado com o ORMS.

O protótipo a desenvolver irá permitir essa integração através de um método diferente, recorrendo a processos BPEL (uma tecnologia "Fusion Middleware"), utilizando como suporte a plataforma de integração ORIB v13 e as Oracle AQ como *bus* de integração.

De forma a obter-se um protótipo generalista, foi escolhida a transacção de mensagens de criação ou modificação de transferências (do tipo `TsfDesc`), que devem ser mapeadas para outro formato (do tipo `SODesc`), de forma a puderem ser subscritas pelo SIM, como se pode observar na Figura 4.1. Foi escolhido este tipo de transacção para que o protótipo englobe também o caso em que exista necessidade de conversão entre formatos, que é um cenário bastante comum.

4.3 Arquitectura do Protótipo

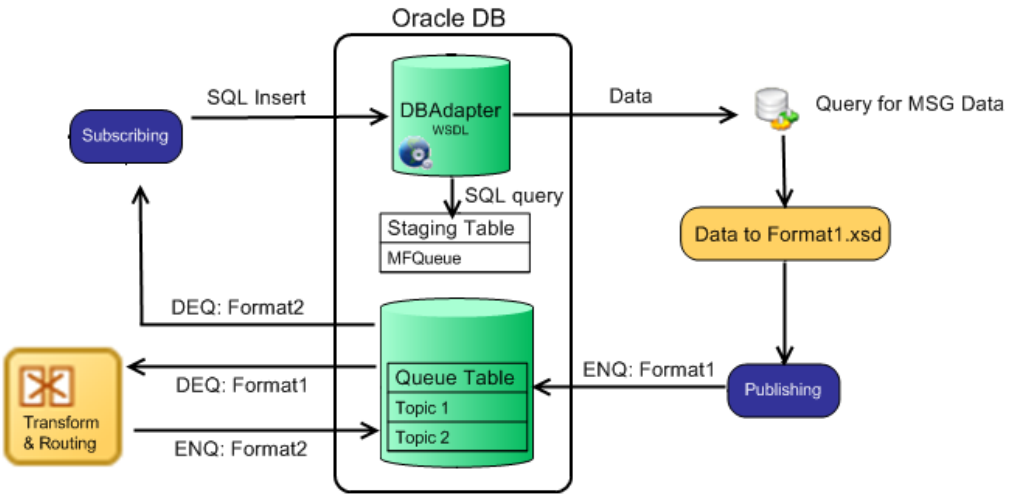

Figura 4.2: Arquitectura do Protótipo

Tabela 4.1: Parâmetros do procedimento `GETNXT()`

Parâmetros	IN/OUT	Tipo
O_status_code	OUT	VARCHAR2,
O_error_msg	OUT	VARCHAR2,
O_message_type	OUT	VARCHAR2,
O_message	OUT	RIB_OBJECT,
O_bus_obj_id	OUT	RIB_BUSOBJID_TBL,
O_routing_info	OUT	RIB_ROUTINGINFO_TBL,
I_num_threads	IN	NUMBER DEFAULT 1,
I_thread_val	IN	NUMBER DEFAULT 1

O ciclo de vida das mensagens de integração pode ser especificado de forma simples, como se observa na Figura 4.2. Os dados necessários à realização da operação são retirados das tabelas da base de dados, de acordo com os dados indicados na tabela temporária (*staging table*) da aplicação publicadora. Possuindo os dados necessários à criação da mensagem, esses dados devem ser mapeados para um formato definido pelo *schema* apropriado, consoante a família da mensagem e a operação a efectuar. Quando os dados se encontram no formato especificado, são inseridos no campo `O_MESSAGE` da estrutura a publicar na base de dados (BD). Mas `O_MESSAGE` representa apenas uma parte da mensagem a ser publicada. Para além dos dados da operação a realizar, devem ser indicados o tipo de mensagem (`message_type`), informações de redireccionamento (`O_routing_info`), entre outros dados necessário, tal como acontecia na plataforma ORIB 12.

A mensagem da tabela temporária da aplicação publicadora é, na plataforma de integração anterior (versão 12), retirada com o recurso ao procedimento armazenado (*stored procedure*) `GETNXT()`, que possui os parâmetros presentes na Tabela 4.1.

Quando uma mensagem se encontra pronta para publicação, o procedimento `GETNXT()` do gestor da família da mensagem examina a *staging table* e, através dos dados contidos nessa tabela, cria os objectos ORIB para publicação. Esses objectos possuem os dados relativos à mensagem a publicar e ainda outras informações como o tipo de mensagem, o seu estado, informação de redireccionamento, entre outras, representadas pelos parâmetros mencionados do procedimento `GETNXT()`. Para o método de integração apresentado neste projecto, serão necessários os dados indicados pelos parâmetros do procedimento `GETNXT()`, mas estes devem ser obtidos manualmente, questionando a base de dados (através de uma *query* SQL), uma vez que não se irá recorrer ao procedimento `GETNXT()`.

Seguidamente, este conjunto de dados é colocado no tópico da Oracle AQ. Caso a aplicação subscritora necessite de transformação para outro formato, a mensagem é retirada da AQ (Advanced Queue). Os dados do campo `O_message` são transformados para o novo formato e a mensagem é colocada novamente na AQ, numa nova fila de espera para

publicação. A aplicação subscritora, após esse processo, retira da AQ a mensagem publicada e cria os registos necessários nas tabelas apropriadas da base de dados da aplicação subscritora. Este processo conclui a integração das duas aplicações.

4.4 Especificação do Protótipo

Após a revisão anterior da arquitectura necessária para uma implementação de integração baseada em processos BPEL, é seguidamente especificado o protótipo implementado. A especificação apresentada serve tanto para implementação deste protótipo, como para outros casos de integração de aplicações "Oracle Retail" com recurso a processos BPEL. Para cada processo, representativo de cada passo necessário à integração, é descrita a especificação geral, fruto do estudo deste tipo de integração e de seguida são descritos os pormenores de especificação para o protótipo em causa.

4.4.1 Recuperação de Informação de Publicação

Segundo o curso normal, poderia usar-se um adaptador que fizesse uma chamada ao procedimento `GETNXT()`, que retornaria o objecto ORIB ao seu adaptador, que por sua vez converteria essa informação numa *string* XML. Esta *string* XML seria então encapsulada num envelope de mensagens ORIB.

No entanto, para este caso de implementação com recurso a AQ, não é possível determinar uma solução baseada no procedimento `GETNXT()`. Uma vez que o parâmetro de saída `O_message`, de tipo interno `RIB_OBJECT`, ao ser mapeado em XML, resulta numa estrutura Oracle que depende dos dados da mensagem de publicação, não pode seguir um XML *schema* definido.

Sendo assim, deve-se recorrer à solução mais trabalhosa, mas mais fiável. A criação de uma *query* SQL que retire da base de dados a informação necessária, campo a campo, das várias tabelas onde a informação se encontra.

Para este protótipo, será necessário obter os dados relativos à criação da transferência. Esses dados são indicado no "ORIB v13 Integration Guide" [Cor08i], como se pode observar na Figura 4.3.

No entanto, é necessário possuir os dados de referência para retirar das tabelas a informação relativa à transferência correcta. Esses dados encontram-se na tabela temporária (*staging table*) `tsf_mfqueue`, como se observa na Figura 4.4 na página 37. Os campos desta tabela fornecem a informação necessária de referência (`tsf_no`), de forma a ser possível retirar das tabelas indicadas na Figura 4.3 a informação necessária à criação da mensagem para publicação.

ORACLE Retail Integration Bus Mappings Report

Application Name: Retail Management System — **TsfDesc.xsd**

Tag Name	Table Name	Column Name	API Req
tsf_no	tsfhead	tsf_no	Yes
doc_type	*	*	Yes
physical_from_loc	store, wh	store, physical_wh	Yes
from_loc_type	item_loc	loc_type	Yes
from_loc	tsfhead	from_loc	Yes
physical_to_loc	store, wh	store, physical_wh	Yes
to_loc_type	item_loc	loc_type	Yes
to_loc	tsfhead	to_loc	Yes
tsf_type	tsfhead	tsf_type	Yes
pick_not_before_date	tsfhead	approval_date	Yes
pick_not_after_date	*	*	Yes
default_order_type	system_options	default_order_type	Yes
priority	*	*	No
break_by_distro	*	*	Yes

Application Name: Retail Management System — **TsfDtl**

Tag Name	Table Name	Column Name	API Req
item	tsfdetail	item	Yes
tsf_qty	tsfdetail	tsf_qty	Yes
price	*	*	No
selling_uom	*	*	No
priority	*	*	No
expedite_flag	*	*	No
store_ord_mult	item_loc	store_ord_mult	No
tsf_po_link_no	tsfdetail	tsf_po_link_no	No
ticket_type_id	item_ticket	ticket_type_id	No
TsfDtlTckt	*	*	*
inv_status	tsfdetail	inv_status	No

Figura 4.3: Tabelas com Informação de Criação de Transferência [Cor08i]

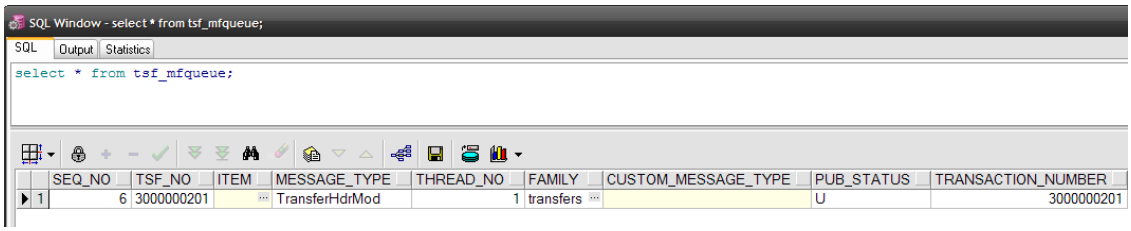

Figura 4.4: Detalhe da Tabela `tsf_mfqueue`

Após terem sido obtidos os dados relativos à transferência criada, é necessário preparar uma mensagem para publicação na AQ. Esse processo encontra-se detalhado de seguida.

4.4.2 Criação da Mensagem de Publicação

A chamada do adaptador de BD que executa a *query* SQL cria automaticamente um documento XML com os dados retirados da base de dados. Esses dados devem ser mapeados para um formato específico de acordo com o tipo de mensagem a publicar.

O mapeamento da informação para o formato XML específico da família da mensagem é efectuado com recurso à ferramenta de mapeamento e transformação do "Oracle JDeveloper". Esta ferramenta permite um mapeamento visual dos vários campos XML de cada um dos documentos. O mapeamento é efectuado ligando, graficamente, o campo de um dos documentos XML com o campo do outro documento XML que este deve assumir. É criado um ficheiro XSL que define o mapeamento entre os documentos XML, sem ser necessário qualquer código por parte do programador. Esta ferramenta permite um mapeamento rápido e fiável graças à utilização de tecnologias e procedimentos baseadas em standards (XML e XSL).

Neste protótipo, a informação retirada da BD, relativa à criação de uma transferência, deve ser mapeada para o *schema* `TsfDesc.xsd` (ver Anexo B). Foram apenas retirados da BD os dados essenciais para demonstrar este processo. O *schema* `TsfDesc.xsd` não ficará completo, mas conterá os dados essenciais, requeridos pela API, como indicado na Figura 4.3.

Numa implementação completa devem ser ainda preenchidos os campos extra de redireccionamento, estado e tipo de mensagem, entre outros, indicados na Tabela 4.1. Neste caso, apenas será preenchido o campo `O_message`, com os dados no formato `TsfDesc.xsd`.

De seguida, deve ser inserida a mensagem correctamente formatada na Oracle AQ. A fila de espera de transferências deve já encontrar-se criada e inicializada para aceitar a mensagem de publicação.

Tabela 4.2: Mapeamento TsfDesc para SODesc

Campos TsfDesc	Campos SODesc
tsf_no	distro_nbr
doc_type	document_type
physical_from_loc	dc_dest_id
pick_not_before_date	pick_not_before_date
pick_not_after_date	pick_not_after_date
not_after_date	not_after_date
default_order_type	order_type
priority	priority

4.4.3 Transformação entre Formatos

Caso seja necessária uma transformação do formato para outro tipo, a mensagem é retirada da AQ e é feito o mapeamento entre os formatos, de forma a recolocar na AQ a mensagem com o novo formato, pronta para ser retirada e consumida pela aplicação subscritora. No caso específico deste protótipo, a criação de uma transferência pertence ao tipo `TsfDesc`, que deve ser transformada no tipo `SODesc` para integrar a aplicação SIM. Estas transformações entre formatos podem ser consultados no guia de integração do ORIB [Cor08i].

As mensagens de transferência são publicadas pelo ORMS de modo a transferir uma certa quantidade de stock entre lojas ou entrepostos. O SIM apenas aceita mensagens para este efeito do tipo `Stock Order` e não `Transfer` (transferência). É necessária a transformação entre estes dois formatos. A mensagens do tipo `Stock Order` são subscritas pela aplicação SIM. Para além da transformação é necessário indicar a loja específica para onde se quer enviar a mensagem.

Tal como já foi referido, existem XML *schemas* específicos para cada tipo de mensagens. Do lado do ORMS as transferências são definidas pelo *schema* `TsfDesc.xsd`, presente no Anexo B, enquanto do lado da aplicação SIM as *Stock Orders* são definidas pelo *schema* `SODesc.xsd` (ver Anexo B). A transformação entre os diferentes formatos deve ser efectuada através de mapeamento dos campos dos *schemas* do tipo de mensagem aceite por cada aplicação. Na Tabela 4.2 podemos ver o mapeamento entre os campos de cada *schema* XML. Estes são os campos de mapeamento obrigatório. Existem outros elementos correspondentes a outros campos de diferentes tabelas que não exigem mapeamento. Nem todos os dados da transferência são mapeados numa *Stock Order*.

O elemento `TsfDesc` inclui ainda múltiplos elementos possíveis do tipo `TsfDtl`. Estes elementos correspondem aos artigos a serem transferidos. Na Tabela 4.3 pode-se verificar o mapeamento para o elemento `SODtl` do tipo `SODesc`.

O campo `dest_id` do elemento SODtl deve ser mapeado de `physical_to_loc` do elemento `TsfDesc`. Isto significa que todos os elementos SODtl terão o mesmo valor

Tabela 4.3: Mapeamento TsfDtl para SODtl

Campo TsfDtl	Campo SODtl
	dest_id
item	item_id
tsf_qty	requested_unit_qty
price	retail_price
selling_uom	selling_uom
priority	priority
ticket_type_id	ticket_type
store_ord_mult	store_ord_mult
expedite_flag	expedite_flag

para os campos `dest_id`.

Para o protótipo em questão não foram mapeados todos os campos, mas apenas os obrigatórios, necessários para servir o exemplo generalista.

Após ser feita a conversão entre os formatos, a mensagem deve ser novamente colocada numa fila de espera, desta vez diferente da anterior. A mensagem, com os dados do envelope que lhe correspondem, deve ser inserida numa fila de espera subscrita pela aplicação SIM. Em muitos casos, este não é um passo necessário. Caso a mensagem que se encontra na fila de espera esteja no formato aceite tanto pela aplicação publicadora como pela aplicação subscritora, esta transformação é desnecessária e o subscritor pode imediatamente retirar da AQ a mensagem já publicada.

4.4.4 Subscrição e Inserção em BD

Depois de efectuada a transformação entre os formatos, de forma a que a aplicação SIM possa subscrever a mensagem no formato que aceita, pode ser feito o *dequeue* da mensagem. A aplicação subscritora retira da Oracle AQ a mensagem publicada e cria os registos necessários nas tabelas apropriadas da sua base de dados.

A criação dos registos nas tabelas de BD da aplicação SIM encontram-se também descritas no "RIB 13 Integration Guides" [Cor08i]. Com os dados presentes na mensagem, que segue o formato do *schema* `SODesc.xsd`, devem ser criados registos nas tabelas da base de dados da aplicação SIM, de acordo com a relação indicada nas Tabelas 4.4 e 4.5.

Tabela 4.4: Tabelas de BD dos Dados de SODesc

Elemento de SODesc	Tabela BD	Coluna da Tabela
distro_nbr	RK_ALLOCATIONS	ALLOCATION_ID
dc_dest_id	RK_ALLOCATIONS	WAREHOUSE_ID
pick_not_before_date	DO_CTL_INV	NOT_AFTER_DATE
pick_not_after_date	RK_ALLOCATIONS	DELIVERY_DATE

A inserção dos dados nas tabelas apropriadas, conclui o processo de integração.

Tabela 4.5: Tabelas de BD dos Dados de SODtl

Elemento de SODtl	Tabela	Coluna
dest_id	RK_ALLOCATIONS	STORE_ID
item_id	RK_ALLOCATIONS	ITEM_ID
requested_unit_qty	RK_ALLOCATIONS	QUANTITY

4.4.5 Variáveis e Pormenores de Especificação

Nos processos BPEL, deverão estar presentes diversas variáveis, que serão descritas seguidamente.

TsfBDOutput Esta variável está directamente associada ao WS que acede à base de dados e executa a *query* SQL para retirar a informação relacionada com a transferência. O formato depende da própria *query* a ser executada e neste protótipo assume o seguinte formato:

```
TSF_MSG_TYPE
<complexType name="TransfersDBOutput">
  <sequence>
    <element name="TSF_NO" type="decimal" nillable="true"/>
    <element name="MESSAGE_TYPE" type="string" nillable="true"/>
    <element name="PUB_STATUS" type="string" nillable="true"/>
    <element name="ITEM" type="string" nillable="true"/>
    <element name="TSF_QTY" type="decimal" nillable="true"/>
    <element name="FROM_LOC" type="decimal" nillable="true"/>
    <element name="FROM_LOC_TYPE" type="string" nillable="true"/>
    <element name="TO_LOC" type="decimal" nillable="true"/>
    <element name="TO_LOC_TYPE" type="string" nillable="true"/>
    <element name="TSF_TYPE" type="string" nillable="true"/>
    <element name="APPROVAL_DATE" type="dateTime" nillable="true"/>
    <element name="DELIVERY_DATE" type="dateTime" nillable="true"/>
    <element name="DEFAULT_ORDER_TYPE" type="string" nillable="true"/>
  </sequence>
</complexType>
```

TsfDesc/SODesc As variáveis mais comuns presentes nos processos de publicação e transformação serão `TsfDesc/SODesc`, especificadas pelos XML *schemas* descritos nos Apêndices B.1 e B.2, respectivamente.

TsfDescS/SODescS Esta variável deverá ser uma conversão de `TsfDesc/SODesc` para o formato *String*. Esta variável é necessária uma vez que a inserção da mensagem na AQ apenas é possível se a mensagem estiver no formato `String`.

Para a configuração dos adaptadores, deve ser tido em conta o *schema* de base de dados a ser acedido (ORMS/SIM), consoante a aplicação Retail a ser acedida. O utilizador deve corresponder à aplicação e deve ser um utilizador normal com acesso suficiente ao *schema*

da aplicação, de forma a manipular os dados das tabelas necessárias à integração. Para os adaptadores de AQ, é usado o mesmo utilizador para a publicação e subscrição. Este não é um procedimento habitual, mas é necessário devido à pouca experiência com este tipo de *Queuing*. Num trabalho futuro, devem ser criados utilizadores independentes para cada uma das aplicações a integrar. É necessário, no entanto, que estes utilizadores possuam privilégios de manipulação da AQ.

Para finalizar, a transformação entre os formatos é efectuada de forma simplista, utilizando os dados estritamente necessários para garantir a integração das duas aplicações.

4.5 Resumo

Neste capítulo foi apresentada a arquitectura de integração com recurso a processos BPEL e a sua especificação de forma a integrar aplicações "Oracle Retail". Em cada processo, foi efectuada ainda a especificação para o protótipo específico a desenvolver. A implementação encontra-se descrita no capítulo seguinte.

Capítulo 5

implementação do Protótipo

Neste capítulo encontra-se descrita a implementação do protótipo especificado no capítulo anterior. A implementação encontra-se dividida nas várias fases de implementação. É feito um resumo da preparação da máquina virtual com a instalação das plataformas e ambientes necessários à execução dos processos BPEL. Seguidamente, é descrita a configuração das Oracle AQ e finalmente é descrita a implementação dos processos BPEL implementados para o protótipo desenvolvido.

5.1 Introdução

Para implementar este protótipo foi necessário criar um ambiente de desenvolvimento e produção próprio, tendo como base uma máquina virtual com SO baseado em Unix, de forma a instalar os componentes necessários à execução dos processos BPEL. Esta máquina virtual exigiu ainda a instalação de uma base de dados e de servidores de aplicações "Oracle Retail". A configuração das Oracle AQ encontra-se também detalhada neste capítulo. Esta configuração foi necessária uma vez que a estrutura Oracle AQ representa o *bus* de integração das aplicações. Finalmente, é descrita a implementação de cada processo BPEL, desde o fluxo de publicação à transformação da mensagem colocada na AQ. A implementação do processo final não foi implementada, pelo que apenas existe a sua especificação no capítulo anterior.

5.2 Configuração de VM

Foi tumultuosa a criação do ambiente propício à implementação do protótipo, uma vez que exigiu a instalação e configuração de diversas aplicações e *middleware* da Oracle. Foram encontrados diversos obstáculos durante esse procedimento, mas que foram

ultrapassados sem grande prejuízo, excepto no que se refere ao tempo disponível para o projecto em questão.

Para a implementação do protótipo de utilização da plataforma de integração Oracle, foi necessário criar um ambiente de desenvolvimento e implementação apropriado a esse propósito. Para tal, foi necessário proceder à instalação e configuração de diversas aplicações Oracle de base de dados e *middleware*, específicas para o efeito.

Preparação do SO para instalação de software Oracle

Para instalar a plataforma de base de dados Oracle DB 10.1.2.2 foi necessário, previamente, configurar o SO.

Inicialmente, procedeu-se à configuração dos ficheiros "hosts", tanto na VM como na máquina física, de forma a permitir a comunicação entre a máquina física e a máquina virtual. Essa configuração consiste simplesmente em adicionar o IP da VM seguido do nome da máquina virtual (*hostname*).

Seguidamente, foi necessário alterar o ficheiro de configuração do "kernel" denominado /etc/sysctl.conf, com os parâmetros necessários à instalação da base de dados Oracle e do "Application Server". Foi ainda necessário alterar os parâmetros de segurança no ficheiro /etc/security/limits.conf.

Para além destas configurações iniciais, foi ainda necessário instalar os pacotes de software necessários à instalação das aplicações Oracle. Estas dependências podem ser consultadas nos manuais de instalação dos produtos Oracle.

Após estar devidamente configurado o Sistema Operativo, procedeu-se à criação de um grupo de instalação, do qual fazem parte três utilizadores (um para cada tipo de instalação).

```
> groupadd oinstall
```

Para instalação da "Oracle RDBMS 10g Release 2 Enterprise Edition 10.2.0.2" (actualização 10.2.0.3):

```
> useradd -g oinstall -G dba oracle
> passwd oracle
( ORACLE password = oracle )
```

Para instalação do "Oracle Application Server Oracle Application Server Forms and Reports 10g version 10.1.2" (actualização 10.1.2.2):

```
> useradd -g oinstall -G dba oracleapp
> passwd oracleapp
( ORACLE password = oracleapp )
```

Para instalação do "Oracle Application Server 10g version 10.1.3.1" (actualização 10.1.3.3):

```
> useradd -g oinstall -G dba oracleapp2
> passwd oracleapp2
( ORACLE password = oracleapp2 )
```

Instalação da plataforma de gestão de base de dados "Oracle RDBMS 10g Release 2 Enterprise Edition 10.2.0.2" (actualização 10.2.0.3)

A instalação da base de dados foi realizada com o utilizador "oracle". Procedeu-se, inicialmente, à edição do ficheiro .bash_profile (/home/oracle) com as variáveis de ambiente necessárias, presentes no manual de instalação.

A instalação da base de dados correu sem problemas, através da interface "Oracle Universal Installer". Após a instalação foi necessário actualizar o ficheiro /etc/oratab com a linha: RMS12:/oracle/db:Y

A actualização para a versão 10.2.0.3 foi efectuada de seguida, sem obstáculos importantes.

Concluída a instalação, foi ainda necessário actualizar o software com as *patches* 5397953, 5648872 e 5721821 com recurso à ferramenta OPatch.

Instalação de "Oracle Application Server Oracle Application Server Forms and Reports 10g version 10.1.2" (actualização 10.1.2.2)

Todas as dependências deste software foram anteriormente cumpridas, à excepção de algumas alterações nos parâmetros de "kernel".

A instalação é também baseada no "Oracle Universal Installer", assim como a actualização para a versão 10.1.2.2.

Foram ainda necessárias as instalações de actualizações, devido a dependências do ORMS v13, utilizando novamente a ferramenta OPatch para instalação dos "patches" 5861907 e 5632264.

Instalação de "Oracle Retail Merchandising System" versão 13

Os pacotes de instalação do ORMS v13 encontram-se divididos entre ficheiros de esquema de BD, ficheiros "batch" e ficheiros do servidor de aplicação.

Para iniciar a instalação do ORMS v13 foi necessário criar um directório e descompactar os ficheiros incluídos (rms13dbschema.zip, rms13batch.zip, rms13appserver.zip). Ao descompactar estes três ficheiros, foi criada a seguintes estrutura:

rms/
- dbschema
- batch
- application

Seguidamente, foi necessário executar os seguintes passos, tendo em consideração que $ORACLE_HOME se refere à localização do OAS e ORACLE_SID ao nome atribuído à base de dados:

- Copiar `rms/dbschema/create_db/init.ora` para $ORACLE_HOME/pfile e renomear para `init$ORACLE_SID.ora`;
- Modificar os parâmetros desse ficheiro de acordo com as instruções nele incluídas na forma de comentários;
- Criar ligação simbólica de `$ORACLE_HOME/pfile/init$ORACLE_SID.ora` para `$ORACLE_HOME/dbs/init$ORACLE_SID.ora`;
- Modificar o ficheiro `rms/dbschema/create_db/crdb1.sql` de acordo com as instruções nele incluídas na forma de comentários e executar no "SQL*Plus";
- Modificar o ficheiro `rms/dbschema/create_db/crdb2.sql` de acordo com as instruções nele incluídas na forma de comentários e executar no "SQL*Plus";
- Modificar o ficheiro `rms/dbschema/create_db/crdb3.sql` de acordo com as instruções nele incluídas na forma de comentários e executar no "SQL*Plus";
- Configurar os ficheiros `listener.ora` e `tnsnames.ora` correctamente.

De seguida, foi necessário executar o ficheiro `rms/dbschema/install.sh`, o qual procedeu à instalação do esquema de base de dados do ORMS v13. A instalação dos ficheiros "batch" foi efectuada a partir do ficheiro `rms/batch/install.sh`.

Após a instalação dos ficheiros "batch", foi necessário alterar o ficheiro `$ORACLE_HOME/guicommon/tk/admin/T2kMotif.rgb`, de forma a possuir a seguinte linha:

```
Tk2Motif*fontMapCs:  iso8859-2=UTF8
```

Essa alteração permite executar o instalador da aplicação (`install.sh`), que conclui a instalação da aplicação ORMS v13.

Instalação de "Oracle Application Server 10g version 10.1.3.1" (actualização 10.1.3.3)

Para instalar este software, bastou executar o ficheiro de instalação e aplicar a actualização 10.1.3.3.

Para além dessas instalações, foi ainda necessário aplicar os "patches" internos 5398506 e 5632264, com recurso à ferramenta apropriada, OPatch.

Instalação de "Oracle Retail Integration Bus" versão 13

Antes da instalação do ORIB v13 foi necessário criar as instâncias oc4j (Oracle Containers for Java) necessárias ao *deployment* da aplicação ORIB v13 e das aplicações "Oracle Retail".

A criação dessas instâncias é feita com recurso aos seguintes comandos, a partir da directoria $ORACLE_HOME/bin/:

```
createinstance -instanceName rib-rms-oc4j-instance
createinstance -instanceName rib-tafr-oc4j-instance
createinstance -instanceName rib-func-artifact-oc4j-instance
```

Seguidamente, é necessário editar o ficheiro `opmn.xml` e adicionar às instâncias definidas, as propriedades:

```
-Xms500M -Xmx900M
-Doc4j.jmx.security.proxy.off=true
<data id=''oc4j-options'' value=''-userThreads''/>
```

Assim, cada instância deverá possuir o seguinte aspecto:

```
...
<process-type id="rib-rpm-oc4j-instance" module-id="OC4J"
status="enabled">
<module-data>
<category id="start-parameters">
 <data id="java-options" value="-server
  -Doc4j.jmx.security.proxy.off=true -Xms500M -Xmx900M
  -Djava.security.policy=
    \$ORACLE_HOME/j2ee/rib-rpm-oc4j-instance/config/java2.policy
    -Djava.awt.headless=true
  -Dhttp.webdir.enable=false"/>
 <data id="oc4j-options" value="-userThreads"/>
</category>
...
```

De forma a serem atribuídas as novas propriedades às instâncias oc4j, é necessário reiniciar o componente OPMN, através do comando `opmnctl reload`. Assim, as instâncias OC4J adquirem as propriedades indicadas e podem ser inicializadas a partir do comando `opmnctl startall`.

Para ser possível implementar e utilizar as Oracle AQ, é necessário criar o utilizador de gestão das mesmas, com o acesso e permissões apropriadas aos pacotes Oracle AQ. Devem ser atribuídas, pelo menos, as seguintes permissões:

```
CREATE USER RIBAQ IDENTIFIED BY RIBAQ
DEFAULT TABLESPACE AQJMS
TEMPORARY TABLESPACE TEMP;
GRANT CONNECT TO RIBAQ;
GRANT RESOURCE TO RIBAQ;
GRANT CREATE SESSION TO RIBAQ;
```

```
GRANT EXECUTE ON SYS.DBMS_AQ TO RIBAQ;
GRANT EXECUTE ON SYS.DBMS_AQADM TO RIBAQ;
GRANT EXECUTE ON SYS.DBMS_AQIN TO RIBAQ;
GRANT EXECUTE ON SYS.DBMS_AQJMS TO RIBAQ;
GRANT AQ_ADMINISTRATOR_ROLE TO RIBAQ;
```

Após esta tarefa, é necessário expandir a distribuição do "ORIB Kernel". Para tal, é necessário criar um directório que será o directório raiz da aplicação ORIB v13, denominado `rib-home`. Nesse directório, deve ser colocado o ficheiro incluido na distribuição ORIB v13, denominado `RibKernel13.0.0ForAll13.x.xApps_eng_ga.tar` e extraído o seu conteúdo.

Seguidamente, também incluído com na distribuição ORIB v13, deve ser copiado `RibFuncArtifact13.0.0ForAll13.0.0Apps_eng_ga.tar`, sem ser descompactado, para o directório `rib-home/download-home/rib-func-artifacts`. Para o ORMS e SIM, assim como para as aplicações a utilizar com o ORIB v13 para as quais foram criadas instâncias oc4j, devem ser obtidos os ficheiros ".tar". Estes ficheiros devem ser colocados no directório `rib-home/download-home/all-rib-apps`, sem serem descompactados.

Finalmente, para instalar a aplicação ORIB v13, deve ser executado o ficheiro de instalação `rib-installer.sh`, situado no directório `rib-home`, com a certeza de que as instâncias `opmn` anteriormente criadas se encontram activas.

Seguindo os passos indicados pelo instalador e escolhendo as opções apropriadas à situação em causa, é instalado o ORIB v13.

Instalação de "Oracle BPEL Process Manager 10g Release 3" (10.1.3.1.0)

Após ser descompactado o ficheiro `.cpio`, obtêm-se uma estrutura de directórios que contêm os ficheiros de instalação BPEL. Para instalar o "Oracle BPEL Process Manager 10g" é necessário criar o *schema* de BD, a partir da execução do ficheiro `irca.sh`, situado no directório `<INSTALL_DIR>/install/soa_schemas/irca`. É pedida a criação de uma password de acesso ao *schema* BPEL.

Após ser criado o *schema* BPEL, deve ser executado o instalador BPEL denominado (`install.sh`), situado na raiz dos directórios de instalação.

Estes passos concluem a criação do ambiente necessário à execução dos processos BPEL de integração das aplicações "Oracle Retail".

5.3 Configuração de Oracle AQ

As filas de espera que irão alojar as mensagens de integração, são baseadas na tecnologias Oracle AQ, descritas nos capítulos anteriores. De forma a possuirmos uma estrutura

de filas de espera para as mensagens de integração, é necessário executar um conjunto de passos.

Inicialmente, devem ser criados os papeis de administrador (`my_aq_adm_role`) e utilizador (`my_aq_user_role`) da AQ. Isso é conseguido através do seguinte *script* SQL:

```
CREATE ROLE my_aq_adm_role;
GRANT CONNECT, RESOURCE, aq_administrator_role TO my_aq_adm_role;

CREATE ROLE my_aq_user_role;
GRANT CREATE SESSION, aq_user_role TO my_aq_user_role;
EXEC DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE(privilege => 'ENQUEUE_ANY',
grantee => 'my_aq_user_role', admin_option => FALSE);
EXEC DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE(privilege => 'DEQUEUE_ANY',
grantee => 'my_aq_user_role', admin_option => FALSE);
```

É necessário criar um administrador de AQ, com acesso e permissões privilegiadas a todas as filas de espera, de forma a manter o controlo da toda a estrutura das AQ, para criação, eliminação, modificação ou resolução de erros que possam ocorrer nas AQ. Para criar o administrador AQ, deve ser iniciada a sessão com o utilizador de sistema, com privilégios e acesso total à base de dados, no modo *System Database Analyser* (SYSDBA). Este utilizador pode então criar o administrador de AQ, com recurso ao seguinte comando SQL:

```
CREATE USER aqadm IDENTIFIED BY aqadm DEFAULT TABLESPACE system
TEMPORARY TABLESPACE temp;

GRANT my_aq_adm_role TO aqadm;

GRANT CONNECT, RESOURCE to aqadm;
GRANT EXECUTE ON SYS.DBMS_AQ to aqadm;
GRANT EXECUTE ON SYS.DBMS_AQADM to aqadm;
GRANT EXECUTE ON SYS.DBMS_AQIN to aqadm;
commit;
```

Para podermos colocar e retirar mensagens das filas de espera, é necessário criar um outro utilizador. Este utilizador servirá como publicador e subscritor de mensagens das aplicações a integrar. Foi criado um único utilizador para os dois tipos de acção (*enqueue/dequeue*), de forma a simplificar a implementação do protótipo. Numa integração real, deve ser criado um utilizador deste tipo para cada aplicação a integrar.

A criação deste tipo de utilizador deve ser realizada com recurso ao seguinte *script* SQL:

```
CREATE USER aquser IDENTIFIED BY aquser DEFAULT TABLESPACE system
TEMPORARY TABLESPACE temp;

GRANT my_aq_user_role TO aquser;
```

Antes de puderem ser criadas as filas de espera para as transferências e *Stock Orders*, deve ser criado o tipo da mensagem que as filas de espera vão suportar. Lembrando o envelope descrito na Tabela 4.1 do Capítulo 4, encontram-se os campos que deverão ser suportados na mensagem de cada fila de espera, uma vez que conterão a informação necessária, para além da mensagem em si, como referido na especificação do protótipo. Não são considerados os campos `I_num_threads` e `I_thread_val` na implementação deste protótipo.

Para criação do objecto que representará o tipo de mensagem das filas de espera, basta executar o seguinte *script* SQL:

```
CREATE TYPE tsf_msg_type AS OBJECT(
O_status_code VARCHAR2,
O_error_msg VARCHAR2,
O_message_type VARCHAR2,
O_MESSAGE CLOB,
O_BUS_OBJ_ID RIB_BUSOBJID_TBL,
O_ROUTING_INFO RIB_ROUTINGINFO_TBL
);
/
GRANT EXECUTE ON tsf_msg_type TO my_aq_user_role;
```

Após estar criado o tipo de mensagem utilizado, é necessário criar as tabelas de filas de espera. Estas tabelas possuem a especificação do formato da mensagem da fila de espera, assim como permissões de acesso às filas de espera, restrições, *triggers*, entre outros pormenores de especificação. A *Queue Table* pode ser criada através da execução do seguinte *script*:

```
EXEC DBMS_AQADM.CREATE_QUEUE_TABLE(
queue_table => 'tsfs_mcqueue_tbl',
queue_payload_type => 'aqadm.tsf_msg_type',
multiple_consumers => TRUE
);

EXEC DBMS_AQADM.CREATE_QUEUE_TABLE(
queue_table => 'so_mcqueue_tbl',
queue_payload_type => 'aqadm.tsf_msg_type',
multiple_consumers => TRUE
);
```

O tipo de mensagem da tabela de fila de espera é designado através do parâmetro `queue_payload_type`. Como se pode observar, existe um parâmetro denominado de `multiple_consumers` que designa se a tabela possui filas de espera com múltiplos consumidores. Isto significa que deverão ser especificados os subscritores (que pela própria definição se conclui que pode ser mais que um) e a mensagem apenas será retirada da fila de espera assim que todos os subscritores tiverem retirado uma cópia da mensagem.

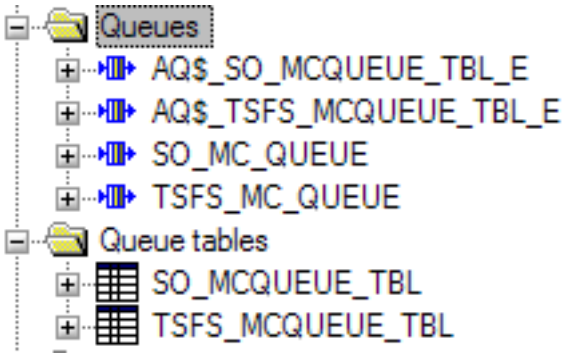

Figura 5.1: Filas de Espera Criadas e Tabelas Respectivas

Para este caso, foram criadas *Queue Tables* com o mesmo tipo de mensagem, uma vez que o envelope possui o mesmo formato para as aplicações a integrar. A discriminação do tipo de mensagem apenas é feita observando o conteúdo do campo `O_message`, que é transparente para a *Queue Table*, uma vez que possui o mesmo tipo (`CLOB`).

Seguidamente, são criadas as filas de espera de transferências e *Stock Orders* e inicializadas, indicando a tabela onde serão inseridas as filas de espera:

```
EXEC DBMS_AQADM.CREATE_QUEUE(
queue_name => 'so_mc_queue',
queue_table => 'so_mcqueue_tbl'
);
EXEC DBMS_AQADM.CREATE_QUEUE(
queue_name => 'tsfs_mc_queue',
queue_table => 'tsfs_mcqueue_tbl'
);
EXEC DBMS_AQADM.START_QUEUE(
queue_name => 'tsfs_mc_queue'
);
EXEC DBMS_AQADM.START_QUEUE(
queue_name => 'so_mc_queue'
);
```

Como se pode observar na Figura 5.1, as tabelas e respectivas filas de espera AQ ficam criadas e prontas a ser utilizadas. Observa-se ainda que são automaticamente criadas filas de espera de excepção para as tabelas de filas de espera criadas, denominadas `AQ$_SO_MCQUEUE_TBL_E` e `AQ$_TSFS_MCQUEUE_TBL_E`.

Para poder ser retirada a mensagem da fila de espera pela publicação subscritora, devem ser designados os subscritores das filas de espera. Isso é feito com recurso ao seguinte *script*:

```
DBMS_AQADM.ADD_SUBSCRIBER(
queue_name => 'tsfs_mc_queue',
subscriber => sys.aq$_agent('aquser', null, null));
```

```
DBMS_AQADM.ADD_SUBSCRIBER(
queue_name => 'so_mc_queue',
subscriber => sys.aq$_agent('aquser', null, null));
```

Desta forma, e como referido anteriormente, o utilizador `aquser` subscreve ambas as filas de espera e pode retirar mensagens das filas de espera.

A subscrição de AQ deve ser efectuada sempre no parâmetro `subscriber`, do procedimento `sys.aq$_agent(user, address, protocol))`.

A sintaxe desta função de designação de agentes de AQ, possui os seguintes parâmetros:

```
TYPE = aq$_agent IS OBJECT (
name            VARCHAR2(30),
address         VARCHAR2(1024),
protocol        NUMBER)
```

Todos os consumidores adicionados como subscritores a uma fila de espera de múltiplos consumidores devem possuir valores únicos para os parâmetros de AQ$_AGENT. Podem ser adicionados mais que um subscritor repetidamente utilizando o procedimento `DBMS_AQADM.ADD_SUBSCRIBER` até um máximo de 1024 subscritores para uma fila de espera de múltiplos consumidores.
Dois subscritores não podem possuir os mesmos valores para os atributos `NAME`, `ADDRESS` e `PROTOCOL` para o tipo AQ$_AGENT. Pelo menos um dos três atributos deve ser diferente entre pares de subscritores.

Após estarem devidamente criadas, configuradas e inicializadas as Oracle AQ, segue-se a implementação dos processos BPEL, que farão uso das mesmas. Esta implementação encontra-se descrita em pormenor nas secções seguintes.

5.4 Processos BPEL

Estes processos são apenas relativos à implementação do protótipo para o caso descrito no capítulo anterior. Estas implementações são baseadas na arquitectura descrita, para o caso específico de troca de mensagens de criação de transferências na aplicação ORMS e transformação para o formato *Stock Order* aceite pela aplicação SIM. Para tal, é necessário criar o processo de recuperação da informação da criação da transferência no ORMS a partir do *schema* respectivo da BD, conversão no formato Oracle `TsfDesc.xsd` e colocação na respectiva fila de espera da AQ. Outro processo necessário centra-se na transformação do formato da transferência, retirando da AQ, convertendo o formato e voltando a colocar na nova fila de espera.

O processo final não foi implementado devido à restrição de tempo e a dificuldades emergentes da instalação do ambiente, mas é descrita a especificação da sua implementação, que consiste em retirar a mensagem no novo formato da AQ e criar os registos no *schema* da DB da aplicação SIM.

5.4.1 Publicação

O processo BPEL de publicação da mensagem de criação da transferência, comporta diferentes componentes. A presença de um adaptador de cliente refere-se ao servidor BPEL, que necessita desta ligação para poder operar o processo. Existem neste processo dois adaptadores directamente relacionados com a implementação do processo de integração.

O adaptador de DB TransfersDB é responsável por retirar os dados relativos à transferência, como descrito na especificação do protótipo. Estes dados são retirados com recurso à seguinte *query* SQL:

```
select q.tsf_no,
     q.message_type,
     q.pub_status,
     qd.item,
     qd.tsf_qty,
     qh.from_loc,
     qh.from_loc_type,
     qh.to_loc,
     qh.to_loc_type,
     qh.tsf_type,
     qh.approval_date,
     qh.delivery_date,
     qsys.default_order_type

     from tsf_mfqueue q, tsfdetail qd, tsfhead qh,
          item_loc ql, system_options qsys
     where q.seq_no = (select min(q2.seq_no)
                         from tsf_mfqueue q2
                        where q2.pub_status = 'U')
           and q.tsf_no = qd.tsf_no
           and q.tsf_no = qh.tsf_no
           and (qh.from_loc_type = ql.loc_type
                and ql.item = qd.item
                and qh.from_loc = ql.loc)
```

Como se pode observar, os dados são provenientes de várias tabelas. Essas tabelas contêm a informação relativa à criação da transferência na aplicação ORMS, como especificado no capítulo anterior. A necessidade destes dados é observável na Figura 4.3. Esta Figura descreve os campos das tabelas onde se encontra a informação necessária.

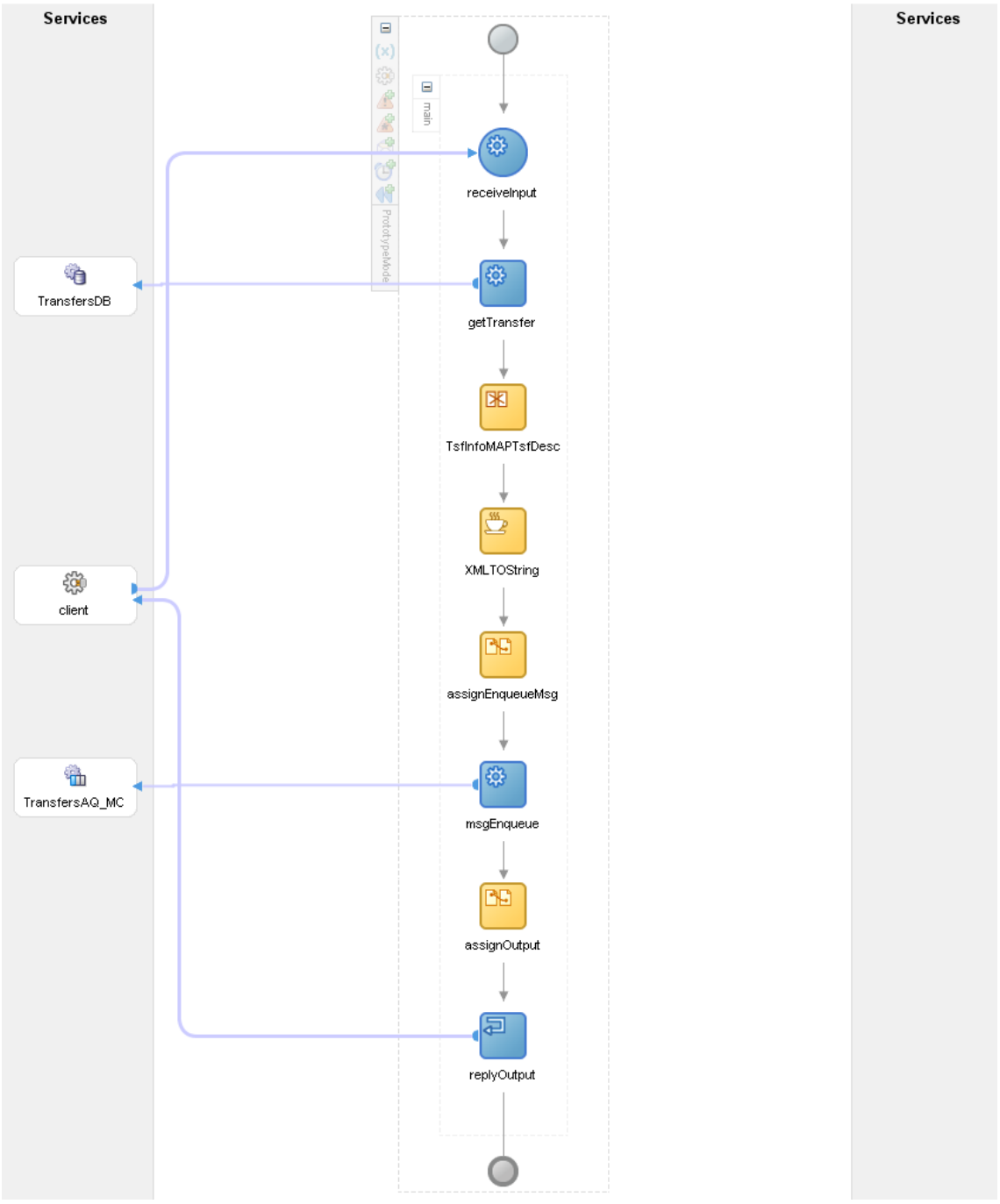

Figura 5.2: Processo BPEL de Publicação

Como referido, os dados de referência encontram-se no registo de `tsf_mfqueue` (ver Figura 4.4). Esta *query* SQL retira a primeira mensagem inserida na tabela `tsf_mfqueue` e compara o valor de referência da transferência (`tsf_no`) com o campo respectivo das outras tabelas, de forma a retirar a informação de criação dessa transferência em particular.

Depois de retirados os dados necessários relativos à transferência, é necessário colocar esses mesmos dados num formato *standard* Oracle. Esse formato encontra-se descrito no *schema* `TsfDesc.xsd`, presente no anexo B.1. Utilizando esse mesmo *schema*, é efectuado o mapeamento da informação da transferência para esse formato específico. A Figura 5.3 representa o mapeamento referido. Este mapeamento é facilmente realizado através da ferramenta disponibilizada pela aplicação "JDeveloper", como referido anteriormente.

O mapeamento segue o standard Oracle descrito na Figura 4.3 da página 36, presente nos relatórios de mapeamento do "ORIB 13 Integration Guides".

É necessário inserir os dados agora formatados (`TsfDesc`) no campo `O_message` do envelope da mensagem a ser colocada na fila de espera AQ. Para tal, uma vez que o campo `O_message` é do tipo `CLOB`, será necessário converter os dados em formato XML no tipo *string*.

Para a conversão de XML em *String*, foi criado um componente *Java Embedding*, que possui código Java executável no seu interior. Foi então necessário criar esse código de conversão, que consiste no seguinte:

```
String tsfDescString = new String();

try{

javax.xml.transform.Transformer transformer =
javax.xml.transform.TransformerFactory.newInstance().newTransformer();

java.io.StringWriter stringWriter = new java.io.StringWriter(2048);

transformer.transform(new javax.xml.transform.dom.DOMSource(
(org.w3c.dom.Node)getVariableData("TsfDesc", "TsfDesc","/TsfDesc")),
new javax.xml.transform.stream.StreamResult(stringWriter));

StringBuffer buffer =  stringWriter.getBuffer();

tsfDescString = buffer.toString();
}
catch(javax.xml.transform.TransformerException ex)
{
   ex.printStackTrace();
}
setVariableData("tsfDescS", tsfDescString);
```

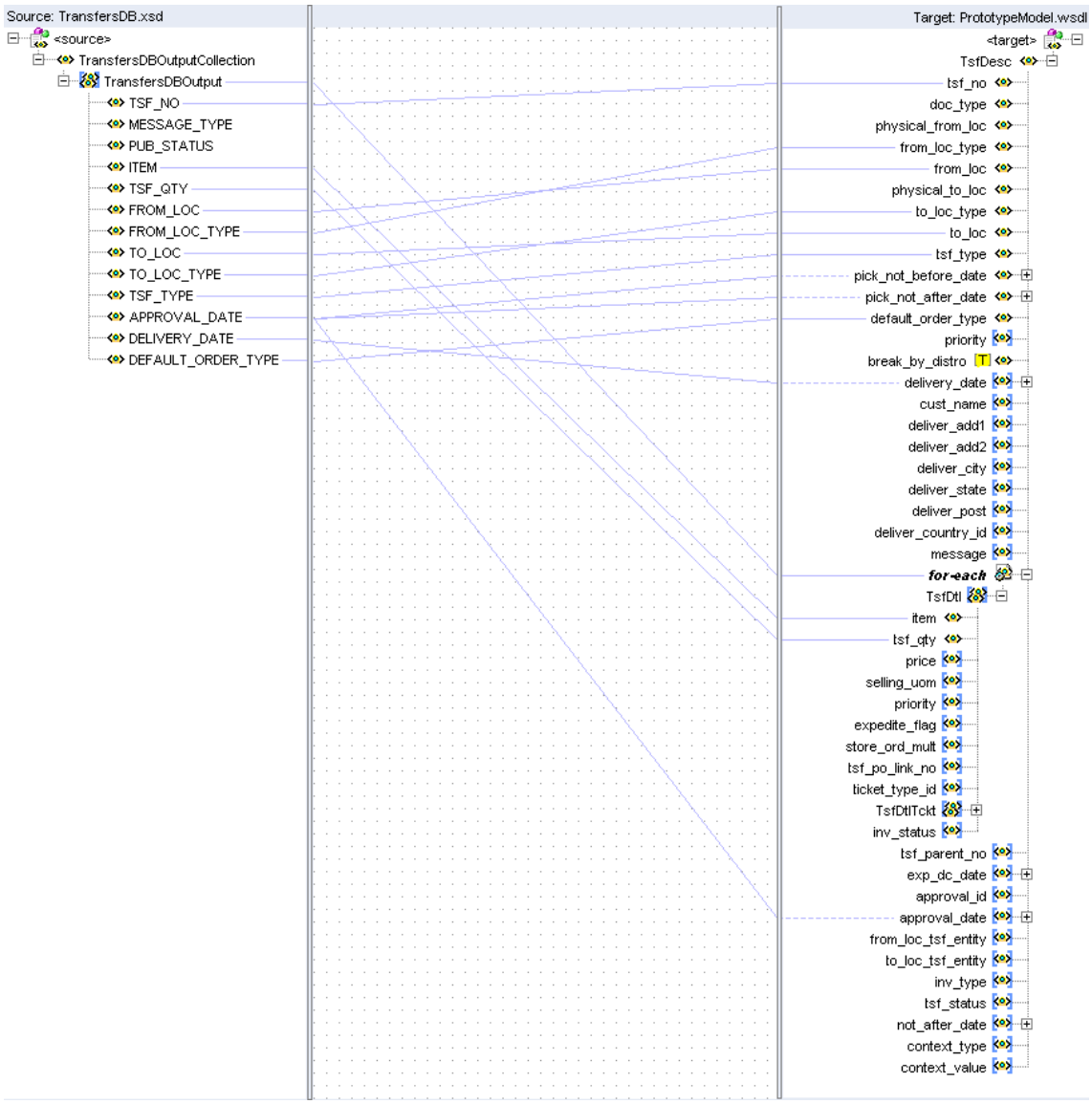

Figura 5.3: Mapeamento dos Dados da BD no Formato TsfDesc

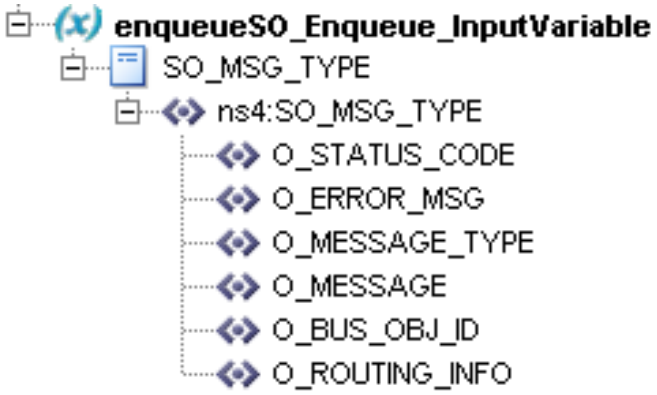

Figura 5.4: Variável de Inserção do Envelope na Fila de Espera de Transferências

Inicialmente, é criada uma variável do tipo *String*, denominada `tsfDescString`, que será o formato para o qual deve ser convertido o documento XML. Seguidamente, é criado um objecto do tipo `Transformer` [Mic03b] e outro tipo de objecto da classe `StringWriter` [Mic03a]. O objecto `Transformer` possui o método `transform()` que permite a conversão de uma fonte XML noutro formato. Neste caso, o segundo formato será uma *String*. Para tal, o primeiro argumento do método `transform()` é a variável que contém o ficheiro XML `TsfDesc`, acedido a partir do método interno BPEL `getVariableData()`. O segundo argumento representa o segundo formato da conversão, que irá ser representado pelo objecto `StringWriter`.

O documento XML é assim convertido e os dados do documento são colocados no objecto `StringWriter`. Esses dados são acedido através do método `getBuffer()` e são convertidos no formato *String* através do método `toString()`. O documento encontra-se agora convertido no formato *String*. Esta *String* representativa do documento XML original é então colocada na variável de destino `tsfDescS`, através do método interno da API BPEL `setVariableData()`, de forma a poder ser utilizada no processo BPEL.

O componente assignEnqueueMsg apenas atribui o valor dessa variável (que contém os dados da mensagem a publicar) ao campo `O_message` do envelope a colocar na AQ. O componente `msgEnqueue` é responsável por invocar o adaptador da AQ. A variável que contêm o envelope da mensagem possui o formato especificado na fila de espera, como se pode observar na Figura 5.4.

Como se pode observar, ainda na Figura 5.4, existem outros campos para além de `O_message`. Esses campos, como referido na especificação (na Secção 4.4), não são preenchidos para a implementação do protótipo. O adaptador TransfersAQ_MC é responsável pela inserção da mensagem na fila de espera de transferências `TSFS_MC_QUEUE`.

Após a implementação deste processo, fica colocada na `TSFS_MC_QUEUE` a mensagem de criação de transferência com o formato específico.

5.4.2 Transformação de Formatos

Para este caso específico de implementação, o protótipo deve possuir um processo BPEL de transformação entre os formatos `TsfDesc` e `SODesc`, como referido na especificação do mesmo (ver a Secção 4.4).

O processo BPEL para transformação entre os formatos requer que a mensagem anteriormente inserida da AQ seja retirada, o formato da mesma convertido para `SODesc` e inserida a mensagem no novo formato numa nova AQ, como se observa na Figura 5.5, que representa o processo BPEL em questão.

O adaptador TsfAQ é responsável por retirar da fila de espera `TSFS_MC_QUEUE` a mensagem publicada. O conteúdo do campo `O_message`, da mensagem retirada da AQ (*dequeue*), é colocado na variável associada ao componente getTsfMsg. A variável contida no componente getTsfMsg é do tipo `TsfDesc`. Isto porque a mensagem do campo `O_message`, ao ser retirada com recurso ao adaptador de AQ, é definida como possuindo o formato especificado no *schema* `TsfDesc.xsd`, explicitado no Anexo B.1.

De seguida, o componente de transformação `TsfDescMAPSODesc` é responsável pela conversão entre os dois formatos (`TsfDesc/SODesc`). Esse mapeamento é efectuado de acordo com a especificação do guia de integração do ORIB v13 e como já foi descrito anteriormente na Tabela 4.2. O mapeamento pode ser observado na Figura 5.6.

A conversão do documento XML no formato `SODesc` para o tipo *String* é necessária, mais uma vez, para colocar o conteúdo da mensagem no campo `O_Message` do envelope da nova mensagem a inserir na AQ:

```
String SODescString = new String();

try{

javax.xml.transform.Transformer transformer =
javax.xml.transform.TransformerFactory.newInstance().newTransformer();

java.io.StringWriter stringWriter = new java.io.StringWriter(2048);

transformer.transform(new javax.xml.transform.dom.DOMSource(
(org.w3c.dom.Node)getVariableData("SODesc", "SODesc","/SODesc")),
new javax.xml.transform.stream.StreamResult(stringWriter));

StringBuffer buffer =  stringWriter.getBuffer();
SODescString = buffer.toString();
}
catch(javax.xml.transform.TransformerException ex)
{
   ex.printStackTrace();
}
setVariableData("SODescS", SODescString);
```

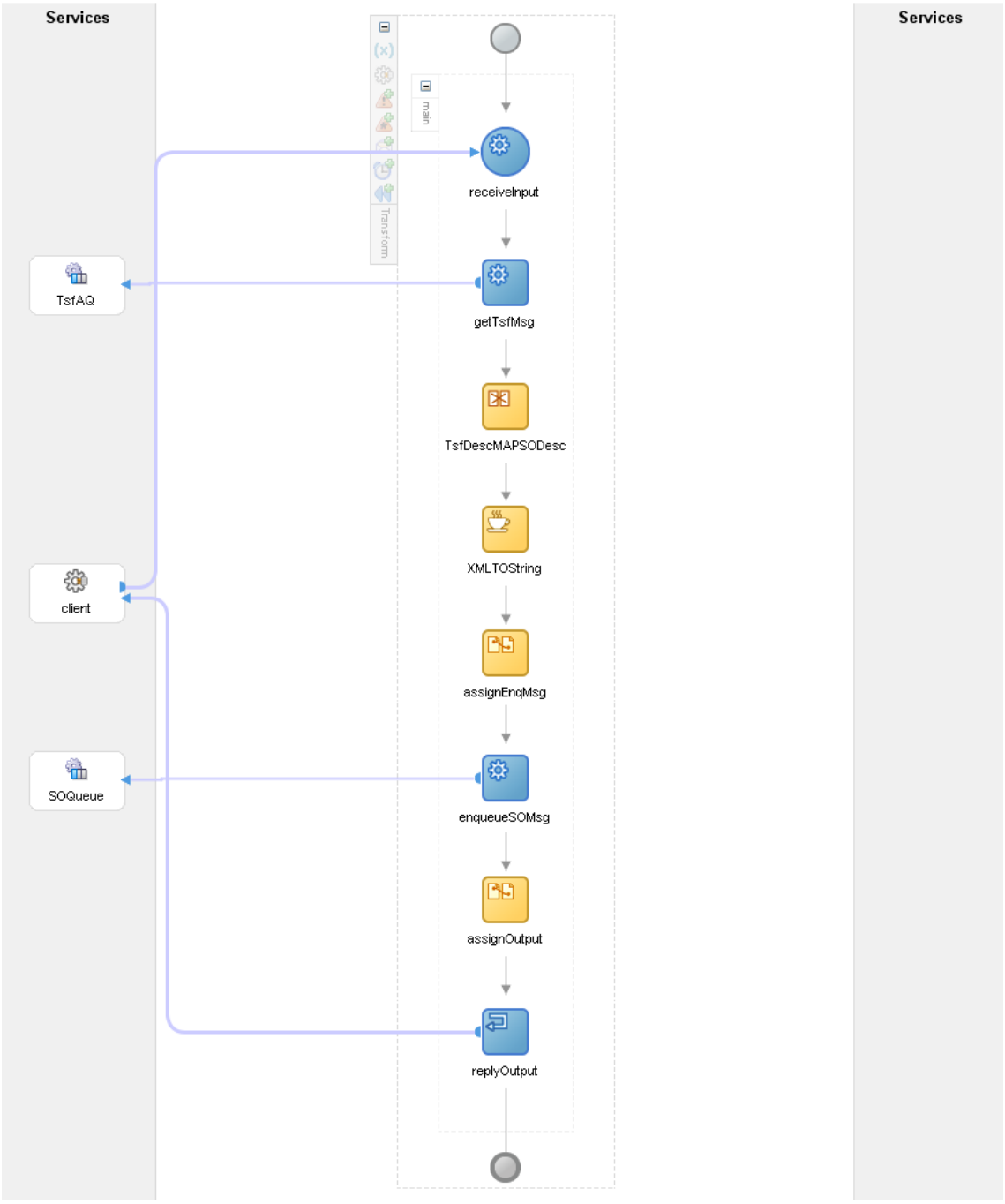

Figura 5.5: Processo BPEL de Transformação de Formatos

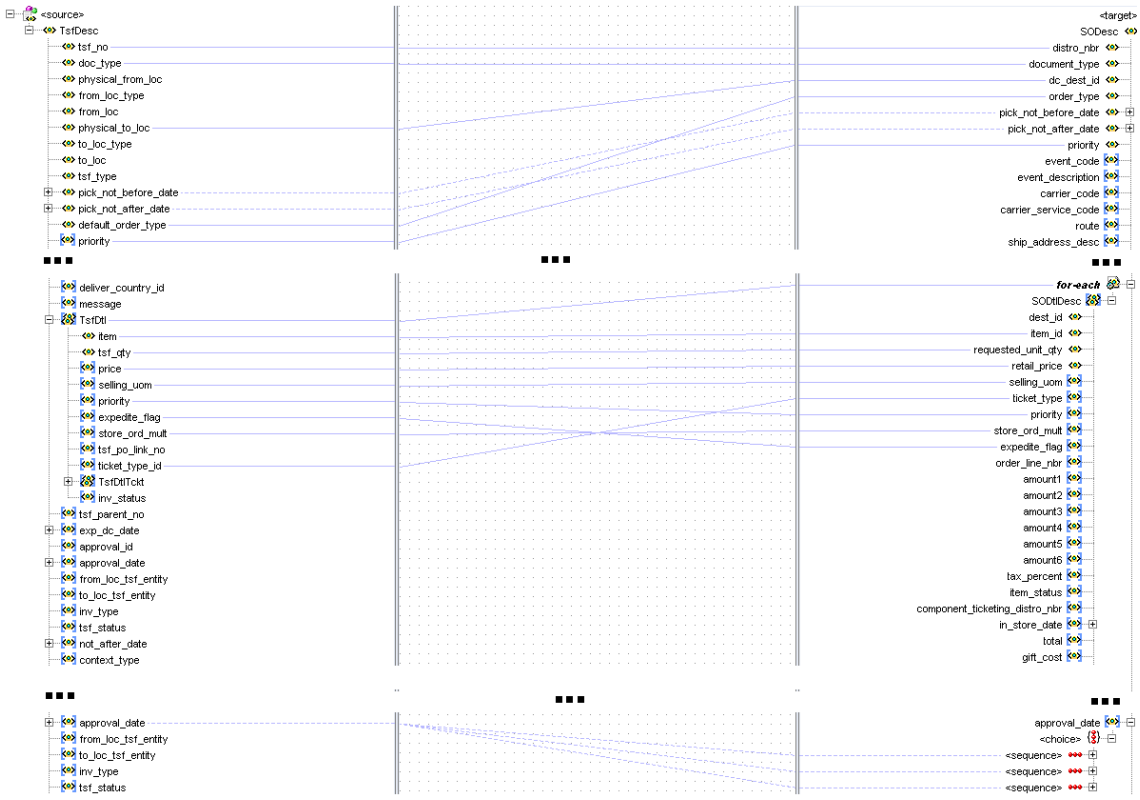

Figura 5.6: Mapeamento do Formato TsfDesc para o Formato SODesc

Este processo é idêntico ao anteriormente descrito, mas neste caso a mensagem encontra-se no formato `SODesc` e será colocada, já convertida no tipo *String*, na variável `SODescS`.

Essa variável é então atribuída ao componente enqueueSOMsg, através da actividade de cópia de variáveis assignEnqMsg. Esta cópia atribui o valor de `SODescS`[1] ao campo `O_message` do envelope a ser colocado na nova fila de espera.

O componente enqueueSOMsg encontra-se relacionado com o adaptador de AQ `SOQueue`, que é responsável por inserir a mensagem neste novo formato (`SODesc`), encapsulada no envelope, na fila de espera correspondente. Neste caso será a `SO_MC_QUEUE` anteriormente configurada e inicializada.

Após a execução deste processo, deverá existir na `SO_MC_QUEUE` uma mensagem com o envelope de dados, entre eles o campo `O_message` com a mensagem publicada no formato `SODesc`.

5.4.3 Subscrição e Inserção na BD

Este processo não foi implementado por restrições temporais. No entanto encontra-se especificado no capítulo anterior.

5.5 Resumo

Neste capítulo, foi descrita a configuração da máquina virtual, com todas as plataformas de BD e servidores de aplicações necessários à execução de processos BPEL para integração de aplicações "Oracle Retail" (versão 13), com recurso a tecnologias "Fusion Middleware". A configuração da máquina virtual exigiu a instalação de várias aplicações, com todos os detalhes de configuração necessários do Sistema Operativo. Seguiu-se a configuração das Oracle AQ, de forma a criar o *bus* de integração para troca de mensagens. Finalmente, foi descrita a implementação dos processos BPEL relativas ao protótipo desenvolvido, descrevendo os componentes envolvidos no desenvolvimento de cada processo. Ficou em falta a implementação do processo de subscrição e inserção dos dados na segunda aplicação a integrar, que deverá ser concluido após a data de entrega deste relatório. No capítulo seguinte, será demonstrada e avaliada a execução dos processos BPEL implementados.

[1] A mensagem XML no formato `SODesc` convertida em *String*.

Capítulo 6

Exploração e Avaliação

Este capítulo destina-se a avaliar a execução dos processos BPEL. É inicialmente descrita a criação de uma transferência de teste para a execução dos processos implementados. Seguidamente, é descrita a execução dos processos com base na transferência criada. Este capítulo termina com uma avaliação da execução destes processos, discutindo as vantagens em relação às outras plataformas de integração de aplicações "Oracle Retail".

6.1 Introdução

Para executar os processos BPEL, é necessário criar uma transferência, como descrito neste capítulo. É necessário criar uma transferência de forma a que esta possa ser tratada e integrada em ambas as aplicações a integrar. A criação da transferência na aplicação ORMS irá preencher as tabelas temporárias com a informação necessária à criação da mensagem de integração. Essa informação deve ser processada pelos processos BPEL implementados que irão sincronizar essa informação com a segunda aplicação a integrar (SIM). A aplicação SIM é notificada da existência e pormenores da transferência (`Transfer`), convertida numa `Stock Order` (formato aceite pelo SIM). Após a execução dos processos BPEL implementados, é efectuada uma avaliação dessa mesma execução, determinando a possibilidade e viabilidade de uma integração através deste método. Este método de integração é também comparado e destacadas as suas vantagens, em relação a implementações de integração das plataformas ORIB v12 e ORIB v13.

6.2 Criação da Transferência

Para obtermos uma transferência de teste, é necessário executar a aplicação ORMS e criar uma transferência nova. A criação de uma transferência é realizada a partir do menu

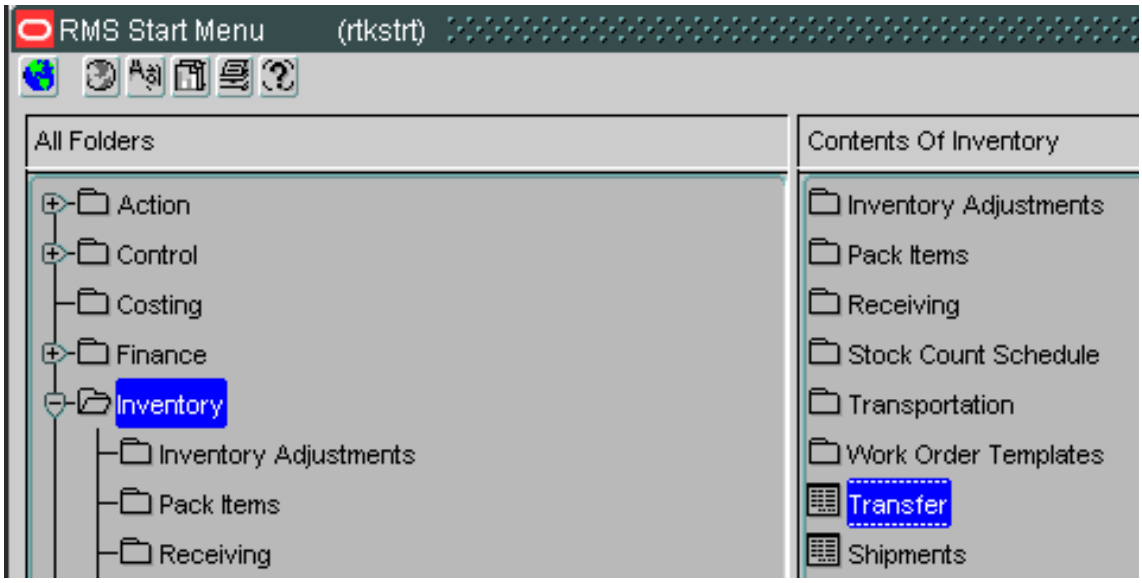

Figura 6.1: Acesso às Transferência na Aplicação ORMS

"Inventory" e clicando em "Transfer", como se pode observar na Figura 6.1. Esta opção dá acesso à criação de transferência, como se pode observar na Figura 6.2.

Para criar uma transferência é necessário designar a origem da transferência e o seu destino. Para este protótipo, foi escolhida a loja "Eagan" como localização de origem, e o armazém "W. Catalog VWH 13" como destino da transferência dos artigos, como se observa na Figura 6.3.

De seguida, é necessário indicar os artigos a transferir. Com uma rápida busca à base de dados, encontram-se os artigos "Test Item 100000307", "Test Item 100000631" e "Test Item 100000964" pertencentes à loja "Eagan". São então adicionados os artigos a esta transferência, com quantidades atribuídas de 30, 40 e 55, respectivamente, como se observa na Figura 6.4.

Os artigos têm que pertencer à loja de origem, uma vez que nos encontramos a efectuar uma transferência de artigos entre duas localizações, sendo necessário que esses artigos existam na localização de origem. Verifica-se facilmente a localização desses artigos como pertencentes à loja de destino, navegando até à opção "*Location*", no menu de visualização dos detalhes do artigo (ver Figura 6.5)

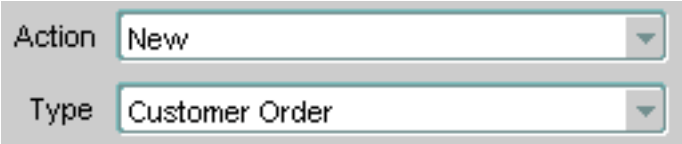

Figura 6.2: Criação de Nova Transferência

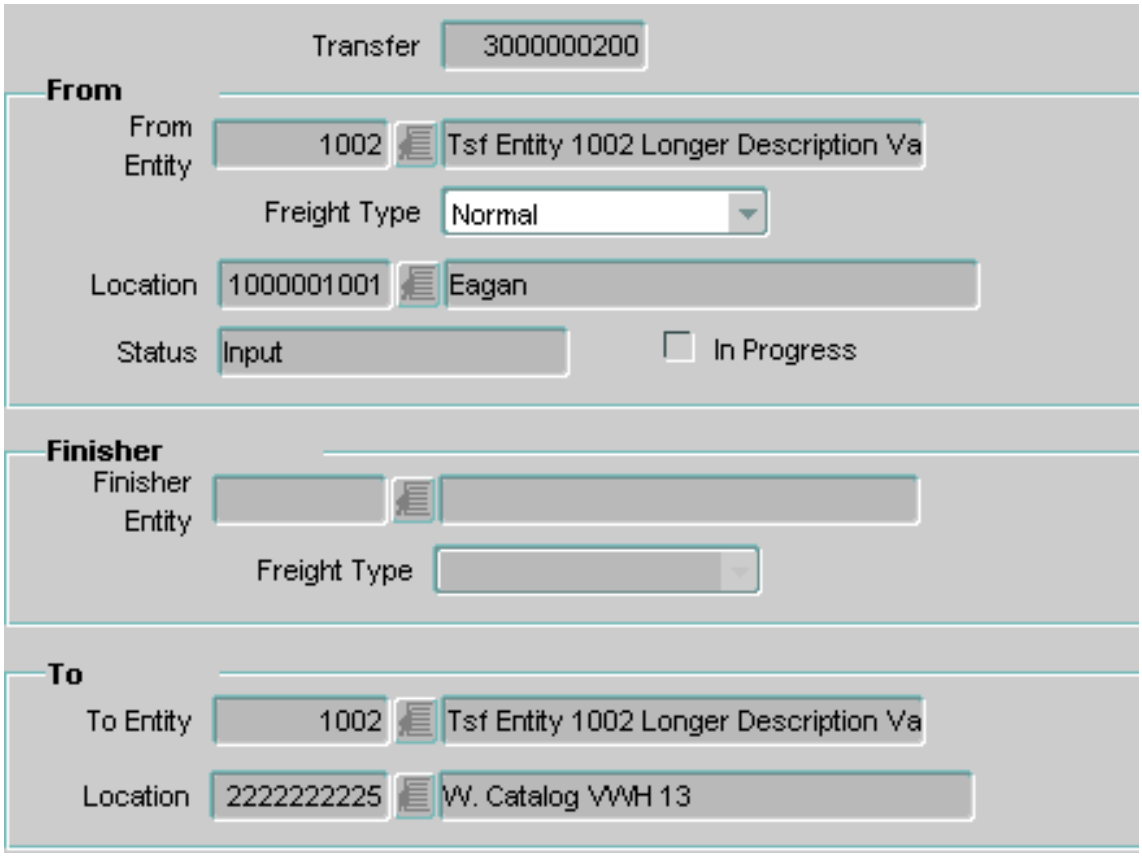

Figura 6.3: Designação da Origem e Destino da Transferência

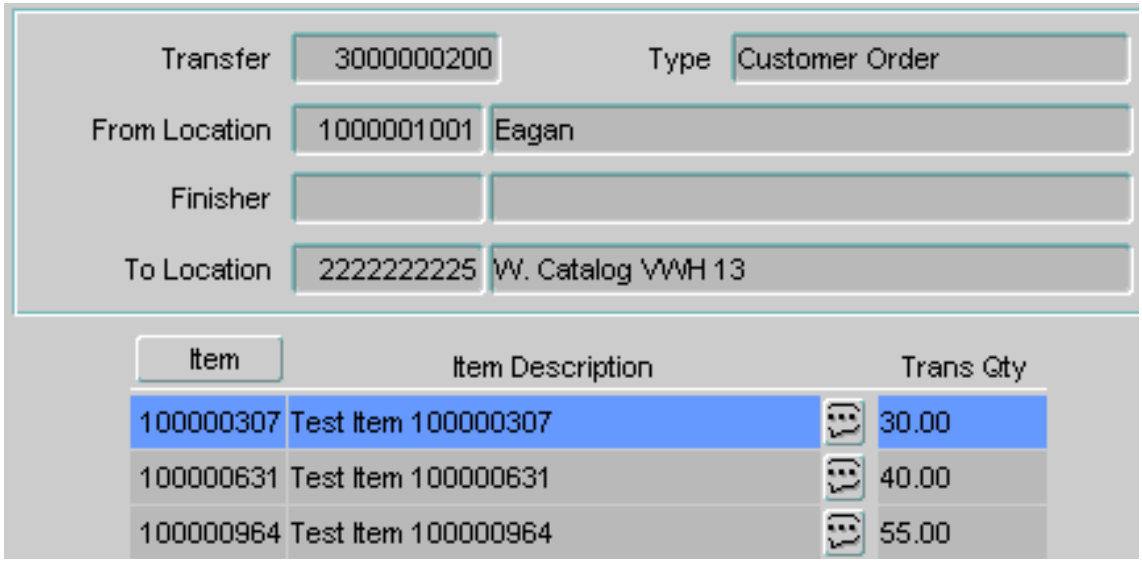

Figura 6.4: Designação dos Artigos e Quantidades a Transferir

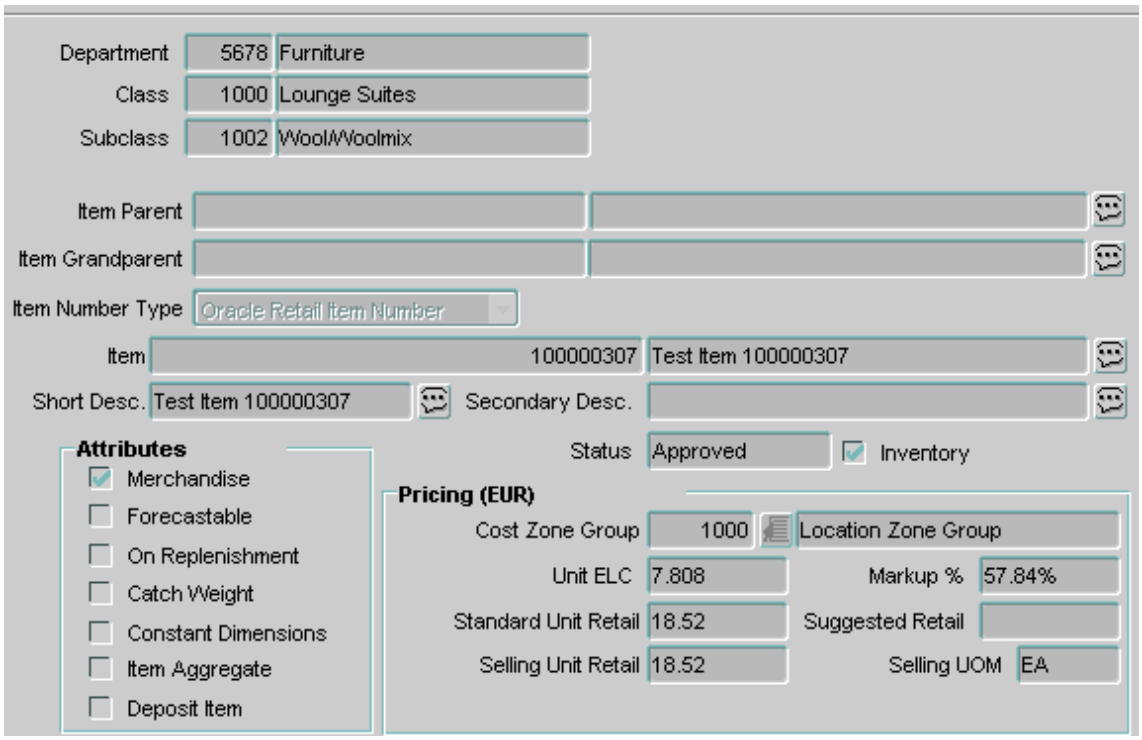

Figura 6.5: Detalhes do Artigo "Test Item 100000307"

No ecrã "*Location*", como se pode observar na Figura 6.6, facilmente se verifica que a loja "Eagan" possui esse mesmo artigo. O mesmo acontece com os restantes artigos atribuídos à transferência.

Após criada e aprovada a transferência, é criado um registo na tabela `tsf_mfqueue` com os dados de referência da transferência aprovada e criada, como se verifica na Figura 6.7, incluindo o campo de referência `tsf_no`. Este campo é importante para que os dados da transferência sejam facilmente obtidos das tabelas onde a informação da mesma se encontra.

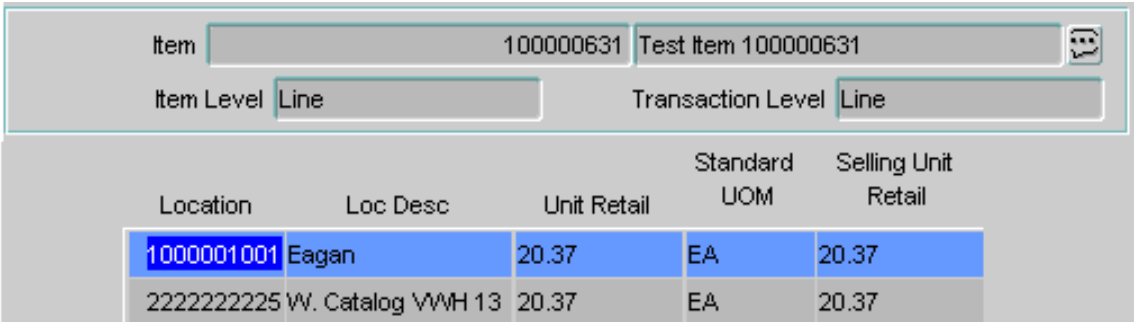

Figura 6.6: Detalhes da Localização do Artigo "Test Item 100000307"

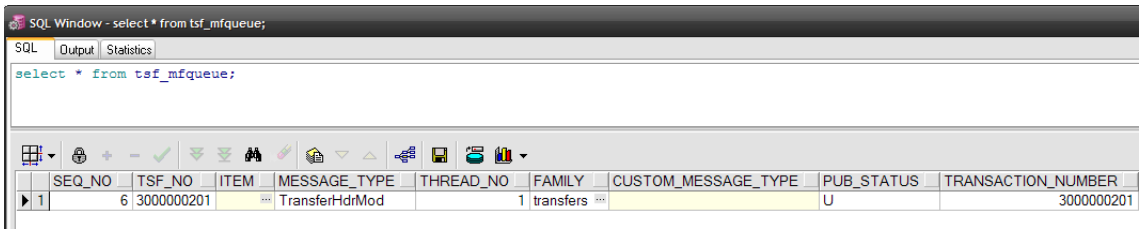

Figura 6.7: Detalhe da Tabela `tsf_mfqueue`

Com a transferência criada e devidamente aprovada, é agora possível executar os processos BPEL, de forma a que a informação desta transferência seja partilhada com a aplicação SIM.

6.3 Execução dos Processos BPEL

Após a implementação dos processos, foi feito o *deployment* dos mesmos no servidor BPEL. A execução do processo de publicação (ver Secção 5.4.1) termina com a colocação de uma mensagem na fila de espera `TSFS_MC_QUEUE`, pertencente à tabela `TSFS_MCQUEUE_TBL`. Este processo origina vários documentos XML aquando da sua execução. O componente que retira os dados da BD, relativos à transferência tem como resultado o documento XML descrito no Anexo A.1. Este documento atribui os valores retirados da base de dados à variável do componente getTransfer. Esta variável é posteriormente mapeada para o formato `TsfDesc`.

Durante a execução deste mesmo processo, o componente de mapeamento dos dados da transferências retirados da BD (tsfInfoMAPTsfDesc) origina correctamente o XML resultante da transformação no formato `TsfDesc`, como se pode observar no Anexo A.2. Posteriormente, durante a execução deste mesmo processo, é gerado um novo XML, através da execução do componente msgEnqueue, como se pode observar no Anexo A.3. Após a execução do processo 5.4.1, recorrendo à ferramenta "PL/SQL Developer", pode verificar-se que a mensagem é publicada na AQ `TSFS_MC_QUEUE`, como se pode observar na Figura 6.8.

Observando o campo destacado a vermelho na Figura 6.8, verifica-se que a mensagem foi correctamente inserida, com o campo `O_message` preenchido com a mensagem publicada no formato `TsfDesc`. Verifica-se ainda que, apesar do campo `O_message` ser do tipo `CLOB`, contendo a informação no tipo *String*, a mensagem é correctamente identificada como sendo um documento XML, tal como se pode observar na Figura 6.9.

A execução do processo BPEL de transformação de formatos (ver Secção 5.4.2) retira a mensagem da *enqueue* `TSFS_MC_QUEUE` no formato `TsfDesc`. De seguida efectua o mapeamento para o formato `SODesc` e volta a colocar a mensagem na fila de es-

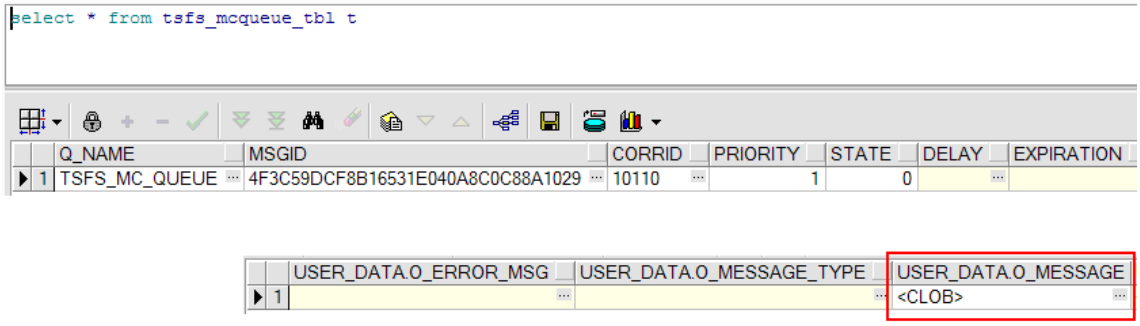

Figura 6.8: Mensagem Inserida em `TSFS_MC_QUEUE`

pera `SO_MC_QUEUE`. O mapeamento entre os formatos `TsfDesc` e `SODesc`, implementado pelo componente tsfMAPSODesc gera correctamente o XML baseado no *schema* `SODesc.xsd`, descrito no Anexo A.4.

A execução deste processo contêm ainda o componente msgEnqueue, que insere a mensagem na fila de espera `SO_MC_QUEUE`, gerando o XML discriminado no Anexo A.5. A mensagem é colocada desta vez na fila de espera `SO_MC_QUEUE`. De novo, recorrendo ao "PL/SQL Developer", pode confirmar-se a inserção da mensagem na AQ, como se observa na Figura 6.10. Tal como na mensagem anterior, do tipo `TsfDesc.xsd`, o campo `O_message`, destacado a vermelho possui a mensagem publicada num documento XML, desta vez no formato `SODesc.xsd`, como se verifica na Figura 6.11.

Fica em falta a implementação e execução do processo BPEL de subscrição, que deveria de seguida retirar a nova mensagem da fila de espera `SO_MC_QUEUE` e inserir os registos nas tabelas apropriadas, como descrito na especificação, na Secção 4.4.4 do capítulo 4.

A possibilidade de inserir documentos XML na fila de espera, permite que a mensagem mantenha o seu formato rigidamente definido e facilita a sua manipulação após ser retirada. Assim, torna-se possível retirar conteúdo do campo `O_message`, da mensagem da AQ, directamente para o formato `TsfDesc.xsd`, `SODesc.xsd`, ou outro formato específico do tipo de mensagem que se queira publicar na AQ. A utilização do XML possui aqui um papel importante, no sentido em que permite *enqueue/dequeue* de mensagens das filas de espera AQ, com campos definidos num formato específico.

6.4 Avaliação da Integração com Processos BPEL

A integração com recurso a processos BPEL apresenta grandes diferenças face à implementação baseada em ORIB v12, a plataforma de integração anterior "Oracle Retail". Como já tinha sido descrito no Capítulo 2, especificamente na Secção 2.5, a plataforma de integração ORIB v12 exige a existência de dois métodos distintos de implementação.

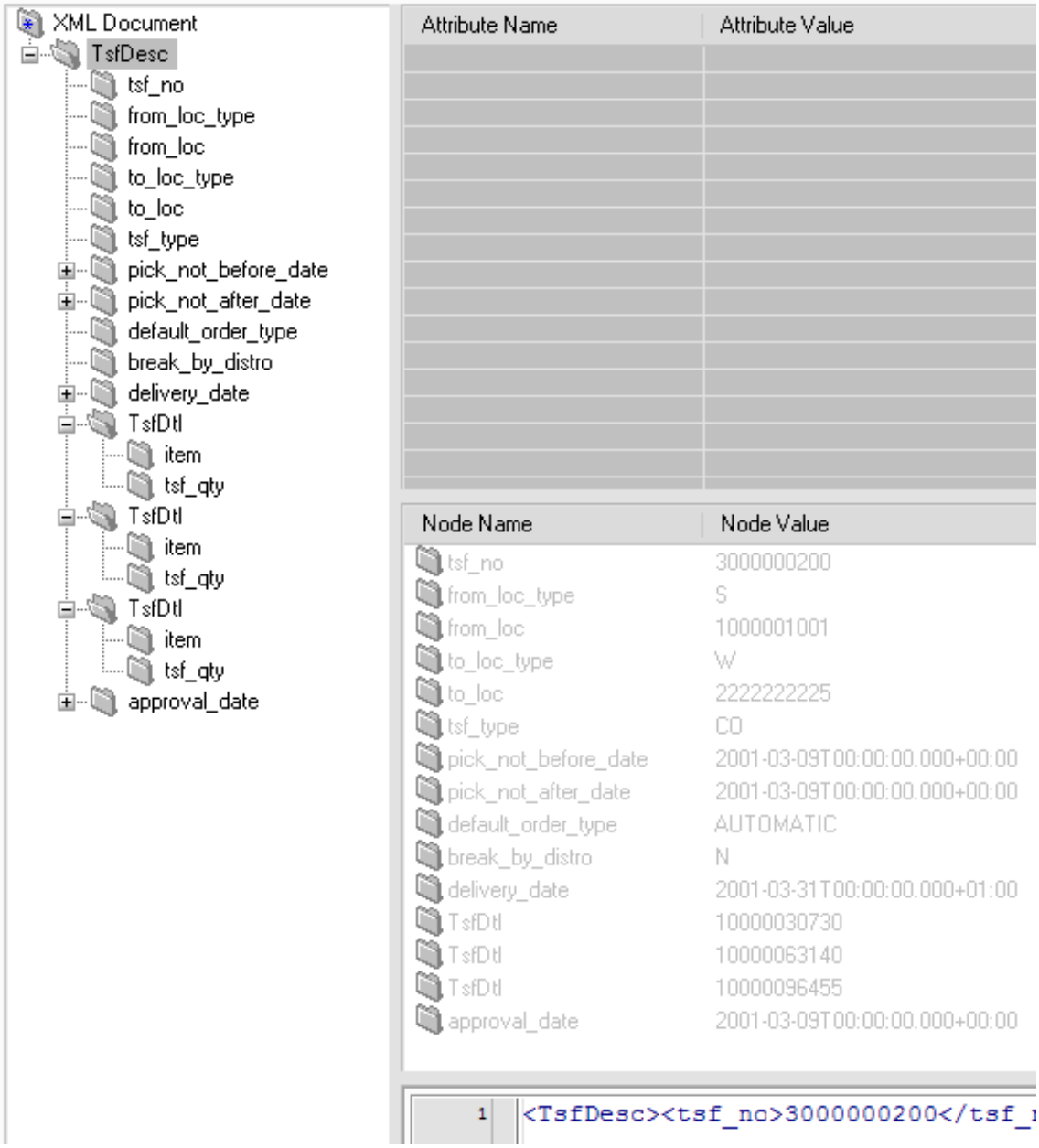

Figura 6.9: Campo `O_message` da Mensagem em `TSFS_MC_QUEUE`

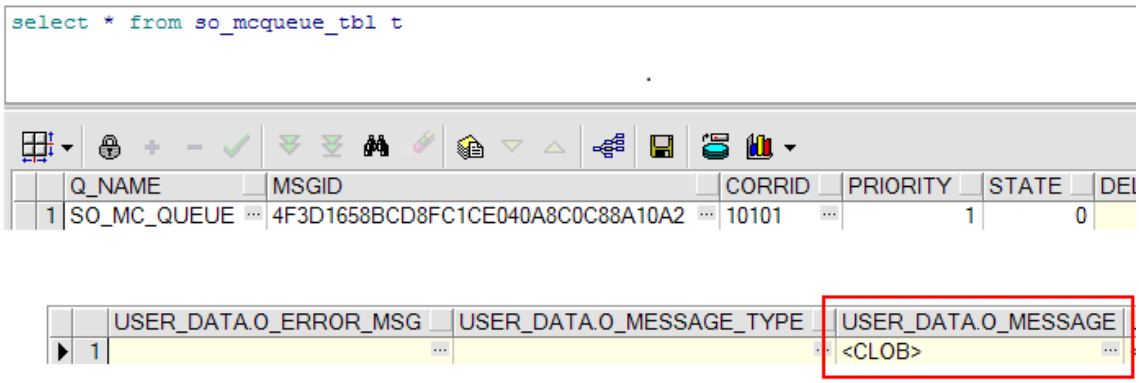

Figura 6.10: Mensagem Inserida em `SO_MC_QUEUE`

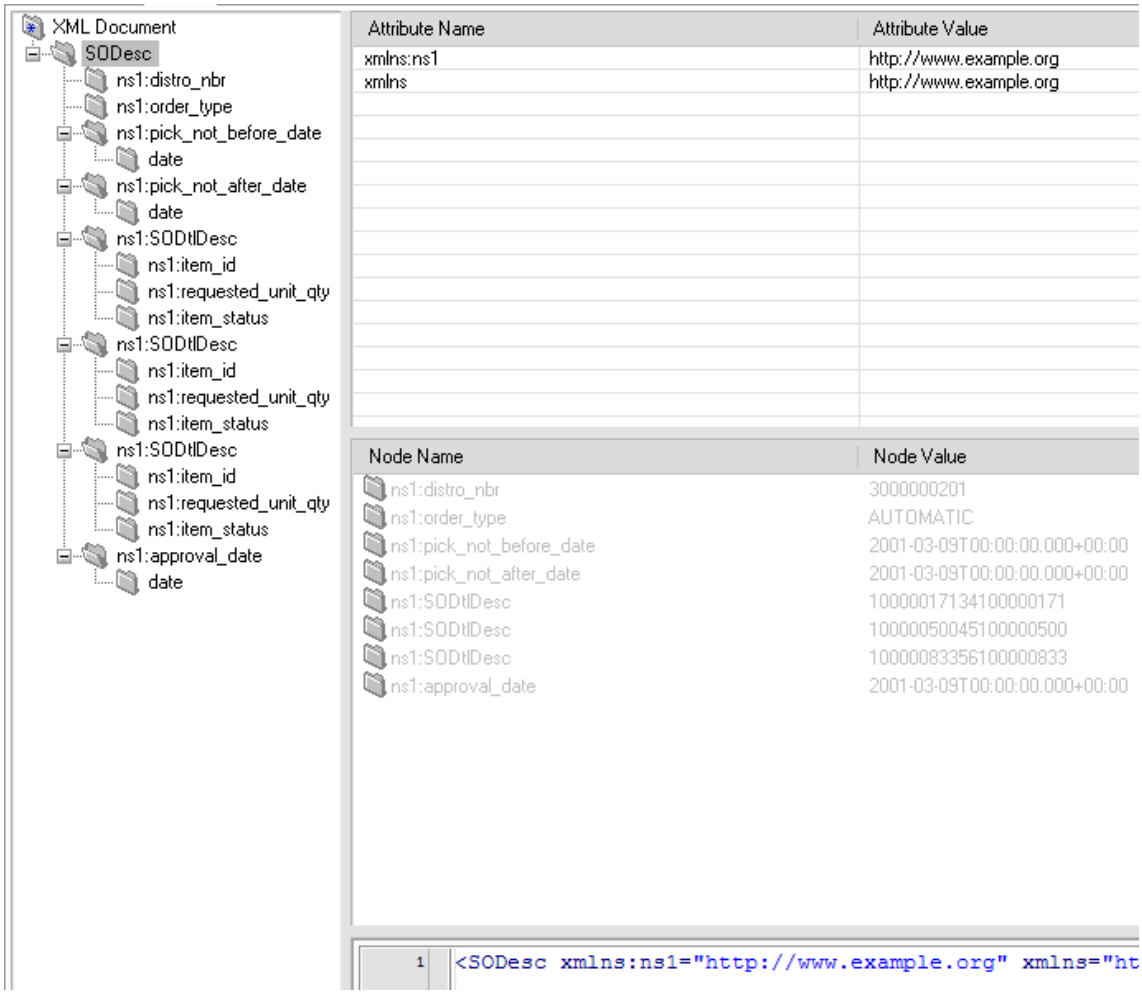

Figura 6.11: Campo `O_message` da Mensagem em `SO_MC_QUEUE`

Um método para a integração de aplicações J2EE (ver Secção 2.5.2 na Página 16) e outro para aplicações não J2EE (ver Secção 2.5.1 na Página 14).

A integração de aplicações J2EE baseada na plataforma ORIB v12 exige o uso de EJB e MDB, ao passo que uma integração de aplicações não J2EE exige a utilização de componentes e*Way. Ambas as implementações utilizam `JMS Queues` como *bus* de integração.

Na versão 13 do ORIB, a diferenciação entre implementações J2EE e não J2EE não existe. A implementação de aplicações baseadas ou não em J2EE passa a ser tratada apenas com a utilização de EJB para a publicação e MDB para a subscrição e TAFR. No entanto, a configuração do TAFR e a utilização de EJB e MDB continuam a exigir o dispêndio de tempo em termos de código a ser escrito. É necessário configurar e personalizar os "Java Beans" para publicação e subscrição e a configuração de MDB para TAFR pode, por vezes, tornar-se uma tarefa complicada e morosa.

Com uma integração baseada em processos BPEL será possível reduzir a produção de código ao mínimo. A utilização de "JDeveloper" oferece um ambiente gráfico *user-friendly* que facilita bastante a implementação de processos BPEL. A existência de adaptadores e actividades facilmente inseridas no processo por um meio simples de *drag-and-drop* e a configuração auxiliada destes componentes, tornam a implementação destes processos mais rápida, mais fiável e mais fácil.

A configuração de TAFR para mapeamento entre formatos diferentes de aplicações "Oracle Retail" que exigem formatos distintos de mensagens, como já foi referido, é facilitada pela ferramenta de mapeamento do "JDeveloper". Torna-se assim obsoleta a necessidade de produção de código susceptível a erros em MDB. A unificação do método de integração na versão 13 do ORIB é uma vantagem em relação à plataforma anterior. No entanto, essa vantagem mantem-se com a implementação baseada em processos BPEL. Não existe igualmente a distinção entre aplicações baseadas ou não em J2EE, uma vez que os processos BPEL para integração de aplicações "Oracle Retail" recorrem a adaptadores de DB e AQ baseados em WS.

A existência de um sistema de filas de espera baseadas em Oracle AQ é também uma vantagem da plataforma ORIB v13, igualmente utilizável na implementação de processos BPEL. Os adaptadores AQ baseados em WS permitem facilmente efectuar operações de inserção e remoção (*enqueue/dequeue*) de mensagens nos tópicos das AQ. Estes adaptadores são inteiramente personalizáveis e permitem configurar propriedades das AQ, como propagação, *scheduling*, modos de *dequeuing*. Permite ainda definir a lista de receptores da mensagem para mensagens de múltiplos consumidores e a definição de subscritores baseados em regras, de forma a que estes recebam apenas mensagens com propriedades ou conteúdos específicos. Como facilmente se verifica, os adaptadores AQ em processos BPEL encontram-se personalizáveis o suficiente para um uso completo de todas as suas propriedades.

Ainda referente à utilização de Oracle AQ, não existem medições oficiais pela Oracle quanto à sua eficiência. É certo que a utilização de uma tecnologia suportada pela mesma empresa que detém as aplicações a integrar, supostamente deverá fornecer melhorias no suporte à implementação de integração das aplicações "Oracle Retail". No entanto, a eficiência das Oracle AQ, um sistema de filas de espera baseadas em BD, em comparação com "JMS Queues", uma implementação baseada em ficheiros, deve ser um factor a ter em conta. Num ambiente empresarial onde ocorrem milhares, se não milhões de transacções por dia, o sobrecarregamento da BD devido à existência das Oracle AQ (com a adição das transacções que requer) afecta a eficiência de todo o sistema. É assim necessário saber se a diferença entre as implementações (JMS Queues/AQ) não compromete a eficiência do sistema de modo significativo.

A utilização de WS e do formato XML para ligações e transferência de informação entre componentes representa mais uma vantagem da implementação baseada em processos BPEL. Os WS são cada vez mais aceites e completamente implementáveis na integração de aplicações "Oracle Retail", logo será um factor a ter em conta e um componente de utilização quase imperativa, dado que existe essa possibilidade. O formato XML representa um standard que permite facilmente comunicar entre componentes internos ao processo BPEL, assim como manter documentos de formato rígido para mensagens de integração trocadas entre as aplicações "Oracle Retail". O uso de XML e WS facilita ainda a integração de aplicações "Oracle Retail" com outras aplicações Oracle baseadas em Fusion Middleware.

6.5 Resumo

Neste capítulo, foi inicialmente demonstrada a execução dos processos BPEL implementados, com base na criação de uma transferência na aplicação ORMS. Da sua execução comprovou-se a viabilidade de uma integração baseada neste método, sendo feita a avaliação do mesmo após a descrição da execução. No capítulo seguinte, são retiradas as conclusões deste projecto, tanto em relação ao desenvolvimento do protótipo e do estudo das plataformas de integração existentes, como do processo de estágio em si.

Capítulo 7

Conclusões

Neste capítulo serão apresentadas as conclusões retiradas, tanto ao nível do projecto em si, como do estágio. As conclusões relativas ao projecto centram-se no desenvolvimento do protótipo e nas vantagens da abordagem de integração implementada, em relação a outras plataformas existentes no pacote de aplicações "Oracle Retail", tanto da versão 12 como da versão 13. As conclusões acerca do estágio prendem-se com o ambiente de trabalho, disponibilização de documentação, formação oferecida no âmbito do estágio e relação com os elementos da empresa. No final deste capítulo, é ainda dada uma perspectiva sobre o trabalho futuro a realizar no âmbito deste projecto. Este capítulo conclui o relatório de estágio de avaliação da plataforma de integração "Oracle Retail".

7.1 Sobre o Projecto

Este projecto possuiu como base a investigação de um método de integração baseado em tecnologias recentes. Destas tecnologias destaca-se o desenvolvimento de processos BPEL e o sistema de filas de espera "Oracle Advanced Queues". Como em todos os projectos de investigação, existem sempre obstáculos inerentes à aprendizagem e descoberta de novas tecnologias, acrescentando ainda o facto de este projecto assentar em métodos de integração não explorados anteriormente, para o caso particular de aplicações "Oracle Retail". Esta inexistência de anteriores implementações ou artigos relacionados tornou os obstáculos por vezes mais difíceis de ultrapassar.

Assim sendo, o planeamento inicialmente previsto não foi seguido com precisão. O lançamento da plataforma de integração ORIB v13 no final do mês de Abril, exigiu que a sua instalação e a instalação do ambiente de suporte[1] fossem realizadas de forma a executar os processos BPEL. Os processos foram desenvolvidos desde o início, sendo

[1] VM com Oracle RDBMS + OASF&R 10.1.2.2 + OAS 10.1.3.3 + ORMS13 + ORIB13 + BPEL PM

suspensos aquando do aparecimento desta nova versão das aplicações. Esta instalação tardia do ambiente revelou-se mais complicada do que inicialmente previsto, devido à inexperiência de instalação de ambientes Oracle. Obviamente, os objectivos propostos não foram alcançados na sua totalidade, uma vez que ficou por implementar o terceiro processo BPEL de inserção dos dados na segunda aplicação a integrar. No entanto, a especificação do protótipo foi finalizada e a sua implementação será continuada, de forma a entregar à empresa o valor total desta investigação. Estando finalizado o protótipo, este deverá ser refinado e optimizado de forma a permitir uma implementação, com o objectivo de chegar à fase de produção para integrações em sistemas de clientes da empresa.

Com este projecto foi possível determinar um novo método de integração das aplicações de retalho Oracle, graças à investigação destas novas tecnologias e da potencialidade das mesmas a nível de integração de aplicações "Oracle Retail". A utilização de processos BPEL permite uma integração baseada em WS, por sua vez descritos no formato XML. Isto permite não só integrar as aplicações "Oracle Retail" entre si, mas ainda vir a integrar estas aplicações com outros módulos Oracle fora da área de retalho. Isto deve-se à implementação baseada em WS, possibilitando que outras aplicações Oracle façam uso das tecnologias "Fusion Middleware" para comunicar com o *middleware* das aplicações "Oracle Retail".

Actualmente, como foi referido, a versão 13 das aplicações "Oracle Retail" recorrem à implementação de integração baseada em EJB e MDB. Foi obtida, assim, uma antevisão do que será a integração de aplicações "Oracle Retail" nas próximas versões destas aplicações, com o recurso a estas tecnologias "Fusion Middleware".

Este método de integração é, no entanto um pouco complexo, sendo necessárias várias alterações em cada um dos processos de forma a poder ser reutilizado nos outros casos de integração, entre mensagens e aplicações diferentes. Estas desvantagens de implementação deverão ser resolvidas na próxima versão das aplicações "Oracle Retail" e respectiva plataforma de integração. Esta versão já deverá possuir uma integração baseada nestas tecnologias "Fusion Middleware", nomeadamente a utilização de processos BPEL, que deverão substituir os "Java Beans", e do sistema de filas de espera Oracle AQ.

O valor final deste projecto para a empresa, consiste em fornecer uma visão antecipada da implementação de tecnologias "Fusion Middleware" na integração das aplicações "Oracle Retail". Permite ainda a familiarização com tecnologias nunca antes utilizadas na empresa, de forma a preparar os elementos do núcleo de integração para a eventual e inevitável utilização das mesmas em versões futuras da plataforma de integração "Oracle Retail". Para além destes factores, este protótipo permite obter uma perspectiva para o futuro da integração, baseada em Web Services. A optimização deste protótipo e posterior implementação em projectos futuros, irá fornecer um método de integração inovador nesta área de negócio, colocando a empresa um passo à frente na integração de aplicações "Oracle Retail".

7.2 Sobre o Estágio

O ambiente interno na empresa onde foi efectuado o estágio permitiu uma autonomia total sobre o projecto em si, não descurando um espírito de entreajuda entre os elementos da empresa. Sempre que necessário, por ser imperativo lidar com plataformas, ambientes e tecnologias nunca antes experimentados pelo autor do projecto, foi demonstrada uma disponibilidade célere por qualquer elemento da organização, de forma a auxiliar no que fosse necessário e possível a nível de *know-how* interno. Esse auxílio proveio ainda da base de conhecimento da organização, da qual foi possível obter a documentação necessária.

No entanto, uma vez que as tecnologias, para além de recentes, nunca haviam sido implementadas numa abordagem de integração, a autonomia e o esforço de aprendizagem ditaram o sucesso do projecto, dentro do possível. Noutra perspectiva, este projecto permitiu a adquirir experiência de trabalho com plataformas de DB e com implementações de integração de alto nível dentro de empresas de média e grande dimensão. A experiência obtida com implementações de aplicações "Oracle Retail" veio a fornecer uma bagagem necessária quando se entra no mundo de aplicações orientadas a bases de dados de grande calibre. Esta experiência foi também possibilitada devido à formação intensiva fornecida pela empresa nas aplicações "Oracle Retail".

7.3 Trabalho Futuro

Como em qualquer projecto de investigação, existe sempre algo a melhorar. Neste projecto não foi possível terminar o protótipo, pelo que o último processo BPEL, embora especificado, deverá ser terminado após a entrega deste relatório de projecto.

As melhorias mais assinaláveis em relação aos processos desenvolvidos emergem da necessidade de existência de diferentes utilizadores passíveis de operar as AQ. A criação de utilizadores distintos para cada aplicação é necessária, caso este método de integração entre em produção para projectos reais. Antes ainda de ser passível de desenvolvimento e posterior produção, estes processos devem ser unificados. A unificação dos processos de publicação, transformação e subscrição permitirá uma integração segura e fiável.

Uma vez que este é apenas um protótipo inicial, deve-se melhorar pequenos pormenores de implementação, sejam eles relacionados com a optimização do processo BPEL em si ou relativos a operações de inserção e remoção de mensagens das AQ. O elevado número de transacções por hora derivado de uma abordagem de filas de espera, baseadas em BD, exige que sejam efectuadas medições de desempenho e eficiência do sistema, para garantir a fiabilidade da integração, uma vez que todas as aplicações integradas possuem ainda as suas próprias transacções de BD.

A afinação do protótipo e o desenvolvimento de novos processos BPEL optimizados especificamente para cada aplicação a integrar, podem trazer mais valias a nível de desempenho. No entanto, este protótipo pretende ser generalista e, a partir do exemplo em que se baseia, ser reutilizável para qualquer integração entre aplicações "Oracle Retail", com pequenos ajustes nos adaptadores de BD e de AQ e nos formatos de mensagens específicos das aplicações a integrar.

Como se pode observar, esta é apenas a base para a integração de aplicações "Oracle Retail" baseadas em processos BPEL e Oracle AQ. Estas são tecnologias poderosas que irão, no futuro, facilitar a integração ao nível de Oracle. No entanto, exigem um estudo mais aprofundado e preciso, que deverá ser continuado.

Referências

[AS06] Ricardo Alexandre Sousa. J2EE - RIB Integration. Technical report, Wipro Technologies, Novembro 2006.

[Aut08a] Allround Automations. Allround Automations PL/SQL Developer 7.1, 2008. http://www.allroundautomations.nl/plsqldev71.html.

[Aut08b] Allround Automations. PL/SQL Developer - The IDE, 2008. http://www.allroundautomations.com/plside.html.

[Bar05] Sidhartha Bardoloye. Enterprise Application Integration. Technical report, Wipro Technologies, Junho 2005.

[Con08] GreenHat Consulting. Fonte de imagem greenHat Consulting p2p, 2008. http://www.greenhatconsulting.com/images/eGov_point2point.gif.

[Cor02a] Oracle Corporation. AQ Intro, 2002. http://download-uk.oracle.com/docs/cd/B10501_01/appdev.920/a96587/qintro.htm.

[Cor02b] Oracle Corporation. AQ Tutorials, 2002. http://download-west.oracle.com/docs/cd/A97630_01/appdev.920/a96587/apexampl.htm#62214.

[Cor05a] Oracle Corporation. Oracle JDBC FAQ, 2005. http://www.oracle.com/technology/tech/java/sqlj_jdbc/htdocs/jdbc_faq.htm.

[Cor05b] Oracle Corporation. Streams Advanced Queuing: Overview and Tutorial. Technical report, Oracle Corporation, Junho 2005.

[Cor08a] Oracle Corporation. JD Edwards Enterprise One, 2008. http://www.oracle.com/applications/jdedwards-enterprise-one.html.

[Cor08b] Oracle Corporation. Oracle and Stellent, 2008. http://www.oracle.com/stellent/index.html.

[Cor08c] Oracle Corporation. Oracle Applications - Fusion Middleware, 2008. http://www.oracle.com/applications/fusion.html.

[Cor08d] Oracle Corporation. Oracle BPEL, 2008. http://www.oracle.com/technology/products/ias/bpel.

[Cor08e] Oracle Corporation. Oracle E-Business Suite, 2008. http://www.oracle.com/applications/e-business-suite.html.

[Cor08f] Oracle Corporation. Oracle Retail, 2008. http://www.oracle.com/.

[Cor08g] Oracle Corporation. Oracle Retail, 2008. http://www.oracle.com/partners/home/pf/global/retail/auth/index.html.

[Cor08h] Oracle Corporation. PeopleSoft Enterprise, 2008. http://www.oracle.com/applications/peoplesoft-enterprise.html.

[Cor08i] Oracle Corporation. Rib 13.0.x Integration Guides. Technical report, Oracle Corporation, Abril 2008.

[Cor08j] Oracle Corporation. Siebel CRM Software, Customer Relationship Management, Business Analytics, 2008. http://www.oracle.com/siebel/index.html.

[Cou01] Bill Coulam. PL/SQL Developer by Allround Automations software review, 2001. http://www.orafaq.com/tools/allround/developer.htm.

[Ena08] Enabler. Enterprise Application Integration, Junho 2008. Documentação interna Wipro Retail.

[FBVRR05] Marta Fernandes Borlido e Vitor Rangel Rodrigues. EAI and SOA Business Benefits. Technical report, Wipro Technologies, Outubro 2005.

[Fen03] James Fenner. Enterprise Application Integration Techniques. Technical report, University College London, 2003.

[GDKB04] John Garmany e Donald K. Burleson. *Oracle Application Server 10g Architecture and Administration*. McGraw-Hill/Osborne, Julho 2004. http://www.devshed.com/c/a/Oracle/Oracle\penalty\@M\hskip\z@skip\discretionary{-}{}{}\penalty\@M\hskip\z@skip-Application\penalty\@M\hskip\z@skip\discretionary{-}{}{}\penalty\@M\hskip\z@skip-Server\penalty\@M\hskip\z@skip\discretionary{-}{}{}\penalty\@M\hskip\z@skip-10g\penalty\@M\hskip\z@skip\discretionary{-}{}{}\penalty\@M\hskip\z@skip-Architecture\penalty\@M\hskip\z@skip\discretionary{-}{}{}\penalty\@M\hskip\z@skip-and\penalty\@M\hskip\z@skip\discretionary{-}{}{}\penalty\@M\hskip\z@skip-Administration/.

[GRSDB04] Rick Greenwald, Robert Stackowiak e Donald Bales. *Oracle Application Server 10g Essentials*. O'REILLY, First edition, 2004.

[He04] Hao He. Implementing REST Web Services: Best Practices and Guidelines, 2004. http://www.xml.com/pub/a/2004/08/11/rest.html.

[Kis07] A. N. V. R. K. R. Kishore. EAI with Web Services. Technical report, Wipro Technologies, Julho 2007.

[MG08] Object Management Group. CORBA FAQ, 2008. http://www.omg.org/gettingstarted/corbafaq.htm.

[Mic03a] Sun Microsystems. Class StringWriter (JavaTM 2 Platform Std. Ed. v1.4.2), 2003. http://java.sun.com/j2se/1.4.2/docs/api/java/io/StringWriter.html.

[Mic03b] Sun Microsystems. Class Transformer (JavaTM 2 Platform Std. Ed. v1.4.2), 2003. http://java.sun.com/j2se/1.4.2/docs/api/javax/xml/transform/Transformer.html.

[Mic08a] Sun Microsystems. Enterprise javabeans technology, 2008. http://java.sun.com/products/ejb/.

[Mic08b] Sun Microsystems. Sun Java, 2008. http://java.sun.com/.

[OJG07] Adérito Oliveira e Joana Gonzalez. SOA, ESB and BPEL, Outubro 2007. Sessão de informação, documento interno Wipro Retail.

[Pra08] Lakshmi Prasanna. Java Messaging Service - an Overview. Technical report, Wipro Technologies, Junho 2008.

[Rei07] Cláudio Reis. Oracle Retail Integration - RIB + RSL, Outubro 2007. Documentação interna Wipro Retail.

[Rei08] Cláudio Reis. Retail Integration Bus v13, Abril 2008. Documentação interna Wipro Retail.

[RJG02] Raghu Ramakrishnan e Johannes Gehrke. *Database Management Systems*. McGraw-Hill, 2002.

[Ruo08] Hiro Ruo. Chapter 24 - Distributed Component Object Model, 2008. http://docs.rinet.ru/ZhPP/ch24.htm.

[Sch08] Shay Schmeltzer. Oracle JDeveloper Overview. Technical report, Oracle Corporation, Janeiro 2008.

[SDB04] Robert Stackowiak e Donald Bales. *Oracle Application Server 10g Essentials*. Oreilly & Associates Inc, 2004.

[Sea08] SearchSOA.com. What is UDDI?, 2008. http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci508228,00.html.

[See00] SeeBeyond. e*Gate Integrator - The eBusiness Integration Platform. Technical report, SeeBeyond, November 2000.

[Smi05] J. Smith. *Virtual Machines: Versatile Platforms for Systems & Processes*. MORGAN KAUFMANN, 2005.

[SRC08] John Stearns e Roberto Chinnici. An Introduction to the Java EE 5 Platform, 2008. http://java.sun.com/developer/technicalArticles/J2EE/intro_ee5/.

[Swa04] Steven Swafford. Product review: PL/SQL Developer, 2004. http://aspalliance.com/614.

[Sys08a] Cisco Systems. A simple definition of Electronic Data Interchange (EDI), 2008. http://www.freewaycommerce.co.uk/define-electronic-data-interchange.html.

[Sys08b] Cisco Systems. SSL: Introduction to Secure Sockets Layer, 2008. http://www.cisco.com/en/US/netsol/ns340/ns394/ns50/ns140/networking_solutions_white_paper09186a0080136858.shtml.

[Tec08] Wipro Technologies. Wipro Technologies Website, 2008. http://www.wipro.com.

[TEFGK02] P. Th. Eugster, P. Felber, R. Guerraoui e A. M. Kermarrec. The Many Faces of Publish/Subscribe. Technical report, Swiss Federal Institute of Technology, Lausanne, CH and Bell Laboratories, Murray Hill, USA and Microsoft Research Ltd., Cambridge, UK, Abril 2002.

[VMW08] VMWare. VMware Server Features, Free Virtualization Server Consolidation, 2008. http://www.vmware.com/products/server/features.html.

[Wik08a] Wikipedia. Application Programming Interface, 2008. http://en.wikipedia.org/wiki/API.

[Wik08b] Wikipedia. Message-Oriented Middleware, 2008. http://en.wikipedia.org/wiki/Message_Oriented_Middleware.

[Wik08c] Wikipedia. S/MIME, 2008. http://en.wikipedia.org/wiki/SMIME.

Anexo A

Resultado da Excecução BPEL

A.1 Componente `getTransfer`

```
<messages>
 <getTransfer_TransfersDB_InputVariable>
  <part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                    name="TransfersDBInput_msg">
  <TransfersDBInput
  xmlns="http://xmlns.oracle.com/pcbpel/adapter/db/TransfersDB"/>
  </part>
 </getTransfer_TransfersDB_InputVariable>
 <getTransfer_TransfersDB_OutputVariable>
  <part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                       name="response-headers">
  []
  </part>
  <part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                             name="TransfersDBOutputCollection">
  <TransfersDBOutputCollection
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://xmlns.oracle.com/pcbpel/adapter/db/TransfersDB">
    <TransfersDBOutput>
      <TSF_NO>3000000200</TSF_NO>
      <MESSAGE_TYPE>TransferUnapp</MESSAGE_TYPE>
      <PUB_STATUS>U</PUB_STATUS>
      <ITEM>100000307</ITEM>
      <TSF_QTY>30</TSF_QTY>
      <FROM_LOC>1000001001</FROM_LOC>
      <FROM_LOC_TYPE>S</FROM_LOC_TYPE>
      <TO_LOC>2222222225</TO_LOC>
      <TO_LOC_TYPE>W</TO_LOC_TYPE>
      <TSF_TYPE>CO</TSF_TYPE>
      <APPROVAL_DATE>2001-03-09T00:00:00.000+00:00</APPROVAL_DATE>
      <DELIVERY_DATE>2001-03-31T00:00:00.000+01:00</DELIVERY_DATE>
      <DEFAULT_ORDER_TYPE>AUTOMATIC</DEFAULT_ORDER_TYPE>
    </TransfersDBOutput>
    <TransfersDBOutput>
      <TSF_NO>3000000200</TSF_NO>
      <MESSAGE_TYPE>TransferUnapp</MESSAGE_TYPE>
      <PUB_STATUS>U</PUB_STATUS>
      <ITEM>100000631</ITEM>
      <TSF_QTY>40</TSF_QTY>
      <FROM_LOC>1000001001</FROM_LOC>
      <FROM_LOC_TYPE>S</FROM_LOC_TYPE>
```

```
      <TO_LOC>2222222225</TO_LOC>
      <TO_LOC_TYPE>W</TO_LOC_TYPE>
      <TSF_TYPE>CO</TSF_TYPE>
      <APPROVAL_DATE>2001-03-09T00:00:00.000+00:00</APPROVAL_DATE>
      <DELIVERY_DATE>2001-03-31T00:00:00.000+01:00</DELIVERY_DATE>
      <DEFAULT_ORDER_TYPE>AUTOMATIC</DEFAULT_ORDER_TYPE>
   </TransfersDBOutput>
   <TransfersDBOutput>
      <TSF_NO>3000000200</TSF_NO>
      <MESSAGE_TYPE>TransferUnapp</MESSAGE_TYPE>
      <PUB_STATUS>U</PUB_STATUS>
      <ITEM>100000964</ITEM>
      <TSF_QTY>55</TSF_QTY>
      <FROM_LOC>1000001001</FROM_LOC>
      <FROM_LOC_TYPE>S</FROM_LOC_TYPE>
      <TO_LOC>2222222225</TO_LOC>
      <TO_LOC_TYPE>W</TO_LOC_TYPE>
      <TSF_TYPE>CO</TSF_TYPE>
      <APPROVAL_DATE>2001-03-09T00:00:00.000+00:00</APPROVAL_DATE>
      <DELIVERY_DATE>2001-03-31T00:00:00.000+01:00</DELIVERY_DATE>
      <DEFAULT_ORDER_TYPE>AUTOMATIC</DEFAULT_ORDER_TYPE>
     </TransfersDBOutput>
   </TransfersDBOutputCollection>
  </part>
 </getTransfer_TransfersDB_OutputVariable>
</messages>
```

A.2 Componente `tsfInfoMAPTsfDesc`

```
<TsfDesc>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="TsfDesc">
 <TsfDesc>
   <tsf_no>3000000201</tsf_no>
   <from_loc_type>S</from_loc_type>
   <from_loc>1000000019</from_loc>
   <to_loc_type>S</to_loc_type>
   <to_loc>1000000007</to_loc>
   <tsf_type>RAC</tsf_type>
   <pick_not_before_date>
      <date>2001-03-09T00:00:00.000+00:00</date>
   </pick_not_before_date>
   <pick_not_after_date>
      <date>2001-03-09T00:00:00.000+00:00</date>
   </pick_not_after_date>
   <default_order_type>AUTOMATIC</default_order_type>
   <break_by_distro>N</break_by_distro>
   <delivery_date>
      <date/>
   </delivery_date>
   <TsfDtl>
      <item>100000171</item>
      <tsf_qty>34</tsf_qty>
   </TsfDtl>
   <TsfDtl>
      <item>100000500</item>
      <tsf_qty>45</tsf_qty>
   </TsfDtl>
```

```
   <TsfDtl>
      <item>100000833</item>
      <tsf_qty>56</tsf_qty>
   </TsfDtl>
   <approval_date>
      <date>2001-03-09T00:00:00.000+00:00</date>
   </approval_date>
 </TsfDesc>
</part>
</TsfDesc>
```

A.3 Componente `msgEnqueue` de Transferências

```
<msgEnqueue_Enqueue_InputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   name="TSF_MSG_TYPE">
   <TSF_MSG_TYPE xmlns="http://xmlns.oracle.com/xdb/AQADM">
   <O_STATUS_CODE xmlns=""/>
   <O_ERROR_MSG xmlns=""/>
   <O_MESSAGE_TYPE xmlns=""/>
   <O_MESSAGE xmlns="">
    <?xml version = '1.0' encoding = 'UTF-8'?>
<TsfDesc>
<tsf_no>3000000200</tsf_no>
<from_loc_type>S</from_loc_type>
<from_loc>1000001001</from_loc>
<to_loc_type>W</to_loc_type>
<to_loc>2222222225</to_loc>
<tsf_type>CO</tsf_type>
<pick_not_before_date>
<date>2001-03-09T00:00:00.000+00:00</date>
</pick_not_before_date>
<pick_not_after_date>
<date>2001-03-09T00:00:00.000+00:00</date>
</pick_not_after_date>
<default_order_type>AUTOMATIC</default_order_type>
<break_by_distro>N</break_by_distro>
<delivery_date>
<date>2001-03-31T00:00:00.000+01:00</date>
</delivery_date>
<TsfDtl>
<item>100000307</item>
<tsf_qty>30</tsf_qty>
</TsfDtl>
<TsfDtl>
<item>100000631</item>
<tsf_qty>40</tsf_qty>
</TsfDtl>
<TsfDtl>
<item>100000964</item>
<tsf_qty>55</tsf_qty>
</TsfDtl>
<approval_date>
<date>2001-03-09T00:00:00.000+00:00</date>
</approval_date>
</TsfDesc>
```

```
 </O_MESSAGE>
   <O_BUS_OBJ_ID xmlns=""/>
   <O_ROUTING_INFO xmlns=""/>
</TSF_MSG_TYPE>
</part></msgEnqueue_Enqueue_InputVariable>
```

A.4 Componente `tsfMAPSODesc`

```
<SODesc>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="SODesc">
 <SODesc>
   <distro_nbr>3000000201</distro_nbr>
   <pick_not_before_date>
      <date>2001-03-09T00:00:00.000+00:00</date>
   </pick_not_before_date>
   <pick_not_after_date>
      <date>2001-03-09T00:00:00.000+00:00</date>
   </pick_not_after_date>
   <order_type>AUTOMATIC</order_type>
   <delivery_date>
      <date/>
   </delivery_date>
   <SODtlDesc>
      <item_id>100000171</item_id>
      <requested_unit_qty>34</requested_unit_qty>
   </SODtlDesc>
   <SODtlDesc>
      <item_id>100000500</item_id>
      <requested_unit_qty>45</requested_unit_qty>
   </SODtlDesc>
   <SODtlDesc>
      <item_id>100000833</item_id>
      <requested_unit_qty>56</requested_unit_qty>
   </SODtlDesc>
   <approval_date>
      <date>2001-03-09T00:00:00.000+00:00</date>
   </approval_date>
 </SODesc>
</part>
</SODesc>
```

A.5 Componente `msgEnqueue` de *Stock Orders*

```
<enqueueSO_Enqueue_InputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    name="SO_MSG_TYPE">
    <SO_MSG_TYPE xmlns="http://xmlns.oracle.com/xdb/AQADM">
   <O_STATUS_CODE xmlns=""/>
   <O_ERROR_MSG xmlns=""/>
   <O_MESSAGE_TYPE xmlns=""/>
   <O_MESSAGE xmlns="">
      <?xml version = '1.0' encoding = 'UTF-8'?>
<SODesc xmlns:ns1="http://www.example.org"
                     xmlns="http://www.example.org">
```

```
<ns1:distro_nbr>3000000201</ns1:distro_nbr>
<ns1:order_type>AUTOMATIC</ns1:order_type>
<ns1:pick_not_before_date>
<date xmlns="">2001-03-09T00:00:00.000+00:00</date>
</ns1:pick_not_before_date>
<ns1:pick_not_after_date>
<date xmlns="">2001-03-09T00:00:00.000+00:00</date>
</ns1:pick_not_after_date>
<ns1:SODtlDesc>
<ns1:item_id>100000171</ns1:item_id>
<ns1:requested_unit_qty>34</ns1:requested_unit_qty>
<ns1:item_status>100000171</ns1:item_status>
</ns1:SODtlDesc>
<ns1:SODtlDesc>
<ns1:item_id>100000500</ns1:item_id>
<ns1:requested_unit_qty>45</ns1:requested_unit_qty>
<ns1:item_status>100000500</ns1:item_status>
</ns1:SODtlDesc>
<ns1:SODtlDesc>
<ns1:item_id>100000833</ns1:item_id>
<ns1:requested_unit_qty>56</ns1:requested_unit_qty>
<ns1:item_status>100000833</ns1:item_status>
</ns1:SODtlDesc>
<ns1:approval_date>
<date xmlns="">2001-03-09T00:00:00.000+00:00</date>
</ns1:approval_date>
</SODesc>
  </O_MESSAGE>
   <O_BUS_OBJ_ID xmlns=""/>
   <O_ROUTING_INFO xmlns=""/>
</SO_MSG_TYPE>
</part></enqueueSO_Enqueue_InputVariable>
```

Anexo B

XML *Schemas*

B.1 TsfDesc.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.example.org"
            targetNamespace="http://www.example.org"
            elementFormDefault="qualified">
  <xs:include schemaLocation="RIBDate.xsd"/>
  <xs:element name="TsfDtlTckt">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="comp_item" type="varchar225"/>
        <xs:element name="comp_price" type="number20" minOccurs="0"/>
        <xs:element name="comp_selling_uom" type="varchar24" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="TsfDtl">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="item" type="varchar225"/>
        <xs:element name="tsf_qty" type="number12"/>
        <xs:element name="price" type="number20" minOccurs="0"/>
        <xs:element name="selling_uom" type="varchar24" minOccurs="0"/>
        <xs:element name="priority" type="number4" minOccurs="0"/>
        <xs:element name="expedite_flag" type="varchar21" minOccurs="0"/>
        <xs:element name="store_ord_mult" type="varchar21" minOccurs="0"/>
        <xs:element name="tsf_po_link_no" type="number10" minOccurs="0"/>
        <xs:element name="ticket_type_id" type="varchar24" minOccurs="0"/>
        <xs:element ref="TsfDtlTckt" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="inv_status" type="number2" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="TsfDesc">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="tsf_no" type="number10"/>
        <xs:element name="doc_type" type="varchar21"/>
        <xs:element name="physical_from_loc" type="number10"/>
        <xs:element name="from_loc_type" type="varchar21"/>
        <xs:element name="from_loc" type="varchar210"/>
        <xs:element name="physical_to_loc" type="number10"/>
```

```
      <xs:element name="to_loc_type" type="varchar21"/>
      <xs:element name="to_loc" type="varchar210"/>
      <xs:element name="tsf_type" type="varchar26"/>
      <xs:element name="pick_not_before_date" type="RIBDate"/>
      <xs:element name="pick_not_after_date" type="RIBDate"/>
      <xs:element name="default_order_type" type="varchar29"/>
      <xs:element name="priority" type="number4" minOccurs="0"/>
      <xs:element name="break_by_distro" type="varchar21"/>
      <xs:element name="delivery_date" type="RIBDate" minOccurs="0"/>
      <xs:element name="cust_name" type="varchar240" minOccurs="0"/>
      <xs:element name="deliver_add1" type="varchar2240" minOccurs="0"/>
      <xs:element name="deliver_add2" type="varchar2240" minOccurs="0"/>
      <xs:element name="deliver_city" type="varchar2120" minOccurs="0"/>
      <xs:element name="deliver_state" type="varchar23" minOccurs="0"/>
      <xs:element name="deliver_post" type="varchar230" minOccurs="0"/>
      <xs:element name="deliver_country_id" type="varchar23" minOccurs="0"/>
      <xs:element name="message" type="varchar22000" minOccurs="0"/>
      <xs:element ref="TsfDtl" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="tsf_parent_no" type="number10" minOccurs="0"/>
      <xs:element name="exp_dc_date" type="RIBDate" minOccurs="0"/>
      <xs:element name="approval_id" type="varchar230" minOccurs="0"/>
      <xs:element name="approval_date" type="RIBDate" minOccurs="0"/>
      <xs:element name="from_loc_tsf_entity" type="number10" minOccurs="0"/>
      <xs:element name="to_loc_tsf_entity" type="number10" minOccurs="0"/>
      <xs:element name="inv_type" type="varchar26" minOccurs="0"/>
      <xs:element name="tsf_status" type="varchar21" minOccurs="0"/>
      <xs:element name="not_after_date" type="RIBDate" minOccurs="0"/>
      <xs:element name="context_type" type="varchar26" minOccurs="0"/>
      <xs:element name="context_value" type="varchar225" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:simpleType name="number12">
  <xs:restriction base="xs:decimal">
    <xs:totalDigits value="12"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="varchar225">
  <xs:restriction base="xs:string">
    <xs:maxLength value="25"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="varchar210">
  <xs:restriction base="xs:string">
    <xs:maxLength value="10"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="varchar2240">
  <xs:restriction base="xs:string">
    <xs:maxLength value="240"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="number4">
  <xs:restriction base="xs:integer">
    <xs:totalDigits value="4"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="varchar230">
  <xs:restriction base="xs:string">
    <xs:maxLength value="30"/>
```

```
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar23">
    <xs:restriction base="xs:string">
      <xs:maxLength value="3"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar2120">
    <xs:restriction base="xs:string">
      <xs:maxLength value="120"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar26">
    <xs:restriction base="xs:string">
      <xs:maxLength value="6"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="number2">
    <xs:restriction base="xs:integer">
      <xs:totalDigits value="2"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar24">
    <xs:restriction base="xs:string">
      <xs:maxLength value="4"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar22000">
    <xs:restriction base="xs:string">
      <xs:maxLength value="2000"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar240">
    <xs:restriction base="xs:string">
      <xs:maxLength value="40"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="number20">
    <xs:restriction base="xs:decimal">
      <xs:totalDigits value="20"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="number10">
    <xs:restriction base="xs:long">
      <xs:totalDigits value="10"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar21">
    <xs:restriction base="xs:string">
      <xs:maxLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar29">
    <xs:restriction base="xs:string">
      <xs:maxLength value="9"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

B.2 SODesc.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.example.org"
           targetNamespace="http://www.example.org"
           elementFormDefault="qualified">
  <xs:include schemaLocation="RIBDate.xsd"/>
  <xs:element name="SODesc">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="distro_nbr" type="varchar210"/>
        <xs:element name="document_type" type="varchar21"/>
        <xs:element name="dc_dest_id" type="varchar210"/>
        <xs:element name="order_type" type="varchar29"/>
        <xs:element name="pick_not_before_date" type="RIBDate"/>
        <xs:element name="pick_not_after_date" type="RIBDate"/>
        <xs:element name="priority" type="number4"/>
        <xs:element name="event_code" type="varchar26" minOccurs="0"/>
        <xs:element name="event_description" type="varchar21000" minOccurs="0"/>
        <xs:element name="carrier_code" type="varchar24" minOccurs="0"/>
        <xs:element name="carrier_service_code" type="varchar26" minOccurs="0"/>
        <xs:element name="route" type="varchar210" minOccurs="0"/>
        <xs:element name="ship_address_desc" type="varchar230" minOccurs="0"/>
        <xs:element name="ship_address1" type="varchar2240" minOccurs="0"/>
        <xs:element name="ship_address2" type="varchar2240" minOccurs="0"/>
        <xs:element name="ship_address3" type="varchar2240" minOccurs="0"/>
        <xs:element name="ship_address4" type="varchar2240" minOccurs="0"/>
        <xs:element name="ship_address5" type="varchar2240" minOccurs="0"/>
        <xs:element name="ship_city" type="varchar2120" minOccurs="0"/>
        <xs:element name="ship_state" type="varchar23" minOccurs="0"/>
        <xs:element name="ship_zip" type="varchar230" minOccurs="0"/>
        <xs:element name="ship_country_id" type="varchar23" minOccurs="0"/>
        <xs:element name="bill_address_desc" type="varchar230" minOccurs="0"/>
        <xs:element name="bill_address1" type="varchar2240" minOccurs="0"/>
        <xs:element name="bill_address2" type="varchar2240" minOccurs="0"/>
        <xs:element name="bill_address3" type="varchar2240" minOccurs="0"/>
        <xs:element name="bill_address4" type="varchar2240" minOccurs="0"/>
        <xs:element name="bill_address5" type="varchar2240" minOccurs="0"/>
        <xs:element name="bill_city" type="varchar2120" minOccurs="0"/>
        <xs:element name="bill_state" type="varchar23" minOccurs="0"/>
        <xs:element name="bill_zip" type="varchar230" minOccurs="0"/>
        <xs:element name="bill_country_id" type="varchar23" minOccurs="0"/>
        <xs:element name="break_by_distro" type="varchar21" minOccurs="0"/>
        <xs:element name="rollback_allocation" type="varchar21" minOccurs="0"/>
        <xs:element name="cartonization" type="varchar21" minOccurs="0"/>
        <xs:element name="consumer_direct" type="varchar21" minOccurs="0"/>
        <xs:element name="message" type="varchar2300" minOccurs="0"/>
        <xs:element name="chute_type" type="varchar26" minOccurs="0"/>
        <xs:element name="po_nbr" type="varchar210" minOccurs="0"/>
        <xs:element name="parent_customer_order" type="varchar220" minOccurs="0"/>
        <xs:element name="pick_complete" type="varchar21" minOccurs="0"/>
        <xs:element name="order_line_costs" type="varchar21" minOccurs="0"/>
        <xs:element name="container_type" type="varchar26" minOccurs="0"/>
        <xs:element name="amount1" type="number16" minOccurs="0"/>
        <xs:element name="amount2" type="number16" minOccurs="0"/>
        <xs:element name="amount3" type="number16" minOccurs="0"/>
        <xs:element name="amount4" type="number16" minOccurs="0"/>
        <xs:element name="amount5" type="number16" minOccurs="0"/>
```

```
      <xs:element name="amount6" type="number16" minOccurs="0"/>
      <xs:element name="sub_total" type="number12" minOccurs="0"/>
      <xs:element name="gift_cost_total" type="number12" minOccurs="0"/>
      <xs:element name="promotion_total" type="number12" minOccurs="0"/>
      <xs:element name="total" type="number12" minOccurs="0"/>
      <xs:element name="tax_total" type="number12" minOccurs="0"/>
      <xs:element name="shipping_total" type="number12" minOccurs="0"/>
      <xs:element name="order_uda1" type="varchar210" minOccurs="0"/>
      <xs:element name="order_uda2" type="varchar210" minOccurs="0"/>
      <xs:element name="order_uda3" type="varchar210" minOccurs="0"/>
      <xs:element name="order_uda4" type="varchar210" minOccurs="0"/>
      <xs:element name="order_uda5" type="varchar210" minOccurs="0"/>
      <xs:element name="order_uda6" type="varchar210" minOccurs="0"/>
      <xs:element name="order_uda7" type="varchar210" minOccurs="0"/>
      <xs:element name="order_uda8" type="varchar210" minOccurs="0"/>
      <xs:element name="order_uda9" type="varchar210" minOccurs="0"/>
      <xs:element name="order_uda10" type="varchar210" minOccurs="0"/>
      <xs:element name="dl_comment" type="varchar230" minOccurs="0"/>
      <xs:element name="ship_date" type="RIBDate" minOccurs="0"/>
      <xs:element ref="SODtlDesc" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="not_after_date" type="RIBDate" minOccurs="0"/>
      <xs:element name="distro_parent_nbr" type="varchar210" minOccurs="0"/>
      <xs:element name="exp_dc_date" type="RIBDate" minOccurs="0"/>
      <xs:element name="approval_id" type="varchar2120" minOccurs="0"/>
      <xs:element name="approval_date" type="RIBDate" minOccurs="0"/>
      <xs:element name="from_loc_tsf_entity" type="number10" minOccurs="0"/>
      <xs:element name="to_loc_tsf_entity" type="number10" minOccurs="0"/>
      <xs:element name="inv_type" type="varchar26" minOccurs="0"/>
      <xs:element name="so_status" type="varchar21" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="SODtlTcktDesc">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="distro_nbr" type="varchar210"/>
      <xs:element name="document_type" type="varchar21" minOccurs="0"/>
      <xs:element name="item_id" type="varchar225" minOccurs="0"/>
      <xs:element name="dest_id" type="varchar210" minOccurs="0"/>
      <xs:element name="component_item_id" type="varchar225"/>
      <xs:element name="retail_price" type="number20"/>
      <xs:element name="selling_uom" type="varchar24"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="SODtlDesc">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="dest_id" type="varchar210"/>
      <xs:element name="item_id" type="varchar225"/>
      <xs:element name="requested_unit_qty" type="number12"/>
      <xs:element name="retail_price" type="number20"/>
      <xs:element name="selling_uom" type="varchar24" minOccurs="0"/>
      <xs:element name="ticket_type" type="varchar24" minOccurs="0"/>
      <xs:element name="priority" type="number4" minOccurs="0"/>
      <xs:element name="store_ord_mult" type="varchar21" minOccurs="0"/>
      <xs:element name="expedite_flag" type="varchar21" minOccurs="0"/>
      <xs:element name="order_line_nbr" type="number3" minOccurs="0"/>
      <xs:element name="amount1" type="number16" minOccurs="0"/>
      <xs:element name="amount2" type="number16" minOccurs="0"/>
```

```
      <xs:element name="amount3" type="number16" minOccurs="0"/>
      <xs:element name="amount4" type="number16" minOccurs="0"/>
      <xs:element name="amount5" type="number16" minOccurs="0"/>
      <xs:element name="amount6" type="number16" minOccurs="0"/>
      <xs:element name="tax_percent" type="number12" minOccurs="0"/>
      <xs:element name="item_status" type="varchar21" minOccurs="0"/>
      <xs:element name="component_ticketing_distro_nbr" type="varchar210" minOccurs="0"/>
      <xs:element name="in_store_date" type="RIBDate" minOccurs="0"/>
      <xs:element name="total" type="number12" minOccurs="0"/>
      <xs:element name="gift_cost" type="number12" minOccurs="0"/>
      <xs:element name="shipping" type="number12" minOccurs="0"/>
      <xs:element name="promotion" type="number10" minOccurs="0"/>
      <xs:element name="sub_total" type="number12" minOccurs="0"/>
      <xs:element name="tax" type="number12" minOccurs="0"/>
      <xs:element name="order_detail_uda1" type="varchar210" minOccurs="0"/>
      <xs:element name="order_detail_uda2" type="varchar210" minOccurs="0"/>
      <xs:element name="order_detail_uda3" type="varchar210" minOccurs="0"/>
      <xs:element name="order_detail_uda4" type="varchar210" minOccurs="0"/>
      <xs:element name="order_detail_uda5" type="varchar210" minOccurs="0"/>
      <xs:element ref="SODtlTcktDesc" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:simpleType name="number3">
  <xs:restriction base="xs:integer">
    <xs:totalDigits value="3"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="varchar225">
  <xs:restriction base="xs:string">
    <xs:maxLength value="25"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="number12">
  <xs:restriction base="xs:decimal">
    <xs:totalDigits value="12"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="varchar21000">
  <xs:restriction base="xs:string">
    <xs:maxLength value="1000"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="number16">
  <xs:restriction base="xs:decimal">
    <xs:totalDigits value="16"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="varchar210">
  <xs:restriction base="xs:string">
    <xs:maxLength value="10"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="varchar2240">
  <xs:restriction base="xs:string">
    <xs:maxLength value="240"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="varchar220">
  <xs:restriction base="xs:string">
```

```
      <xs:maxLength value="20"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="number4">
    <xs:restriction base="xs:integer">
      <xs:totalDigits value="4"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar230">
    <xs:restriction base="xs:string">
      <xs:maxLength value="30"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar23">
    <xs:restriction base="xs:string">
      <xs:maxLength value="3"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar2120">
    <xs:restriction base="xs:string">
      <xs:maxLength value="120"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar26">
    <xs:restriction base="xs:string">
      <xs:maxLength value="6"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar24">
    <xs:restriction base="xs:string">
      <xs:maxLength value="4"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar2300">
    <xs:restriction base="xs:string">
      <xs:maxLength value="300"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="number20">
    <xs:restriction base="xs:decimal">
      <xs:totalDigits value="20"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="number10">
    <xs:restriction base="xs:long">
      <xs:totalDigits value="10"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar21">
    <xs:restriction base="xs:string">
      <xs:maxLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="varchar29">
    <xs:restriction base="xs:string">
      <xs:maxLength value="9"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

B.3 RIBDate.xsd

```
<xs:schema xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
jaxb:extensionBindingPrefixes="xjc"
jaxb:version="1.0">

<xs:complexType name="RIBDate">
<xs:sequence>
<xs:choice>
<xs:group ref="full-date"/>
<xs:group ref="date-string"/>
<xs:group ref="date-obj"/>
</xs:choice>
</xs:sequence>
</xs:complexType>
<xs:group name="full-date">
<xs:sequence>
<xs:element name="year" type="FourDigitInt"/>
<xs:element name="month" type="TwoDigitInt"/>
<xs:element name="day" type="TwoDigitInt"/>
<xs:sequence minOccurs="0">
<xs:element name="hour" type="TwoDigitInt"/>
<xs:element name="minute" type="TwoDigitInt"/>
<xs:element name="second" type="TwoDigitInt"/>
</xs:sequence>
</xs:sequence>
</xs:group>
<xs:group name="date-string">
<xs:sequence>
<xs:element name="date-value" type="xs:string"/>
<xs:element name="date-format" type="xs:string" minOccurs="0"/>
</xs:sequence>
</xs:group>
<xs:group name="date-obj">
<xs:sequence>
<xs:element name="date" type="xs:dateTime"/>
</xs:sequence>
</xs:group>
<xs:simpleType name="TwoDigitInt">
<xs:restriction base="xs:int">
<xs:totalDigits value="2"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="FourDigitInt">
<xs:restriction base="xs:int">
<xs:totalDigits value="4"/>
</xs:restriction>
</xs:simpleType>
</xs:schema>
```