Faculdade de Engenharia da Universidade do Porto

Mestrado Integrado em Engenharia Informática e Computação

Melhoria na lealdade de um retalhista perante os seus clientes

Relatório do Estágio Curricular do MIEIC
realizado na ENABLER/WIPRO
2007/2008

Hugo João Gonçalves e Ramos da Silva Lopes

Orientador na FEUP: Dr. João Mendes Moreira
Orientador na Enabler/WIPRO: Eng. Ricardo Nogueira

Julho de 2008

A confidencialidade deste documento deverá ter a duração de 2 anos, de forma a salvaguardar os interesses da empresa ENABLER/WIPRO.

Resumo

O presente documento tem como objectivo descrever o trabalho realizado no estágio intitulado por *Melhoria na lealdade de um retalhista perante os seus clientes*, realizado na empresa *Enabler Wipro*, de Fevereiro a Julho de 2008.

O projecto está inserido no centro de competência de *Enterprise Application Integration* (EAI), responsável pela integração de diferentes sistemas de um retalhista, em especial entre o sistema implementado pela *Enabler Wipro* e os sistemas legados existentes, ou a implementar nos clientes.

Com a realização deste estágio era pretendido, não só o desenvolvimento de um projecto, compreendendo as fases que o envolvem, mas também a possibilidade iniciar um contacto mais próximo com o mundo empresarial, especialmente numa área tão complexa como é o retalho. A oportunidade dada de trabalhar junto de pessoas competentes na área do retalho foi muito importante, reduzindo o período de aprendizagem e fortalecendo os conhecimentos.

Este estágio foi realizado, com a ajuda de diversas ferramentas inerentes à área de integração como é o caso do SeeBeyond Egate e do PL/SQL Developer.

O objectivo principal deste relatório será descrever todos os processos envolvidos no problema da integração num sistema de retalho. Este será constituído por vários capítulos, cujo objectivo será descrever nomeadamente os sistemas a serem implementados, os tipos de integração possíveis, a decisão do melhor a ser utilizado no presente problema, descrição do trabalho realizado, e finalmente, obter alguma ilacções acerca do trabalho efectuado.

Agradecimentos

Este estágio não seria possível sem o contributo fundamental de algumas pessoas, que me apoiaram em todos os momentos, bons e maus.

Aos meus pais, por todo o esforço e dedicação ao longo destes cinco anos, obrigado por estarem sempre comigo.

Um obrigado também aos meus orientadores, o Ricardo que me ensinou muito e que aturou as minhas dúvidas constantes e ao Dr. João Mendes Moreira por todo o apoio dado a este trabalho.

Aos meus colegas de estágio, que foram fundamentais na integração na empresa e pelos momentos de boa disposição diários na Enabler.

Por fim, os meus agradecimentos a todos os colegas da Enabler que acompanharam e ajudaram, especialmente à Carla Almeida, que acompanhou a integração do início ao fim e foi sempre um elemento de apoio em qualquer situação de dúvida.

Índice de Conteúdos

Lista de Figuras

Lista de Tabelas

1 Introdução

O presente documento tem como objectivo descrever as actividades desenvolvidas no contexto do estágio curricular do Mestrado Integrado em Engenharia Informática e Computação, realizado na empresa *Enabler Wipro*, entre Fevereiro e Julho de 2008.

1.1 Apresentação da Instituição de Estágio

A *Enabler Wipro* é uma empresa de referência no desenho, implementação e suporte de sistemas de informação para retalho. A sua grande experiência no sector do retalho é um dos seus maiores pontos-chave, que se reflecte na boa carteira de clientes de que dispõe, como será referido mais à frente nesta secção.

A *Enabler* propõe-se oferecer aos seus clientes um conhecimento forte de retalho, uma atitude orientada aos resultados, de elevado nível de conhecimento por parte dos seus colaboradores, investigação e desenvolvimento e acima de tudo, a procura da excelência em cada projecto.

A empresa tem um cariz cada vez mais internacional, focando-se em alcançar o estatuto de líder europeu, especialista na implementação e suporte de sistemas de retalho.

História

A *Enabler* foi criada em 1997, fruto da emancipação planeada do departamento de Sistemas de Informação (IS/IT) da *Modelo Continente Hipermercados* (MCH), pertencente ao G*rupo Sonae* na área de Retalho.

A *Sonae* é o maior grupo privado português, com interesses em diversas áreas de negócio: retalho alimentar e não alimentar, desenvolvimento e gestão de centros comerciais, telecomunicações fixas e móveis, media, Internet e novas tecnologias, bem como turismo, construção imobiliária, logística, seguros, entre outras. O grupo opera em 8 países, com um volume de negócios consolidado que ronda os 4,384 milhões de euros.

A *MCH* é a actual líder do mercado de retalho em Portugal. Surge em 1985, através de uma *junção* entre a *Sonae* e a *Promodes*, abrindo nessa altura o primeiro hipermercado em Portugal, e dando origem a uma revolução no sector de retalho em Portugal, até aí dominado por pequenos operadores tradicionais. A empresa opera hoje em três diferentes cadeias de base alimentar – *Continente* (hipermercados), *Modelo* (hipermercados) e *Modelo Bonjour* (supermercados) – e em vários formatos de retalho especializado – *Sport Zone* (equipamento e vestuário desportivo), *Worten* (electrodomésticos e electrónica de consumo), *Vobis* (equipamento informático), *Modalfa* (vestuário), *MaxMat* (construção, bricolage e jardim), *Zippy* (vestuário de bebé e criança), *Star* (agências de viagens), *Worten Mobile* (telecomunicações móveis) e *Área Saúde* (parafarmácias).

Assente no conhecimento adquirido durante anos nos Sistemas de Informação da *MCH*, a *Enabler* foi criada como uma empresa independente da *MCH*, fornecedora de serviços na área de Retalho.

Liderada pelo Engenheiro António Murta, a empresa continuou a trabalhar para a *MCH*, conseguindo apresentar um crescimento sustentado.

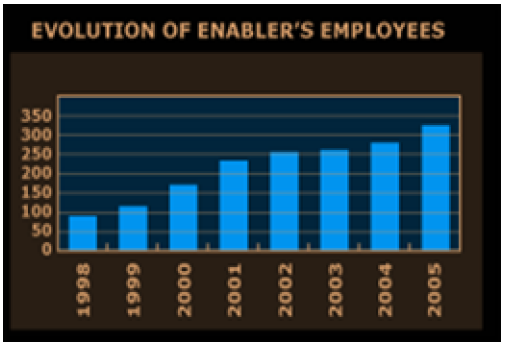
A partir de 1999, surgem os primeiros projectos fora de Portugal. O primeiro foi desenvolvido para o retalhista italiano *Despar*, seguindo-se rapidamente novos projectos para outros grandes retalhistas europeus, como a *Debenhams* (UK) e a *Ahold* (Espanha).

Em 2000, o volume de vendas era já de 15 milhões de euros e pela mesma altura a *Enabler* abre o seu primeiro escritório fora de Portugal, criando a *Enabler UK*.

Desde a sua fundação, a *Enabler* conseguiu um crescimento significativo e sustentado, focando-se nos retalhistas líderes dos principais mercados mundiais. Desta forma conseguiu um vasto leque de clientes em todo o mundo, levando a um aumento das vendas e dos lucros.

Assim, depois de uma base sólida em Portugal, a *Enabler* expandiu-se para um negócio completamente global. Possui neste momento escritórios nos principais países europeus, nomeadamente: Portugal, Reino Unido, Alemanha, Itália, França e Espanha e alargou as suas operações para todo o Mundo, possuindo clientes na América do Norte, América Latina (possui um escritório no Brasil, responsável por esta área), Médio Oriente e Ásia Pacifico.

Na figura 1 é possível observar o ritmo de crescimento da *Enabler* ao nível de número de empregados e de volume de vendas. A *Enabler* conta com mais de 300 trabalhadores, número que tem vindo a aumentar de ano para ano. Com este crescimento, a Enabler tem também conseguido uma facturação elevada, que ultrapassa já os 30 milhões de Euros anuais.



Evolução do número empregados

Evolução do volume de vendas

Figura 1 – Crescimento da *Enabler*

Em 2006, a *Enabler* encontrava-se num período de estagnação de crescimento. A empresa tornou-se grande demais e não tinha base de sustentação para continuar o ritmo de

crescimento alcançado nos anos anteriores. Nesse momento, dá-se a venda da empresa à companhia indiana *Wipro Technologies*.

A *Wipro Technologies* é a segunda maior empresa de Tecnologias de Informação (TI) da Índia, com uma facturação anual superior a 3000 milhões de Euros. Criada em 1980, é uma extensão da empresa *Wipro Limited*.

A *Western India Vegetable Products Limited* (*Wipro Limited*) foi fundada em 1945. Começou por desenvolver produtos à base de óleo de girassol (imagem que ainda hoje está representada no símbolo da empresa). Por volta da década de 1970, e já sob a liderança de Azim Premji, que assumiu o controlo da empresa com apenas 21 anos, começa a direccionar os seus objectivos para a área tecnológica.

Em 1975 cria o primeiro computador de origem indiana, e em 1980 surge a *Wipro Technologies*, para prestar serviços na área das Tecnologias de Informação.

Actualmente, com mais de 80 mil trabalhadores, a *Wipro Technologies* tornou-se numa das maiores empresas de prestação de serviços na área das TI do mundo, oferecendo serviços como comércio electrónico (*e-commerce)*, reengenharia de processos de negócios, migração de sistemas, manutenção de sistemas legados e integração de sistemas.

Com a compra da *Enabler*, a *Wipro* reforça então o sector de retalho, e entra no mercado europeu e latino-americano. Por sua vez, a divisão da *Wipro* correspondente à aquisição da *Enabler* aproveita o poder e os conhecimentos do seu novo grupo para conseguir entrar em novos mercados e conquistar novos clientes.

Outra das mais valias aproveitadas pela *Enabler* é o conhecimento na área de qualidade da *Wipro*, que tem vindo a aumentar constantemente, como é possível observar na figura 2. Em 1995, a *Wipro* alcança o certificado de qualidade ISO 9001. Em 1997, recebe o certificado *Capability Maturity Model* (*CMM*) de nível 3, e, em 1998, o certificado de nível 5. Este certificado representa o nível máximo de maturidade da qualidade possível a uma empresa. A *Wipro Technologies* foi a primeira empresa de serviços de software a nível mundial a obter este certificado, tendo posteriormente obtido ainda os níveis máximos nos certificados *People Capability Maturity Model* (*PCMM*) e *Capability Maturity Model Integration* (*CMMi*).

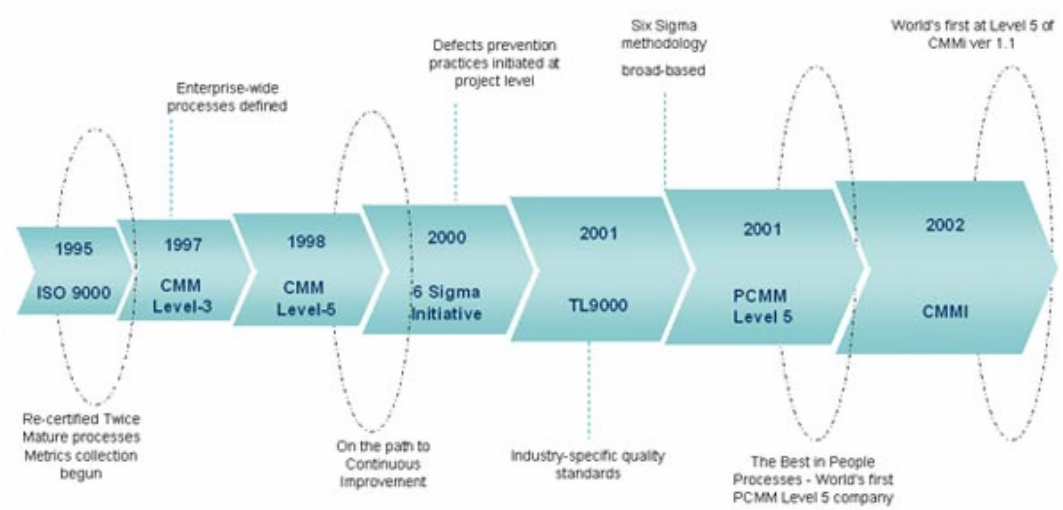


Figura 2 – Evolução dos certificados de qualidade da *Wipro*

Por sua vez a *Enabler* possui o certificado *ISO 9001:2000*, conseguido em 2003, e espera em breve alcançar novos índices de qualidade.

A *Enabler* é agora denominada *Enabler Wipro* (figura 3), tendo alcançado um ano de parceria e envolvimento, ao mesmo tempo que comemora dez anos de existência. O futuro apresenta-se de um novo período de crescimento, assente numa estrutura mais sólida e capaz de crescer e evoluir mais facilmente.

Figura 3 – Actual símbolo da Enabler Wipro

Neste novo processo de expansão, a aposta é na conquista de novos mercados. A empresa conseguiu entrar pela primeira vez no mercado africano, com a *Nakumatt* (Quénia). O próximo objectivo é a consolidação no mercado norte-americano, que representa uma parte significativa do mercado de retalho mundial, e onde é possível encontrar as maiores empresas do sector.

Estrutura da Empresa

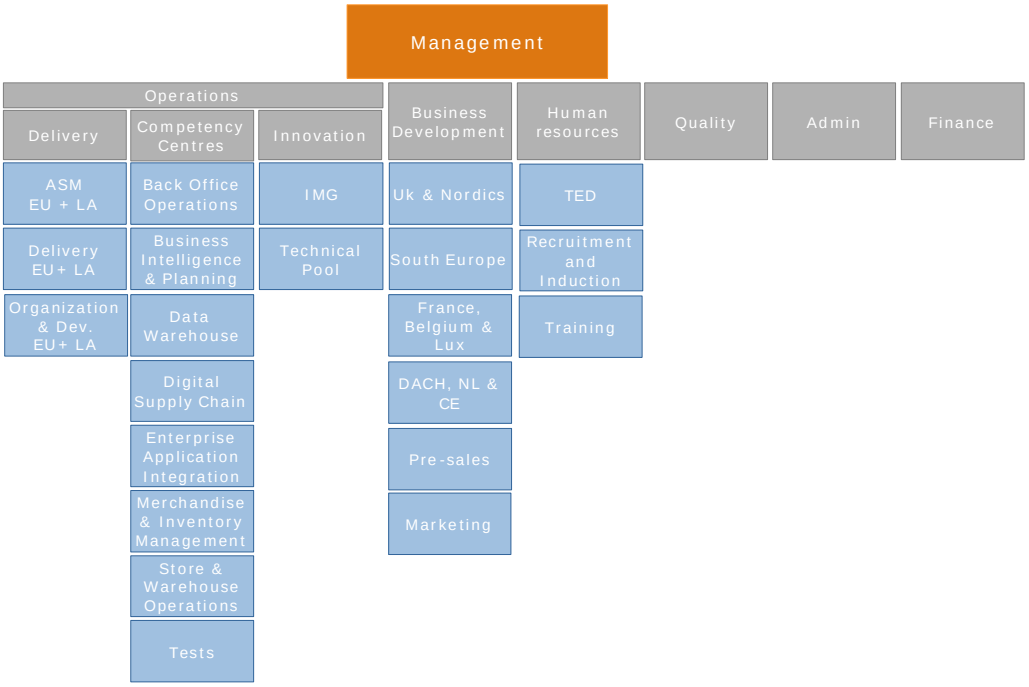


Figura 4 – Organigrama da Enabler

A Enabler encontra-se dividida em áreas de negócio, como é possível observar na figura 4. A nível operacional a Enabler encontra-se dividida em três áreas: Entregas, Centros de Competência e Inovação.

Na área de operações, a Enabler encontra-se organizada em Centros de Competência (*CC*). Cada uma destas estruturas é responsável por gerir a alocação e qualidade dos seus recursos nas valências específicas, e captar, reter e assegurar o desenvolvimento dos seus recursos. Têm também como objectivo apoiar a pré-venda e elaboração de propostas, assegurar a qualidade do desenvolvimento e gerir projectos.

Os *Centros de Competências* existentes são:

- Back Office Operations (BOO)
- Business Intelligence (BI)
- Data Warehousing (DW)
- Digital Supply Chain (DSC)
- **Enterprise Application Integration (EAI)**
- Merchandise Management (M&M)
- Store & Warehouse Management (SWM)

- Tests

Dentro da *Enabler* o estagiário foi integrado no centro de competência *Enterprise Application Integration (EAI)*. Este centro é responsável pela integração dos diferentes sistemas existentes numa empresa para que possam comunicar entre eles de forma fiável e segura.

O projecto teve como objectivo a implementação de uma solução de retalho num retalhista internacional.

1.2 Organização e Temas Abordados no Presente Relatório

Este relatório encontra-se dividido em sete capítulos:

No segundo capítulo será efectuado o enquadramento do projecto, apresentando os principais conceitos que o envolvem.

No terceiro capítulo será efectuada uma revisão tecnológica, onde serão descritas as principais ferramentas e metodologias utilizadas durante o estágio.

No quarto e quinto capítulo será apresentada a solução e o trabalho realizado, com especial destaque para a área de integração dos processos.

No sexto capítulo serão apresentados os testes, como resultados da implementação.

No sétimo e último capítulo serão apresentadas as conclusões do trabalho realizado e apresentadas perspectivas do trabalho futuro.

2 Enquadramento do Projecto

Antes de ser elaborada uma descrição mais completa do trabalho efectuado, é importante descrever o contexto em que o projecto foi elaborado e as grandes características que envolvem um projecto na *Enabler Wipro*. Será descrito o que é o retalho, os processos típicos numa cadeia de abastecimento, a descrição do problema, o cliente, as ferramentas e produtos utilizados no projecto, a maneira com que foi gerido o projecto e, finalmente o plano de trabalho definido inicialmente.

2.1 Retalho / Sistema de Gestão de Retalho

Um retalhista tem como objectivo principal oferecer, num único local, uma vasta gama de produtos a consumidores finais.

Definição de Retalho – Venda directa de produtos aos consumidores

Figura 5 – Cadeia de Abastecimento

A cadeia de abastecimento (figura 5) é um sistema de organizações, pessoas, tecnologias, actividades, informação e recursos envolvidos no movimento de um produto ou serviço do fornecedor para o cliente. As diferentes actividades da cadeia de abastecimento transformam os recursos naturais, matérias-primas e componentes num determinado produto acabado que deverá servir os interesses do cliente.

A cadeia de abastecimento, quando planeada, desenhada e executada eficientemente, constitui a chave para atingir valores elevados de desempenho operacional. Na prática, este desempenho pode ser verificado na satisfação dos clientes e no aumento da produtividade. Torna-se assim importante, uma gestão eficiente dos materiais e informações que fluem dentro de uma empresa, bem como aqueles que são transaccionados com parceiros de negócios.

De um modo geral, a gestão da cadeia de abastecimento de uma determinada empresa é baseada em dois tipos de aplicações:

- As aplicações que permitem a troca de informações entre parceiros;
- As aplicações que realizam o planeamento e optimização da cadeia de abastecimento.

O objectivo deste planeamento é garantir uma correspondência entre a procura e a oferta, contribuindo assim para a diminuição de stocks, tendo em conta as restrições induzidas pela capacidade produtiva e pela logística de produção. As aplicações deste tipo necessitam normalmente de informações que estão localizadas nos sistemas transaccionais das empresas (ERPs). Estas informações advêm das transacções internas da empresa e da troca de informação existente entre a empresa e os seus parceiros. Estas trocas de informação poderão ser realizadas de modo automático, através de uma integração geral entre cliente e fornecedor, ou de modo manual, através do acesso a portais que disponibilizam uma significativa quantidade de informação logística, quer de um lado, quer do outro. Considerando então uma empresa que pretende planear as suas actividades, tendo como objectivo de negócio a redução de stocks ao longo da sua linha da cadeia. Para garantir o planeamento necessário, tendo em conta todos os constrangimentos, a empresa terá de agregar diversas infomações, tais como prazos, sotcks existentes, datas de entregas, entre outras. Torna-se então essencial haver uma ligação entre a empresa e os seus parceiros de negócio, de modo a que estas operações sejam efectuadas com uma maior rentabilidade.

2.2 Processos na Cadeia de Abastecimento

Saber gerir a cadeia de abastecimento é garantir que a produção não pára. É essa a pressão da competitividade, o esforço e responsabilidade de um gestor logístico moderno, ser capaz de "tirar um coelho da cartola" quando os imponderáveis acontecem. Na economia global, em que o fornecimento deixou de ser uma questão de proximidade, só os mais capazes conseguem enfrentar as mudanças constantes, planeando, colaborando e preparando planos alternativos para todos os cenários[1]. De todo o ciclo de vida de um produto de acordo com a cadeia de abastecimento, poder-se-ão distinguir os seguintes passos[2]:

§ criação de ordens de compra;

§ produção e distribuição por parte do fornecedor;

§ operações de entrada no entreposto;

§ operações de saída do entreposto;

[1] *http://www.logisticamoderna.com/home/anteriores.asp?id=40#EmFoco*

[2] Note-se que em todos os passos descritos apenas são envolvidos o ERP e o sistema de gestão de entreposto. O sistema de gestão da loja está fora do âmbito do projecto, apesar de também partilhar informação com o ERP.

- controlo de inventário;
- transferências e alocações;
- retorno do excesso de mercadoria para os fornecedores.

Criação de Ordens de Compra

Esta operação consiste na criação de um determinado tipo de encomenda, de modo a que seja possível notificar o fornecedor da necessidade da compra de artigos ou matérias-primas. Esta ordem poderá ser criada manualmente, ou então automaticamente através da detecção por parte do sistema do alcance do stock mínimo de um determinado artigo. Tipicamente esta ordem de compra é criada num ERP, sendo depois integrada com o sistema de gestão de entreposto. Esta integração é necessária para ser visível no entreposto que é esperada nova mercadoria.

Produção e distribuição do fornecedor

Este processo advém da recepção por parte do fornecedor de uma ordem de compra criada pela retalhista. Esta ordem de compra poderá ser recebida através da integração dos sistemas do fornecedor e do retalhista, ou então poderão também ser utilizados métodos clássicos de comunicação como o caso do *e-mail*, correio, fax, etc.

Operações de entrada no entreposto

Aqui podem-se englobar todos os processos necessários para dar entrada da mercadoria no armazém. Uma vez que o processo de entrada de mercadoria é composto por vários sub-processos, passa-se de seguida a descrevê-los.

Pré-Recepção

Pré-Recepção é o processo que é executado quando um ASN (Notificação de chegada/saída de mercadoria do entreposto – *Advanced Shippment Notice*) está calendarizado no sistema de gestão do entreposto e a mercadoria chega ao armazém. Este processo terminando, indica o ponto em que os bens são removidos do veículo para a área de descargas do armazém. Caso não exista nenhum ASN calendarizado, este terá de ser primeiro criado, para posteriormente ficar datado.

Dentro desta solução mais geral, terá de existir um ASN ou, caso contrário, a carga será rejeitada. Estas notificações podem ser mapeadas a partir do ERP, criadas pelo fornecedor através de transacções EDI (*Electronic Data Interchange*), ou terão de ser criadas manualmente no sistema de gestão de entreposto para permitir que os operadores recebam a carga. Contudo, quando criadas manualmente, estas notificações são criadas com base numa ordem de compra já existente no sistema.

Esta criação pode ser calendarizada numa determinada data de modo que o entreposto se prepare para a recepção, fazendo com que os recursos necessários estejam disponíveis para a recepção da mercadoria. Caso não o seja, é necessário registar a data no dia de chegada da mercadoria.

Recepção

Recepção, como o nome indica, denomina o processo onde o produto é fisicamente recebido na área de recepção do entreposto e inserido no inventário do sistema de gestão do entreposto em questão. Esta recepção suporta um determinado número de processos, incluindo a validação de acordo com o que deveria ser recebido com os detalhes da ordem de compra.

A recepção da mercadoria será envolvida pelos seguintes passos:

- O operador efectua verificações de etiquetas do material, e introduz a respectiva porta de recepção.
- É gerado automaticamente o número identificador do ASN. Se este não estiver alocado a uma porta do entreposto, terá de ser introduzida essa informação manualmente.
- O operador será informado se a mercadoria provém de vendedores locais ouestrangeiros. Este processo poderá ser importante, visto que poderão existir diferentes processos de negócio para cada caso.
- O número do identificador de cada palete (LPN) correspondente à mercadoria é automaticamente gerado pelo sistema. Isto permite criar apenas uma etiquetagem, aquando da recepção de uma palete.
- Terá de ser feito a leitura de cada SKU para o sistema através do código de barras. Se para cada código existir mais que um SKU, o operador é obrigado a escolher o tipo da mercadoria, de entre uma lista, em que cada produto possui esse mesmo código de barras (EAN).
- Se este SKU existir em várias ordens de compra para o ASN que foi recebido, o sistema de gestão de entreposto notifica o operador para que este possa escolher a combinação certa para este SKU. Se este SKU não exitir, o ASN é configurado para não permitir a recepção e a mercadoria é posta de lado para uma posterior verificação de processo.
- O operador é notificado para introduzir a quantidade de items recebidos. Se esta exceder a quantidade existente no pedido de compra, o sistema não irá aceitar a recepção do excesso de mercadoria. No caso de fornecedores locais, a mercadoria em excesso é retornada imediatamente e no caso de fornecedores mais distantes ou estrangeiros, a mercadoria é colocada de lado para posterior processamento.
- Caso o SKU necessite de uma data de expiração, esta terá de ser introduzida. Se esta, aquando da validação existir fora da janela de aceitação o sistema automaticamente rejeita essa recepção.
- Caso todas as validações sejam realizadas com sucesso, o sistema cria o LPN no estado ‘30’ e localiza fisicamente a mercadoria na plataforma de recepção. Neste

momento o ASN e a ordem de compra já estarão com as quantidades recebidas actualizadas.

É neste ponto que caso seja necessário, alguma da mercadoria poderá ficar alocada a uma determinada loja. Caso este LPN seja total ou parcialmente alocado a uma loja, o operador recebe uma mensagem a indicar a respectiva alocação para *cross-docking*[3], aquando da alocação de um LPN inteiro a uma loja. Caso um LPN tenha de ser dividido por lojas, o próximo processo será o das alocações para as lojas.

A verificação final dos ASNs é controlado neste nível do processo de entrada. Uma vez que um ASN é completamente descarregado e recebido, o operador poderá fechar este processo.

Movimento após a recepção

Uma vez que é dito aos operadores para mover LPNs da plataforma de recepção, o operador vai seleccionando manualmente um LPN. No caso de este ser suficientemente grande para uma movimentação e caso haja uma pré-alocação identificada pela ordem de compra ou pelo SKU, serão verificadas as ordens de distribuição pelas lojas. Caso este não esteja destinado a um processo de *flowthrough* [4](i.e., já ter como destino uma determinada loja), o LPN em questão é imediatamente alocado para *putaway* (i.e., é guardada como stock ou reserva no entreposto).

Caso o LPN seja bastante pequeno, deverá ser alocado para uma palete para, desta forma, minimizar os custos. Posteriormente a esta alocação deverá ser feito o mesmo processo descrito em cima (*flowthrough* ou *putaway*).

Putaway

Este conceito refere-se à acção de colocar um LPN em stock/reserva. Existem duas aproximações que são usadas nas operaçõesdo cliente:

- *Putaway* por parte do operador: Neste caso os operadores determinam manualmente a localização desta operação a efectuar sobre um LPN.
- *Putaway* por parte do sistema: Aqui são usadas um determinado número de *flags* no sistema, para que este determine uma localização o mais óptima possível. Obviamente que este método é mais eficaz, comparando-o com o método anterior.

[3] Processo que visa a divisão de uma mercadoria recepcionada, para imediata distribuição por outros entrepostos ou lojas. Isto possibilita uma melhor rentabilização do tempo.

[4] Processo em que a chegada da mercadoria a um determinado entreposto, esta rá já alocada para uma movimentação para uma determinada loja.

Operações de saída do entreposto

Aqui podem-se englobar todos os processos necessários para a saída da mercadoria do armazém. Uma vez que o processo de saída de mercadoria é composto por vários sub-processos, passa-se de seguida a descrevê-los.

Reaprivisionamento

Dentro de um armazém, isto corresponde ao processo em que é necessário repor stock de algumas localizações. Este processo serve para optimizar as operações de picking (descritas em seguida), e surge devido à sua quantidade estar abaixo de um determinado nível pré-definido pelo retalhista.

Consolidação de Mercadorias

Este conceito refere-se a um processo executado sobre o pedido de um determinado grupo de lojas, para que seja consolidada toda a mercadoria pedida por cada uma destas. Estes processos poderão ser agendados a efectuar automaticamente no sistema de gestão do entreposto.

Picking

Processo que visa a recolha de stock de um determinado entreposto, para operações de distribuição cujo destino são lojas/entrepostos, ou de mudança de localizações dentro do mesmo entreposto.

Put to Store

Se um LPN está alocado para *flowthrough*, o seu conteúdo será distribuido por cartons[5]. A este processo designa-se por *Put to Store.*

[5] Um *carton* corresponde a um contentor com uma certa capacidade, usado para armazenar os artigos que vão sair do armazém.

Controlo de Inventário

Para este controlo de inventário, terão de ser efectuadas contagens periódicas para se saber se o que está inserido no sistema estará de acordo com a contagem física. Isto para evitar uma diminuição da qualidade e consistência dos dados.

Transferências e Alocações

Esta transferência é criada no ERP sendo depois integrada com o sistema de gestão de entreposto. O seu objectivo é a mudança de localização de uma determinada mercadoria ou artigo. Estas localizações poderão ser lojas ou entrepostos. No caso de serem lojas, deverá então ser criada uma alocação.

Retorno do excesso de mercadoria para os fornecedores

Este tipo de operação é normalmente efectuada aquando da existência de um contrato entre o fornecedor e o cliente. Normalmente é efectuado um retorno deste tipo no caso de produtos alimentares fora do prazo, na mudança de colecções no que toca ao vestuário, ou então caso algum artiga estaja danificado. Esta operação é tipicamente criada no sistema de gestão de entreposto e depois integrada com o ERP, podendo acontecer também o inverso.

2.3 Sistemas de Informação de Retalho

Os sistemas de informação em retalho, se tratarem informação apenas por si só (sem estarem integrados), implicam a repetição de operações que podem levar ao erro. Por este motivo são mais demorados, ocupando muito do tempo dos recursos humanos. Com a integração destes sistemas, a empresa tem mais tempo disponível para se centrar na criação de valor, já que as operações demoram menos tempo e são efectuadas a mais baixo custo. Assim, os recursos e os capitais são alocados para a gestão dos principais activos da empresa: os clientes.[6]

A nível informático, o retalho é encarado como o transporte, manipulação e armazenamento da informação que circula em toda a cadeia de abastecimento. Essa informação engloba desde a mais simples e básica, como a criação de um artigo novo no sistema, até ao processamento de grandes volumes de informação, envolvendo milhares de artigos e transacções.

Para conseguir tratar este volume de informação, e dele conseguir extrair informação necessária para gerir o sistema, é essencial existirem um conjunto de sistemas de informação capazes de suportar esta estrutura, de processar a informação e trocá-la rapidamente. Estes sistemas são neste momento considerados críticos por parte das empresas para a gestão do seu negócio.

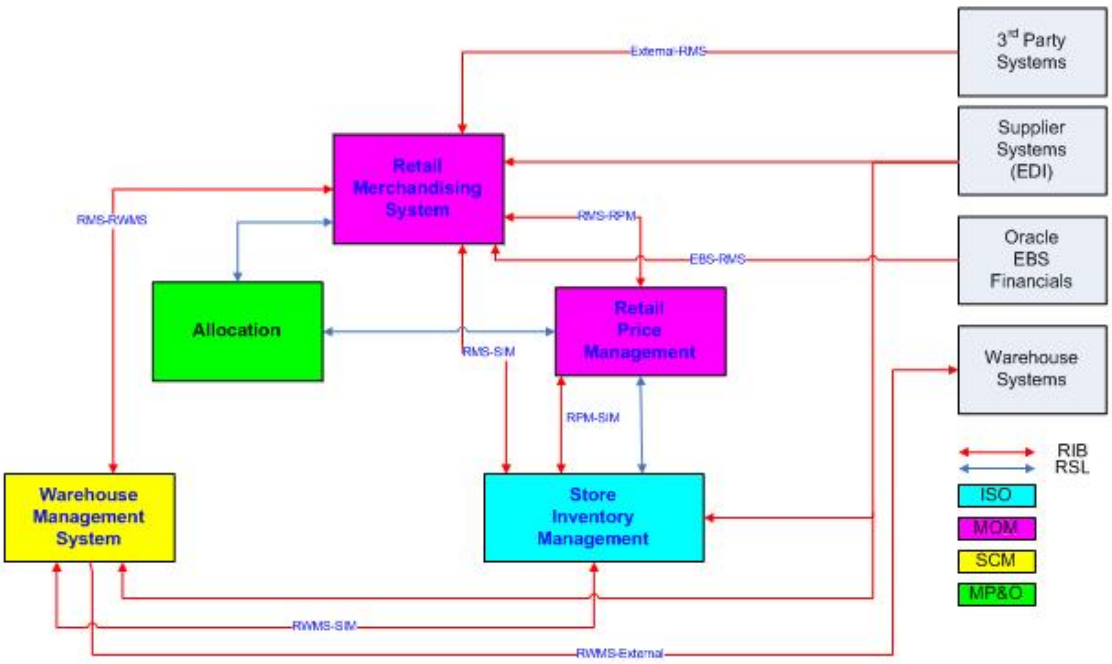


Figura 6 – Arquitectura Geral de um Sistema de Gestão em Retalho

Este tipo de sistemas, interliga vários sub-sistemas, que trabalhando independentemente não são funcionais na àrea do retalho, e não possibilitam a execução dos processos pré-definidos

[6] E-Business: Cadeia de Abastecimento, *http://e-business.blogspot.com/*

pelos seus requisitos. Sendo assim, como é visível na figura 7, estes sistemas são todos interligados, com o objectivo de se constituirem os vários fluxos de informação que circulam num sistema desta natureza.

2.4 Descrição do Problema

O objectivo principal na elaboração deste trabalho é o de aferir a lealdade dos clientes/membros perante uma determinada empresa de retalho.

A primeira fase, e abordagem possível para este problema, passa pela eficiente gestão da cadeia de abastecimento, que poderá ser possibilitada pela implementação e integração de uma plataforma de gestão de entreposto a ser integrada no cliente. Isto possibilitará com que um determinado produto possa estar disponível num determinado lugar, num tempo útil que favorecerá o cliente, e consequentemente o retalhista. Este processo, como será explicado mais em frente, passará pela implementação de ferramentas de gestão que possibilitarão que o retalhista forneça um melhor serviço aos seus clientes, diminuindo também os seus custos (redução de custos - objectivo comum de todos os negócios) logísticos.

Para esta eficiente gestão da cadeia de abastecimento, é necessário existir um sistema que permita a optimização de todo o processo de pedidos de artigos, para que estes cheguem no tempo certo ao local de venda. Caso isto se verifique, poderá ser considerada uma mais valia para o retalhista.

Será então descrito ao longo deste documento o estudo efectuado para a determinação da melhor plataforma a ser implementada conjuntamente com os restantes sistemas já existentes. Todos estes integrados entre si, constituem um grande ganho para o negócio do cliente. O objectivo principal deste documento cairá então principalmente no problema da escolha de um determinado tipo de integração entre o sistema de gestão do entreposto e o ERP que lhe servirá como base. Estes sistemas são respectivamente o *Manhattan Warehouse Management System* (MWMS) e o *Oracle Retail Merchandising System* (ORMS).

2.5 Cliente

The Sultan Center (TSC) é uma grande empresa situada no médio oriente, e está inserida no ramo de distribuição de produtos e de mercadoria pelas mais variadas lojas situadas nessa zona.

Este projecto terá como principal objectivo o alcance por parte desta empresa de um lugar cimeiro, no que toca ao Retalho no Médio Oriente. Por este facto, a empresa referenciada decidiu investir em novos sistemas de informação em todas as áreas em que oferece os seus serviços, com o objectivo de suportar o seu forte crescimento por via da obtenção de economias de escala.

Foi então acordado, que seria implementado o *Oracle Retail suite*, integrado com um sistema de gestão de entreposto.

Ficou então estabelecido a implementação dos seguintes produtos no sistema já existente da empresa:

- *Oracle Retail suite* versão 12;
- *Manhattan Warehouse Management for Open Systems*, versão 2006;

Haverá então a necessidade de integrar todas estas plataformas entre si, de modo a que todos os dados existentes no sistema sejam consistentes. O modo desta integração será mais à frente descrita detalhadamente.

Este projecto consiste no desenvolvimento da interface de integração de dados em todos os fluxos que envolvem o MWMS.

2.6 Ferramentas e produtos utilizados no projecto

A Enabler é especialista num conjunto de aplicações de apoio à integração nos projectos para os seus clientes. O mais utilizado na área de integração é o *Seebeyond eGate Integrator* da *Sun MicroSystems* (figura 7), uma plataforma de integração que oferece funcionalidades de monitorização e gestão. Esta plataforma permite a gestão da actividade e o controlo de todas as componentes que constituem um determinado fluxo de informação a ser integrada em sistemas distintos.

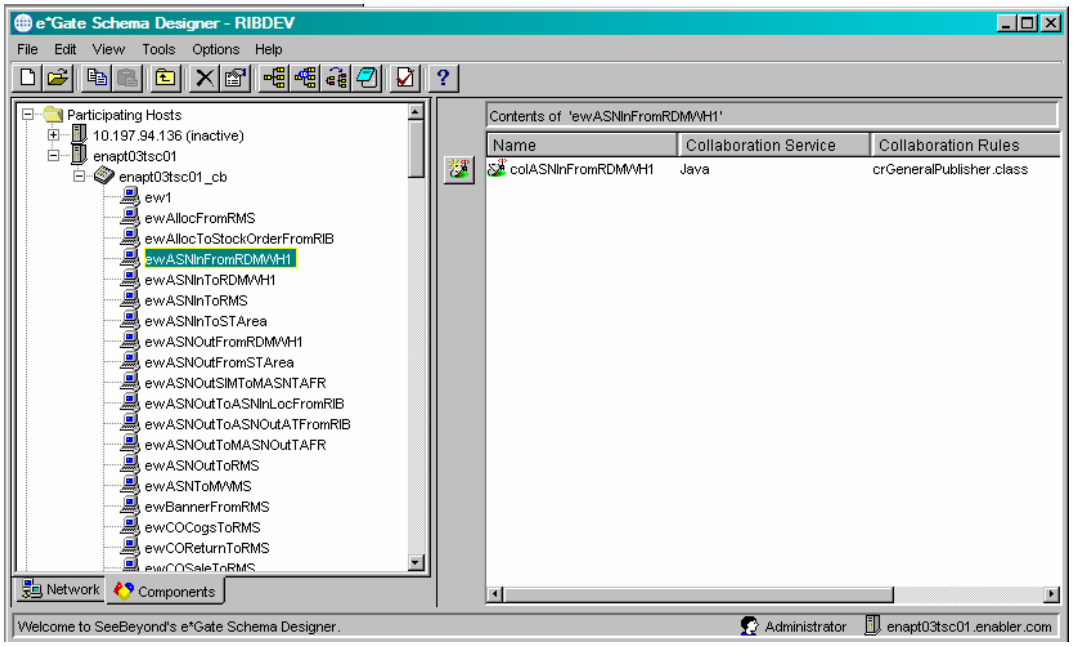


Figura 7 – Seebeyond eGate Integrator

Outra das plataformas mais utilizadas é o *Oracle Retail Integration Bus (ORIB)*, um protocolo de mensagens específico para os sistemas de retalho da *Oracle*, e que permite uma comunicação uniformizada e bem definida entre os produtos da solução *Oracle*. Para além disto, também poderá ser utilizada para integração com outros sistemas Na figura 8 apresentam-se todos os sistemas de integração utilizados em projectos na *Enabler*.

Figura 8 – Sistemas de integração mais utilizados no EAI da Enabler

2.7 Gestão do Projecto

Tipicamente a gestão dos projectos da Enabler divide-os em quatro diferentes fases:

- § Análise e especificação dos requisitos;
- § Definição da arquitectura;
- § Desenho da solução/Implementação;
- § Testes.

De seguida serão explicadas as diferentes fases, tendo como foco o desenvolvimento de projectos de integração.

Análise e definição de requisitos

Esta análise é efectuada juntamente com o cliente sendo criado posteriormente, um documento designado por CRP (*Conference Room Pilot*) que visa a análise dos sistemas e fluxos existentes e a descrição funcional dos fluxos de integração

Pontos Críticos

- Arquitectura actual *versus* arquitectura futura
- Comunicação intra e inter-equipas, equipa de integração proactiva
- Reutilização de fluxos já existentes
- Criação de uma check-list completa com todos os fluxos de integração
- Volumes de dados – obter informação o mais cedo possível
- Entendimento das regras de negócio inerentes a cada fluxo de dados
- Visão exacta das dependências entre fluxos de dados

- Definição de responsabilidades (quem faz o quê em que etapa do fluxo)

- Evitar colocar lógica de negócio complexa nas interfaces

Definição da Arquitectura

Normalmente a arquitectura é definida num documento interno à Enabler designado por *Functional Requirement Document* (FRD). Estes documentos são tipicamente partilhados com o cliente, e neles será delineada a arquitectura de todos os fluxos a serem desenvolvidos e descritos na fase do desenho da solução.

Na definição da arquitectura, existem três aspectos a ter em conta: a gestão de erros, os sistemas de monitorização e recuperação e a infraestrutura tecnológica.

Gestão de Erros

- **Assíncrona** – de forma a minimizar impacto nos fluxos operacionais;
- **Tentativas** – deve permitir um determinado número de tentativas automáticas, com um número máximo até um limite pré-definido;
- **Dependências** – o mecanismo de tentativas deve assegurar as dependências entre mensagens;
- **Monitorização** – sempre que o mecanismo de tentativas atinge o máximo de pré-definido, a camada de monitorização deve ser avisada.

Monitorização

- **Controlo de fluxos** – fazer um roteamento controlodo das mensagens, permitindo saber onde estão, em que estado se encontram, se estão em erro, etc.;
- **Arquivo de mensagens** – a solução mais completa para permitir republicação de mensagens em casos de recuperação; tem como desvantagens a maior complexidade e requisitos mais elevados de hardware.

Infraestrutura TI

- Processador;
- Memória;
- Espaço em disco;
- Particionamento físico;
- Rede de comunicações.

Desenho da solução

Esta fase tem como principal ponto, a definição de cada solução para os fluxos[7] a testar. Para isso será necessária a especificação de como o sistema será desenvolvido. Essa especificação será feita em documentos técnicos internos à Enabler que tipicamente serão partilhados com o cliente.

De seguida são apresentados factores importantes no desenho da solução:

- Comunicação entre equipas é factor fundamental para o sucesso;
- Documento de descrição técnica de cada fluxo do sistema - TRD;
- Definição do documento de testes unitários – UTD;
- Definição de áreas intermédias nas aplicações satélite (base de dados ou sistemas de ficheiros), de forma a criar uma separação lógica entre a camada de integração e a camada aplicacional;
- Documentação deve ser por fluxo;
- Reutilização de componentes já existentes sempre que possível;
- Ter em consideração o volume, desempenho, latência.

Testes

Esta fase de testes é uma fase muito importante em qualquer ciclo de desenvolvimento, pois poderá identificar falhas graves no sistema para posterior correcção. É esta fase que irá verificar a consistência do sistema em todos os seus aspectos, de acordo com a arquitectura e os requisitos definidos.

Testes unitários

Este tipo de testes serão efectuados imediatamente após a implementação, e visam principalmente o teste de cada uma das componentes independetemente das outras. Por este facto, estes testes serão efectuados passo a passo, de acordo com:

- Documentação dos testes unitários na fase de desenho (template) - UTD;
- Preenchimento do documento de testes unitários (UTD) – Consiste no teste de todos os estados possíveis em que o sistema se possa encontrar num determinado momento;

[7] Estes diferentes fluxos designam diferentes tipos de dados/famílias relativos ao retalho (i.e. ordens de compras, artigos, fornecedores, etc).

- Testes de desempenho e volume nos casos mais relevantes;

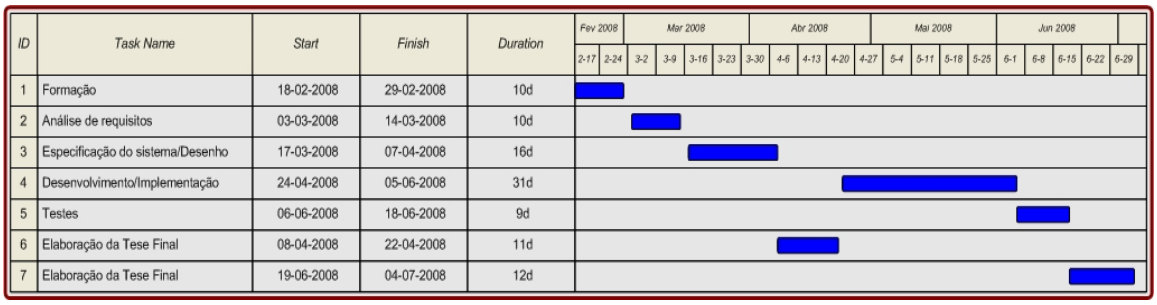
Testes de Integração

Este tipo de testes já tem como visão, o sistema integrado como um todo. Tipicamente estes testes são efectuados no cliente no fim de todas as implementações e, simularão o funcionamente do sistema aquando da existência do ambiente de produção. Tem como principais características:

- Fase de testes mais aprofundada, que determinará se o sistema ficará funcional no seu conjunto;
- Validação do sistema implementado junto de todos os outros sistemas existentes.

2.8 Plano de Trabalho

O estágio decorreu ao longo de aproximadamente quatro meses e meio, de 18 de Fevereiro a 7 de Julho e teve seis grandes períodos. Na figura 9 e no Anexo A (com imsgem de maior dimensão) é apresentado o plano criado.



ID	Task Name	Start	Finish	Duration
1	Formação	18-02-2008	29-02-2008	10d
2	Análise de requisitos	03-03-2008	14-03-2008	10d
3	Especificação do sistema/Desenho	17-03-2008	07-04-2008	16d
4	Desenvolvimento/Implementação	24-04-2008	05-06-2008	31d
5	Testes	06-06-2008	18-06-2008	9d
6	Elaboração da Tese Final	08-04-2008	22-04-2008	11d
7	Elaboração da Tese Final	19-06-2008	04-07-2008	12d

Figura 9 – Plano de Estágio

Período de Formação – Período inicial de introdução à empresa. Contacto com os sistemas, com as metodologias e as tecnologias fundamentais da empresa.

Análise de Requisitos – Estudo do negócio para posterior caracterização dos requisitos necessários. Este período durou aproximadamente dez dias.

Especificação do sistema/Desenho – Período de aproximadamente quinze dias que serviu para a participação na definição e desenvolvimento de processos de integração entre o sistema de gestão de retalho e os sistemas já existentes no cliente.

Desenvolvimento/Implementação – Período de aproximadamente um mês, em que foram desenvolvidos todos os fluxos a serem integrados.

Testes – Período constituido por aproximadamente nove dias, visto que foram sendo executados alguns testes mais minimalistas, aquando da implementação.

Elaboração da Tese Final – Período de elaboração de todas as entregas acerca do estágio.

3 Revisão Tecnológica

Nesta parte do documento irá ser descrito o estado de arte da área onde se insere este projecto.

3.1 Base de Dados Oracle

Um sistema de gestão de retalho deve ser capaz de lidar com elevados volumes de dados, ao mesmo tempo que efectua um determinado processamento.

Neste tipo de sistemas, o processamento de todas as transacções do dia tem de ser feito durante a noite, período em que a loja está fechada, tendo uma janela de tempo muito reduzida, daí que a performance é um aspecto muito importante. Temos como exemplo as entradas e saídas de informação que têm de ser analisadas e reportadas no início do dia seguinte. Além disso é essencial que estes sistemas não falhem, pois estamos a falar de sistemas, cujas falhas poderão originar uma completa incoerência dos dados, no seu conjunto.

O sistema de gestão de base de dados (SGBD) utilizado é então essencial. Por esse motivo, para o *ORMS*, foi escolhida a solução de base de dados da *Oracle*, versão 10g. De fácil integração com os produtos da antiga *Retek* as bases de dados *Oracle* são reconhecidos pela sua fiabilidade, capacidade de tratar grandes volumes de dados e capacidade de programação e alteração.

Para trabalhar com a base de dados utiliza-se a linguagem *PL/SQL*, uma extensão procedimental ao *SQL*.

A ferramenta utilizada para trabalhar com a base de dados é o *PL/SQL Developer*, da *Allround Automations*, exemplificado na figura 10.

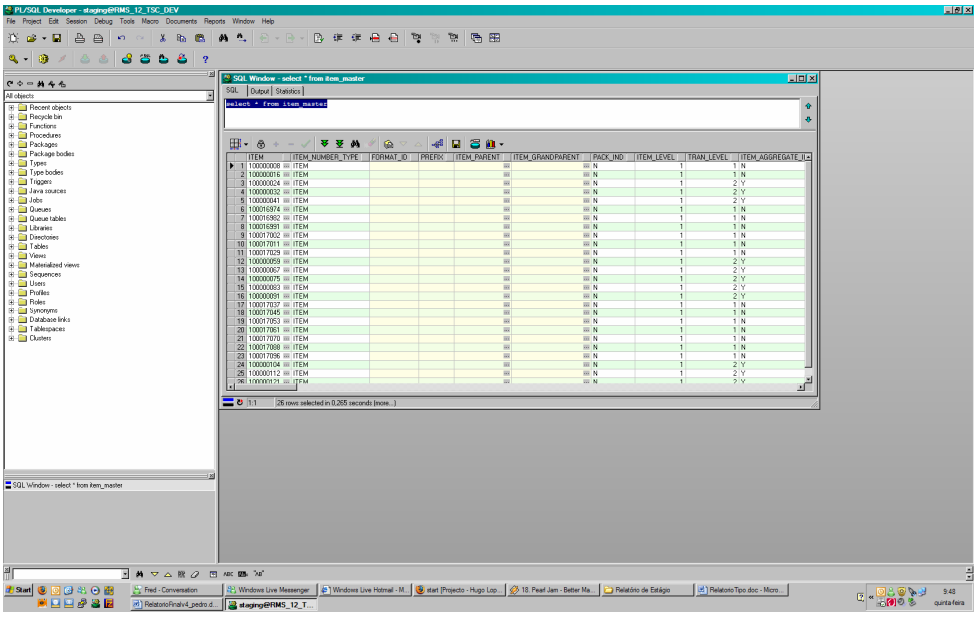


Figura 10 – PL/SQL Developer

3.2 XML

Estimulado pela insatisfação com os formatos existentes (padronizados ou não), o *World Wide Web Consortium* começou a trabalhar em meados da década de 1990 numa linguagem de marcação universal. O princípio do projecto era criar uma linguagem que pudesse ser lida por software, e ser integrada com as restantes linguagens.Os princípios desta nova linguagem, designada por XML são:

- Separação do conteúdo da formatação;
- Simplicidade;
- Possibilidade de criação de *tags* sem limitação;
- Criação de ficheiros com o objectivo de validar a sua estrutura (Designados por *Data Type Definition* - DTD);
- Podem ser reutilizados por bases de dados/sistemas diferentes;

Pela sua portabilidade, uma base de dados pode escrever a sua informação em ficheiros XML, que poderá depois ser usada para integrar os dados com outras aplicações, fazendo com que outra base de dados possa ler a informação contida nesse ficheiro. Ao longo deste documento, será possível observar muitos exemplos desta linguagem (Anexo B). Cada ficheiro XML tem a sua estrutura definida num outro documento que se designa por DTD.

De acordo com *XML, bioinformatics and data integration* (Frederich Achard, 2001), esta linguagem é altamente fléxivel, visto que é muito fácil modificar um DTD. Devido ao facto de tanto o DTD como o XML serem facilmente interpretados pelas organizações, estes poderão ser editados por indivíduos que não possuam uma grande aptidão para computadores.

Por outro lado, de acordo com a mesma publicação, esta linguagem possui algumas fraquezas, como é o caso de cada ficheiro não se encontrar codificado na forma binária, induzindo assim toda a visibilidade dos seus dados, por parte do exterior. Fez-se esta opção, para facilitar a sua edição em qualquer editor de texto, sendo assim desnecessária a utilização de editores próprios, de modo a que fosse possível abrir ficheiros binários.

3.3 Oracle Retail Merchandising System

O *Oracle Retail Merchandising System* é um *ERP* específico para empresas de retalho. Especialmente desenhado para gerir actividades de retalho complexas num ambiente global e através de múltiplos canais de venda, o *Oracle Retail Merchandising System* incorpora três grandes núcleos funcionais: gestão das fundações do negócio, gestão de mercadorias e acompanhamento financeiro da mercadoria.

Estes três núcleos permitem aos retalhistas executar actividades cruciais do dia-a-dia de retalho permitindo aumentar o foco nas decisões chave que permitirão aumentar os lucros e diminuir os custos.

Esta é a principal aplicação implementada pela *Enabler*. Começou por ser escolhida pois era a ferramenta que a *MCH* utilizava, e era aí que se concentravam os conhecimentos da empresa. A partir desse momento e com a sua evolução constante, a *Enabler* continuou a apostar neste produto.

O sistema começou por se chamar RMS (*Retek Merchandising System*™) desenvolvido pela *Retek Information Systems*™, uma empresa norte-americana, líder em soluções na indústria de retalho e que possui uma grande experiência neste tipo de negócio.

Em 2005 foi adquirida pela *Oracle*, numa luta renhida com a *SAP*, para entrar na área de retalho. A estratégia da *Oracle* foi a de dividir a aplicação e criar novas aplicações específicas para a área de retalho e englobadas numa oferta geral de Retalho. A compra do RMS por parte da *Oracle* veio beneficiar a *Enabler*, pois aumentou a exposição do produto a clientes internacionais e, com o posicionamento da *Enabler* junto da *Oracle*, aumentou também a exposição da *Enabler* a novos mercados. Após a aquisição do RMS pela Oracle, este passou-se a designar por *Oracle Retail Merchandising System* (ORMS).

A solução *ORMS* foi desenvolvida para satisfazer as necessidades das empresas de retalho no âmbito de gestão de informação, permitindo 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 secundárias o que permite operar localmente com a respectiva moeda, consolidando, no entanto, os movimentos na moeda primária para fins de relatórios corporativos. Para além das moedas, permite ainda vários idiomas numa única instalação, o que facilita a compreensão dos vários utilizadores de diferentes países.

A informação no ORMS tem por base duas hierarquias fundamentais, a organizacional e a mercadológica. A primeira traduz a estrutura operacional da empresa e é constituída por *Company*, *Chain*, *Area*, *Region*, *District*, *Store* e *Warehouse*. A segunda reflecte o modo como os artigos são geridos/categorizados na empresa, sendo composta por *Company*, *Division* (correspondendo ao departamento, por ex.Vestuário), *Group* (secção, por ex. Vestuário de Verão), *Department* (categoria, por ex.roupa de praia), *Class* (subcategoria ou família) e *Subclass* (subfamília ou unidade base).

Este sistema tem como principais módulos:

- Criação e Manutenção de Items;
- Gama;
- Preços de Custo;

- Preços de Retalho;
- Gestão dos Fornecedores;
- Promoções;
- Custos estimados de armazenamento;
- Ordens de Compra;
- Alocações, por forma a distribuir determinada mercadoria pelas várias lojas;
- Reaprivisionamento de Lojas;
- Gestão de Inventários;

Esta aplicação afere uma grande visibilidade em todos os processos dos retalhistas, como confere Michael Bush em:

”The system allows us to see everything more clearly. For example, we can see easily now what related items people tend to buy, which allows us to make the store layout and design a bit more intelligent. If belts tend to sell with blue jeans, we can see that and display the two items in the same area.”8

Como foi dito, agora com a implementação deste novo sistema da Oracle, é possível verificar quais os produtos que as pessoas tendem a comprar, possibilitando assim uma gestão mais inteligente do negócio.

3.4 Manhattan Warehouse Management for Open Systems

Este sistema vai ser usado na gestão do entreposto.

Overview

O *Manhattan Warehouse Management System* (MWMS) é um sistema que gere todo o processo de entrada, saída e de controlo de inventário relativamente a um determinado entreposto. Este foi desenvolvido por uma empresa média sediada nos Estados Unidos da América designada por *Manhattan Associates*. Durante 17 anos, a MA concentrou-se exclusivamente em ajudar cadeias de abastecimento de diversas empresas, de modo a que estas atingissem custos logísticos mais baixos e, consequentemente lucros mais altos e clientes satisfeitos. Todos os seus 2,300 colaboradores focam-se principalmente na optimização dessa cadeia de abastecimento, trabalhando assim de modo a criar valor no negócio dos seus clientes através de investigação e desenvolvimento de novos produtos.

[8] *http://www.extendedretail.com/pastissue/article.asp?art=25764&issue=147*

Descrição

A arquitectura deste sistema foi especialmente desenhada para criar uma enorme flexibilidade e escabilidade para os clientes. As diferentes camadas deste aplicação poderão estar contidas em diferentes servidores (ou no mesmo) permitindo assim uma grande flexibilidade no que toca por exemplo ao número de entrepostos a usar por uma operadora logística. Estas camadas são:

§ Camada de interface com o utilizador;

- Camada da aplicação Web;
- Camadado servidor aplicacional;
- Camada de Dados;
- Camada de integração.

Interface com o utilizador

A interface do MWMS com o utilizador é corrida num *web-browser* como o *Internet Explorer*, e está bastante intuitiva à sua utilização. Na figura 11 é apresentada essa interface:

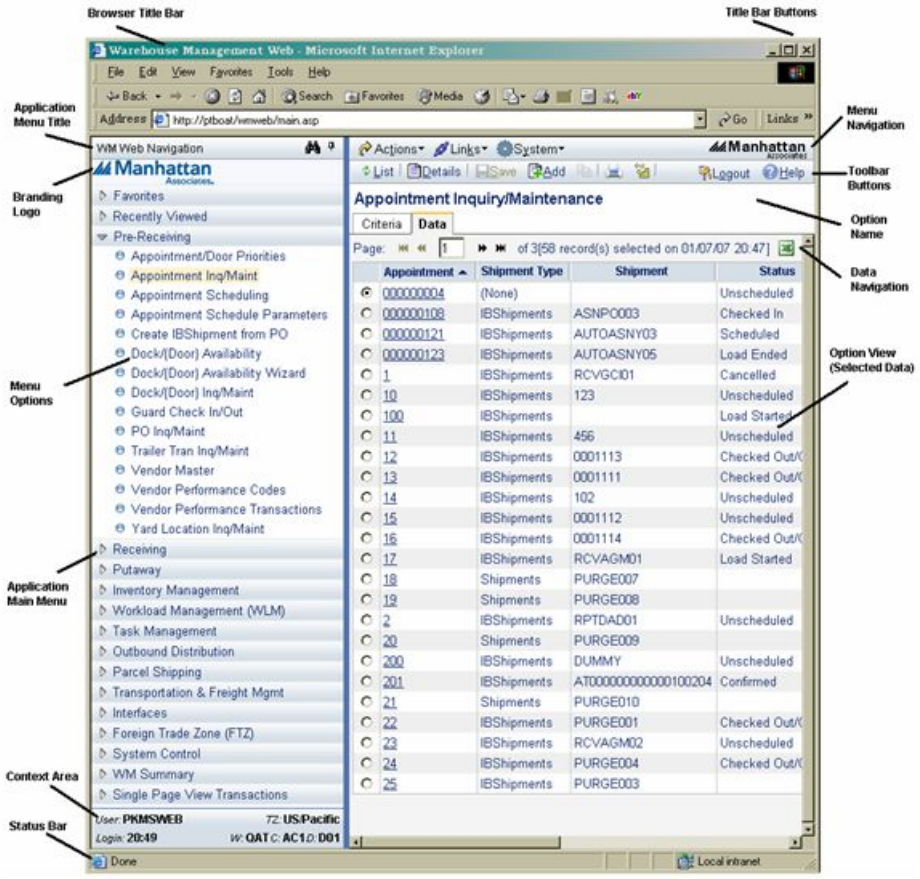


Figura 11 – Interface do MWMS

Aplicação Web

Esta aplicação é responsável por processar os pedidos efectuados pela aplicação cliente. Este sistema corre em *Windows 2003* e é baseado em Internet Information Server 6.0. O servidor usa o XML como forma de representar todos os dados do negócio. Esta camada é então responsável pelas seguintes actividades:

- Gerar dinamicamente todas as páginas usadas no sistema de gestão do entreposto;
- Formatar todos os dados para serem visualizados;
- Suportar várias aplicações clientes;
- Gerar relatórios;

Por outro lado, esta camada apresenta uma boa vantagem, pois visto que corre em *Windows 2003*, é possível a distribuição do servidor por várias máquinas com o objectivo de aumentar a eficiência do mesmo.

Servidor aplicacional

Esta camada é responsável pela execução das operações do entreposto (quer sejam executadas via Rádio Frequência, ou usando o interface Web, e pela execução de operações de gestão/manutenção da aplicação.. O servidor desta aplicação aloja todo a lógica de negócio e é acedido através de *Telnet*[9] no caso da operação de RF e através de um pedido CORBA[10] nos restantes casos.

Base de Dados

Esta base de dados pode ser integrada em *Sql Server 2000*, IBM DB2 ou então nas versões *9i* ou *10g* da *Oracle*. A camada de acesso de dados existente neste produto, disponibiliza uma camada abstracta entre a base de dados e a aplicação.

Integração

Esta camada disponibiliza o *Enterprise Integration Services* (EIS) que está integrado na solução da *Manhattan*. O EIS permite a comunicação e envio de mensagens entre todos os produtos da *Manhattan*, assim como no caso deste projecto, com outros produtos. Esta plataforma será descrita em detalhe mais à frente.

[9] Telnet é um protocolo cliente-servidor de comunicações usado para permitir a comunicação entre computadores ligados numa rede (exemplos: rede local / LAN, Internet), baseado em TCP - http://pt.wikipedia.org/wiki/Telnet

[10] CORBA é a arquitetura padrão para estabelecer e simplificar a troca de dados entre sistemas distribuídos heterogêneos.

Ambientes de implementação existentes

No MWMS existem vários ambientes de implementação, nomeadamente: desenvolvimento, testes e produção.

Ambiente de Desenvolvimento

Este ambiente é usado para assegurar que o software que está a ser desenvolvido é de alta qualidade.

Ambiente de Testes

Este ambiente é usado na parte final do projecto, para determinar que falhas ou brechas se deixaram para modificar e melhorar.

Ambiente de Produção

Este ambiente é o ambiente em que o sistema irá correr aquando da sua utilização no negócio do cliente.

3.5 Oracle Retail Integration Bus

O ORIB é uma solução da *Oracle Retail* responsável por sincronizar dados entre as aplicações OR que poderão fazer parte do sistema de informação de um retalhista, assim como outras aplicações que não pertencem à solução de retalho da Oracle. Esta plataforma poderá utilizar o paradigma da Publicação/Subscrição, de forma a possibilitar o fluxo de dados entre diferentes aplicações.

- Esta plataforma implementa duas camadas principais:
- Integração horizontal das funcionalidades (Recuperação de erros, APIs)

Publicação/Subscrição

Publicação e Subscrição é um paradigma de mensagens assícronas, onde a origem (publicadores) das mensagens não está pré-programada para mandar as suas mensagens para destinos específicos (subscritores). Porém, as mensagens publicadas são caracterizadas em classes, sem o conhecimento de qual o componente que deverá subscrevê-las. Na parte da subscrição, os subscritores estarão aptos a receber determinados tipos de mensagens, sem saberem também qual o publicador das próprias. Esta abordagem garente certamente uma grande escalabilidade e uma topologia de redes mais dinâmica,como pode ser verificado na figura 11.

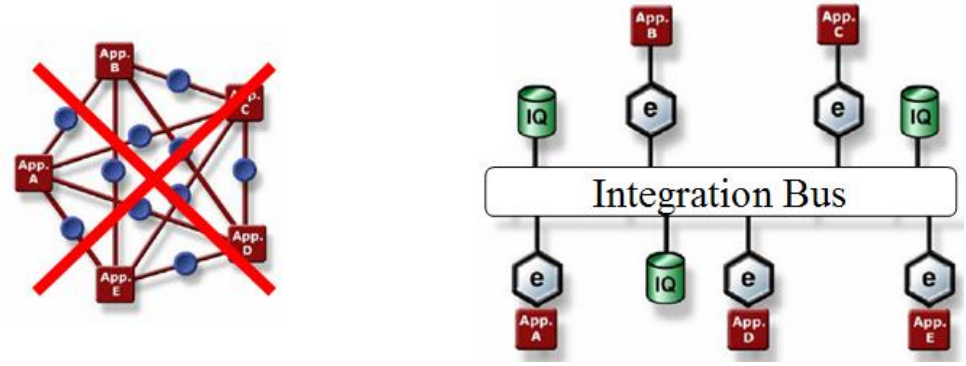


Figura 12 –Antes/Depois do ORIB

#	Antes do RIB	Depois do RIB
1	Grande desorganização na arquitectura do sistema	Mais bem organizado que anteriormente
2	Pouca escalabilidade, devido à criação de um grande número de ligações,	Grande escalabilidade. Apenas necessária a criação de uma única ligação ao RIB para publicar a mensagem

	aquando da existência de uma nova solução no sistema	
3	Dependêmcia física entre todas as aplicações	Não existe qualquer dependência funcional entre as aplicação, pois cada mensagem poderá permanecer numa fila de espera designada por JMS até a aplicação ficar operacional
4	Quando existe uma falha em alguma operação de publicação ou de subscrição, o fluxo terá de enviar novamente a mensagem	Com a existência de um 'hospital', a mensagem será automaticamente colocada nessa ponto até que seja possível executar novamente a operação pretendida (subscrição ou publicação)
5	Não é possível verificar a mensagem antes da subscrição	A qualquer momento é possível verificar o conteúdo da queue e verificar se a mensagem está bem construída.
6[11]	Tipicamente era normal o uso de arquitecturas *point-to-point* com recorrência ao protocolo FTP.	Passou a ser orientado à mensagem, acabando assim com o enorme número de transferências de ficheiros a efectuar por FTP
7	Diversas mensagens para diferentes aplicações	A mesma mensagem é facilmente integrada em todas as aplicações

Tabela 1 – Comparação: utilização/não utilização do RIB

Para que os subscritores apenas recebam as mensagens a si destinadas, terá de haver uma filtragem das mensagens. A filtragem é realizada através de uma abordagem baseada em tópicos criados na fila de espera. Nesta plataforma, as mensagens são então publicadas nos respectivos tópicos. Tipicamente cada tópico é criado de acordo com um determinado fluxo (ex: fluxo dos items). Assim sendo, após a publicação de mensagens num determinado tópico, a componente que vai subscrever esse tópico, receberá todas as mensagens nele existentes.

JMS

Java Message Service, ou *JMS*, é uma API da linguagem Java orientada à manipulação de mensagens. Através desta API, é possível fazer com que duas ou mais aplicações comuniquem entre si, através da troca de mensagens.

Esta JMS suporta dois tipos de implementação: modelo ponto-a-ponto e modelo publicação/subscrição.

No modelo ponto-a-ponto (figura 13), um "produtor" (*producer*) envia mensagens para uma fila e um "consumidor" (*consumer*) lê essas mensagens. Neste caso, o produtor conhece o

[11] *http://download.oracle.com/docs/cd/B31315_01/rib/pdf/120/rib -120-tag.pdf*

destino da mensagem e envia diretamente a mensagem para a fila do *consumidor*. Este modelo possui as seguintes características:

- apenas um consumidor poderá ler a mensagem;
- não é necessário que o produtor esteja activo no momento em que o consumidor lê a mensagem, assim como não é necessário que o consumidor também o esteja no momento que o produtor envia a mensagem;
- quando lê uma mensagem com sucesso o consumidor envia um aviso (*acknowledged*) para o produtor.

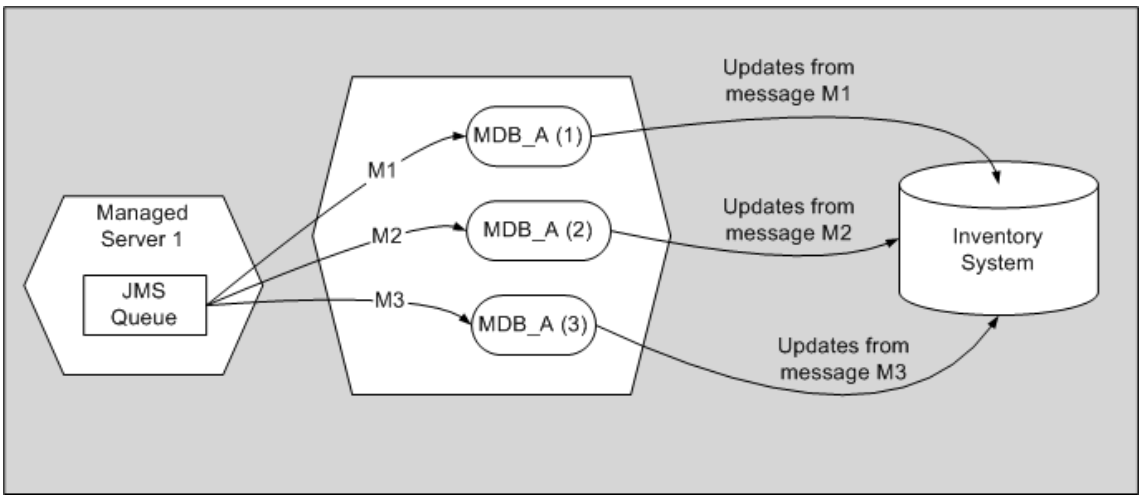


Figura 13 – JMS sob o paradigma *point-to-point*

O modelo de publicação/subscrição (figura 14) suporta a publicação de mensagens para um determinado tópico de mensagens (*message topic*). O(s) subscritores podem receber mensagens de um determinado tópico. Neste modelo, nem o publicador nem o subscritor sabem um do outro, visto não ser uma arquitectura transparente. As características deste modelo são:

- vários subscritores poderão subscrever a mesma mensagem;
- Não existe nenhuma dependência funcional entre os publicadores e os subscritores. Quer isto dizer que mensagens poderão ser publicadas, independentemente se o outro componente se encontrar ou não *on-line*.

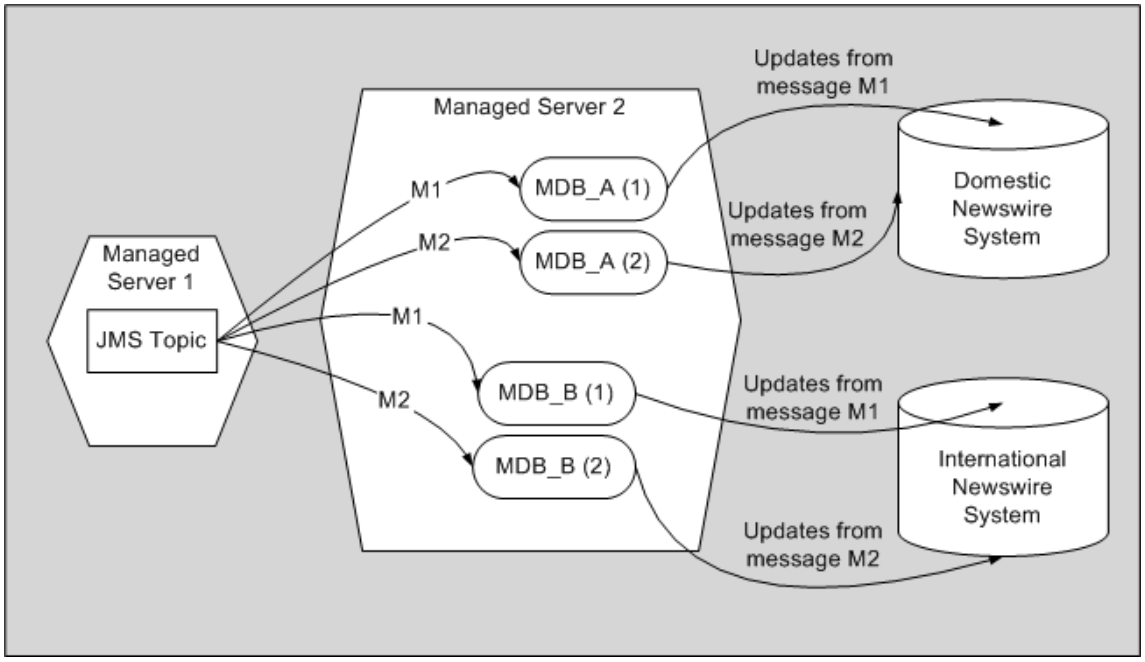


Figura 14 – JMS sob o paradigma publicação/subscrição

3.6 Manhattan Enterprise Integration Services

Introdução

Sistemas de gestão da cadeia de abastecimento são por natureza o suporte à comunicação dos mais variados componentes pertencentes a um sistema de retalho. Por exemplo, as ordens de compra, os artigos e outros dados críticos vêm de uma plataforma central de gestão. Este exemplo requer assim uma operação completa de integração com outros sistemas de modo a permitir o funcionamento do sistema como um todo. Em resposta a esta necessidade, a *Manhattan Associates* desenvolvou uma plataforma de integração designada por *Enterprise Integration Services* (EIS).

Objectivos

Com o aparecimento de uma nova geração de produtos desenvolvidos pela *Manhattan Associates* (como por exemplo ERPs) focados principalmente nos baixos custos de implementação, foi necessário a utilização da mesma arquitectura de integração por parte de todos eles. Uma solução ponto-a-ponto não preenchiria estes requisitos de custos, pois:

- Para cada produto desenvolvido, seria necessária a implementação de uma camada de comunicação;
- Cada subscrição teria de ser gerida por cada aplicação;
- Cada documento teria de ser transferido para todas as restantes plataformas, o que poderia causar grande latência no sistema.

Características

Assim sendo foi desenvolvida esta plataforma com o objectivo de resolver todos os problemas descritos atrás. Esta plataforma permite a comunicação através dos protocolos de HTTP ou FTP para o caso de XML, e apenas de FTP, no caso de *Flat Files*. Para a concepção de tal plataforma, foram utilizados os seguintes princípios:

- Minimização da criação de mensagens, visto que cada sistema poderá subscrever e consequentemente integrar a mesma mensagem;
- Possível tratamento de erros nas transferências, sem perdas de mensagem e de conteúdo, visto fazer automaticamente o recarregamento da mensagem automaticamente;
- Escalabilidade[12];
- *System Logging*[13].

[12] escalabilidade é uma característica desejável em todo o sistema, numa rede ou num processo, que indica a sua habilidade de manipular uma determinada quantidade de trabalho de forma uniforme, ou estar preparado para o crescimento do mesmo - *http://pt.wikipedia.org/wiki/Escalabilidade*

[13] Possibilidade de efectuar todo o processo de verificação do estado dos componentes constituintes do sistema, através da leitura de ficheiros.

4 Descrição do Trabalho Realizado

Neste capítulo serão descritas todas as decisões efectuadas, assim como a estrutura física do sistema e de todos os seus componentes

4.1 Análise e Escolha da Arquitectura

Neste secção serão descritos três diferentes tipos de arquitecturas, com vista à execução dos objectivos.

Seguindo a investigação feita pelo estagiário, durante a realização do seu projecto, foi descoberta a possibilidade de ser efectuada a integração de três maneiras diferentes: via ficheiros XML, via flat files e via acesso directo à base de dados. Será descrito em baixo cada um destes tipos de integração para posteriormente haver uma discussão das vantagens e desvantagens de cada um. Estas possíveis soluções surgiram na pesquisa de documentação interna da Enabler referente a outros projectos semelhantes na àrea da integração.

Integração via ficheiros XML

Subscrição do Manhattan

Para este caso de integração, no caso de integração entre o MWMS e o ORMS, teremos de ter três tipos de adaptadores: o publicador do ORMS, o transformador e o subscritor para o Manhattan. Para que a integração seja feita via ficheiros XML, terá então de existir uma regra para cada um desses adaptadores.

Neste caso, como o fluxo se dá do lado do ORMS para o MWMS, ter-se-à no adaptador de publicação uma regra que determina se o adaptador irá verificar periodicamente a existência de alterações na base de dados, como já foi referido, através das tabelas auxiliares que simulam uma fila de espera. Caso exista alguma alteração efectuada, é então criada a mensagem e colocada no tópico da JMS de acordo com a regra do adaptador de publicação.

É então agora a vez do adaptador transformador (TAFR) de subscrever a mensagem do tópico, executar um determinado mapeamento definido na regra do transformador e colocar a mensagem noutro tópico, que deverá ser subscrito pelo adaptador referente ao MWMS. Este *modus operandi* é genérico sempre que existe a necessidade de proceder a alterações na mensagem entre o publicador e o subscritor.

Posteriormente, a regra do subscritor vai determinar que deverá ser subscrita uma mensagem existente num determinado tópico após o mapeamento. Essa mensagem é então escrita num ficheiro XML e colocada numa determinada pasta, que é definida num componente chamado "*connection point*", através da execução da regra deste componente. Para além da pasta, também os nomes dos ficheiros seguem um determinado padrão: *fluxoddmmyyyyhhmmss.xml*, em que *ddmmyyyy* significa a data e *hhmmss* a hora da criação do ficheiro.

Posteriormente, através da execução calendarizada de scripts, são copiados através de FTP todos os ficheiros existentes nessa pasta para uma pasta existente no servidor do MWMS. O EIS estando configurado para aceder a essa pasta, irá verificar a existência de novos ficheiros, criando em seguida uma mensagem a partir do ficheiro XML e colocando-a numa fila de espera interna. Todos os ficheiros XML integrados no EIS são validados contra a sua estrutura (incluindo o tipo de dados), garantindo que apenas ficheiros com uma estrutura válida são integrados. Possuindo uma estrutura válida, as mensagens são processadas, e irão ser preenchidas umas tabelas intermediárias. É a partir dessas tabelas que o WMS preenche o seu modelo de dados de negócio.

Após a conclusão do preenchimento das tabelas intermediárias do MWMS, são executados uns *scripts* existentes no MWMS (denominados de *bridges*) que irão verificar a integridade dos dados, e caso esteja correcta, todos os dados serão inseridos no sistema. Toda a arquitectura deste processo está representada na figura 15.

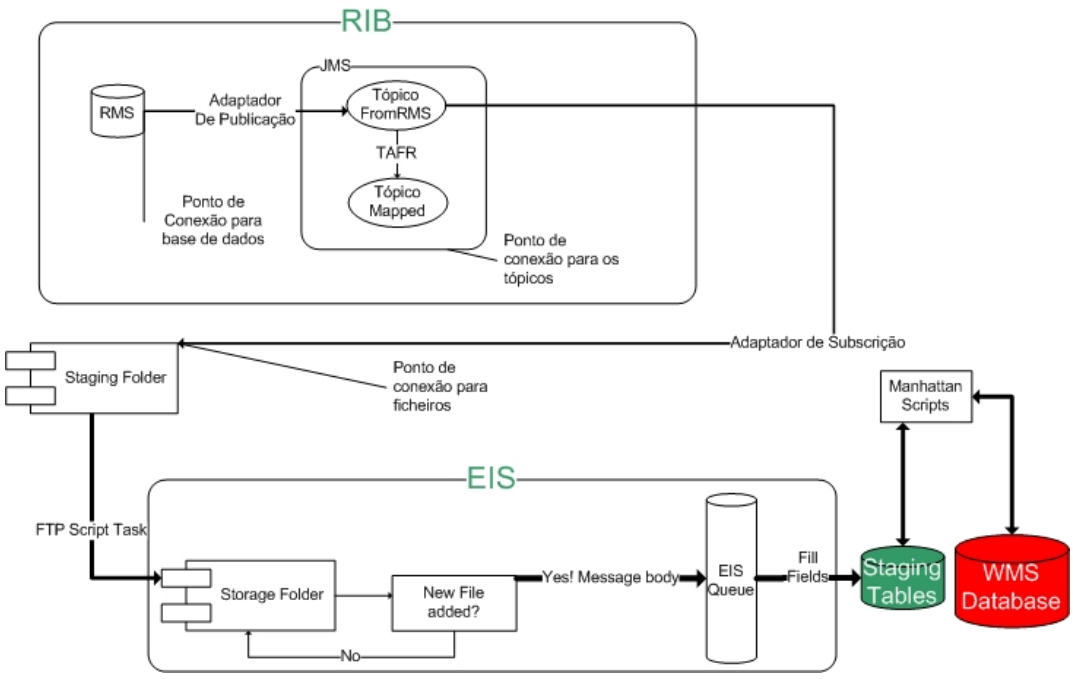


Figura 15 – Processo através de XML

Aquando da inicialização do adaptador de publicação ou de subscrição, caso o seu ponto de conexão seja a base de dados, terá então de existir um método GETNXT (publicação) como já foi referido na secção anterior.

O método GETNXT tem como objectivo a verificação na parte do publicador da existência de alterações registadas na tabela auxiliar (fila de espera) designada, por exemplo, por SUPPLIER_MFQUEUE.

Caso seja encontrado algum registo na tabela, proceder-se-à então à construção de um objecto que possui toda a informação relativa a um determinado fluxo. Esse objecto possuí atributos, assim como outros objectos dentro de si, de acordo com os dados com necessidade de serem integrados. É esse objecto que será posto no tópico a que está ligado o adaptador, na respectiva JMS, através da regra de publicação.

Caso a mensagem tenha de ser mapeada num novo modelo de dados (caso os dois sistemas possuam tabelas e relações diferentes), será então necessário um tópico na JMS para que seja publicada uma nova mensagem segundo a regra pertencente ao adaptador de transformação. Esta regra irá realizar o mapeamento de acordo com os DTDs de cada sistema.

Caso se dê o acontecimento de algum tipo de erro, quer por alguma componente estar desconfigurada, ou pelo servidor estar em baixo, o status na tabela auxiliar será alterado para 'H' e uma tabela de erros será preenchida. O objectivo desta tabela será armazenar as mensagens que contenham erros, e permitir a sucessão de várias tentivas de execução da publicação/subscrição através de um adaptador.

É posteriormente chamada uma regra para que seja possível a transformação da mensagem existente num determinado tópico da JMS para um ficheiro em XML. Esta apenas usa a biblioteca do ORIB para que seja possível memorizar a mensagem construída pelo método.

Publicação do Manhattan

No caso dos fluxos em que o MWMS serve como publicador, existirá uma *script* que copia o ficheiro existente na pasta do EIS para uma pasta a ser processada pelo RIB. Para além disto, os componentes de subscrição detectam em seguida a existência de novos ficheiros nessa pasta. Caso exista algum ficheiro, esse será introduzido no tópico correspondente pela regra existente no adaptador de publicação.

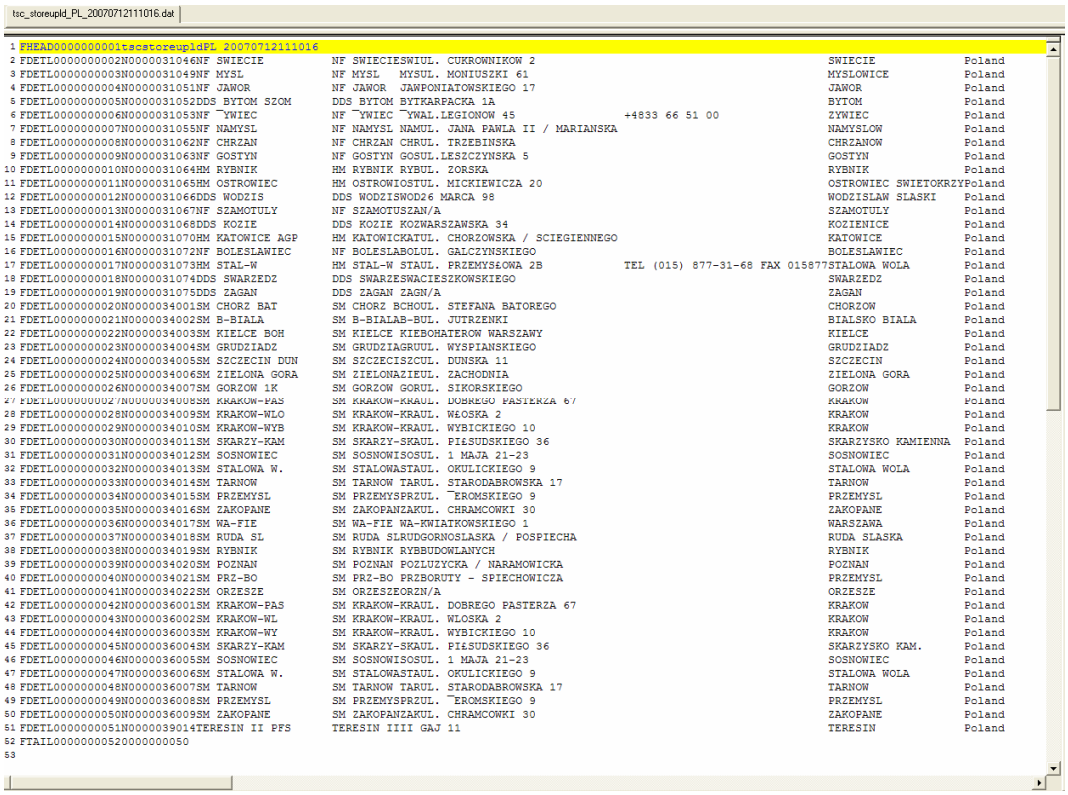
Caso a mensagem tenha de ser mapeada num novo modelo de dados (caso os dois sistemas possuam tabelas e relações diferentes), será então necessário um tópico na JMS para que seja publicada uma nova mensagem segundo a regra pertencente ao adaptador de transformação. Esta regra irá realizar o mapeamento de acordo com os DTDs de cada sistema.

No que toca à subscrição do lado do ORMS, esta será efectuada por um processo *standard*, usando o método da base de dados CONSUME. Este método irá introduzir o conteúdo da mensagem em tabelas da base de dados.

Este método irá primeiro inserir a informação com as chaves numa tabela de controle, cujo principal objectivo é manter um histórico de todas as transacções realizadas. Após esta operação, são então inseridos, alterados ou apagados os dados nas tabelas do RMS, ficando assim a informação disponível actualizada. O tipo de operação a efectuar será determinado pelo tipo da mensagem recebida no tópico da JMS. Poderá ser visto um exemplo de uma mensagem existente na JMS no Anexo C. De acordo com este anexo a mensagem possui então um cabeçalho com o objectivo de indicar o tipo de fluxo a que pertence e o tipo de mensagem que contém (criação, alteração ou remoção).

Integração via flat files

Em termos de arquitectura, este método é muito semelhante ao anterior, pelo que não se torna necessário descrever o processo. A principal diferença consiste na estrutura do ficheiro que comunica com as duas plataformas. Estes ficheiros designados por flat files, são caracterizados principalmente pela sua simplicidade. São ficheiros sequenciais, em que cada posição será reservada para um campo da tabela da base de dados. Na figura 16 é apresentado um exemplo deste tipo de ficheiro.



```
tsc_storeupld_PL_20070712111016.dat
 1 FHEAD0000000001tscstoreupldPL 20070712111016
 2 FDETL0000000002N0000031046NF SWIECIE          NF SWIECIESWIUL. CUKROWNIKOW 2                                        SWIECIE              Poland
 3 FDETL0000000003N0000031049NF MYSL             NF MYSL   MYSUL. MONIUSZKI 61                                         MYSLOWICE            Poland
 4 FDETL0000000004N0000031051NF JAWOR            NF JAWOR  JAWPONIATOWSKIEGO 17                                        JAWOR                Poland
 5 FDETL0000000005N0000031052DDS BYTOM SZOM      DDS BYTOM BYTKARPACKA 1A                                              BYTOM                Poland
 6 FDETL0000000006N0000031053NF ¯YWIEC           NF ¯YWIEC ¯YWAL.LEGIONOW 45              +4833 66 51 00               ZYWIEC               Poland
 7 FDETL0000000007N0000031055NF NAMYSL           NF NAMYSL NAMUL. JANA PAWLA II / MARIANSKA                            NAMYSLOW             Poland
 8 FDETL0000000008N0000031062NF CHRZAN           NF CHRZAN CHRUL. TRZEBINSKA                                           CHRZANOW             Poland
 9 FDETL0000000009N0000031063NF GOSTYN           NF GOSTYN GOSUL.LESZCZYNSKA 5                                         GOSTYN               Poland
10 FDETL0000000010N0000031064HM RYBNIK           HM RYBNIK RYBUL. ZORSKA                                               RYBNIK               Poland
11 FDETL0000000011N0000031065HM OSTROWIEC        HM OSTROWIOSTUL. MICKIEWICZA 20                                       OSTROWIEC SWIETOKRZYPoland
12 FDETL0000000012N0000031066DDS WODZIS          DDS WODZISWOD26 MARCA 98                                              WODZISLAW SLASKI     Poland
13 FDETL0000000013N0000031067NF SZAMOTULY        NF SZAMOTUSZAN/A                                                      SZAMOTULY            Poland
14 FDETL0000000014N0000031068DDS KOZIE           DDS KOZIE KOZWARSZAWSKA 34                                            KOZIENICE            Poland
15 FDETL0000000015N0000031070HM KATOWICE AGP     HM KATOWICKATUL. CHORZOWSKA / SCIEGIENNEGO                            KATOWICE             Poland
16 FDETL0000000016N0000031072NF BOLESLAWIEC      NF BOLESLABOLUL. GALCZYNSKIEGO                                        BOLESLAWIEC          Poland
17 FDETL0000000017N0000031073HM STAL-W           HM STAL-W STAUL. PRZEMYS£OWA 2B          TEL (015) 877-31-68 FAX 015877STALOWA WOLA        Poland
18 FDETL0000000018N0000031074DDS SWARZEDZ        DDS SWARZESWACIESZKOWSKIEGO                                           SWARZEDZ             Poland
19 FDETL0000000019N0000031075DDS ZAGAN           DDS ZAGAN ZAGN/A                                                      ZAGAN                Poland
20 FDETL0000000020N0000034001SM CHORZ BAT        SM CHORZ BCHOUL. STEFANA BATOREGO                                     CHORZOW              Poland
21 FDETL0000000021N0000034002SM B-BIALA          SM B-BIALAB-BUL. JUTRZENKI                                            BIALSKO BIALA        Poland
22 FDETL0000000022N0000034003SM KIELCE BOH       SM KIELCE KIEBOHATEROW WARSZAWY                                       KIELCE               Poland
23 FDETL0000000023N0000034004SM GRUDZIADZ        SM GRUDZIAGRUUL. WYSPIANSKIEGO                                        GRUDZIADZ            Poland
24 FDETL0000000024N0000034005SM SZCZECIN DUN     SM SZCZECISZCUL. DUNSKA 11                                            SZCZECIN             Poland
25 FDETL0000000025N0000034006SM ZIELONA GORA     SM ZIELONAZIEUL. ZACHODNIA                                            ZIELONA GORA         Poland
26 FDETL0000000026N0000034007SM GORZOW 1K        SM GORZOW GORUL. SIKORSKIEGO                                          GORZOW               Poland
27 FDETL0000000027N0000034008SM KRAKOW-PAS       SM KRAKOW-KRAUL. DOBREGO PASTERZA 67                                  KRAKOW               Poland
28 FDETL0000000028N0000034009SM KRAKOW-WLO       SM KRAKOW-KRAUL. W£OSKA 2                                             KRAKOW               Poland
29 FDETL0000000029N0000034010SM KRAKOW-WYB       SM KRAKOW-KRAUL. WYBICKIEGO 10                                        KRAKOW               Poland
30 FDETL0000000030N0000034011SM SKARZY-KAM       SM SKARZY-SKAUL. PI£SUDSKIEGO 36                                      SKARZYSKO KAMIENNA   Poland
31 FDETL0000000031N0000034012SM SOSNOWIEC        SM SOSNOWISOSUL. 1 MAJA 21-23                                         SOSNOWIEC            Poland
32 FDETL0000000032N0000034013SM STALOWA W.       SM STALOWASTAUL. OKULICKIEGO 9                                        STALOWA WOLA         Poland
33 FDETL0000000033N0000034014SM TARNOW           SM TARNOW TARUL. STARODABROWSKA 17                                    TARNOW               Poland
34 FDETL0000000034N0000034015SM PRZEMYSL         SM PRZEMYSPRZUL. ¯EROMSKIEGO 9                                        PRZEMYSL             Poland
35 FDETL0000000035N0000034016SM ZAKOPANE         SM ZAKOPANZAKUL. CHRAMCOWKI 30                                        ZAKOPANE             Poland
36 FDETL0000000036N0000034017SM WA-FIE           SM WA-FIE WA-KWIATKOWSKIEGO 1                                         WARSZAWA             Poland
37 FDETL0000000037N0000034018SM RUDA SL          SM RUDA SLRUDGORNOSLASKA / POSPIECHA                                  RUDA SLASKA          Poland
38 FDETL0000000038N0000034019SM RYBNIK           SM RYBNIK RYBBUDOWLANYCH                                              RYBNIK               Poland
39 FDETL0000000039N0000034020SM POZNAN           SM POZNAN POZLUZYCKA / NARAMOWICKA                                    POZNAN               Poland
40 FDETL0000000040N0000034021SM PRZ-BO           SM PRZ-BO PRZBORUTY - SPIECHOWICZA                                    PRZEMYSL             Poland
41 FDETL0000000041N0000034022SM ORZESZE          SM ORZESZEORZN/A                                                      ORZESZE              Poland
42 FDETL0000000042N0000036001SM KRAKOW-PAS       SM KRAKOW-KRAUL. DOBREGO PASTERZA 67                                  KRAKOW               Poland
43 FDETL0000000043N0000036002SM KRAKOW-WL        SM KRAKOW-KRAUL. WLOSKA 2                                             KRAKOW               Poland
44 FDETL0000000044N0000036003SM KRAKOW-WY        SM KRAKOW-KRAUL. WYBICKIEGO 10                                        KRAKOW               Poland
45 FDETL0000000045N0000036004SM SKARZY-KAM       SM SKARZY-SKAUL. PI£SUDSKIEGO 36                                      SKARZYSKO KAM.       Poland
46 FDETL0000000046N0000036005SM SOSNOWIEC        SM SOSNOWISOSUL. 1 MAJA 21-23                                         SOSNOWIEC            Poland
47 FDETL0000000047N0000036006SM STALOWA W.       SM STALOWASTAUL. OKULICKIEGO 9                                        STALOWA WOLA         Poland
48 FDETL0000000048N0000036007SM TARNOW           SM TARNOW TARUL. STARODABROWSKA 17                                    TARNOW               Poland
49 FDETL0000000049N0000036008SM PRZEMYSL         SM PRZEMYSPRZUL. ¯EROMSKIEGO 9                                        PRZEMYSL             Poland
50 FDETL0000000050N0000036009SM ZAKOPANE         SM ZAKOPANZAKUL. CHRAMCOWKI 30                                        ZAKOPANE             Poland
51 FDETL0000000051N0000039014TERESIN II PFS      TERESIN IIII GAJ 11                                                   TERESIN              Poland
52 FTAIL0000000520000000050
53
```

Figura 16 – Exemplo de um flat file

Integração directa na base de dados

Neste tipo de integração não é usado o EIS, pois os dados são colocados directamente nas tabelas intermediárias do MWMS.

Subscrição do Manhattan

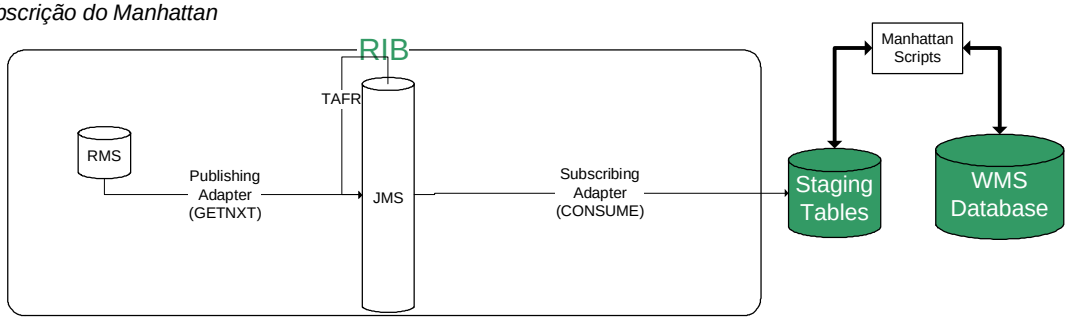


Figura 17 – Arquitectura de acesso directo à base de dado s (subscrição MWMS)

No que toca a este tipo de arquitectura, existiriam dois procedimentos na base de dados com vista à publicação e à subscrição de dados por parte do MWMS ou do ORMS. A subscrição do MWMS, seria feita através da chamada do método CONSUME existente na base de dados, pelo adaptador de subscrição (figura 17). Este adaptador irá subscrever a mensagem já alterada pelo adaptador de transformação para que esta possua o mesmo formato do modelo de dados do MWMS, garantindo assim que os dados sejam compatíveis ao serem integrados.

Publicação do Manhattan

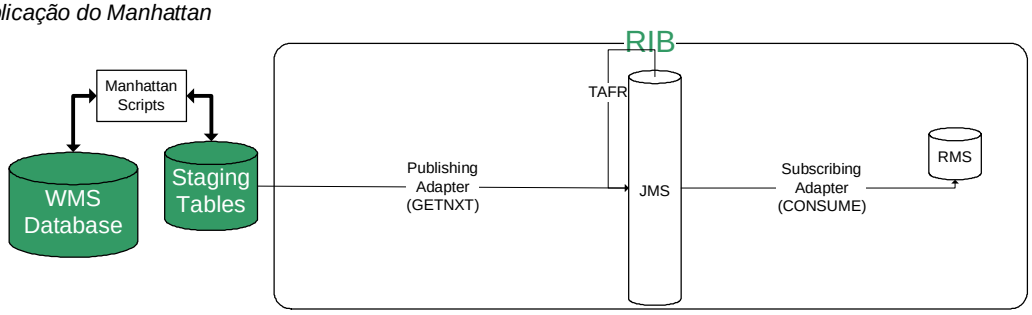


Figura 18 – Arquitectura de acesso directo à base de dados (publicação MWMS)

Aquando da execução dos fluxos de publicação do MWMS, seria necessária a criação de *triggers* para as tabelas intermediárias de forma a inserir as alterações efectuadas numa tabela auxiliar. Seria também necessária a criação de um procedimento na base de dados, para que sejam processadas todas as alterações efectuadas. O algoritmo deste procedimento é semelhante ao já descrito na integração através de ficheiros XML. Posteriormente a mensagem será colocada num tópico específico para que o transformador a subscreva e crie outra mensagem de acordo com o modelo de dados do ORMS. Esta nova mensagem será então publicada num novo tópico, e ficando pronta a ser subscrita pelo adaptador de subscrição. Este adaptador chamará o método CONSUME que irá preencher, apagar ou alterar dados nas tabelas do ORMS (figura 18).

Comparação entre os métodos e Conclusão

Para a definição da arquitectura a utilizar, foi necessária a comparação entre os três métodos. As métricas usadas para efectuar esta análise foram as seguintes: tempo de transacção, organização/eficácia. O tempo de transacção consistirá no tempo que passa desde a inserção de informação no sistema, até à integração dessa informação no outro sistema.

A organização ou eficácia corresponderia a toda a estruturação da arquitectura, assim como a facilidade por parte de um utilizador mais experiente, de compreender os ficheiros a integrar em ambos os sistemas. Uma maior eficiência neste caso, fará com que a informação esteja muito melhor organizada, em comparação com casos de pouco eficiência.

Em termos de tempo de transacção, poderá ser admitido que a solução de acesso directo à base de dados é algo superior às outras soluções, visto que não cria ficheiros, e não usa o EIS. Experiências de medição de tempo, indicaram que esta solução pode reduzir até 70% o tempo total da transacção em relação aos outros tipos de integração. Já no que toca ao factor da eficiência, em comparação com as outras duas formas de integração, este tipo de integração possui uma maior probabilidade de ocorrência de erros aquando da inserção da informação directamente na base de dados, visto não existir nenhum processo que valida a consistência dos dados antes da integração (ficaria a depender do programador), assim como a compatibilidade de tipos. Assim sendo, poderão ocorrer erros aquando da inserção ou da utilização dos dados num dos sistemas a integrar. No caso do XML, esta validação é executada, antes da integração propriamente dita.

No que toca à solução de *flat files*, apesar da informção estar menos organizada, como foi possível verificar nafigura 15, estes não sofrem a verificação do esquema XML. Logo são mais fracos no que toca à organização, mas por outro lado possuem uma maior velocidade de processamento. Apesar de tudo, a diferença do tempo de processamento não foi muito significativa em relação à arquitectura que visa a utilização de XML.

Por fim, a solução que visa a utilização de ficheiros XML tem a grande vantagem de fazer a verificação dos dados de cada mensagem, mas pode-se tornar algo lenta. Isto, devido ao facto de estarem interligados muitos processos na arquitectura que visa esse tipo de integração. No entanto este tipo de integração torna o sistema independente do modelo de dados, tornado possível a alteração do modelo de dados do WMS, sem que fosse necessário alterar o processo de integração. Na figura 19 é possível visualizar um pequeno diagrama que permite analisar comparativamente os três tipos de arquitecturas:

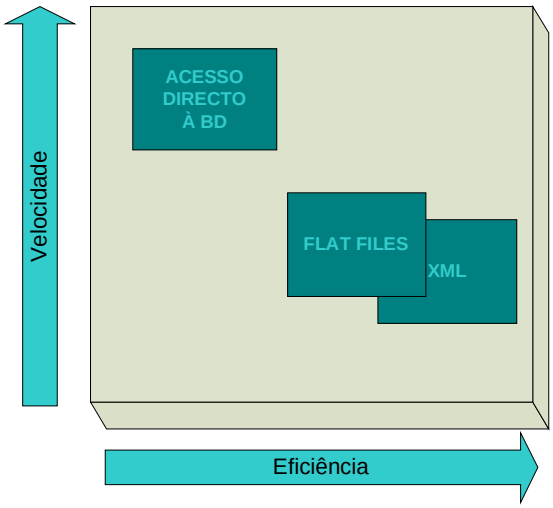


Figura 19 – Análise comparativa da velocidade e da eficiência entre os três tipos de arquitectura

Para proceder à análise foram efectuadas uma série de experiências com o objectivo de medir o tempo de escrita por parte das duas soluções (a solução que visava a utilização de *flat files* fora excluida das experiências[14]). Para isso, foi definido um modelo de medição que consiste no tempo que decorre desde a subscrição da mensagem da JMS até à criação do ficheiro XML na pasta local. Isto foi possível através da plataforma *SeeBeyond* Egate que permite a gestão e monitorização de todos os componentes. Assim sendo, para a mesma mensagem temos:

1. Acesso directo à base de dados[15] durou cerca de 14 segundos após a activação do componente de subscrição.

2. Solução utilizando ficheiros XML durou cerca de 21 segundos após activação do componente de subscrição.

Através da análise comprovou-se que o acesso directo é relativamente mais rápido. Por outro lado, a utilização de XML é totalmente indicada para ser realizada a integração, visto que reduz os custos de implementação e diminui o "uso do papel" por parte das empresas para integrar todos os seus dados. Diminuindo este último factor, obviamente que a empresa passa a ter uma gestão muito mais eficaz (Fernando Zaidan, 2007)16.

[14] Isto devido ao facto de esta obrigar ao desenvolvimento de novas componentes algo complexas, para o tempo disponível.

[15] Estes testes foram efectuados com a ajuda de tabelas auxiliares criadas manualmente, para q ue não fosse afectado o ambiente de testes, caso houvesse inconsistência de dados.

[16] Sistemasde informação empresariais: integração de sistemas interorganizacionais via XML,

http://www.administradores.com.br/producao_academica/sistemas_de_informacoes_empre sariais_integracao_de _sistemas_interorganizacionais_via_xml/152/

Visto que no âmbito deste projecto o factor mais importante seria a consistência dos dados para minimizar os erros, e que a velocidade de integração poderá ser do tipo near real-time17, como foi negociado com o cliente, optou-se pela utilização de ficheiros XML. Tudo isto, devido à sua vantagem em termos de organização e eficiência, face às outras possibilidades. Além disso, a utilização deste *standard* possui também uma fácil manipulação de dados, visto existirem muitas ferramentas que processam documentos desse tipo e muitas linguagens de programação já suportam nativamente o *standard* XML.

4.2 Oracle Retail Integration Bus

Nesta secção será descrita a parte mais funcional do tratamento de mensagens por parte do ORIB.

Mensagens

Cada mensagem do ORIB pertence a uma determinada família de mensagens, sendo que cada família possui informação acerca de uma componente de negócio: Artigos, Ordens de Compra, Transferências, etc. Para cada família existirá um ou mais tipos de mensagens consoante a operação que foi efectuada. Assim, para apagar um artigo, ter-se-à uma mensagem do tipo *ItemDel*, enquanto que para a sua criação, a mensagem será *ItemCre*. Na figura 20 apresenta-se um exemplo de uma mensagem:

```
<RibMessages>
<publishetname>etItemLocFromRMS</publishetname>
<ribMessage> <family>itemloc</family>
<type>ItemLocDel</type> <ribmessageID>2008-05-15
16:27:29.764 BST</ribmessageID> <publishTime>2008-05-
15 16:27:29.764 BST</publishTime> <messageDa ta>
&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-
8&quot;?&gt;
&lt;ItemLocRef&gt;
      &lt;item&gt;item1&lt;/item&gt;
      &lt;ItemLocPhysRef&gt;
        &lt;physical_loc&gt;1&lt;/physical_loc&gt;
               &lt;ItemLocVirtRef&gt;
                      &lt;loc&gt;111&lt;/loc&gt;
               &lt;/ItemLocVirtRef&gt;
       &lt;/ItemLocPhysRef&gt;
&lt;/ItemLocRef&gt;
</messageData>
</ribMessage>
</RibMessages>
```

Figura 20 – Exemplo de uma mensagem XML na JMS

[17] Neste caso, em média significa cerca de quinze segundos.

Como pode ser verificado, cada mensagem encontra-se no formato XML. Esta mensagem refere-se então ao fluxo da localização dos artigos (*ItemLoc*) e é referente à remoção (*ItemLocDel*). O tópico em que está publicada denomina-se por *etItemLocFromRMS*.

Ciclo da Mensagem

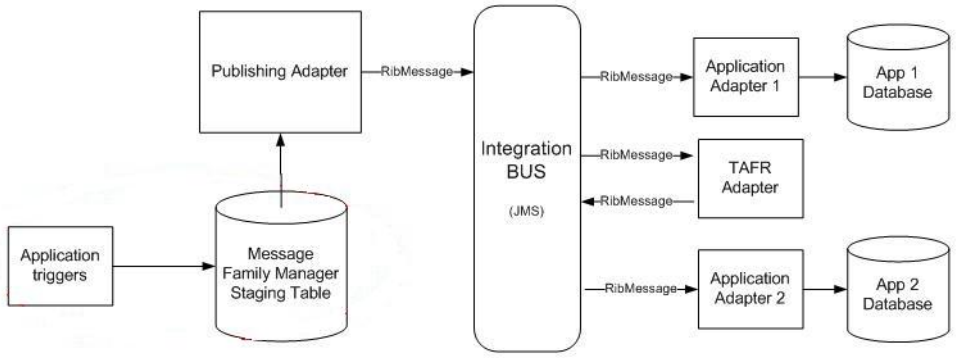


Figura 21 – Ciclo da Mensagem

De acordo com a figura 21, cada mensagem passa sobre um determinado número de componentes desenvolvidos na plataforma de integração:

- *Triggers* - Têm o objectivo de detectar alterações em determinadas tabelas do lado do sistema publicador e colocar todas essas operações numa tabela auxiliar normalmente designada por *nome_do_fluxo*MFQUEUE com a respectiva identificação do objecto em questão. Este processo é normalmente é efectuado através da execução de um procedimento designado por ADDTOQ.

- Adaptador de publicação - Processos que poderão tar sempre a correr e que chamarão uma função designada por GETNXT, com o objectivo de verificar a existência de dados na tabela auxiliar do fluxo correspondente ao *package* de onde é chamada a função. Esta função retorna o tipo Oracle da mensagem, que será depois convertido em XML e colocado no respectivo tópico existente na fila de espera. Caso a integração seja feita, por exemplo, via ficheiros em vez de uma ligação à base de dados, este adaptador irá invocar uma componente designada por *collaboration rule*, que é desenvolvida em java e executa toda a parte de criação e manipulação de ficheiros.

Porém poderá existir um ponto de conexão[18] diferente, por exemplo para ficheiros. Assim sendo, em vez de ser detectada a existência de novos dados na tabela auxiliar, é verificada a existência de ficheiros XML ou de *Flat Files* numa determinada pasta, prontos a serem publicados na fila de espera. Cada adaptador terá então um ponto de conexão e um tópico onde as suas mensagens são publicadas ou subscritas.

- Adaptador TAFR - O objectivo deste tipo de adaptadores é transformar tipos de mensagens, através da invocação de classes Java, e voltá-las a colocar noutro tópico, para que possam ser subscritas por outros sistemas.

[18] Pontos referentes a cada componente, com o objectivo de por exemplo abrir uma sessão de acesso a uma base de dados, à escrita/leitura de ficheiros ou à escrita/leitura dos tópicos existentes na JMS.

- Adaptador de subscrição - Estes processos poderão também estar constantemente a correr, subscrevendo de um determinado tópico uma mensagem, transformando-a depois num tipo Oracle para que se possa chamar um método designado por *Consume* existente no package referente à família a que pertence a mensagem. Os campos serão depois validados e introduzidos na tabela final ou em tabelas intermediárias para que outros sistemas possam integrar a informação. Tal como o adaptador de publicação, se este fizer a integração através de ficheiros, chamarà também uma determinada regra, que será desenvolvida em Java.

Transformadores de Mensagens

Estes transformadores de mensagens são desenvolvidos em código Java e são chamados por um adaptador de transformação (TAFR – figura 22), cujo o objectivo é subscrever a mensagem de um(a) tópico, executar o mapeamento sobre essa mensagem e finalmente, colocar a mensagem noutro tópico. Estes transformadores poderão, tipicamente, conter mais que um ponto de conexão, de maneira a permitir que seja possível uma ligação por exemplo a um tópico da JMS, a uma pasta em que serão colocados os ficheiros, ou então poderá ligar-se a base de dados. Esta multiplicidade de ligações servirão muitas vezes para complementar os dados que são subscritos por este adaptador, que serão posteriormente publicados após a transformação, através de um outro ponto de conexão.

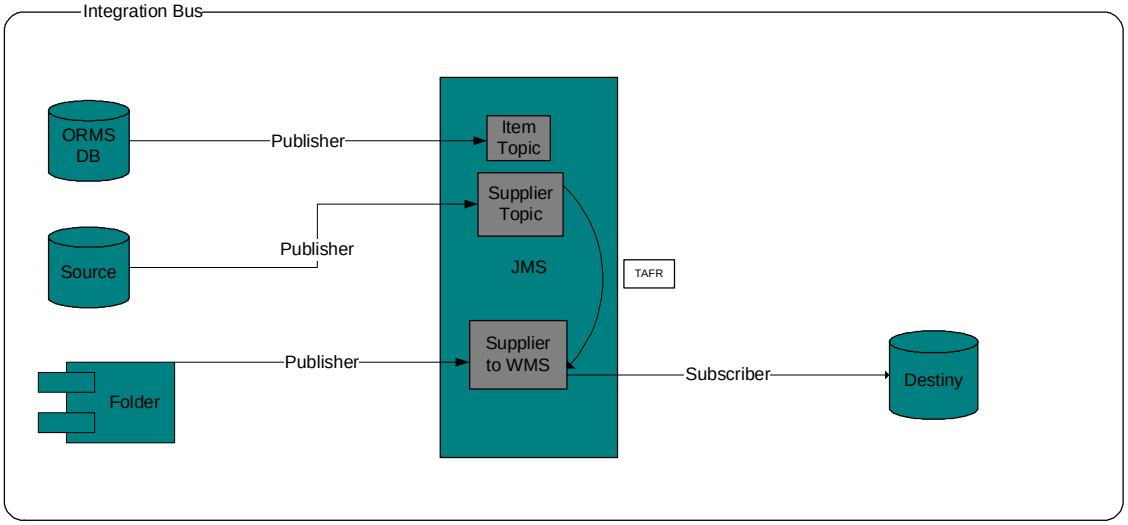


Figura 22 – Utilização de Transformadores

De seguida são demonstrados os três diferentes tipos dos pontos de conexão, de modo a que seja possível uma explicação mais clara. Como pode ser verificado, o tópico *SupplierToWMS* subscreve um adaptador de publicação, cujo ponto de conexão é do tipo de ficheiro, e subscreve também um adaptador transformador (TAFR). Este adaptador de transformação vai transformar a mensagem existente no tópico do *Supplier*, para posteriormente publicar a mensagem já transformada no tópico *SupplierToWMS*. Por fim este tópico será subscrito

directamente por procedimentos existentes na base de dados do ORMS (o caso deste projecto).

4.3 Manhattan Enterprise Integration Services

Arquitectura e Vista Geral

Em termos de funcionalidade, o EIS terá o objectivo de através das mensagens XML existentes numa determinada pasta do servidor, fazer a sua integração numas tabelas intermédias para verificação de erros e, posteriormente fazer a inserção em tabelas definitivas. O único problema que poderia afectar o desempenho do EIS seria a grande quantidade de mensagens a ser integrada, mas isso não será efectivamente um problema, visto que o ORMS gera mensagens com uma cadência controlada, e o EIS suporta facilmente a cadência gerada.

O EIS contém também uma fila de espera FIFO que disponibiliza espaço físico e informação para reencaminhamento das mensagens. O formato das mensagens poderá estar em texto ou em binário, e não existe limite de tamanho para cada uma delas. As mensagens são colocadas nesta fila, após o processo das mensagens XML, como poderá ser visto numa vista geral da arquitectura física o EIS (figura 23).

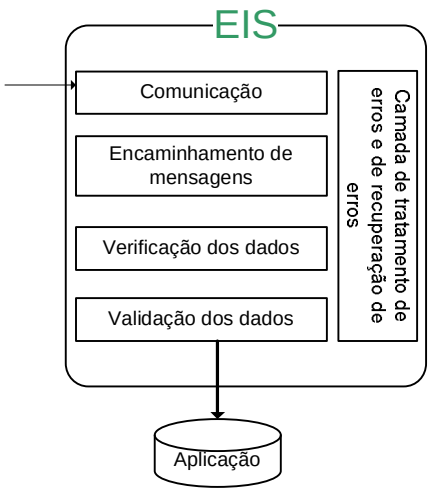


Figura 23 – Vista Geral do EIS

O EIS possui então as seguintes camadas:

- Comunicação: Camada que usa protocolos de comunicação (FTP, http, Sockets, etc.) para transferir ficheiros para o servidor onde se encontra a plataforma.
- Encaminhamento de Mensagem: É feito através de informação existente no cabeçalho do ficheiro XML.
- Verificação dos Dados: Através do esquema XML.
- Validação dos Dados: Verifica os tipos de dados, o seu comprimento.

- Camada de Recuperação: Permite a alteração do estado do sistema para que seja recebida uma mensagem que foi perdida no passado.

Processos/Operações

Operações Outbound

Para cada aplicação destino, é criado um serviço no EIS para comunicar em modo *Outbound*, com o respectivo protocolo e configurações. Este modo de comunicação, por seu lado encontra-se sempre num determinado estado: inicialização, espera ou envio (figura 24).

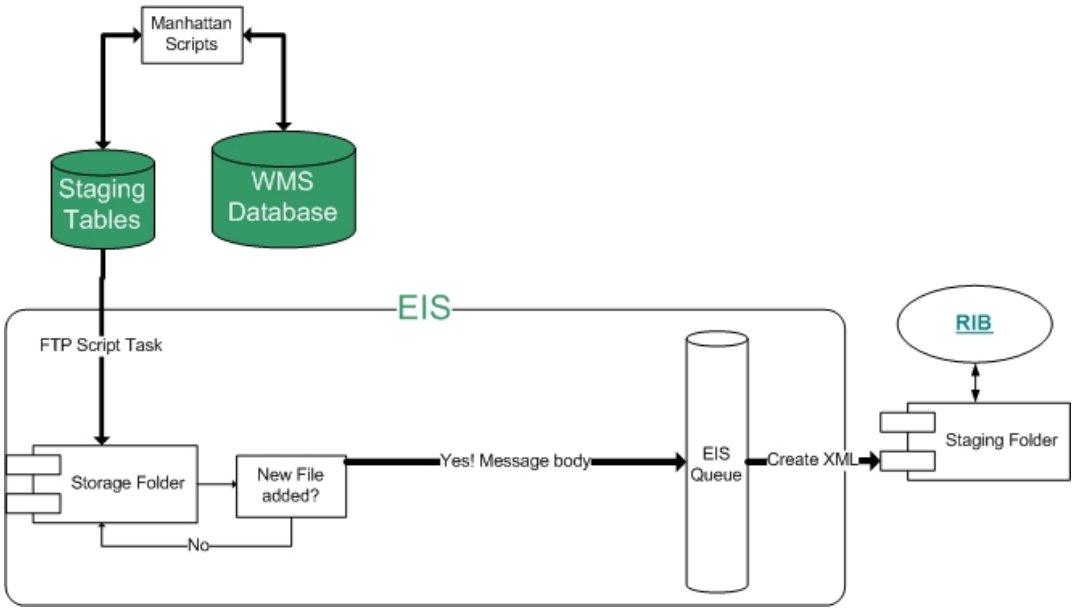


Figura 24 – Operações Outbound do WMS

- Inicialização: No início, o serviço lê as suas configurações para determinar as definições de comunicação da aplicação.
- Espera: Se não existirem mensagens na fila de espera para serem processadas, o serviço EIS vai para um estado de espera. Este poderá ser configurado para verificar periodicamente a existência de mensagens na fila de espera.
- Envio: Este processo é constituido por quatro passos principais:
    - Ler a mensagem da fila de espera: As mensagens com o estado *READY* são lidas da fila e processadas, uma de cada vez, por ordem decrescente de prioridade. Para mensagens com as mesmas prioridades, são sempre processadas primeiro as mais antigas.
    - Enviar a mensagem para o MWMS: A mensagem é então enviada para a aplicação destino, usando o protocolo pré-definido (no nosso caso o FTP).

- Espera pelo sinal *ACK* (*acknowledge*) por parte da aplicação destino: para garantir que a mensagem foi integrada com sucesso.
- Actualizar o estado da mensagem existente ainda na fila para: *Delivered*, *Failed* ou *Retry*.

Operações Inbound

Para cada aplicação de origem, um serviço do EIS será criado para comunicar no modo *Inbound* com o protocolo pré-determinado na configuração. O serviço *Inbound* está sempre num dos três estados seguintes: inicialização, espera e recepção (figura 25).

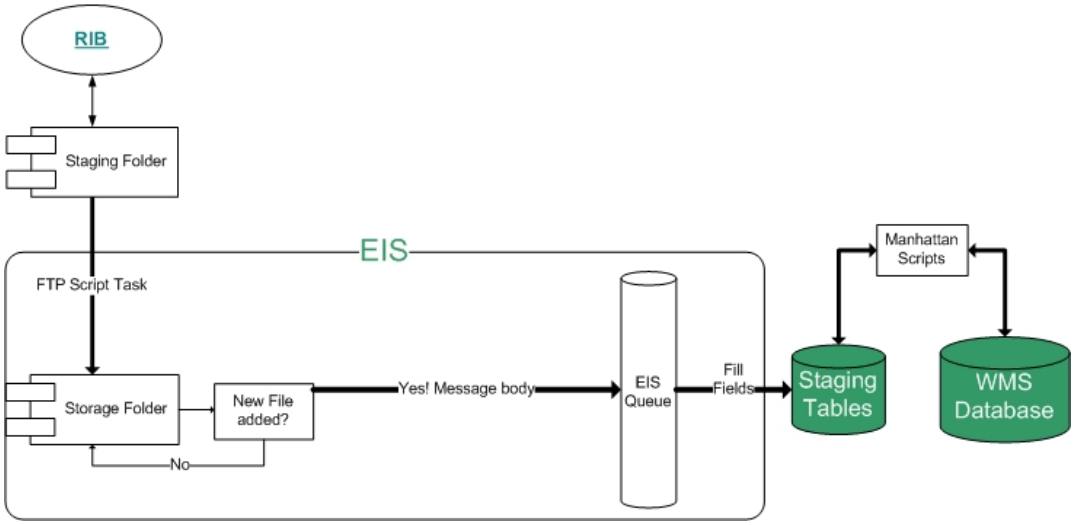


Figura 25 – Operações Inbound do WMS

- Inicialização: No início, o serviço lê as suas configurações para determinar as definições de comunicação da aplicação.
- Espera: Se não existirem mensagens na fila de espera para serem processadas, o serviço EIS vai para um estado de espera. Este poderá ser configurado para verificar periodicamente a existência de mensagens na fila de espera.
- Recepção: Este processo é constituido por três passos principais:
  - Ler a mensagem da aplicação de origem: Mensagens recebidas numa pasta através do protocolo pré-definido.
  - Guardar a mensagem na fila de espera do EIS: As mensagens guardadas na fila de espera têm inicialmente um estado *READY*.

- Emissão do sinal *ACK (acknowledge)* por parte do EIS: Para garantir que a mensagem foi recebida e guardada na fila de espera com sucesso.

5 Análise dos Fluxos Desenvolvidos

Como foi referido atrás, o ORIB possuí vários componentes para que seja possível uma integração robusta. Terá de existir um conjunto destes componentes para cada fluxo existente entre o MWMS e o ORMS (figura 26). De seguida serão ilustrados esses fluxos:

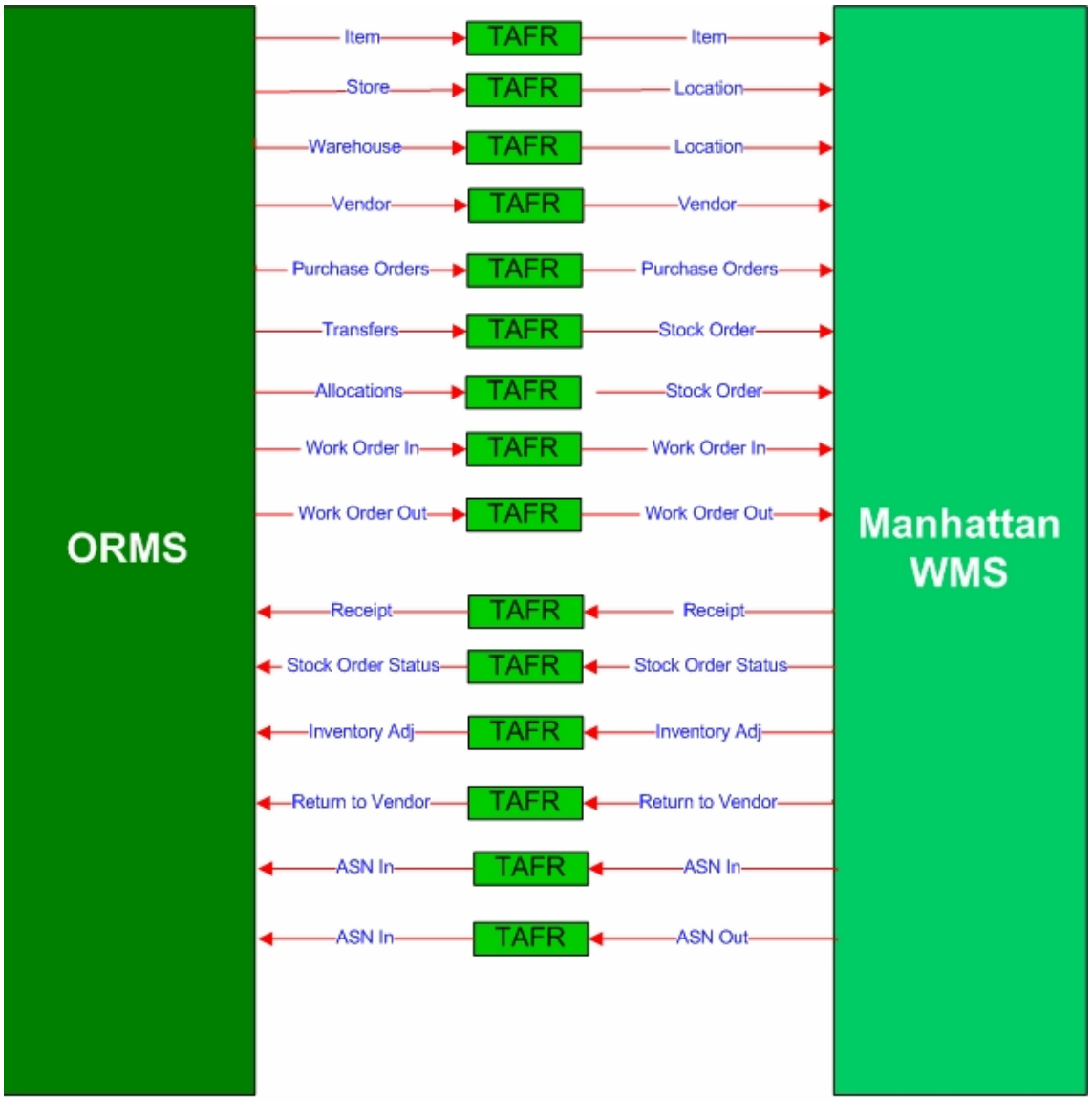


Figura 26 – Operações Inbound do WMS

Alguns destes fluxos, não foram realizados pelo estagiário, logo não estarão descritos no presente documento.

5.1 Artigos

Todas as operações (inserções, modificações e remoções) sobre os items, terão de ser enviadas para o MWMS. Como origem dessa informação, obviamente teremos o ORMS, como indica a figura 27.

Após uma análise cuidadosa do RIB e do EIS, conclui-se que o fluxo tal como se apresenta na figura 27 seria o mais adequado para a realização da integração visto ser necessária a transformação da mensagem entre os dois sistemas (componente 2):

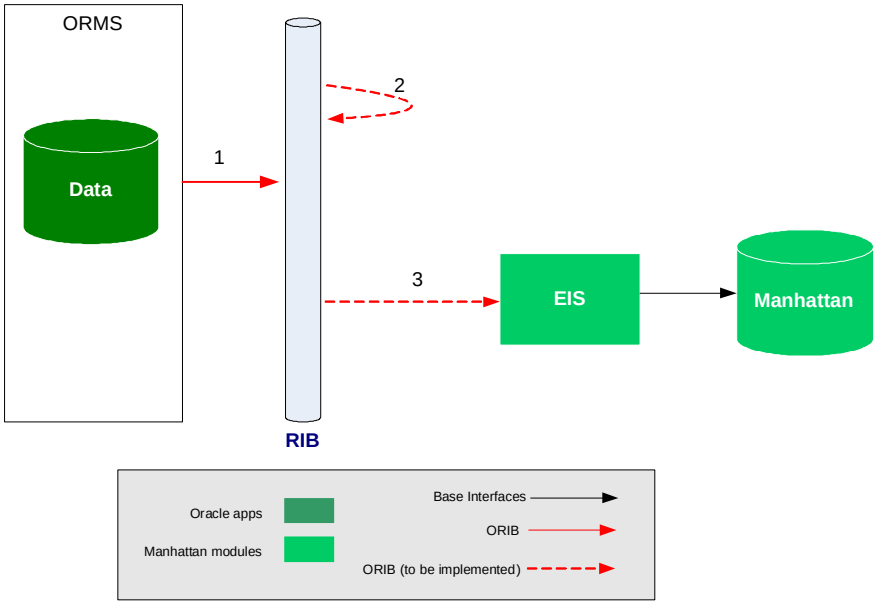


Figura 27 – Fluxo dos Artigos

Segundo a figura 26, ter-se-ão os seguintes prossupostos:

1. Os dados relativos a um artigo são publicados a partir do RMS, através de uma interface já implementada pela Oracle.
2. A mensagem será depois subscrita por um adaptador transformador que terá de ser implementado, e posteriormente será colocada no ORIB.
3. Um novo adaptador de subscrição será criado para subscrever a mensagem do ORIB e inseri-la no sistema de ficheiros do EIS.

Os objectos a serem criados, serão então os dois adaptadores, e os restantes componentes necessários para o seu bom funcionamento (pontos de conexão e tópicos para os quais haverá publicação/subscrição por parte destes dois objectos): ewItemsToMItemsTAFR, ewItemsToMWMS – Componentes 2 e 3 respectivamente na figura 27. Como foi indicado atrás, cada adaptador tem de ter associado a si uma regra, que deverá ser desenvolvida em JAVA.

Assim sendo, a regra do TAFR (2), transforma os seus dados de acordo com o modelo específico do Manhattan, colocando novamente a mensagem na JMS

O componente de subscrição (3) para o Manhattan irá posteriormente colocar a mensagem num ficheiro XML, numa pasta local, para que depois seja transferido para o sistema de ficheiros do EIS, por FTP.

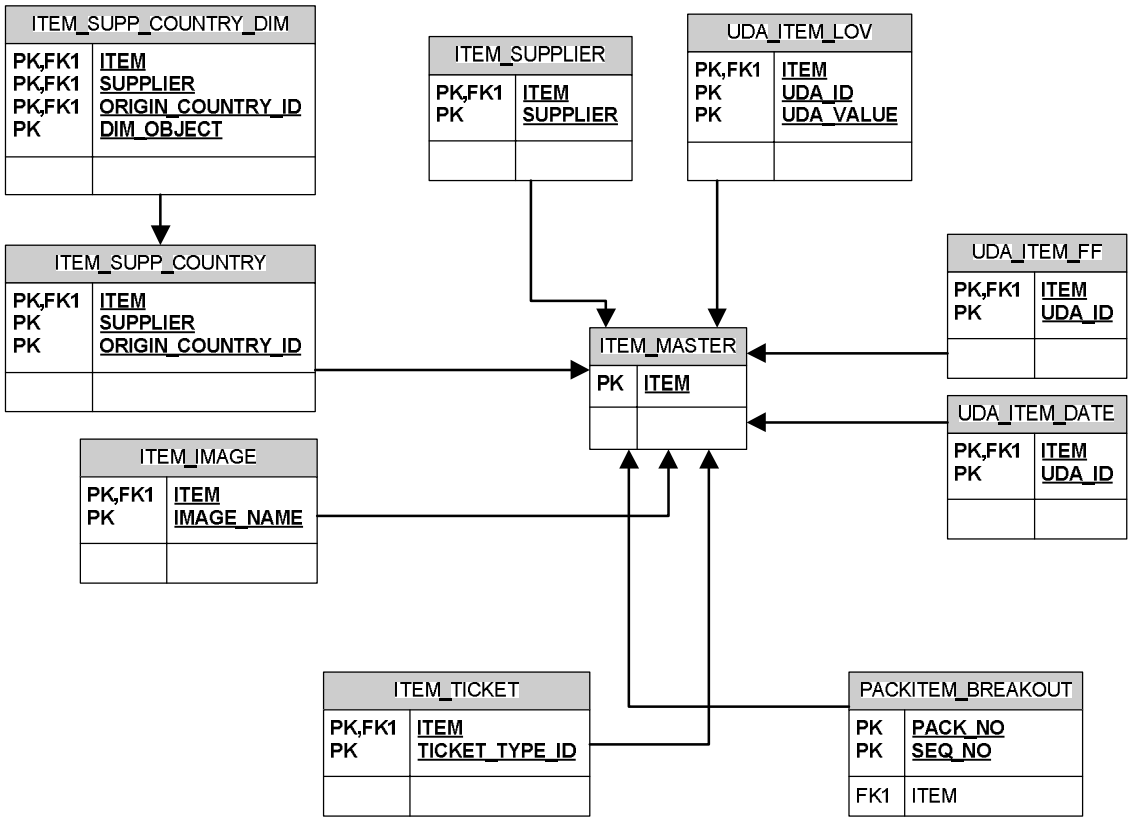


Figura 28 – Diagrama de Classes dos Artigos

Como pode ser verificado na figura 28, existe uma certa complexidade de dados na parte dos artigos, no lado do RMS. Existindo alguma alteração em alguma destas tabelas, será "disparado" um *trigger*, com o objectivo de inserir a alteração na seguinte tabela:

Attribute Name	Attribute Type	Null (Y/N)	Key (Y/N)	Field Description
SEQ_NO	NUMBER(15)	N	Y	The sequence in which the records was placed in the ITEM_MFQUEUE. Used to order the publication of messages to the RIB.
ITEM	VARCHAR2(25)	N	N	Uniquely identifies an item.
SUPPLIER	NUMBER(10)	Y	N	Item and supplier uniquely identifies an tem_supplier;
COUNTRY_ID	VARCHAR2(3)	Y	N	Item, supplier and origin country_id uniquely identifies an Item_supp_country.
DIM_OBJECT	VARCHAR2(6)	Y	N	Item, supplier, origin_country_id and dim_object uniquely identifies an Item_supp_country_dim.
REF_ITEM	VARCHAR2(25)	Y	N	Ref_item uniquely identifies an item's UPC.
PACK_COMP	VARCHAR2(25)	Y	N	Item and pack_comp uniquely identifies a packitem's item components.
IMAGE_NAME	VARCHAR2(120)	Y	N	Item and image_name uniquely identifies an item_image.
UDA_ID	NUMBER(5)	Y	N	Item and uda_id uniquely identifies an uda_item_ff or an uda_item_date.
UDA_VALUE	NUMBER(3)	Y	N	Item, uda_id and Uda_value uniquely identifies an Uda_item_lov.
MESSAGE_TYPE	VARCHAR2(15)	N	N	Describes the action in RMS that is causing the message to be published to the RIB.
FAMILY	VARCHAR2(30)	N	N	The functional area that this transaction belongs to, in this case, item.
CUSTOM_MESSAGE_TYPE	VARCHAR2(1)	Y	N	Not used by RMS

Attribute Name	Attribute Type	Null (Y/N)	Key (Y/N)	Field Description
PUB_STATUS	VARCHAR2(1)	N	N	Set to 'U'npublished upon insertion into the table. Set to 'H'ospital when a non-fatal error is encountered during the publication process.
TRANSACTION_NUMBER	NUMBER(10)	Y	N	Business transaction key.
TRANSACTION_TIME_STAMP	DATE	Y	N	Time of record creation.
TICKET_TYPE_ID	VARCHAR2(4)	Y	N	This column will be populated for item ticket messages. The item and ticket make up the primary key for these messages.
THREAD_NO	NUMBER(4)	N	N	The thread number on which the item will be published. Each item has one and only one thread number.
APPROVE_IND	VARCHAR2(1)	N	N	Indicates whether the status of the item has changed to Approved.

Tabela 2 – Tabela auxiliar dos artigos

O campo principal desta tabela auxiliary, será o MESSAGE_TYPE, pois terá como conteúdo o tipo de mensagem criada. Os restantes campos, serão chaves estrangeiras para a tabela que foi alterada. O restante processo será executado segundo a arquitectura descrita no capítulo anterior. De seguida serão apresentados os componentes desenvolvidos para este fluxo.

Componente	Nome	Componente Base?
Adaptador de Publicação	ewItemsFromRMS	Sim
Adaptador de Transformação	ewItemsToMItemsTAFR	Não
Adaptador de Subscrição	ewItemsToMWMS	Não

Regra de Transformação	crItemsToMItemsTAFR.java	Não
Regra de Subscrição	crItemsToMWMS.java	Não
Ficheiro de Configuração	component.xml	Sim, mas alterado posteriormente

Tabela 3 – Componentes para um determinado fluxo

O ficheiro de configuração *component.xml* está em anexo, e tem como objectivo a descrição de todos os adaptadores.

Nome do Trigger	Tipo de Trigger	Descrição
EC_TABLE_IEM_AIUDR	AFTER INSERT,UPDATE OR DELETE	Trigger to detect the creation, update or deletion of records in the ITEM_MASTER table.
EC_TABLE_ISP_AIUDR	AFTER INSERT,UPDATE OR DELETE	Trigger to detect the creation, update or deletion of records in the ITEM_SUPPLIER table.
EC_TABLE_ISC_AIUDR	AFTER INSERT,UPDATE OR DELETE	Trigger to detect the creation, update or deletion of records in the ITEM_SUPP_COUNTRY table.
EC_TABLE_ISD_AIUDR	AFTER INSERT,UPDATE OR DELETE	Trigger to detect the creation, update or deletion of records in the ITEM_SUPP_COUNTRY_DIM table.
EC_TABLE_PKS_AIUDR	AFTER INSERT,UPDATE OR DELETE	Trigger to detect the creation, update or deletion of records in the PACKITEM_BREAKOUT table.
EC_TABLE_PKS_IUDS	AFTER INSERT,UPDATE OR DELETE	Trigger to detect the creation, update or deletion of records in the PACKITEM_BREAKOUT table.

Nome do Trigger	Tipo de Trigger	Descrição
EC_TABLE_UIT_AIUDR	AFTER INSERT,UPDATE OR DELETE	Trigger to detect the creation, update or deletion of records in the UDA_ITEM_DATE table.
EC_TABLE_UIF_AIUDR	AFTER INSERT,UPDATE OR DELETE	Trigger to detect the creation, update or deletion of records in the UDA_ITEM_FF table.
EC_TABLE_UIL_AIUDR	AFTER INSERT,UPDATE OR DELETE	Trigger to detect the creation, update or deletion of records in the UDA_ITEM_LOV table.

Tabela 4 – *Triggers* para detectar alterações no fluxo dos artigos

São estes *triggers* na tabela 4 que irão detectar as alterações efectuadas nas tabelas de origem do ORMS. Na figura 29 é apresentado um exemplo de um algoritmo genérico para um *trigger*:

```
if DELETING OR INSERTING OR UPDATING then

      if addtoq(L_error_message,L_message_type,Keys (..)) = FALSE then
        raise PROGRAM_ERROR;
      end if;

end if;
```

Figura 29 – Algoritmo dos triggers

Basicamente, irá ser chamado o procedimento, já falado, designado por *addtoq* com o objectivo de preencher a MFQUEUE de cada fluxo.

A transformação irá ser feita com base no seguinte algoritmo (crItemsToMItemsTAFR.java):

```
// The transformMessage method is responsible for the message transformation and filtering.

public com.retek.rib.util.RibMessageInternal transformMessage (RibMessageInternal msg) throws
Exception
```

```
{


  // Init variables


  // Map all RibMessage envelope attributes from the In Message(msg) to the Out Message(MsgOut) – refer to the template


          If msg.Type in [all messages types for items] {  // process this message

            // Transform message

           …

           tempXML = "<!DOCTYPE VendorMaster SYSTEM \"" + rp.getDtdUrlDefault() + "VendorMaster.dtd\">";

            tempXML = tempXML + getPayloadOut();//getpayloadout() will map the message

          }


          // Use this line to add the tranformed RibMessage to the JMS

          MsgOut.setMessageData(tempXML);

          MsgOut.setCustomData(msg.getCustomData());

          getetdRibMsgsOut().addRibMessage(MsgOut.getRibMessage());

          …

}


public boolean executeBusinessRules() throws Exception

{

…

          RibMessagesWrapper out = new RibMessagesWrapper(getetdRibMsgsOut());
```

```
// instead of using the sendRibMessagesForAllTopics(out) method to publish the Messages into the respective topic, we will use the send(...) method of the tafrHelper object.

        if (out.countRibMessage() > 0) {

          tafrHelper.send(out);

        };

...

}
```

Figura 30 – Algoritmo de uma regra de um transformador

A mensagem será depois enviada para um determinado local, de acordo com a configuração do ponto de conexão para este adaptador. Neste caso, estará a apontar para o tópico *etItemsFromRIB*, existente na JMS. Este pressuposto poderá ser alterado nas configurações do adaptador (figura 31).

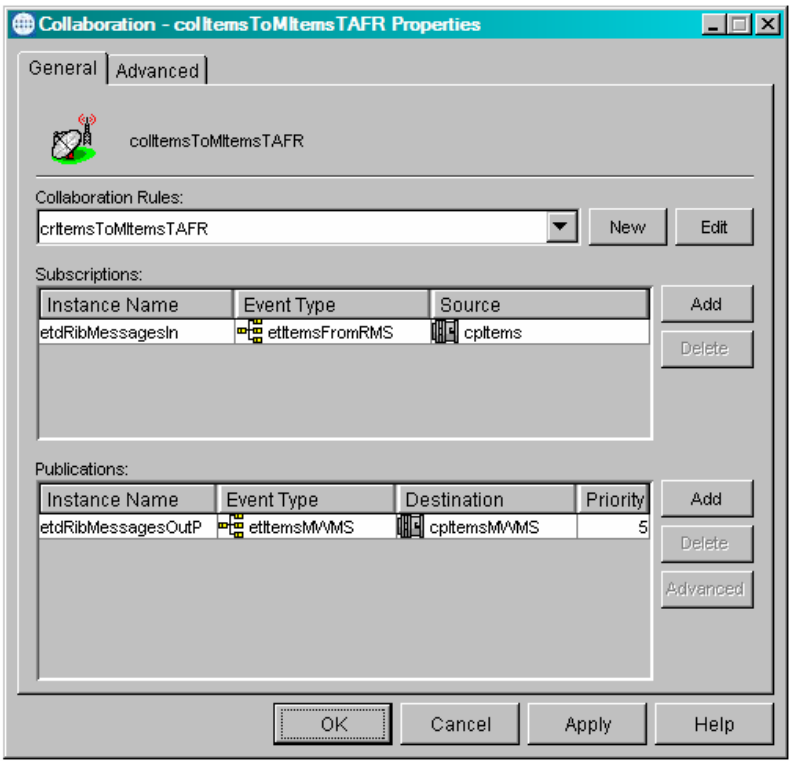


Figura 31 – Configuração do adaptador de transformação

É na configuração do adaptador, como é demonstrado na figura31, que será determinado o tópico da JMS para a qual irá ser publicada a mensagem produzida pelo próprio.

Em relação à subscrição por parte do MWMS, será chamado a regra crItemsTOMWMS.java pelo adaptador, que contém um ponto de conexão direccionado para um sistema de ficheiros (cpWriteToFile), como pode ser visível na figura 31. É através deste formulário que será criado e configurado o ponto de conexão respectivo.

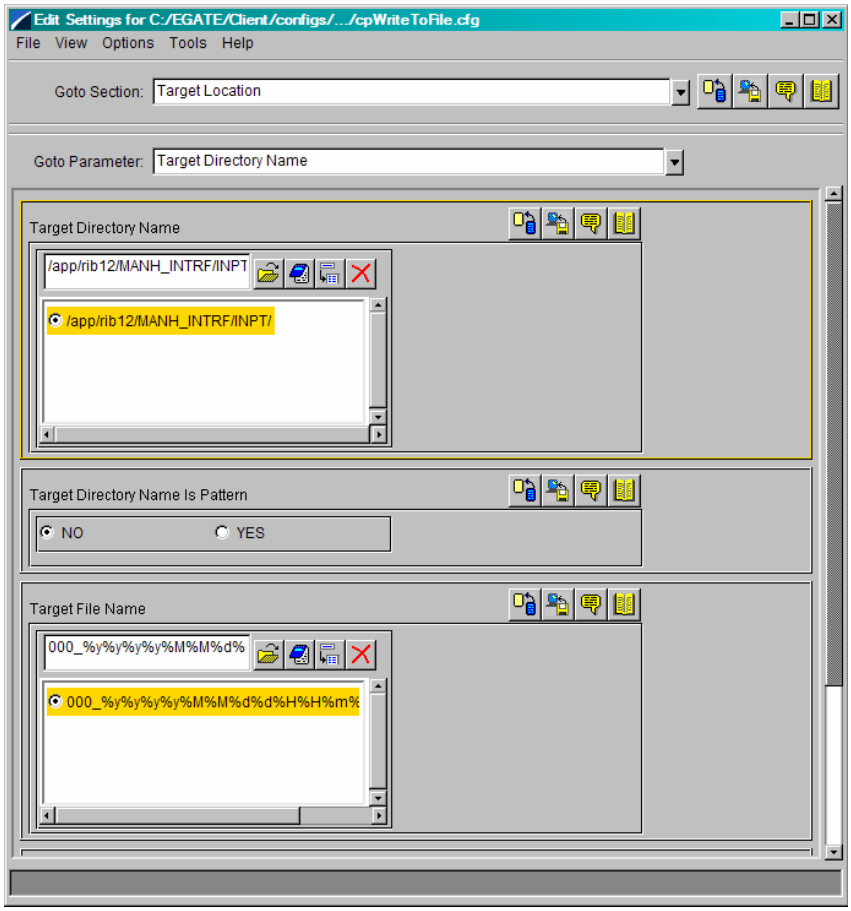


Figura 32 – Configuração do ponto de conexão do subscritor

5.2 Ordens de Compra

A ordem de compra é um documento comercial submetido pelo comprador ao vendedor indicando o tipo, quantidade e preço acordado para os produtos ou serviços que serão prestados.

O ORMS publica uma ordem de compra com vários detalhes (figura 33). Cada detalhe, é um artigo numa determinada quantidade.

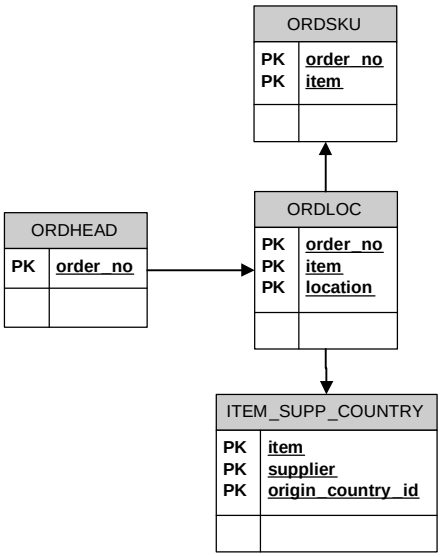


Figura 33 – Diagrama de Classes das ordens de compra

Foi utilizado o seguinte fluxo para integração das ordens de compra entreo o ORMS e o MWMS (figura 34)

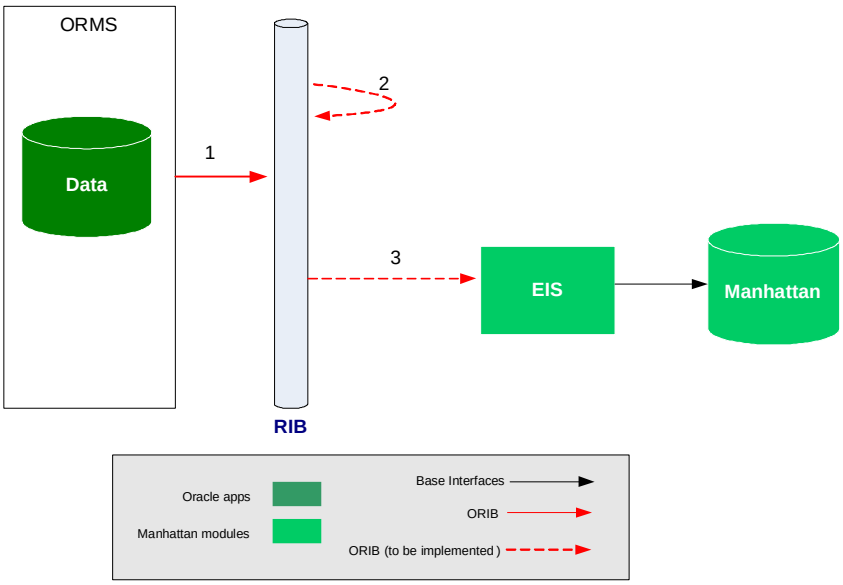


Figura 34 – Fluxo das Ordens de Compra

1. A ordem de compra é publicada do ORMS para o ORIB por interfaces já existentes.
2. A mensagem será transformada pelo TAFR.

3. Um novo subscritor será criado com o objectivo de obter a mensagem da JMS e colocá-la no sistema de ficheiros do EIS.

No que toca à publicação por parte do RMS, será necessária a seguinte configuração no ficheiro *component.xml* (figura 35).

```
<eway name="ewOrderFromRMS">
    <collaboration name="colOrderFromRMS">
        <adaptorComponent>

<class>com.retek.rib.collab.general.OracleObjectPublisherComponentImpl</class>
            <messageFamily name="Order">
<translatorClass>com.retek.rib.pubtrans.OrderObjectTranslator</translatorClass>
                <storedProc>
                    <signature>{call
RMSMFM_ORDER.GETNXT(?,?,?,?,?,?,?,?)}</signature>
                </storedProc>
            </messageFamily>
        </adaptorComponent>
    </collaboration>
</eway>
```

Figura 35 – Exemplo do component.xml

Como já foi referido, no caso dos publicadores, é necessária a referência de uma classe (a ser utilizada na construção da mensagem) com o conteúdo das tabelas de um determinado fluxo (neste caso, ordens de compras). Este processo será executado sobre o tipo *Oracle* devolvido pelo procedimento existente na base de dados e desenvolvido em PL/SQL, designado por *getnxt*. A servir de base a este processo, existirá uma tabela designada por *order_mfqueue* com o seguinte aspecto:

Attribute Name	Attribute Type	Null (Y/N)	Key (Y/N)	Field Description
SEQ_NO	NUMBER(15)	No		Sequence Number
ORDER_NO	NUMBER(8)	Yes		Purchase Order Number
MESSAGE_TYPE	VARCHAR2(15)	Yes		Message Type
FAMILY	VARCHAR2(30)	Yes		Message Family
PUB_STATUS	VARCHAR2(1)	Yes		Message Status
TRANSACTION_TIME_STAMP	DATE	Yes		Current Date and Time

Tabela 5 – Tabela auxiliar para as ordens de compras

De seguida é apresentado o algoritmo do *getnxt*:

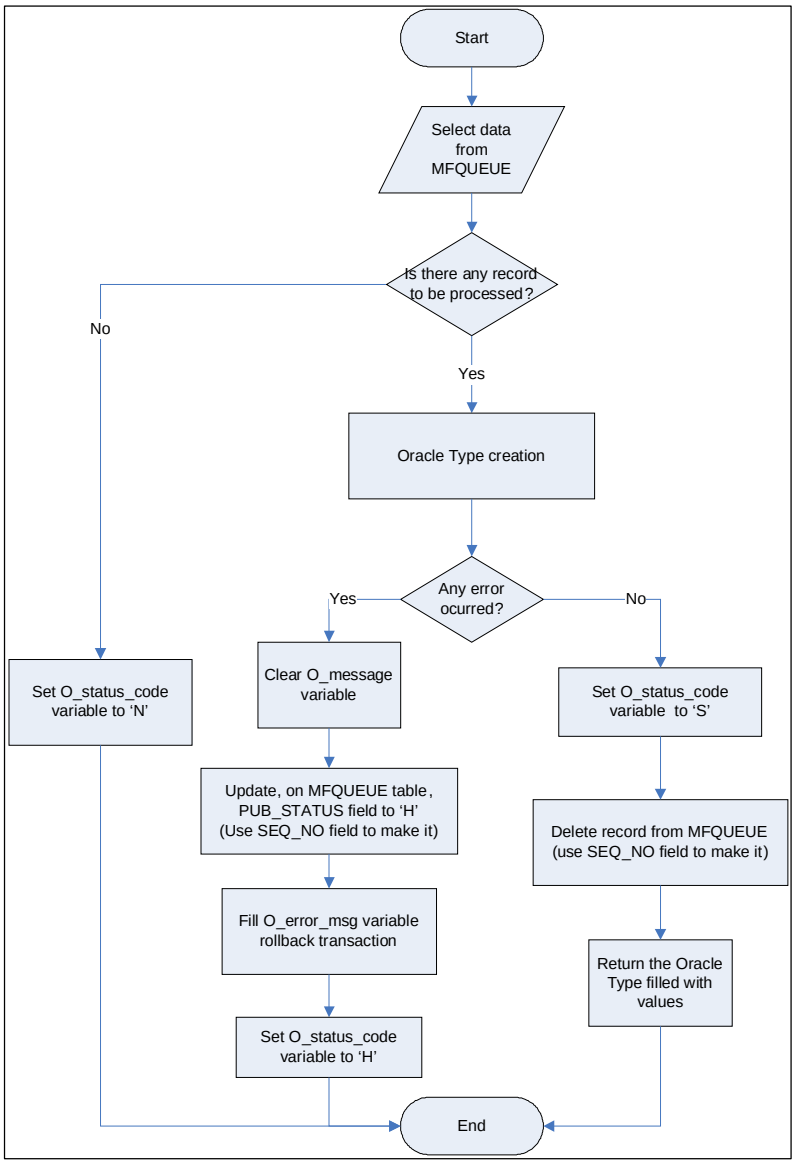


Figura 36 – Algoritmo do procedimento getnxt

Pelo que se pode retirar da figura 36, vemos que no processo é verificada a existência de dados na tabela auxiliar, neste caso designada por *order_mfqueue*. Caso existam dados, é então criado um *Oracle Type*[19], de acordo com o tipo da mensagem especificada. Esta informação acerca do tipo da mensagem existe na tabela *mfqueue*. De seguida será atribuida toda a informação a integrar à estrutura criada e apagado o registo da tabela auxiliar. Caso exista algum tipo de erro, o estado de publicação na tabela auxiliar encontrar-se-à em 'H', indicando que os respectivos dados a integrar se encontram numa tabela que regista todos os erros, denominada por *rib_message*. Caso tudo corra como esperado, será então devolvida a

[19] Classe contruida em PL/SQL com o objectivo de poder ser usada para criar objectos com determinados atributos, como em todas as linguagens orientadas a objectos.

estrutura preenchida para a classe tradutora, desenvolvida em Java, para que seja criada e colocada na JMS a mensagem XML.

5.3 Ajuste de Inventário

Foi desenvolvido o seguinte fluxo para as mensagens relativas ao fluxo de ajuste de inventário (figura 37).

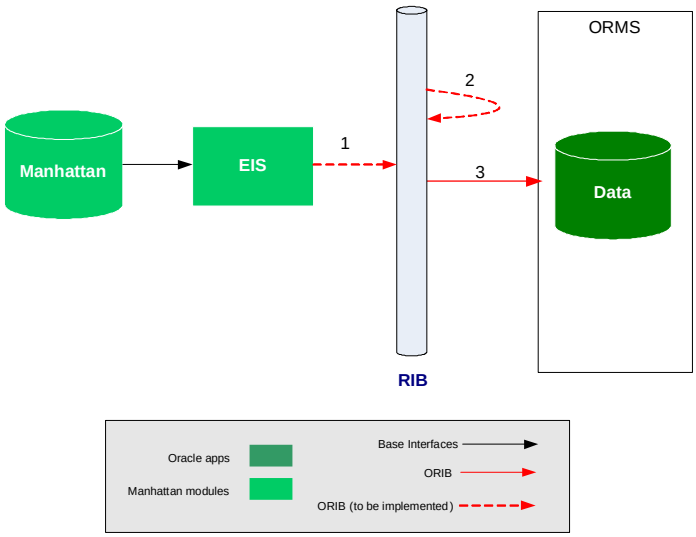


Figura 37 – Fluxo do ajuste de inventário

1. O EIS publica o ficheiro numa pasta local ao servidor, sendo posteriormente colocado numa pasta do ORIB. Um novo adaptador de publicação publica a mensagem no ORIB.

2. A mensagem é recebida por um novo adaptador de transformação e re-inserido no ORIB.

3. Através da utilização de um subscritor já existente, a informação é inserida no ORMS

Os objectos a serem criados serão, então, os dois adaptadores e os restantes objectos necessários ao seu bom funcionamento: (1) ewInvAdjustFromMWMS, (2) ewMInvAdjustToInvAdjustTAFR. Como foi indicado atrás, cada adaptador tem de ter associado a si uma regra que deverá ser desenvolvida em JAVA.

Assim sendo, a regra do TAFR (2) transforma os seus dados, de acordo com o modelo específico do ORMS, colocando novamente a mensagem na JMS.

Antes da mensgem ser transformada o componente de publicação (1) do Manhattan irá utilizar o ficheiro XML publicado pelo EIS, criar uma mensagem e, para além disso, colocá-lo na JMS/ORIB.

Na tabela 6 serão apresentados os componentes desenvolvidos para este fluxo.

Componente	Nome	Componente Base?
Adaptador de Publicação	ewInvAdjustFromMWMS	Não
Adaptador de Transformação	ewInvAdjustToMInvAdjustTAFR	Não
Adaptador de Subscrição	ewInvAdjustToMWMS	Sim
Regra de Transformação	crInvAdjustToMInvAdjustTAFR.java	Não
Regra de Publicação	crInvAdjustFromMWMS.java	Não
Ficheiro de Configuração	component.xml	Sim, mas alterado posteriormente

Tabela 6 – Componentes a desenvolver no fluxo de ajuste de inventário

O ficheiro de configuração *component.xml* está em anexo, e tem como objectivo a descrição de todos os adaptadores.

O adaptador de publicação terá a configuração que poderá ser vista na figura 38:

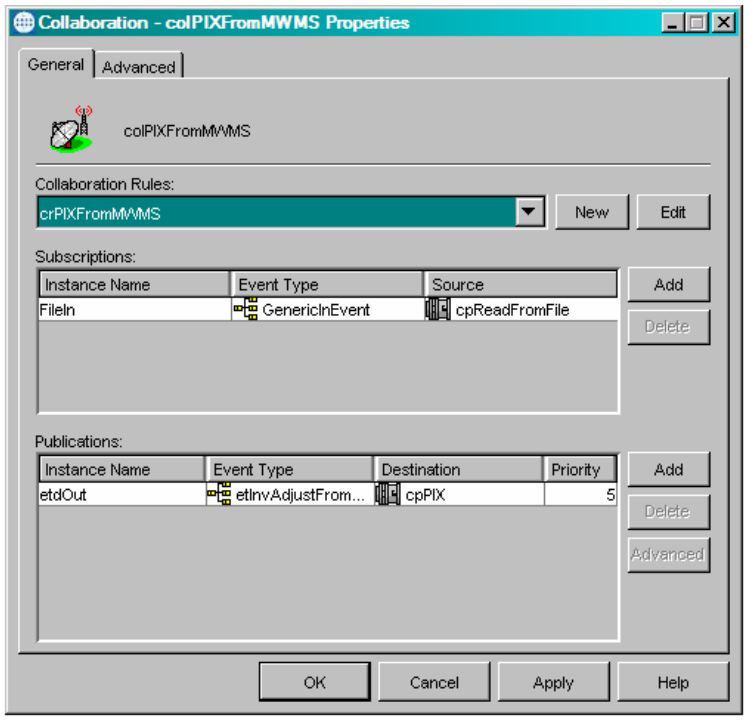


Figura 38 – Configuração do adaptador de publicação

Este adaptador irá utilizar o ponto de conexão *cpReadFromFile*, que está configurado para verificar a existência de ficheiros numa determinada pasta. Caso existam, serão utilizados na regra que irá publicar a mensagem no ORIB, no respectivo tópico designado por: *etInvAdjustFromMWMS*.

Esta regra, desenvolvida na tecnologia JAVA, abre o ficheiro XML e transforma-o numa mensagem que será aceite pelo ORIB, introduzindo-a na fila de espera, posteriormente.

O adaptador de transformação terá a seguinte configuração:

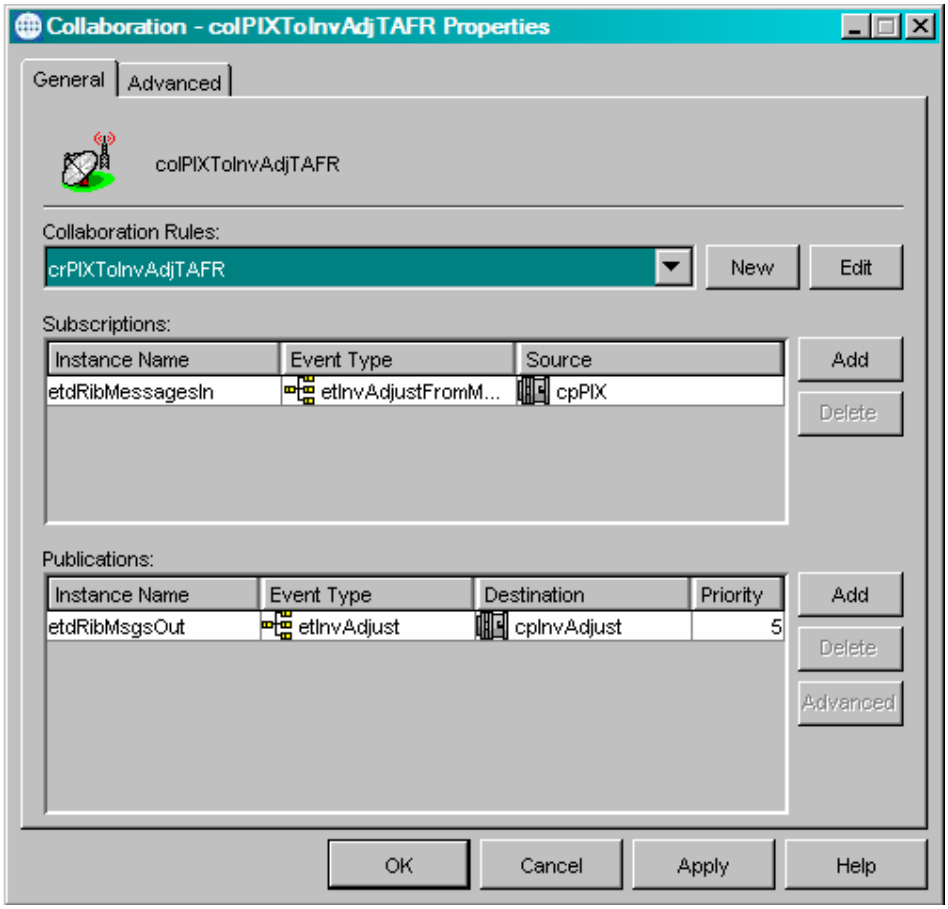


Figura 39 – Configuração do adaptador de transformação

Este adaptador irá subscrever do tópico *etInvAdjustFromMWMS*, transformar a mensagem no formato do MWMS, e depois inserir a mensagem no tópico *etInvAdjust*. Para tal operação, ter-se-à a regra da transformação com o típico algoritmo de transformação, em que subscreve a mensagem no tópico especificado, mapeia a mensagem de acordo com os dados do ORMS, e publica a mensagem num outro tópico.

Os pontos de conexão em questão, como ligarão a um tópico da JMS, irão conter o endereço IP e a porta, que identificará a sua localização (figura 40).

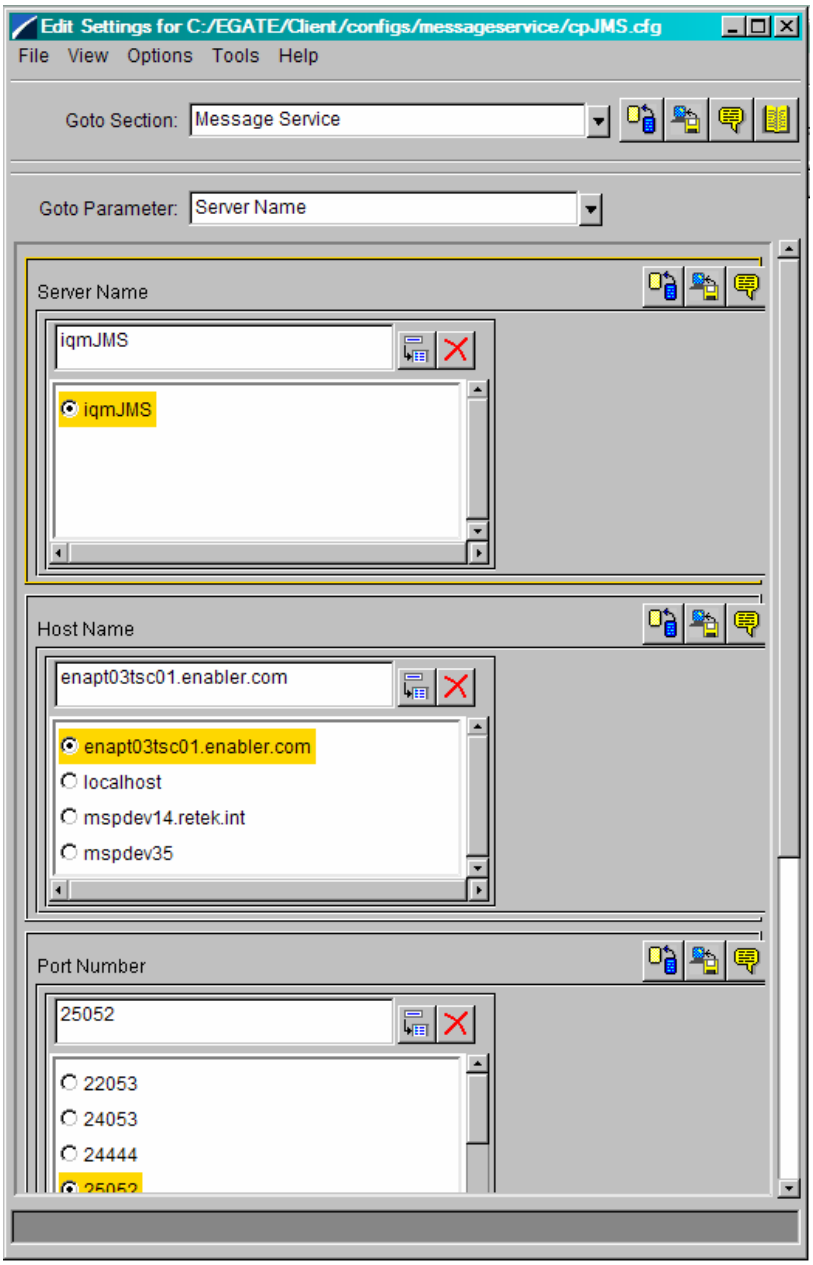


Figura 40 – Ponto de Conexão para a JMS

Após todo este processo, através de interfaces de subscrição de mensagens, a mensagem já transformada irá ser inserida automaticamente no sistema.

Para que tal facto se verifique, a configuração deste adaptador de subscrição deverá existir num ficheiro designado por *component.xml*. A configuração deste adaptador apresenta-se na figura 41.

```
<eway name="ewInvAdjustToRMS">
    <collaboration name="colInvAdjustToRMS">
        <adaptorComponent>

<class>com.retek.rib.collab.general.OracleObjectSubscriberComponentImpl</class>
            <messageFamily name="InvAdjust">
                <storedProc>
                    <signature>{call
RMSSUB_INVADJUST.CONSUME(?,?,?,?)}</signature>
                </storedProc>
                <messageType name="INVADJUSTCRE">

<oracleObject>RIB_INVADJUSTDESC_REC</oracleObject>
                </messageType>
            </messageFamily>
        </adaptorComponent>
    </collaboration>
</eway>
```

Figura 41 – Configuração do adaptador de subscrição do fluxo ajuste de inventário

Nesta configuração deverá ser indicada a classe que implementa todos os objectos de subscrição para ser possível o acesso à JMS e consequentemente, se poder obter a mensagem. Por outro lado, a regra usada neste adaptador será uma regra base (*default)* do sistema, pois o acesso será feito ao ORMS. De seguida deverá ser especificado o package PL/SQL que deverá ser chamado, os tipos de mensagens existentes, e finalmente os *Oracle Types* utilizados no package. Por outro lado, o ponto de conexão usado será também o mesmo para todos os fluxos que comuniquem com o ORMS. A sua designação é *cpToAndFromRMS.*

O algoritmo do procedimento chamado apresenta-se na figura 42.

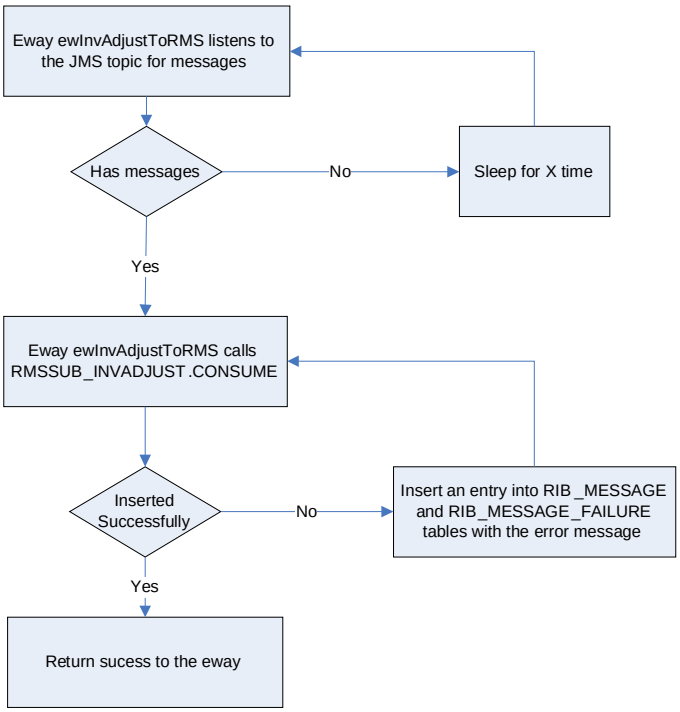


Figura 42 – Algoritmo do método consume

Assim sendo, o adaptador chamará o método da base de dados, e sempre que este detectar a existência de mensagens na JMS, transforma a mensagem num tipo *Oracle*, para posteriormente inseri-la nas respectivas tabelas do ORMS. Caso existam falhas na inserção dos dados nas tabelas, estas serão inseridas em tabelas auxiliares (*ORIB Hospital*) para processamento de erros.

Esta tabela auxiliar designada na plataforma de integração por *ORIB Hospital*, conterá todas as mensagens cuja execução do fluxo falhou para posteriormente serem retentadas a publicação ou subscrição em falta, com o intuíto de serem integradas no sistema.

6 Testes e Resultados

Em relação aos testes, foi criado um documento para cada tipo de fluxo desenvolvido, com o objectivo de facilitar a execução dos testes unitários. Este documento indica todos os pormenores a serem executados durante e aquando do fim do desenvolvimento de todos os componentes. Caso todos os componentes executem de acordo com o especificado no dito documento, então poder-se-à proceder à instalação na máquina do cliente de todos os constituintes de cada um dos fluxos.

Estes testes unitários seguirão tipicamente os passos listados:

1. Inicialização de todos os adaptadores;
2. Inserção/Alteração/Remoção de um objecto relativo a um determinado fluxo;
3. Verificar a execução dos *triggers*;
4. Verificar a mensagem depois da publicação do adaptador de publicação;
5. Verificar a mensagem após a sua transformação;
6. Verificar a criação de ficheiros na pasta intermédia e sua consistência;
7. Finalizar o processo (tarefa a executar pelos consultores da Manhattan e pelo *stream leader Manhattan* pertencente à ENABLER/WIPRO – Eng. Ricardo Nogueira).

Tipicamente, como primeiro passo ter-se-à a ligação do adaptador de publicação e respectiva verificação de erros. Estes poderão ser causados devido à não existência de alguma das componentes, assim como à existência de erros em alguma delas. Para verificar o sucesso da inicialização deste tipo de adaptadores, foi usada a verificação do seu *log* (figura 43).

```
10:08:25.588 EWY  I  -136563008 (svcmain.cxx:104): ServiceMain(): Started
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property:
DataSource.class=oracle.jdbc.xa.client.OracleXADataSource
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadCo  nfiguration(): Loaded property: DataSource.DriverType=thin
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property:
DataSource.ServerName=enapt03tsc01.enabler.com
10:08:25.597 CFG  I  -3473085 60 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property: DataSource.PortNumber=1521
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property: DataSource.Data    baseName=RMS12
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property: DataSource.user_name=rms12
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfig  uration(): Loaded property: DataSource.password=05C820800000
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property: DataSource.timeout=300
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx :1442): EBobConnectorProperties.loadConfiguration(): Loaded property: connector.type=DB
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property:
connector.class=com.stc.eways.jdbcx.DbConnector
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property: connector.transaction_mode=Automatic
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfigurat  ion(): Loaded property:
connector.Connection_Establishment_Mode=Automatic
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property:
connector.Connection_Inactivity_Timeout=0
10:08:25.597 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property:
connector.Connection_Verification_Interval=300
10:08:25.603 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loa  ded property: General_Settings.Connection_Type=Topic
10:08:25.603 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property: General_Settings.Transaction_Type=XA  -
compliant
10:08:25.603 CFG  I  -347308560 (jav a_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property:
General_Settings.Delivery_Mode=Persistent
10:08:25.603 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property:
General_Settings.Maximum_Number_of_Bytes_to_read=16777216
10:08:25.603 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property:
General_Settings.Default_Outgoing_Message_Type=Text
10:08:25.604 CFG  I  -347308560 ( java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property:
General_Settings.Factory_Class_Name=com.stc.common.collabService.SBYNJMSFactory
10:08:25.604 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.load  Configuration(): Loaded property: Message_Service.Server_Name=iqmJMS
10:08:25.604 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property:
Message_Service.Host_Name=enapt03tsc01.enabler.com
10:08:25.604 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded property: Message_Service.Port_Number=25052
10:08:25.604 CFG  I  -347308560 (java_extensions.cxx:1442): EBobConnectorProperties.loadConfiguration(): Loaded prope  rty:
Message_Service.Maximum_Message_Cache_Size=1
10:08:25.866 CDB  I  -347308560 (java_extensions.cxx:1442): SessionFactory.createSession(): Invoking class oracle.jdbc.xa.client.OracleXADataSource setter methods
10:08:25.878 CDB  I  -347308560 (java_extensi ons.cxx:1442): SessionFactory.createSession(): DatabaseName = RMS12
10:08:25.878 CDB  I  -347308560 (java_extensions.cxx:1442): SessionFactory.createSession(): DriverType = thin
10:08:25.878 CDB  I  -347308560 (java_extensions.cxx:1442): SessionFactory.creat  eSession(): PortNumber = 1521
10:08:25.878 CDB  I  -347308560 (java_extensions.cxx:1442): SessionFactory.createSession(): ServerName = enapt03tsc01.enabler.com
```

Figura 43 – Log de um adaptador

De maneira a testar um fluxo completo, por exemplo o fluxo das ordens de compras, será necessária a introdução ou alteração de uma ordem de compra no ambiente do ORMS. Assim sendo, foi efectuada uma alteração na ordem de compra 5016 (já existente), para que fossem testados os *triggers* de inserção na tabela auxiliar para tal fluxo (figura 44).

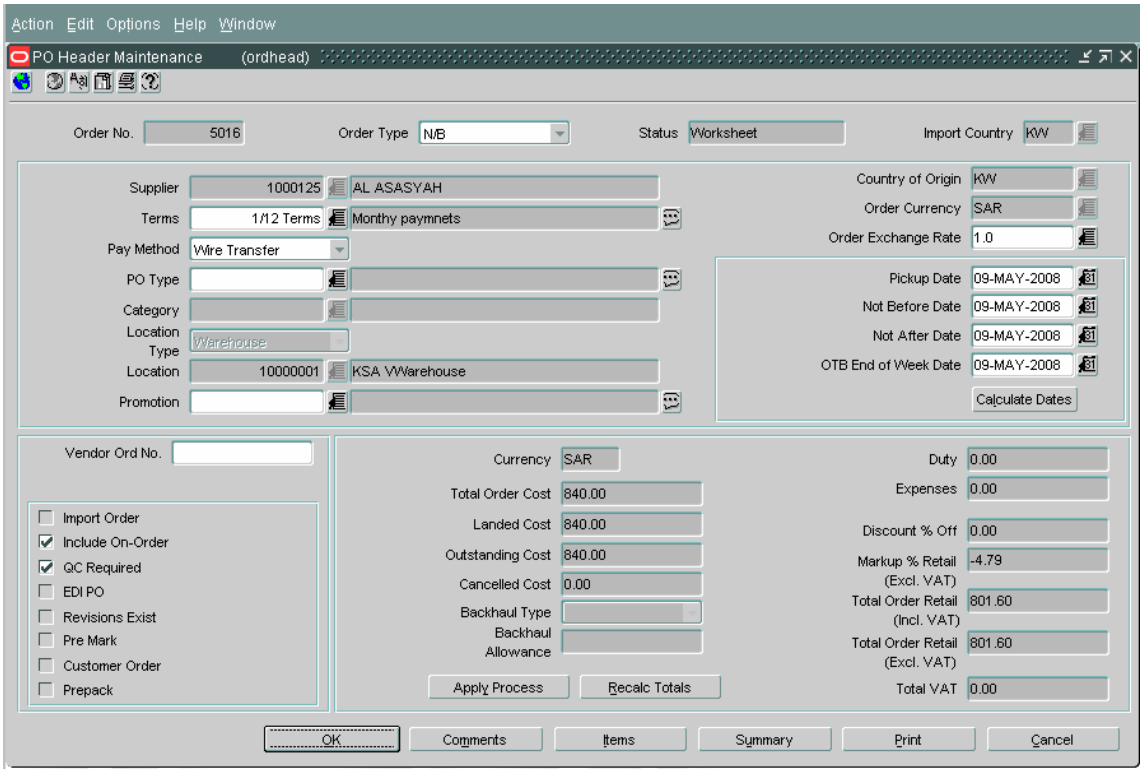


Figura 44 – ORMS – Alteração de uma ordem de compra

De seguida será necessária a verificação da tabela auxiliar, para a existência da linha, cujo campo *order_no* tenha o valor 5016, como se pode verificar na figura 45.

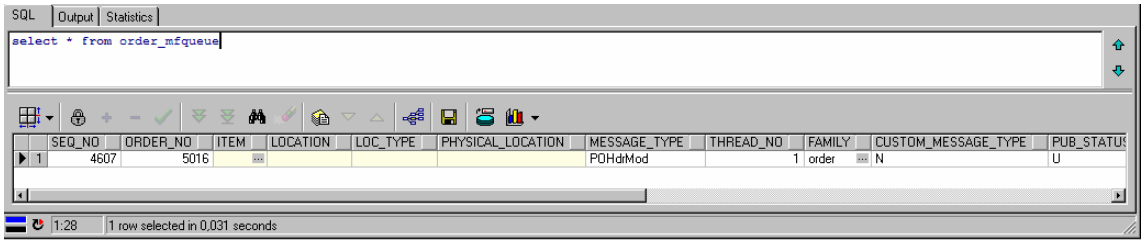


Figura 45 – Tabela ORDER_MFQUEUE

Após a activação do adaptador que vai publicar a mensagem no tópico *etOrdersFromRMS*, é possível verificar a existência da mensagem na JMS, como se exemplifica na figura 46, e a remoção dos dados referenciados na figura 45 da tabela auxiliar.

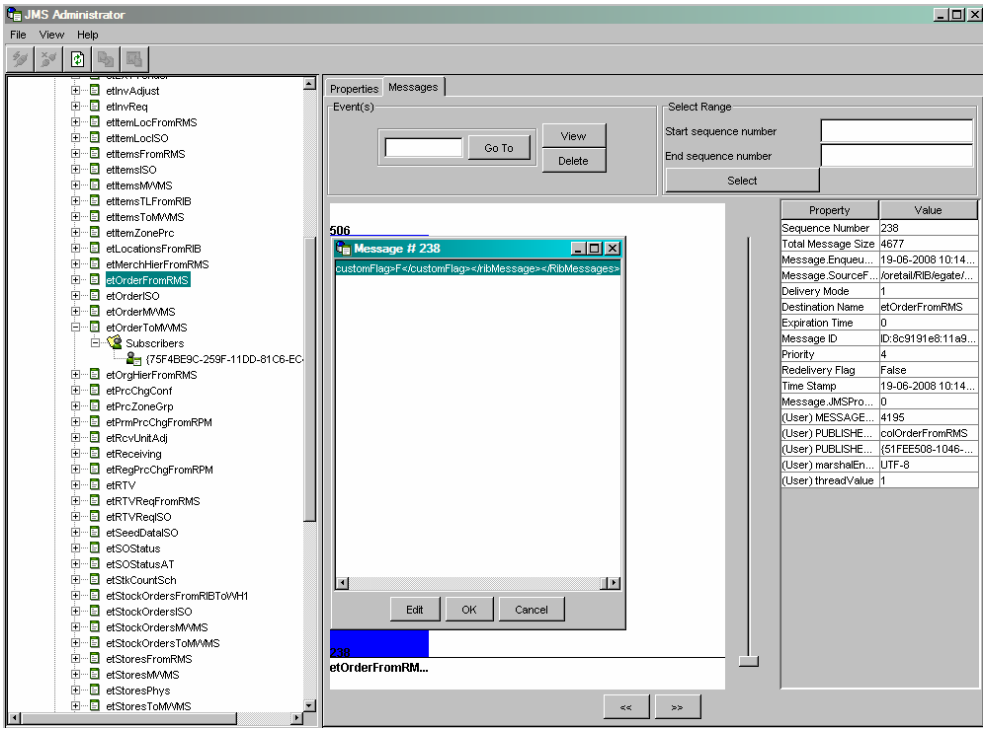


Figura 46 – Mensagem publicada na JMS

Na figura 47 é visível o conteúdo e estrutura da mensagem publicada pelo adaptador de publicação referente ao ORMS.

<?xml version="1.0" encoding="UTF-8"?><RibMessages><publishetname></publishetname><ribMessage><family>order</family><type>POCre</type><id>452</id><ribmessageID>10.3.1|ewOrderFromRMS|colOrderFromRMS|2008.06.19 13:43:51.513|2</ribmessageID><routingInfo><name>to_phys_loc</name><value>12</va lue><detail><dtl_name>to_phys_loc_type</dtl_name><dtl_value>S</dtl_value></detail></routingInfo><publishTime>2008 -06-19 13:43:51.513 GMT+03:00</publishTime><messageData><?xml version="1.0" encoding="UTF -8"?><!DOCTYPE PODesc SY STEM 'http://retail:7780/dtd/PODesc.dtd'><PODesc><doc_type>P</doc_type><order_no> **5016**</order_no><order_type>N/B</order_type><order_type_desc>N/B</order_type_desc><supplier>200012< /supplier><qc_ind>N</qc_ind><not_before_date><year>2008</year><month>06</month><day>01</day><hour>00</hour><minute>00</minute><second>00</second></not_before_da te><not_after_date><year>2008</year><month>06</month><day>01</day><hour>00</hour><minute>00</minute><second>00</second></not_after_date><otb_eow_date><year>2008</year><month>06</month><day>06</day><hour>00</hour><minute>00</minute><second>00</second></otb_eow_date><earliest_ship_date><year>2008</year><month>06</m onth><day>01</day><hour>00</hour><minute>00</minute><second>00</second></earliest_ship_date><latest_ship_date><year>2008</year><month>08</month><day>30</day> <hour>00</hour><minute>00</minute><second>00</second></latest_ship_date><terms>1001</terms><terms_code>30DAYS</terms_code><freight_terms>1001</freight_terms><cust_order>N&l t;/cust_order><payment_method>OA</payment_method><payment_method_desc>Open Account</payment_method_desc><ship_method>41</ship_method><ship_method_desc>Air, container</ship_method_desc><purchase_type> FOB</purchase_type><purchase_type_desc>Free on Board</purchase_type_desc><status>A</status><ship_pay_method>CC</ship_pay_method><ship_pay_method_desc>Collect</ship_pay_method_desc><fob_trans_res>C C</fob_trans_res><fob_trans_res_code_desc>Country</fob_trans_res_code_desc><fob_trans_res_desc>kw</fob_trans_res_desc><fob_title_pass>CI</fob_title_pass><fob_title_pass_code_desc>City</fob_title_pass_code_desc><fob_title_pass_desc>kw</fob_title_pass_desc><exchange_rate>0.881703</exchange_rate><bill_to_id>1004</bill_to_id><po_type>001</po_type><po_type_desc>KUWAIT ORDER</po_type_desc><pre_mark _ind>N</pre_mark_ind><currency_code>EUR</currency_code><pickup_loc>1</pickup_loc><pickup_no>1</pickup_no><pickup_date><year>2008</year><month>06</month><day>01</day>< hour>00</hour><minute>00</minute><second>00</second></pickup_date><PODtl><item>100044334</item><physical_location_type>S</physical_location_type><physical_location>12</physical_l ocation><physical_qty_ordered>2</physical_qty_ordered><unit_cost>3</unit_cost><origin_country_id>KW</origin_country_id><supp_pack_size>1</supp_pack_size><earliest_ship_date><year>2008</year& gt;<month>06</month><day>01</day><hour>00</hour><minute>00</minute><second>00</second></earliest_ship_date><latest_ship_date><year>2008</year><month>08</month>< lt;day>30</day><hour>00</hour><minute>00</minute><second>00</second></latest_ship_date><pickup_loc>1</pickup_loc><pickup_no>1</pickup_no><packing_method>FLAT</packing_metho d><round_lvl>C</round_lvl><POVirtualDtl><location_type>S</loc

Figura 47 – Exemplo de uma mensagem publicada na JMS

Como o destino desta mensagem será o MWMS, será necessária a sua transformação, de modo a que seja viável a sua integração no sistema de gestão de entreposto. Assim sendo foi ligado o adaptador transformador na plataforma de gestão *(SeeBeyond Egate*) que irá subscrever a mensagem do ORMS do tópico *etOrdersFromRMS*, transformá-la e posteriormente publicá-la no tópico *etOrderToMWMS*. Na figura 48 encontra-se a mensagem depois de publicada.

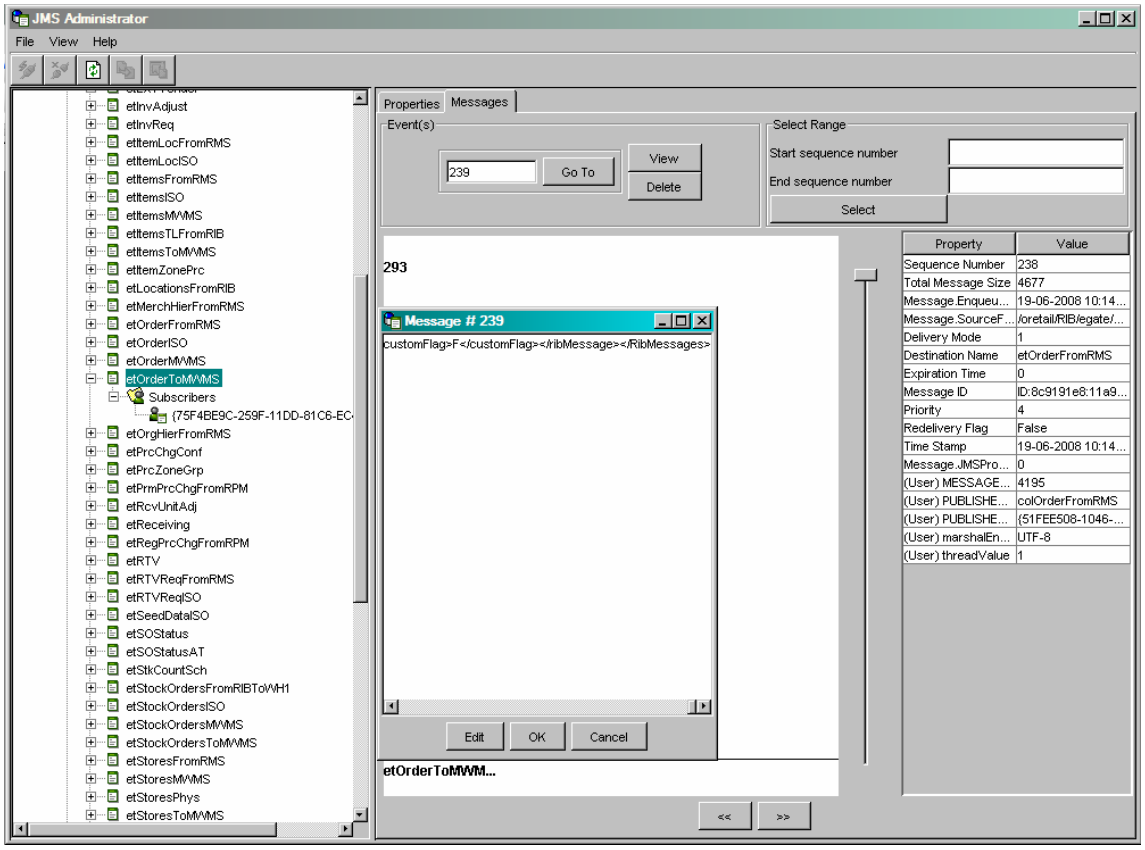


Figura 48 – Mensagem alterada na JMS

Após a subscrição da mensagem transformada por parte do adaptador final, serão então criados os ficheiros XML na pasta pré-definida para a qual o EIS irá realizar a ponte entre os dois sistemas (figura 49).

```
-rw-rw-rw-   1 oregate    oretail       3948 Jun 25 12:05 000_ordermwms_20080625120555324.xml
-rw-rw-rw-   1 oregate    oretail       3952 Jun 25 12:05 000_ordermwms_20080625120555627.xml
-rw-rw-rw-   1 oregate    oretail       1799 Jun 25 12:05 000_ordermwms_20080625120556867.xml
-rw-rw-rw-   1 oregate    oretail       2855 Jun 25 12:05 000_ordermwms_20080625120556195.xml
-rw-rw-rw-   1 oregate    oretail       2858 Jun 25 12:05 000_ordermwms_20080625120556602.xml
-rw-rw-rw-   1 oregate    oretail       2855 Jun 25 12:05 000_ordermwms_20080625120557178.xml
-rw-rw-rw-   1 oregate    oretail       2858 Jun 25 12:05 000_ordermwms_20080625120557740.xml
-rw-rw-rw-   1 oregate    oretail       6119 Jun 25 12:06 000_ordermwms_20080625120615613.xml
-rw-rw-rw-   1 oregate    oretail       1788 Jun 25 12:06 000_ordermwms_2008062512061626.xml
-rw-rw-rw-   1 oregate    oretail       1788 Jun 25 12:06 000_ordermwms_20080625120616487.xml
-rw-rw-rw-   1 oregate    oretail       1792 Jun 25 12:06 000_ordermwms_20080625120616810.xml
-rw-rw-rw-   1 oregate    oretail       1792 Jun 25 12:06 000_ordermwms_20080625120617619.xml
-rw-rw-rw-   1 oregate    oretail       1792 Jun 25 12:06 000_ordermwms_20080625120617182.xml
-rw-rw-rw-   1 oregate    oretail       1786 Jun 25 12:06 000_ordermwms_2008062512061897.xml
-rw-rw-rw-   1 oregate    oretail       2855 Jun 25 12:21 000_ordermwms_20080625122135150.xml
-rw-rw-rw-   1 oregate    oretail       2858 Jun 25 12:21 000_ordermwms_20080625122135419.xml
-rw-rw-rw-   1 oregate    oretail       2855 Jun 25 14:10 000_ordermwms_20080625141053941.xml
-rw-rw-rw-   1 oregate    oretail       2858 Jun 25 14:10 000_ordermwms_20080625141054175.xml
```

Figura 49 – Ficheiros XML publicados na pasta local

Estes ficheiros, contêm uma mensagem XML significativamente diferente da que foi inicialmente introduzida na JMS. Isto, devido ao facto da sua transformação e mapeamento serem bem efectuados (figura 50).

```
<PurchaseOrderBridge xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.2">
    <PurchaseOrder>
        <PONbr>5016</PONbr>
        <ToLocation>12</ToLocation>
        <POHeaderFields>
            <CasesShipped>10</CasesShipped>
            <UnitsOrdered>120</UnitsOrdered>
            <TotalWeight>120.0</TotalWeight>
            <StatusCode>90</StatusCode>
            <Volume>120.0</Volume>
        </POHeaderFields>
        <ListOfPODetails>
            <PODetail>
                <SKUDefinition>
                    <Company>Comp IMM</Company>
                    <Division>DIV IMM</Division>
                    <SkuID>100044334</SkuID>
                </SKUDefinition>
                <SequenceNbr>0</SequenceNbr>
                <PONbr>5016</PONbr>
                <PODetailFields>
                    <CasesShipped>2</CasesShipped>
                    <StdCaseQuantity>2</StdCaseQuantity>
                    <VendorTier>1</VendorTier>
                    <VendorHeight>1</VendorHeight>
                    <StandardSubPackQty>12</StandardSubPackQty>
                    <StandardPackQty>12</StandardPackQty>
                    <UnitsOrdered>2</UnitsOrdered>
                    <NbrofPackforCatchWt>0</NbrofPackforCatchWt>
                    <PriceTicketAvailable>0</PriceTicketAvailable>
                    <RetailPrice>0</RetailPrice>
                    <TotalUnloadTime>0</TotalUnloadTime>
                    <TotalCatchWeight>0</TotalCatchWeight>
                </PODetailFields>
            </PODetail>
        </ListOfPODetails>
    </PurchaseOrder>
</PurchaseOrderBridge>
```

Figura 50 – Ficheiro XML de acordo com o schema do MW MS

7 Conclusão

O estágio iniciado em Fevereiro deixava antever um caminho difícil para percorrer e muito trabalho pela frente. Foi exigida a rápida integração num ambiente empresarial competitivo, a adaptação a um negócio e a um conjunto de ferramentas completamente novos.

Para ultrapassar estas dificuldades em muito contribuiu o período de formação. Este período permitiu construir uma base de conhecimentos relativos ao negócio de retalho. Esta é uma área em que o conhecimento inicial se resumia às compras do dia-a-dia num destes estabelecimentos. No entanto, atrás das bonitas lojas, está uma enorme operação de deslocação de produtos e mercadorias desde o fornecedor até às lojas, que envolve milhares de transacções.

A integração de sistemas é hoje em dia uma área cada vez mais complexa. Com a transição dos processos de negócio de uma empresa do papel para sistemas informáticos, a integração torna-se ainda mais crítica. É um trabalho escondido, com pouca visibilidade, duro e complexo. Servir de intermediário entre dois sistemas, e agradar a ambos, sem comprometer o desempenho e fiabilidade do sistema geral não é fácil, mas a equipa de integração da Enabler/WIPRO possui o conhecimento necessário para que esta integração possa ser feita com grande sucesso, não só entre os sistemas que implementam, assim como com os chamados sistemas *3rd party*.

Segundo um artigo[20] sobre um estudo acerca da lealdade dos clientes, retirou-se como conclusão que a maior parte dos retalhistas usam novos produtos e descontos para aferir a lealdade dos seus clientes. Assim sendo, como desenvolvimento futuro poderia ser implementado um sistema, criado no topo da cadeia de abastecimento, que gerisse, por exemplo a utilização de um cartão de pontos que poderia garantir algumas vantagens aos seus clientes. Estes pontos poderiam ser adquiridos através da aquisição de novos produtos no estabelecimento do retalhista, e posteriormente trocados por uma vasta gama de artigos ou descontos.

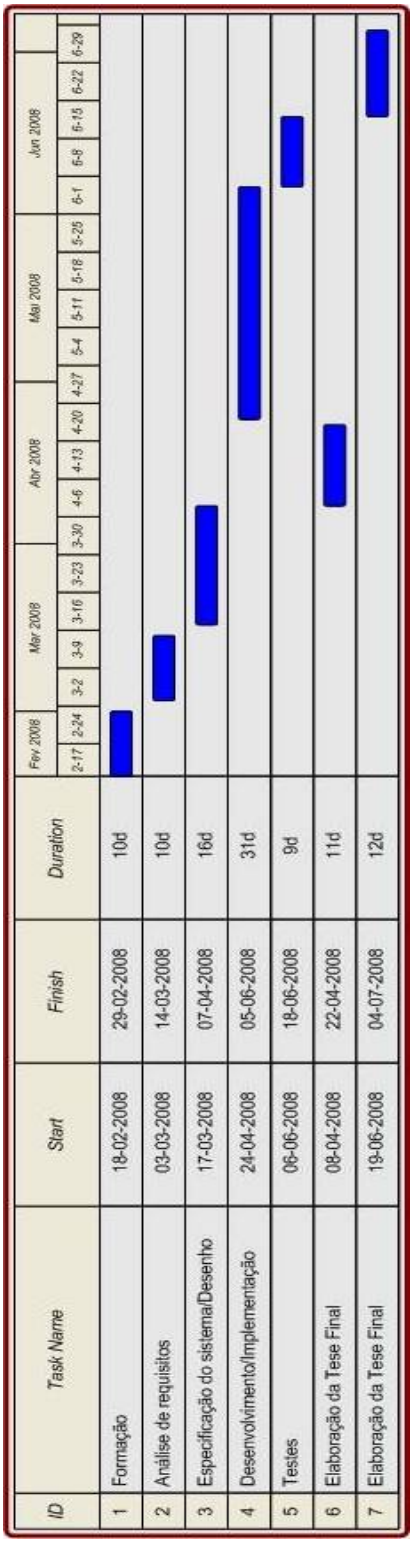
A inserção numa equipa de trabalho experiente contribuiu para a rápida integração do estagiário na empresa e para o seu sucesso. Sem dúvida fundamentais, estes colegas enriqueceram a experiência e tornaram-na muito agradável, tanto a nível profissional como a nível pessoal. A documentação técnica relativa a projectos passados existentes no repositório da empresa foi também uma grande ajuda.

O trabalho decorreu de forma tranquila e dentro dos prazos estabelecidos no plano de trabalho, conseguindo cumprir todos os objectivos propostos, assim como a satisfação do cliente.

---

[20] SAP Portugal: Lealdade dos clientes,
http://www.sap.com/portugal/company/press/press.epx?PressID=8168

ANEXO A: Plano de Estágio



ID	Task Name	Start	Finish	Duration
1	Formação	18-02-2008	29-02-2008	10d
2	Análise de requisitos	03-03-2008	14-03-2008	10d
3	Especificação do sistema/Desenho	17-03-2008	07-04-2008	16d
4	Desenvolvimento/Implementação	24-04-2008	05-06-2008	31d
5	Testes	06-06-2008	18-06-2008	9d
6	Elaboração da Tese Final	08-04-2008	22-04-2008	11d
7	Elaboração da Tese Final	19-06-2008	04-07-2008	12d

ANEXO B: Exemplo de ficheiro XML na pasta local

000_ordemwms_20080625141053941.xml

```
<PurchaseOrderBridge xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.2">
    <PurchaseOrder>
        <PONbr>573</PONbr>
        <ToLocation>MANH1</ToLocation>
        <POHeaderFields>
            <ReceivedFrom>202248</ReceivedFrom>
            <QcHoldUponReceipt>Y</QcHoldUponReceipt>
            <CasesShipped>6</CasesShipped>
            <UnitsOrdered>20</UnitsOrdered>
            <SchedStartRcvDate>2008-06-04T00:00:00.00Z</SchedStartRcvDate>
            <TotalWeight>20.0</TotalWeight>
            <StatusCode>00</StatusCode>
            <Volume>20.0</Volume>
        </POHeaderFields>
        <ListOfPODetails>
            <PODetail>
                <SKUDefinition>
                    <Company>Comp IMM</Company>
                    <Division>DIV IMM</Division>
                    <SkuID>100028107</SkuID>
                </SKUDefinition>
                <SequenceNbr>0</SequenceNbr>
                <PONbr>573</PONbr>
                <PODetailFields>
                    <CasesShipped>4</CasesShipped>
                    <StdCaseQuantity>1</StdCaseQuantity>
                    <VendorTier>1</VendorTier>
                    <VendorHeight>1</VendorHeight>
                    <StandardSubPackQty>1</StandardSubPackQty>
                    <StandardPackQty>1</StandardPackQty>
                    <UnitsOrdered>4</UnitsOrdered>
                    <NbrofPackforCatchWt>0</NbrofPackforCatchWt>
                    <PriceTicketAvailable>0</PriceTicketAvailable>
                    <RetailPrice>0</RetailPrice>
                    <TotalUnloadTime>0</TotalUnloadTime>
                    <TotalCatchWeight>0</TotalCatchWeight>
                </PODetailFields>
            </PODetail>
            <PODetail>
                <SKUDefinition>
                    <Company>Comp IMM</Company>
                    <Division>DIV IMM</Division>
                    <SkuID>100044431</SkuID>
```

ANEXO C: Exemplo de um ficheiro XML na JMS

vendor.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<RibMessages>
    <publishetname></publishetname>
    <ribMessage>
        <family>vendor</family>
        <type>VendorCre</type>
        <id>20001</id>
        <ribmessageID>10.3.1|ewVendorFromRMS|colVendorFromRMS|2008.05.27 15:47:19.330|2</ribmessageID>
        <publishTime>2008-05-27 15:47:19.334 GMT+03:00</publishTime>
        <messageData>
            &lt;?xml version=&quot;1.0&quot; standalone=&apos;no&apos;?&gt;
                &lt;!DOCTYPE VendorDesc PUBLIC
                    &gt;&lt;VendorDesc&gt;
                      &lt;VendorHdrDesc&gt;
                        &lt;supplier&gt;20001&lt;/supplier&gt;
                        &lt;sup_name&gt;Supplier Test Umbrella&lt;/sup_name&gt;
                        &lt;contact_name&gt;Contact name umbrella&lt;/contact_name&gt;
                        &lt;contact_phone&gt;123456&lt;/contact_phone&gt;
                        &lt;sup_status&gt;I&lt;/sup_status&gt;
                        &lt;qc_ind&gt;N&lt;/qc_ind&gt;
                        &lt;vc_ind&gt;N&lt;/vc_ind&gt;
                        &lt;currency_code&gt;USD&lt;/currency_code&gt;
                        &lt;terms&gt;1001&lt;/terms&gt;
                        &lt;freight_terms&gt;1001&lt;/freight_terms&gt;
                        &lt;ret_allow_ind&gt;N&lt;/ret_allow_ind&gt;
                        &lt;ret_auth_req&gt;Y&lt;/ret_auth_req&gt;
                        &lt;edi_po_ind&gt;N&lt;/edi_po_ind&gt;
                        &lt;edi_po_chg&gt;N&lt;/edi_po_chg&gt;
                        &lt;edi_po_confirm&gt;N&lt;/edi_po_confirm&gt;
                        &lt;edi_asn&gt;N&lt;/edi_asn&gt;
                        &lt;edi_sales_rpt_freq&gt;D&lt;/edi_sales_rpt_freq&gt;
                        &lt;edi_supp_available_ind&gt;N&lt;/edi_supp_available_ind&gt;
                        &lt;edi_contract_ind&gt;N&lt;/edi_contract_ind&gt;
                        &lt;edi_invc_ind&gt;N&lt;/edi_invc_ind&gt;
                        &lt;replen_approval_ind&gt;N&lt;/replen_approval_ind&gt;
                        &lt;settlement_code&gt;N&lt;/settlement_code&gt;
                        &lt;pre_mark_ind&gt;N&lt;/pre_mark_ind&gt;
                        &lt;delivery_policy&gt;NEXT&lt;/delivery_policy&gt;
                        &lt;bracket_costing_ind&gt;N&lt;/bracket_costing_ind&gt;
                        &lt;dsd_supplier_ind&gt;N&lt;/dsd_supplier_ind&gt;
                      &lt;/VendorHdrDesc&gt;
                    &lt;/VendorDesc&gt;
        </messageData>
        <customFlag>F</customFlag>
    </ribMessage>
</RibMessages>
```

ANEXO D: Tabela de Acrónimos e Abreviaturas

Abreviatura/Acrónimo	Descrição
API	Application Programming Interface
ASN	Advanced Shipping Notice
DTD	Data Type Definition
EAI	Enterprise Application Integration
EAN	European Article Number
EIS	Enterprise Integration Services
ERP	Enterprise Resource Planning
FRD	Functional Requirement Document
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
JMS	Java Message Service
LPN	License Plate Number
MA	Manhattan Associates
MCH	Modelo Continente Hipermercados
MWMS	Manhattan Warehouse Management System
OR	Oracle Retail
ORIB	Oracle Retail Integration Bus
ORMS	Oracle Retail Merchandise System
PL/SQL	Procedural Language/Structured Query Language
RF	Radio Frequency
SKU	Stock Keeping Unit
TAFR	Transform Address Filter/Router
TI	Tecnologias de Informação
TRD	Technical Requirement Document
UTD	Unit Tests Document
XML	Extensible Markup Language

Bibliografia

1. Enabler Wipro, *http://www.wipro.com/*
2. Modelo Continente SGPS, SA, *http://www.modelocontinente.pt/*
3. "Wipro acquires leading European Retail Solutions Provider, Enabler", *http://www.wipro.com/news/newsitem1/newstory448.htm*
4. Wipro Technologies, *http://www.wipro.com/*
5. Oracle Retail: *http://www.oracle.com/industries/retail/index.html*
6. PL/SQL Developer: *http://www.allroundautomations.nl/plsqldev.html*
7. Manhattan Associates - Supply Chain Management Software Solutions, *http://www.manh.com/*
8. Wikipédia, *http://pt.wikipedia.org*
9. Advanced Information Manager - Warehouse Management by Manhattan Associates, 2006
10. Logistics, *http://logistics.about.com/*
11. Message-Driven EJBs, *http://edocs.bea.com/wls/docs81/ejb/message_beans.html*
12. Extended Retail Solutions, *http://www.extendedretail.com/*
13. Retail Basic Concepts: understand the retail business principles, Enabler 2006
14. Oracle Retail Integration Bus, Technical Architecture Guide, May 2006, *http://download.oracle.com/docs/cd/B31315_01/rib/pdf/120/rib-120-tag.pdf*
15. Sap Portugal: Lealdade dos Clientes, *http://www.sap.com/portugal/company/press/press.epx?PressID=8168*