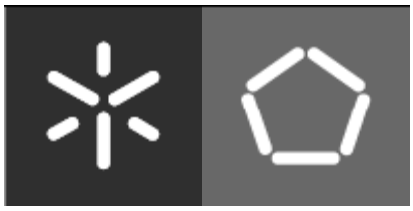

Universidade do Minho
Escola de Engenharia

Implementação de Processos de Negócio em Software Utilizando Transformações de Modelos

Nuno António de Lira Fernandes Faria dos Santos

Dissertação submetida à Universidade do Minho
para obtenção do grau de Mestre em Engenharia e Gestão de Sistemas de Informação, elaborada sob a orientação científica do Professor Ricardo Jorge Silvério de Magalhães Machado

Universidade do Minho
Escola de Engenharia
Departamento de Sistemas de Informação
Guimarães, Outubro de 2009

Agradecimentos

Agradeço ao Professor Doutor Ricardo Machado pela sua excelente tarefa de orientação de todo o processo da dissertação, pela transmissão do seu vasto conhecimento científico, aconselhamento formal e informal, pelo seu empenho em garantir que o trabalho não sofresse desvios de índole temática, pela motivação dada para que os prazos estipulados fossem cumpridos e, por fim, pela indicação e fornecimento de alguma bibliografia importante que se revelou fundamental.

Agradeço à *Bosch Car Multimedia Portugal, Lda.*, em Braga, empresa onde elaborei o caso de estudo desta dissertação, a oportunidade concedida. Agradeço também aos meus colegas do Departamento de Informática da empresa, todos sem excepção, por me proporcionarem uma integração fácil e rápida. Tenho de agradecer especialmente ao Eng. Francisco Duarte, meu "chefe", pela transmissão de conhecimento sobre alguns temas abordados, pela total disponibilidade para discussão sobre várias vertentes da dissertação e pelo aconselhamento do ponto de vista "profissional" e "académico". Não posso deixar de deixar um agradecimento ao Carlos Ribeiro, Marco Couto e José Pedro Araújo, com quem interagi mais durante o tempo em que estive na empresa, pela ajuda por eles fornecida durante a aprendizagem de certas tecnologias em que eu não dominava por completo e pela entreajuda aquando do contacto com tecnologias utilizadas que ainda eram pouco dominadas.

Agradeço aos meus colegas e amigos da licenciatura e também posteriormente do mestrado, não só pela convivência proporcionada ao longo de cinco anos – pois alguma descompressão também é necessária – mas também pela troca de experiências e conhecimentos da elaboração das suas próprias dissertações, que também constituíram contributos para esta dissertação.

Agradeço aos meus amigos pela compreensão da dedicação que a realização deste tipo de trabalhos requer, dado que nem era sempre possível a convivência com eles.

Por fim, mas não menos importante, agradeço à minha família. Desde o facto de me proporcionarem condições durante todo o período estudantil até ao seu apoio total na altura em que decidi tirar uma pós-graduação.

Sumário

Dado que as organizações se regem por processos de negócio, é emergente a necessidade de o desenvolvimento do sistema de informação de uma organização seja direccionado para suportar esses processos. Durante um processo de desenvolvimento de sistemas de informação são conhecidas várias abordagens e metodologias com o objectivo de este seja obtido com cada vez maior acréscimo de qualidade. Essa qualidade deve-se reflectir na definição dos processos do sistema de informação e deve requerer uma metodologia que possua tarefas bem definidas e automatizadas. Uma metodologia possível, a *Business Implementation Methodology* (BIM), propõe transformações dos processos de negócio de forma automatizada para que a implementação dos processos em software consuma menor esforço e menores custos. As quatro fases da metodologia são percorridas desde a definição inicial dos processos de negócio até ao sistema de informação final.

O que distingue nesta fase das outras metodologias é a proposta de utilizar modelos de referência de processos. Dado o contexto, sugere-se a utilização de linguagens de modelação de processos para representação gráfica dos processos de negócio.

Para obtenção do sistema de informação na última fase da metodologia seguida, é proposta a utilização do *Model-Driven Architecture* (MDA). A transformação de um modelo de processos de negócio que seja um *Platform Independent Model* (PIM) para um modelo que seja um *Platform Specific Model* (PSM) permitirá a obtenção da implementação do modelo de processos em software. O resultado da transformação será um modelo composto pelo processo de negócio em BPEL e pela devida integração de aplicações realizadas através do uso de um *Enterprise Service Bus* (ESB).

Palavras-chave: *Business Implementation Methodology*; linguagens de modelação de processos; modelos de referência de processos; MDA; PIM; PSM; ESB

Abstract

Due the fact that organisations rule their business by business processes, it is emerging the need that development of the information system should be oriented to support those processes. During the process of an information system development many approaches are known so it can be achieved with an increase of quality. This quality must be reflected in the definition of the information system's business processes and it requires a methodology that has well defined and automated tasks. A possible methodology, the Business Implementation Methodology (BIM), proposes transformations of business processes in an automated way so software implementation of the processes are less effort consuming. The four methodology phases are executed since the initial definition of the business process to the final information system.

What distinguishes this methodology from other business implementation methodologies in these initial phases is the use of process reference models proposal. In the context, it is suggested the use of business process modelling languages for visual representation of business processes.

For obtaining the information system in the last phase of the methodology, it is proposed the use of OMG's Model-Driven Architecture (MDA). The transformation of a process framework that is a Platform Independent Model (PIM) to a Platform Specific Model (PSM) will allow obtaining software implementation. The result of the transformation will be a model composed by the business process in BPEL and the integration of applications through the use of an Enterprise Service Bus (ESB.

Key words: Business Implementation Methodology; business process modelling languages; process reference models; MDA; PIM; PSM; ESB.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

4SRS	4-Step Rule Set
ARIS	ARchitecture for integrated Information Systems
ATL	Atlas Transformation Language
B2B	Business-to-business
B4P	BPEL4People
BC	Binding Component
BPD	Business Process Diagram
BIM	Business Implementation Methodology
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
BPM	Business Process Management
BPMI	Business Process Management Initiative
BPML	Business Process Modeling Language
BPMN	Business Process Modeling Notation
BPMS	Business Process Management System
BPQL	Business Process Query Language
CE-nets	Condition-event nets
CIM	Computer Independent Model
CM	Car Multimedia
CMM	Capability Maturity Model
CMOF	Complete MOF
CORBA	Common Object Request Broker Architecture
COTS	Commercial off-the-shelf
CPN	Colored Petri Nets
CRM	Customer Relaionship Management
CWM	Common Warehouse Metamodel
DAML-S	DARPA Agent Markup Language for Services
DBMS	Database Management Systems
DC	Delivery Channel
DCOR	Design-Chain Operations Reference
DD	Design Decisions
DES	Discrete Event Systems

DSL	Domain Specific Languages
EAI	Enterprise Integration Integration
EFQM	European Foundation for Quality Management
EJB	Enterprise JavaBeans
EMOF	Essential MOF
EPC	Event-driven Process Chain
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
eTOM	enhanced Telecom Operations Map
HL-net	High-Level Petri-nets
IBM	International Business Machines Corporation
ISO	International Organization for Standardization
ITIL	Information Technology Infrastructure Library
J2EE	Java 2 Platform, Enterprise Edition
JBI	Java Business Integration
JCo	J**a**va Connector
JDBC	Java Database Connectivity
JMS	Java Message Service
MDA	Model-driven Architecture
MDD	Model-driven Development
MDE	Model-driven Engineering
MEP	Message Exchange Pattern
MIC	Model-integrated Computing
MOF	Meta Object Facility
NFR	Non-functional requirements
NMR	Normalized Message Router
OBO	Orchestrated Business Objects
OCL	Object Constraint Language
ODE	Orchestration Director Engine
OMG	Object Management Group
OWL-S	Web Ontology Language for Services
P4	Process 4
PF	Process Framework
PIM	Platform Independent Model
POJO	Plain Old Java Objects

POM	Project Object Model
PrT-net	Predicate/Transition Net
PSM	Platform Specific Model
PSP	Personal Software Process
PT-net	Place-transition net
QVT	Query/Views/Transformations
REL	Regular Expression Language
SA	Service Assembly
SAP R/3	Systeme, Anwendungen und Produkte in der Datenverarbeitung Realtime System Version 3
SCOR	Supply-Chain Operations Reference
SE	Service Engine
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SPEM	Software Process Engineering Metamodel
SU	Service Unit
SWEBoK	Software Engineering Book of Knowledge
TI	Tecnologias de Informação
TS	Technological Spaces
TSP	Team Software Process
UBK	Unternehmensbereich Kraftfahrzeugausrüstung
UBK-RM	Unternehmensbereich Kraftfahrzeugausrüstung – Reference Model
UML	Unified Modeling Language
UML-AD	UML Activity Diagrams
WfMC	Workflow Management Coalition
WPDL	Workflow Process Definition Language
WS	Web Services
WSCI	Web Service Choreography Interface
WSDL	Web Service Description Language
WSFL	Web Services Flow Language
WS-BPEL	Web Services Business Process Execution Language
XLANG	XML LANGuage
XSD	XML Schema
XMI	XML Metadata Interchange

XML	eXtensible Markup Language
XPDL	XML Process Definition Language
YAWL	Yet Another Workflow Language
yEPC	YAWL Event-driven Process Chain

1. Introdução

Este capítulo pretende apresentar o problema relacionado com a temática do desenvolvimento de sistemas de informação em organizações que se regem por processos de negócio. Dada a motivação do trabalho, são enunciados os objectivos a que este trabalho se propôs a alcançar. De seguida é apresentada a abordagem de investigação seguida pelo autor do trabalho. O trabalho relacionado pretende apresentar algum trabalho semelhante já realizado por outros autores bem como a apresentação da estrutura do documento.

O conceito de *Business Process Management* (BPM) [Smith e Fingar 2002] já é um conceito largamente implementado pelas empresas, onde também o termo *Enterprise Resource Planning* (ERP) [Sandoe, Corbitt *et al.* 2001] se encontra inserido no ambiente organizacional, sendo este o sistema normalmente utilizado como suporte aos processos. Dado que as organizações dos clientes já possuem sistemas de informação direccionados para BPM, os requisitos pedidos pelos próprios clientes já têm por base os processos de negócio, pelo que os modelos a utilizar no desenvolvimento de software que cumpram esses requisitos já têm de ser direccionados para processos de negócio.

O desenvolvimento de software deve então ter por base processos de negócio. Isto prende-se com o facto de, em organizações que se regem por processos (*process-oriented organizations*) [Duarte, Fernandes *et al.* 2006], as actividades destas organizações serem focadas na satisfação das necessidades dos clientes. Logo, para suportar estas actividades, também os sistemas de software desenvolvidos devem ser capazes de suportar processos e não apenas uma função específica de um departamento. O crescimento da quantidade de soluções de *Business Process Management Systems* (BPMS) [Smith e Fingar 2002] que surgem no mercado é a constatação desse facto.

A metodologia utilizada neste trabalho para implementação dos processos, a *Business Implementation Methodology* (BIM) [Duarte, Machado *et al.* 2009], tem como característica principal a de utilizar modelos de referência de processos que forneçam as melhores práticas organizacionais do sector e finalizar a implementação de um sistema de software no sistema de informação da organização que garanta a satisfação dos requisitos dos processos do cliente.

A principal motivação deste trabalho prende-se com a última parte. Na implementação do processo em software, tal como nas restantes fases, não existem restrições no que toca a notações e a marcas e instituições, sendo totalmente independente. Para implementar o modelo de processos de negócio em software, uma abordagem *model-driven* pode ser vantajosa, em termos de tempo e de complexidade. A *Model-Driven Architecture* (MDA) [OMG 2003] já é mundialmente reconhecida na sua eficiência no desenvolvimento de

software. A utilização da MDA sugere a concepção de dois modelos. O *Platform Independent Model* (PIM) representa o modelo dos processos de negócio, e o *Platform Specific Model* (PSM) a execução em software do modelo anterior [Duarte, Machado *et al.* 2007]. Resta então saber se, após o modelo de processos estar devidamente definido, a obtenção de software a partir dos conceitos de MDA é efectuada com um nível de automatização das tarefas elevado.

Maximizando o nível de automatização da implementação dos processos de negócio é garantida uma implementação segura e eficiente, originando uma quantidade mínima de erros durante todas as fases de um projecto de implementação, o que por sua vez resulta numa redução do tempo e dos recursos para a empresa.

No entanto, o desenvolvimento de um sistema de informação para a organização continua a ser uma tarefa complicada, devido a implicar a integração de várias tecnologias. A integração de sistemas informáticos numa organização alarga-se a toda a organização e já passa as "fronteiras" da organização, passando por integrar com as aplicações de clientes e fornecedores. A integração inclui cada vez mais aplicações heterogéneas entre si, dado que presentemente o ERP necessita estar integrado com sistemas de gestão de bases de dados, *Customer Relationship Management* (CRM) [Sandoe, Corbitt *et al.* 2001], *datawarehouses*, aplicações de *Business Intelligence,* de apoio a decisões, de escritório e também de suporte a funções específicas. Como é norma em ambientes organizacionais, o tempo dispendido tem que ser cada vez menor, bem como os custos associados. Integrar aplicações da organização e não só, em pouco tempo e com baixos custos, pode, devido a heterogeneidade das aplicações, não ser tarefa fácil. É com esse objectivo que estão sempre a surgir novas normas e abordagens de integração, como *Web Services* (WS) e *Service-Oriented Architecture* (SOA).

1.1 Objectivos e Contributos

Este trabalho consiste em apresentar uma perspectiva de obtenção de software no desenvolvimento de sistemas de informação para organizações que se regem por processos de negócio. A abordagem ao *model-driven development* [Atkinson e Kühne 2003] já provou ser apropriada para desenvolver software. Tal como o próprio titulo deste documento sugere, o principal objectivo deste trabalho de investigação é o de utilizar transformação de modelos na actividade de desenvolvimento de um sistema de informação de uma organização. Para tal, é testada a técnica de transformar processos utilizando os conceitos de *Platform Independent Model* (PIM) e *Platform Specific Model* (PSM) [OMG 2003], do MDA. É com os conceitos de PIM e PSM e utilizando transformação de modelos que se pretende obter um sistema de software

que execute os processos de negócio do sistema de informação. O principal contributo da investigação é então o de sugerir uma abordagem a modelos na fase em que se pretenda transformar um processo de negócio em software. A utilização de modelos PIM e PSM permite que se obtenha software de uma forma perfeitamente automatizada. A metodologia de implementação dos processos seguida pela investigação, a BIM, é um bom suporte para desenvolvimento de software para as organizações que se regem por processos. O trabalho resultante desta investigação pretende ser um exemplo de suporte à utilização de uma metodologia como a BIM.

A tarefa inicial ficou definida como identificar a linguagem de representação de processos de negócio que mais se adequa à *Bosch Car Multimedia Portugal*, *Lda*. Esta tarefa prende-se com o facto de a empresa não ter definida a utilização de qualquer linguagem de modelação de processos, nem conhecimentos profundos das linguagens existentes no mercado. A definição de uma linguagem de representação dos processos de negócio deve ser a tarefa inicial executada antes de se percorrer as fases de uma qualquer metodologia de implementação dos processos, dado que a linguagem escolhida para definir os processos é a utilizada ao longo de todo o desenvolvimento do sistema de informação. Além de ser tido em conta aspectos das características e do funcionamento em termos tecnológicos das próprias linguagens, também são analisados conceitos acerca de aspectos relacionados com os colaboradores da empresa e com o negócio em que esta está inserida. A utilização destes dois aspectos adicionais na comparação das linguagens será também uma das contribuições do trabalho.

Este trabalho também pretende demonstrar a importância do uso de modelos de referência de processos pois, utilizando uma metodologia como a BIM, assegura que os processos reflictam boas práticas organizacionais. A utilização de boas práticas resulta em processos de negócio que trazem maior valor acrescentado para a empresa onde estes funcionam. Os processos do modelo de referência servem de base para posteriormente serem concebidos processos customizados que já respondam aos requisitos específicos do cliente. A utilização dos modelos de referência como base justifica-se tendo em conta que o cliente pode não conhecer boas-práticas inter-organizacionais, ou então não ter a capacidade de zelar correctamente pelo bem da sua organização [Duarte, Machado *et al.* 2009]. Sendo que os processos não só usam boas práticas, mas também representam a lógica do negócio da organização do cliente, é assegurado que o processo está definido da melhor forma possível em termos do seu funcionamento. Este aspecto é essencial na sua transformação em software, pois a execução automática de um processo só traz vantagens se este realmente estiver bem definido.

1.2 Abordagem da Investigação

O trabalho de investigação realizado segue as abordagens habituais para um documento deste género. O trabalho divide-se em duas partes: a parte teórica e a parte prática. A teoria é apresentada primeiro, pois é necessário possuir um conhecimento de base dos conceitos apresentados para melhor se compreender o trabalho prático seguinte. A parte prática engloba a aplicação dos conceitos teóricos em casos reais, num ambiente organizacional.

Para a investigação, foi efectuada uma revisão de literatura no sentido de contextualizar dos conceitos teóricos acerca de linguagens de modelação de processos de negócio, implementação de processos, *model-driven development* e transformação de modelos, de modo a realizar uma análise ao estado da arte destes temas. Pretende-se nesta fase atingir o conhecimento teórico necessário para melhor estruturar a aplicação deste trabalho numa organização que vai servir de caso de demonstração. Todos os conceitos revistos irão servir de base no desenvolvimento da transformação automatizada dos modelos de processos, para se obter processos implementados em sistemas de software. A pesquisa bibliográfica inicialmente é feita para identificação dos principais conceitos associados a cada tema. Ao identificar os conceitos, torna-se necessário também identificar quais os autores mais influentes na área, bem como as instituições que possuam especificações *standard* do tema abordado. Um determinado conceito é sempre descrito conforme a especificação oficial da instituição correspondente, sendo que citações de autores são necessárias para apresentação e discussão dos seus pontos de vista. Será realizada também pesquisa de citações que apresentem perspectivas alternativas. Para melhor fundamentar um ponto de vista, os autores citados por autores identificados na bibliografia também irá ser tido em conta.

Para testar e demonstrar a validade da automatização do desenvolvimento de sistemas, realizou-se um caso de demonstração, utilizando para o efeito recursos e processos de negócio de uma organização, neste caso a *Bosch Car Multimedia* (CM) *Portugal, Lda*., em Braga. É nesta fase que são feitas as experiências para se chegar à solução final e identificar alternativas. Além de utilizar modelos de referência para os processos de negócio, estes foram posteriormente aplicados a processos de negócio efectivamente executados na *Bosch CM*, criando conjuntos de processos que sejam possíveis de ser transformados.

Para além dos processos de negócio, também um grupo de colaboradores pertencentes ao departamento de informática da *Bosch CM* participa activamente no trabalho. Este grupo de colaboradores respondeu a um questionário sobre linguagens de modelação de processos, inserido na actividade de comparação e escolha da linguagem mais apropriada para modelar os processos de negócio da empresa. O questionário era composto por um conjunto de perguntas de escolha múltipla, tanto perguntas em que se pretende apenas uma resposta bem como perguntas com respostas múltiplas. As respostas ao questionário eram anónimas, pois a identidade dos colaboradores não era relevante. Após a recolha das respostas, um estudo estatístico analisa os resultados.

1.3 Trabalho Relacionado

A realização de transformações em linguagens de processos utilizando o MDA já é tema de trabalhos anteriores. [Zhao, Hauser *et al.* 2006] transforma processos em modelo PIM em Diagramas de Actividade UML2 num modelo PSM em BPEL através do método REL (*Regular Expression Language*), tal como [Bézivin, Hammoudi *et al.* 2004] que realiza a mesma transformação utilizando a linguagem de transformação ATL (*ATLAS Transformation Language*) [Bézivin, Dupé *et al.* 2003], ou então [Koehler, Hauser *et al.* 2005] que transforma o modelo utilizando regras OCL (*Object Constraint Language*) [OMG 2006]. Outro tipo de abordagem é realizado por [Bauer, Muller *et al.* 2004] e [Lezoche, Missikoff *et al.* 2008], que começam por conceber um modelo de processos CIM em EPC [Nüttgens, Feld *et al.* 1998], transformando o CIM num modelo de processos PIM em BPMN [White 2004] e finalmente obter um modelo de processos PSM em BPEL. Uma abordagem diferente possuem [Rodriguez, Fernandez-Medina *et al.* 2007] e [Rungworawut e Senivongse 2006] que realizam transformação de um modelo CIM em BPMN em modelos PIM em UML [OMG 2009], em Diagramas de casos de uso e em diagramas de classes, respectivamente.

Uma metodologia para implementação do negócio é a mLEARN [Coelho 2005]. A metodologia *mLEARN*, conhecida pela "metodologia das competências organizacionais", fornece técnicas para representação dos processos de negócio organizacionais, devidamente alinhados com os objectivos estratégicos e também clarifica as responsabilidades na organização. A *mLEARN* assenta numa abordagem integrada e sistémica da organização, numa orientação a objectos organizacionais, apoiado em técnicas interactivas e com a preocupação da gestão do conhecimento organizacional e da mudança.

A mLEARN apenas define tarefas no sentido de se definir uma arquitectura de processos de negócio. No sentido de adicionar implementação de software aos processos, podem ser utilizadas técnicas como o *4-Step Rule Set* (4SRS) [Machado, Fernandes *et al.* 2005]

ou o standard MOF *Query/View/Transformation* (QVT) [OMG 2005] da OMG. O 4SRS é uma técnica para transformar requisitos dos utilizadores em modelos arquitecturais que representam os requisitos do sistema. Associa, para cada objecto encontrado durante a fase da análise, uma determinada categoria: interface, dados e controlo. O objectivo do QVT é obter uma *domain specific language* (DSL) [Deursen, Klint *et al.* 2000] para realizar *queries, views* e transformações em modelos. O QVT permite efectuar relações de correspondência entre modelos, nas quais um código de um modelo é posteriormente transformado e obtém-se um código de um outro modelo.

O conceito de *Process Reference Models* (modelo de referência de processos) é um conceito com o objectivo de fornecer boas-práticas que possam ser utilizadas entre organizações. Integram conceitos de *Business Process Reengineering, benchmarking* e análise de boas-práticas [SCC 2008]. Actualmente, existem *Process Reference Models* disponíveis para as organizações, como S*upply-Chain Operations Reference model* (SCOR) [SCC 2008], *Design-Chain Operations Reference model* (DCOR) [SCC 2006], *Information Technology Infrastructure Library* (ITIL) [ifSMF 2007] ou o *Enhanced Telecommunications Operations Map* (eTOM) [TMForum].

A arquitectura *Spring* [Johnson, Hoeller *et al.* 2005] fornece uma plataforma para executar processos de negócio. É constituída por três camadas: manipulação de interfaces com utilizador (por exemplo, *Web browsers* ou EJB), implementação de regras de entidades e de negócio (por exemplo, um *Application Server*), e persistência de dados (por exemplo, um servidor de bases de dados).

No âmbito de integração de aplicações empresariais, [Hohpe e Woolf 2004] define um conjunto de padrões de integração em organizações ("*Enterprise Integration Patterns*"), onde são definidos requisitos que a integração de aplicações deve obedecer num ambiente organizacional, bem como algumas técnicas e soluções.

Para implementar a integração de aplicações existem alguns *open Enterprise Service Bus* (*open* ESBs). Os *open* ESBs são vantajosos no sentido em que são *open source* e utilizam *open standards*. Alguns exemplos das ferramentas mais conhecidas são o *Mule*, o *Apache ServiceMix*, o *OpenESB*, o *Apache Synapse*, o *JBoss ESB*, o *Apache Tuscany*, o *Fuse ESB*, o *OW2 PEtALS* e o *OpenAdapter* [Rademakers e Dirksen 2008].

Atingindo-se uma determinada quantidade de processos de negócio devidamente modelados, pode ser necessário um sistema de gestão dos processos, que funcione como um repositório. O *BPEL Repository* [Vanhatalo 2006] armazena ficheiros BPEL bem como outros ficheiros XML, WSDL ou XSD. O repositório também permite consultas fáceis

aos dados utilizando a *Object Constraint Language* para executar *queries* dentro do repositório.

De referir ainda algum trabalho já existente na avaliação de linguagens de modelação de processos. Os padrões de *workflow* [Aalst, Hofstede *et al.* 2003] foram compilados a partir de uma análise às linguagens de *workflow* e capturam dependências encontradas na modelação do controlo do fluxo. A abordagem dos vinte padrões de *workflow* traz vantagens na medida em que analisa exemplos de controlo de fluxo que podem ocorrer durante um processo e é possível verificar se é possível na modelação do processo numa determinada linguagem.

Neste trabalho também os colaboradores da organização (neste caso, com funções em projectos se sistemas de informação, software e suporte a processos de negócio) são avaliados no seu conhecimento das linguagens. Para avaliar competências de programadores no que toca a desenvolvimento de sistemas de software há a possibilidade de recorrer à análise nas áreas de conhecimento do *SWEBoK* [Abran, Bourque *et al.* 2004], ou ainda analisar segundo níveis de maturidade da empresa (CMM) [CMMI 2006], do indivíduo (PSP) [Pomeroy-Huff, Mullaney *et al.* 2005], ou da equipa (TSP) [Humphrey 1999], tal como sugere o SEI. O *Personal Software Process* (PSP) pretende classificar o indivíduo, não tanto quanto aos conhecimentos que possui de desenvolvimento de software, mas sim quanto ao processo de desenvolvimento do software. É possível então dizer que o PSP classifica o indivíduo quanto à metodologia utilizada no desenvolvimento de software.

1.4 Estrutura do Documento

Este documento foi definido atendendo a uma estrutura normalmente adoptada de uma dissertação, seguindo os mesmos passos propostos por [Berndtsson, Hansson *et al.* 2007]. Para além deste capítulo onde o problema é introduzido e a contribuição e os objectivos da investigação são estabelecidos, o documento é constituído por um capítulo que descreve o estudo do estado da arte de um conjunto de conceitos identificados com relevância para o trabalho de investigação. Os capítulos seguintes contêm a descrição da proposta, onde são exemplificadas todas as abordagens, bem como a validação e a experimentação dos cenários. É nestes capítulos que é feita a discussão de resultados obtidos. Para finalizar o documento, existe um capítulo de conclusões onde é efectuado um balanço do trabalho realizado, quais as contribuições da investigação e também são apresentadas futuras investigações relacionadas com os conceitos abordados.

No capítulo 2 estão apresentados os conceitos teóricos necessários para a realização do trabalho posterior. Primeiramente, é feita uma análise às principais linguagens de modelação de processos de negócio, à qual se segue uma comparação destas com o objectivo de verificar qual delas se adequa melhor para se utilizar na execução dos processos. Se, para cada linguagem é realizada uma análise às suas principais características e aos aspectos que as diferenciam. Também é apresentada uma primeira abordagem à metodologia de implementação dos processos (BIM), modelos de referência de processos e a plataforma para executar ao nível de software os processos de negócio (ESB). De seguida, é apresentado o método de desenvolvimento de software baseado em modelos (MDD) [Atkinson e Kühne 2003], onde se analisa as principais características, alguns standards (da OMG) e os principais conceitos (como o CIM, PIM e PSM). Por fim, é abordado o tema da transformação de modelos, onde são apresentados conceitos e técnicas de automatização de transposição de modelos para um sistema de software.

No capítulo 3 é dado início ao caso de demonstração. O trabalho tem como caso de demonstração uma empresa do sector industrial, a *Bosch Car Multimedia Portugal, Lda*. Dado que a empresa não tinha ainda adoptado uma linguagem de modelação de processos, procedeu-se à escolha da linguagem mais apropriada. A comparação foi efectuada não só ao nível tecnológico de cada linguagem, mas também ao nível intrínseco dos futuros utilizadores da linguagem e ao nível do próprio negócio em que a empresa está inserida. Acerca da comparação entre as linguagens, no anexo B encontram-se descritos os vinte padrões de *workflow* definidos por [Aalst, Hofstede *et al.* 2003], para melhor se perceber como a primeira comparação foi realizada. Ainda relativamente à comparação, no anexo C está reproduzido o questionário realizado aos colaboradores do Departamento de Informática da *Bosch Car Multimedia Portugal, Lda.* no âmbito da escolha da linguagem segundo o perfil dos utilizadores e, no anexo D, a tabela de respostas do questionário que relata as respostas obtidas ao referido questionário. Escolhida a linguagem, é feita uma apresentação mais detalhada, com as especificações que são necessárias para ter conhecimento na sua utilização. Além do tema das linguagens, neste capítulo também é abordada a forma como os processos modelados na linguagem serão implementados. Para tal, é descrita a metodologia a ser seguida, a *Business Implementation Methodology* (BIM), bem como uma descrição da forma como esta vai ser seguida ao longo do restante trabalho. Para finalizar a primeira parte da demonstração, um conjunto de processos de negócio genéricos, tal como proposto na BIM, junto com a discussão da adopção de modelos de referência, são modelados.

Durante o capítulo 4 são seguidas as restantes fases da metodologia adoptada, bem como é incluída a proposta de investigação relativa à transição para a última das fases, a utilização do MDA para obter uma transformação em software. Numa primeira parte, é discutido como os processos genéricos podem ser utilizados para posterior modelação de processos de negócio "à medida" do negócio do cliente, bem como uma abordagem em que o processo de negócio anteriormente modelado "à medida" resulta num processo de negócio que seja executável no sistema de informação da organização. O referido processo de negócio executável é descrito na linguagem BPEL, pelo que o correspondente código BPEL e WSDL é reproduzido no anexo E. Posteriormente, é discutida uma forma de utilizar os conceitos de PIM e PSM da MDA para se atingir a última fase da BIM, a Implementação. Para demonstrar o resultado, o processo de negócio é implementado numa plataforma de execução dos processos, o *Enterprise Service Bus* (ESB). Todos os componentes do ESB necessários para obter um modelo PSM são descritos e o respectivo código de todos os componentes descritos são reproduzidos no anexo F.

O capítulo 5 pretende apresentar a importância da investigação realizada. É realizado um balanço dos objectivos previamente estabelecidos que foram efectivamente cumpridos, bem como apresentar as contribuições científicas fornecidas pelo documento. Como não existe documento ou investigação perfeita, é apresentado um conjunto de conceitos que podem ser alvo de investigação futura.

No anexo A é reproduzido integralmente uma versão "*draft"* do artigo científico que irá ser, posteriormente, submetido em conferências cujo tema é referente à dissertação, intitulado "*MDA Transformations in Order to Obtain Information Systems*". O artigo foi desenvolvido em conjunto com o Eng. Francisco Duarte, do Departamento de Sistemas de Informação da Universidade do Minho, em Guimarães, e responsável pela secção de Sistemas de Informação do Departamento de Informática da *Bosch Car Multimedia Portugal, Lda.*, e com o Prof. Dr. Ricardo Machado, igualmente do Departamento de Sistemas de In formação da Universidade do Minho e orientador da dissertação de mestrado.

2. Execução de Processos e Transformações de Modelos

O objectivo deste capítulo é fornecer um contexto teórico acerca dos temas abrangidos. O primeiro tema abordado neste capítulo é o das linguagens de modelação de processos de negócio, onde é feita uma breve descrição das principais características das principais linguagens. Conceitos de implementação dos processos de negócio também são apresentados. A metodologia de implementação dos processos é apresentada, bem como as suas fases que a constituem e os principais conceitos a reter. De seguida, é apresentado o método de desenvolvimento de software baseado em modelos, alguns standards e os principais conceitos (como o *Computation Independent Model* - *CIM*, *Platform Independent Model* - PIM e *Platform Specific Model* - PSM) [OMG 2003]. Por fim, é abordado o tema da transformação de modelos.

2.1 Introdução

Este documento é baseado em três conceitos principais, nomeadamente o *Model-Driven Development* (MDD) [Atkinson e Kühne 2003], a transformação de modelos do MDD, e estes aplicados no desenvolvimento de software em organizações orientadas a processos de negócio. O tema dos processos de negócio encontra-se dividido nas linguagens de modelação existentes e na implementação dos processos num sistema de informação.

A Engenharia de Software preocupa-se actualmente em explicar *o quê* que a máquina deve fazer, ao invés de especificar *como* o irá fazer. É neste âmbito que surge o desenvolvimento de sistemas de software baseado em modelos. Um modelo é um conjunto de elementos que descrevem algo que irá ser desenvolvido. Deverá ser abstracto, compreensível, preciso, previsível e com menores custos [Gasevic, Djuric *et al.* 2006]. O MDD surge como abordagem vantajosa no desenvolvimento de software e é com base nessa abordagem que será modelado o sistema de informação. O *Model-Driven Architecture* (MDA) [OMG 2003] é uma iniciativa que surge para que seja possível utilizar os principais standards da *Object Management Group* (OMG) [OMG.org], como por exemplo *Meta Object Facility* (MOF) [OMG 2005] e o *Query/Views/Tranformations* (QVT) [OMG 2005], descritos neste documento.

No *Computation Independent Model* (CIM) [OMG 2003], são especificados o ambiente e os requisitos do sistema, sendo que os detalhes da estrutura do sistema são ocultados. O *Platform Independent Model* (PIM) [OMG 2003] é um modelo que a sua especificação não se altera de uma plataforma para outra. Possui um grau de independência da plataforma de forma a que o modelo possa ser executado em várias plataformas de diferentes tipos. O *Platform Specific Model* (PSM) [OMG 2003] resulta do processo de transformação do PIM para que as especificações do PIM sejam iguais às especificações da plataforma específica.

Para passarmos do PIM para o PSM é necessário um processo de transformação do modelo. Normalmente, o processo de transformação de modelos é realizado seguindo os conceitos de mapeamento e transformação.

As linguagens de representação de negócio permitem modelar os processos, normalmente utilizando para especificar as tarefas e decisões a tomar em relação aos dados. Permitem também verificar o estado do processo, dado estarem relacionados a *Web Services*. A descrição do relacionamento dos processos com os *Web Services* envolventes inclui especificar como um processo de negócio utiliza os *Web Services* para alcançar seus objectivos [Maldaner e Pasqual 2006]. A linguagem de programação que estas linguagens normalmente utilizam é o XML.

Para implementação dos processos de negócio descritos segundo uma qualquer linguagem, uma possível solução é a utilização de um *Enterprise Service Bus* (ESB). No entanto, ainda na fase inicial de "construção" do processo, é proposta a utilização de modelos de referência de processos, de modo a utilizar normas de boas-práticas organizacionais.

2.2 Linguagens de Representação de Processos de Negócio

A escolha de uma linguagem de modelação de processos de negócio mais adequada para a *Bosch Car Multimedia Portugal, Lda.* é o primeiro objectivo proposto. Para tal, foi realizado um levantamento e análise às principais linguagens, como BPML (*Business Process Modeling Language*) [BPMI 2001], BPEL (*Business Process Execution Language*) [OASIS 2007] XPDL (*XML Process Definition Language*) [WfMC 2008], YAWL (*Yet Another Workflow Language*) [Aalst e Hofstede 2002] e CPN (*Colored Petri Nets*) [Jensen 1992]. Para melhor compreensão, é feita também uma descrição da técnica EPC (*Event-driven Process Chain*) [Nüttgens, Feld *et al.* 1998], que é uma técnica de descrição utilizada na modelação das linguagens e é inclusivamente um técnica já conhecida da organização onde se vai realizar o caso de demonstração, e a notação BPMN (*Business Process Modeling Notation*) [White 2004], pois o BPML, BPEL e XPDL são baseados nesta notação.

Event-driven Process Chain (EPC)

O método Event-driven Process Chain (EPC) permite modelar processos de negócio com o propósito de implementação num ERP. É suportado pelo ARIS (*ARchitecture for integrated Infomation Systems*) e pelo SAP R/3 [Nüttgens, Feld et al. 1998]. A abordagem ARIS

foi desenvolvida na universidade de Saarbrucken, Alemanha, em 1992 com o objectivo principal de permitir a descrição e desenvolvimento de sistemas de informação que estivessem integrados à estrutura da organização através de seus processos de negócio [Júnior, Almeida *et al.* 2008]. O EPC obteve um crescimento quanto à sua utilização como linguagem de modelação devido ao sucesso destes dois produtos.

No EPC, os processos são sequências de eventos que desencadeiam (*triggers*) funções, representando as dependências lógicas e temporais das actividades de um processo. Ao introduzir operadores booleanos (AND, OR e XOR), a estrutura EPC pode-se expandir para um complexo *workflow* que ilustra as principais decisões de negócio. A ilustração de fluxos de dados, de unidades organizacionais e de utilização de sistemas TI representa uma expansão do EPC para descrição dos componentes semântica.

BPMN (Business Process Modeling Notation)

BPMN [White 2004] é a especificação standard *da Business Process Management Iniciative* (BPMI) [BPMI.org] para, tal como o próprio nome indica, notações de modelação de processos de negócio. O BPMN é a linguagem para definição de processos de negócios na fase de análise e está incorporada em linguagens como o BPML, BPEL e XPDL [Junior 2005]. De referenciar que, no caso das linguagens BPEL e XPDL, a incorporação do BPMN apenas é possível graças à utilização de meta-modelos de processos de negócio [Harmon 2004].

BPMN tem como principal objectivo o de fornecer uma notação facilmente compreensível por todos os utilizadores empresariais, analistas de negócio que tenham criado os primeiros rascunhos dos processos, técnicos responsáveis pelo desenvolvimento e aplicação da tecnologia que irão realizar os processos e, finalmente, empresários que vão gerir e acompanhar os processos. O BPMN é, então, um mecanismo simples que permite o desenvolvimento dos modelos processos de negócio e garante a complexidade inerente aos processos.

O BPMN define um *Business Process Diagram* (BPD) [Aalst, Dumas *et al.* 2006], que é uma espécie de fluxograma constituído por um conjunto de elementos gráficos. Estes elementos permitem o fácil desenvolvimento de diagramas simples que parecem familiares para a maioria dos analistas de negócio incorporando construções adaptadas à modelação de processos de negócio, tais como AND-split, AND-join, XOR-split, XOR-join [Aalst, Dumas *et al.* 2006].

BPML (Business Process Modeling Language)

O *Business Process Modeling Language* (BPML) é uma meta-linguagem da BPMI [BPMI.org] para modelação de processos de negócio, suportada por empresas como a SAP, Intalio, Sun e Versata. A especificação BPML suporta XML 1.0, XML-Namespaces, XMLSchema 1.0 e XPath 1.0 [Arkin 2002].

BPML fornece um modelo abstracto para processos de negócio colaborativos e transaccionais que abrangem várias aplicações, departamentos e parceiros de negócio. Como abrange várias dimensões da gestão de processos de negócio, o BPML consegue unir as infra-estruturas de TI herdadas com os protocolos B2B (Business-to-business) emergentes, tais como ebXML, RosettaNet e BizTalk [BPMI 2001].

A modelação de BPML é constituída por: actividades (simples ou complexas), processos, contextos, propriedades e sinais. As actividades são componentes de desempenho de funções específicas. Um processo é uma actividade complexa que pode ser invocada por outros processos. Um contexto define um ambiente para a execução das actividades relacionadas, pode ser utilizado para trocar informações e coordenar execuções. Propriedades são utilizadas para troca de informações e só existe num contexto. Os sinais são usados para coordenar a execução das actividades.

BPEL (Business Process Execution Language)

Business Process Execution Language (BPEL), ou formalmente *Web Services Business Process Execution Language* (WS-BPEL) [Oasis 2007], é uma linguagem de especificação de *Web Services* (WS) [Couloris, Dollimore *et al.* 2001] que surge como junção das linguagens WSFL da IBM e XLANG da Microsoft e foi inicialmente denominada de BPEL4WS, passando a ser BPEL por uma questão de simplicidade. Esta convergência incorporou as especificações *WS-Coordination*, *WS-Transaction* e *WS-Security* [Martins 2005] em resposta à especificação BPML (da BPMI) [BPMI.org] mais abrangente.

A norma BPEL fornece uma linguagem que é capaz de especificar processos de negócio e o estado dos seus processos, através da relação destes com os *Web Services*. A descrição do relacionamento dos processos com os *Web Services* envolventes inclui especificar como um processo de negócio utiliza os *Web Services* para alcançar seus objectivos [Maldaner e Pasqual 2006]. Os processos de negócio especificados em BPEL (Figura 1) são executáveis em qualquer ferramenta BPEL, que especifica qual o serviço que o *Web Service* presta ao processo BPEL.

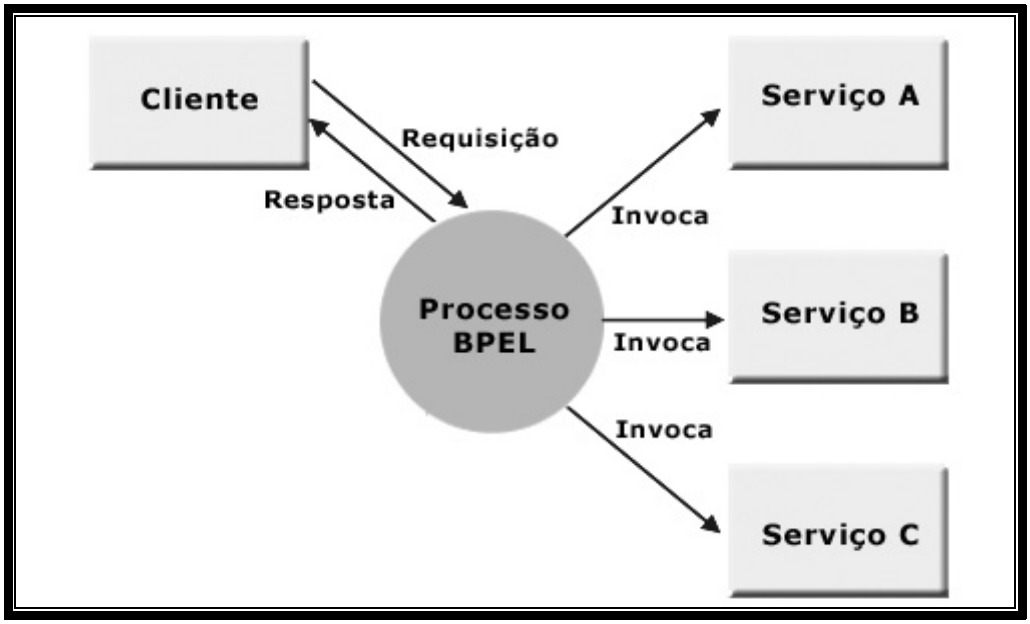

Figura 1. Ilustração de um Processo BPEL [Maldaner e Pasqual 2006]

O Processo BPEL é constituído por 2 tipos de arquivos:

- Um arquivo WSDL [W3C 2007], que contém toda a especificação do tipo de ligação entre as interfaces, propriedades, tipo de portas e operações, mensagens e parte de interesse do processo, incluindo serviços implementados e invocados pelo processo;
- Um arquivo BPEL, que codifica o processo em XML, incluindo todas as actividades e principais variáveis envolvidas. Por herdar as características da Microsoft XLANG e da IBM WSFL, o BPEL utiliza tanto o conceito do cálculo de *pi* [Milner, Parrow *et al.* 1989] como das redes de Petri [Petri 1962], aproveitando as características e vantagens de ambos.

XPDL (XML Process Definition Language)

O *XML Process Definition Language* (XPDL) [WfMC 2008] foi standardizado pelo *Workflow Management Coalition* (WfMC) [WfMC.org] com o objectivo de desenvolver uma linguagem que suportasse a troca de definições dos *workflows* dos processos de negócio.

Em 2008, é lançada a versão final para a versão 2.1 do XPDL [WfMC 2008]. Na sua definição, podemos ver que não se deu apenas a transformação do WPDL numa linguagem baseada em XML, mas também foram adicionados e modificados alguns conceitos em relação ao WPDL. O principal foco desta linguagem é a definição da distribuição de trabalho através de

sua estrutura baseada em grafos [Junior 2005]. O XPDL tem como principais elementos: **Package, Application, Workflow-Process, Activity, Transition, Participant, DataField,** e **DataType** [Aalst 2003].

YAWL (Yet Another Workflow Language)

YAWL [Aalst e Hofstede 2002] foi proposta baseada numa rigorosa análise de sistemas de gestão de *workflow* e linguagens de *workflow* existentes. O YAWL é baseado na teoria de redes de Petri, suportando alguns símbolos diferenciados de outras linguagens. As linguagens baseadas em redes de Petri são melhor executadas quando seguem os padrões de *workflow* baseados em estados [Maldaner e Pasqual 2006]. Contudo, alguns padrões não são facilmente mapeados em redes de Petri (de alto nível). É com essa intenção que então se revela necessário o desenvolvimento de uma nova linguagem, tendo como ponto de partida as redes de Petri e adicionando mecanismos que permitissem um suporte mais directo e intuitivo aos padrões já identificados. É dessa necessidade que surge o YAWL.

Entre todas as linguagens YAWL é a única que segue os 20 padrões impostos pelo P4 (*Process 4* – é um grupo constituído por van der Aalst, ter Hofstede, Kiepuszewski e *o Barriosa Group*), pois a própria linguagem foi proposta justamente por esse grupo. YAWL utiliza uma arquitectura orientada a serviços (Figura 2), que controla os fluxos através de serviços (*YAWL Services*), nos quais as entidades externas oferecem serviços ou requerem serviços.

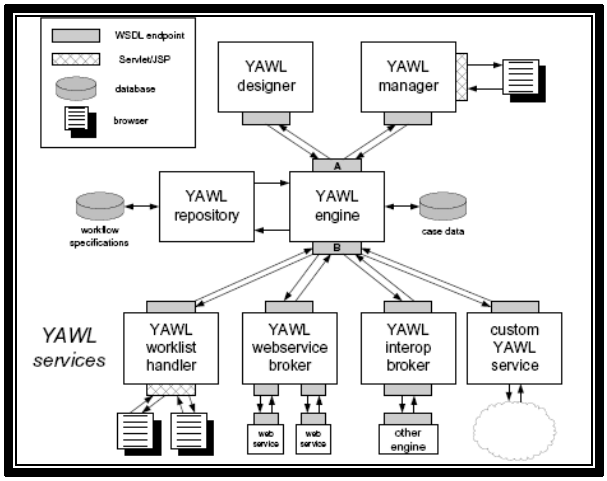

Figura 2. Arquitectura YAWL [Aalst, Aldred *et al.* 2004]

Redes de Petri Coloridas (CPN)

Redes de Petri (*Petri Nets*) são uma classe de grafos generalizados, criados por Carl Adam Petri, no âmbito da sua tese de doutoramento [Petri 1962], para enfrentar concorrência

em sistemas. Porque possuem uma descrição matemática associada à representação gráfica, as redes Petri são uma ferramenta que está bem adaptada, não só para modelação, mas também para a análise e estudo de eventos discretos de sistemas (DES), especialmente aqueles nos quais eventos podem ocorrer simultaneamente. Em outras palavras, utilizando redes de Petri para modelar um sistema conduz a uma descrição matemática desse sistema, que permite o estudo analítico da sua estrutura e propriedades, inclusive aqueles associados a ocorrência simultânea de eventos do sistema.

A representação de uma rede de Petri é um grafo direccionado bipartido. Matematicamente, uma Rede de Petri R=(P, T, I, O) é definida formalmente por: [Oliveira 2006]

- Um conjunto finito P de lugares ("*places*");
- Um conjunto finito T de transições;
- Uma função de entrada I: T → P(P)
- Uma função de saída O: T → P(P*)*

Na sua modelação, um lugar é representado por um círculo, uma transição por uma barra (ou rectângulo) e uma marca (ou *token*) por um ponto no lugar (Figura 3). As ligações são efectuadas por arcos, que terão um determinado peso (a não ser que o peso seja de valor zero).

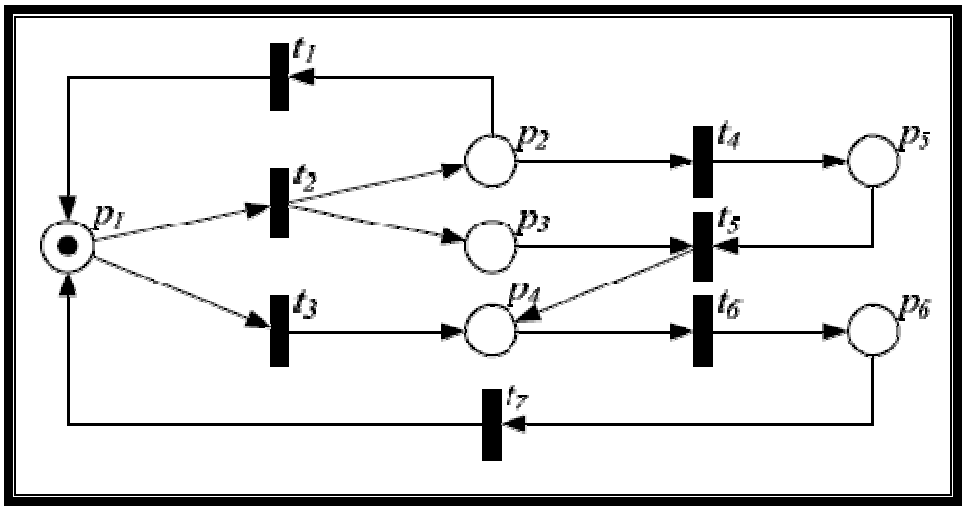

Figura 3. Exemplo de uma Rede de Petri [Oliveira 2006]

A marca, ou *token*, é utilizada para descrever o estado do sistema no lugar onde se encontra, sendo que a normal utilização das redes de Petri é através de condições do estado

dos lugares, isto é, se o lugar está marcado ou não. Daí que uma posterior denominação das redes de Petri originais seja *condition-event nets* (CE-nets).

Posteriormente surgem as redes Lugar-Transição (*Place-transition net* – PT-net), onde os lugares podem ter mais de uma marca e os arcos são ponderados, indicando quantas marcas serão retiradas e/ou inseridas com o disparo da transição à qual o arco é incidente.

O primeiro meta-modelo de redes de Petri de alto nível surgem em 1981, as *Predicate/Transition Nets* (PrT-nets). Estas redes são basicamente a junção das PT-nets e as CE-nets, pelo que possuem os mesmos conceitos destas duas redes.

Às PrT-nets seguiram-se as *time Petri nets*, *colored Petri nets*, *object-oriented Petri nets* e *upgraded Petri nets* [Oliveira 2006].

As primeiras redes de Petri coloridas (CPN) surgem no intuito de resolver limitações das PrT-nets [Jensen 1992]. São directamente inspiradas nas PrT-nets, mas utilizam funções associadas aos arcos, em detrimento de expressões. Em alguns artigos, as CPN podem aparecer denominadas *High-Level Petri-nets* (HL-nets). As CP-nets permitem manipular tipos de dados complexos, pois cada *token* tem associado um valor, que será a cor do *token*, que pode ser analisada e modificada pela transição que estiver a ocorrer. Com CP-nets é possível obter um modelo largo utilizando combinações de pequenos sub-modelos, as interfaces são bem-definidas, bem como a semântica, e os sub-modelos podem ser reutilizados [Jensen 1992].

2.3 Implementação dos Processos

O desenvolvimento de processos de negócio é uma tarefa que vai muito além da sua representação. A definição dos processos deve ser constituída por um conjunto de tarefas, que vão desde os requisitos, modelação ou implementação. Tal como há que ter em conta conceitos como notações, ferramentas (de representação e de execução), arquitecturas, sistema de informação. É nesse sentido que se torna necessária a utilização de uma metodologia para a implementação dos processos. A metodologia apresentada de seguida é constituída por quatro fases de transições entre processos. A metodologia inicia-se com processos previamente definidos que servirão como base, utilizando modelos de referência de processos. A metodologia é finalizada com implementação em software dos processos, onde o uso de um *Enterprise Service Bus* é proposto.

Business Implementation Methodology (BIM)

A BIM [Duarte, Machado *et al.* 2009] é uma metodologia de implementação em software de processos de negócio de uma organização. Esta metodologia propõe o uso de boas-prácticas do domínio do negócio, e permite a customização do processo de modo a que as necessidades da organização sejam atendidas pelo processo em causa. É uma metodologia flexível, que permite várias tecnologias independentes umas das outras, bem como a utilização de produtos de vários vendedores.

O objectivo da BIM é o de servir como um guia para a implementação de sistemas de software para organizações que funcionam orientadas a processos. Permite também o uso de diferentes notações em cada uma das suas fases, pois a transformação dos modelos em cada fase é independente de notações. Também permite a utilização de tecnologias de diversos vendedores, de modo a suportar processos de negócio de um modo real. A BIM utiliza conceitos tais como processos de negócio genéricos, aspectos organizacionais que estão presentes nos processos, a arquitectura do processo e a implementação em software das entidades do negócio.

A BIM é constituída por quatro fases: selecção (*Selection*) dos processos de negócio genéricos adequados; definição (*Definition*) dos processos a utilizar; a concretização (*Concretisation*) do processo no sistema de informação da organização onde vai funcionar; e a implementação em software (*Implementation*) das várias entidades que compõe o processo (Figura 4).

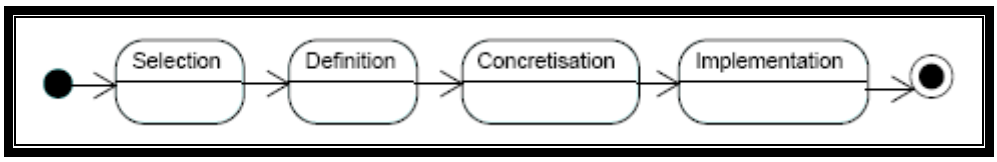

Figura 4. As 4 Fases da BIM [Duarte, Machado *et al.* 2009]

A particularidade que define qual a fase da metodologia em que o processo se encontra é o estado, no sentido em que responda a determinados requisitos da metodologia, da arquitectura dos processos. Definidos os requisitos necessários para finalizar cada fase, é atribuída à arquitectura do processo um estado. Os estados definidos pela metodologia são genéricos (*Generic*), instanciados (*Instantiated*), executáveis (*Runnable*) e implementados em software (*Software Implemented*). A cada estado da arquitectura dos processos (*Process Framework* – PF) corresponde então cada uma das fases da BIM.

Na fase de Selecção, o analista do processo escolhe um modelo de referência de processos adequado para o negócio. O analista vai também, em conjunto com o cliente, definir um conjunto de aspectos organizacionais a incluir no modelo de processos genérico. Esta fase termina quando o modelo genérico está escolhido.

Na fase de Definição, o modelo de processos genérico deve passar a ser instanciado. Para tal, o cliente e o arquitecto do processo devem definir quais as necessidades do processo. De referir que o modelo genérico deve ser sempre o modelo utilizado como base para a instanciação, de modo a que o modelo resultante utilize devidamente um modelo de referência entre organizações [Duarte, Machado *et al.* 2007].

A fase de Concretização já pressupõe que o modelo de processos se encontra na sua representação final, de modo a definir claramente a sua função no sistema de informação da organização. A nível de definição do processo, este não sofre mais alterações. É nesta fase que se decide quais os processos que irão ser implementados em software e quais são puramente manuais. Também devem ser apresentados processos de negócio alternativos que igualmente possam ser concretizados no sistema de informação no imediato.

Na fase de Implementação em Software acontece a obtenção do código-fonte do respectivo processo. Nesta fase apenas são tratados os processos que necessitam suporte em software, conforme decidido na fase anterior. A obtenção do código pode ser feita mapeando directamente a partir do modelo *Runnable*. A implementação do processo é feita a partir de um determinado modelo ou utilizando já código desenvolvido anteriormente e posteriormente customizado de forma a possuir as características desejadas.

Modelos de Referência de Processos

O conceito de *Process Reference Models* (modelos de referência de processos) é um conceito com o objectivo de fornecer boas-práticas que possam ser utilizadas entre organizações. Este conceito surge com o objectivo de as organizações partilharem boas-práticas entre si. Estes modelos podem servir como base para as organizações conceberem os seus próprios processos. Os *process reference models* também podem ser levados em conta num processo de re-definição dos processos de negócio de uma empresa. Aliás, os *process reference models* integram conceitos de *Business Process Reengineering, benchmarking* e análise de boas-práticas [SCC 2008]. Eles devem permitir que se derive o estado futuro dos processos ("*to-be*") a partir do seu estado presente ("*as-is*"), definir metas internas a atingir tendo em conta resultados operacionais de organizações similares e

caracterizar praticas de gestão e soluções que resultem em performances melhores que os seus concorrentes.

Um bem conhecido *process reference model* é o *Supply-Chain Operations Reference model* (SCOR) [SCC 2008] e é um modelo de referência para empresas de produção. O modelo sugere que estas organizações devem operar baseadas em cinco processos de negócio: Planear (*Plan*), Procurar (*Source*), Produzir (*Make*), Entregar (*Deliver*) e Retorno (*Return*).

O *Design-Chain Operations Reference model* (DCOR) [SCC 2006] é um modelo de referência para os processos de desenho dos produtos. Este modelo sugere cinco processos: Planear (*Plan*), Desenho (*Design*), Investigar (*Research*), Integrar (*Integrate*) e Melhorar (*Amend*).

O *Information Technology Infrastructure Library* (ITIL) [ifSMF 2007] possui um conjunto de boas práticas para empresas de Tecnologias de Informação. Foca-se em cinco estados para o serviço prestado pela organização: Estratégia (*Service Strategy*), Desenho (*Service Design*), Transição (*Service Transition*), Operação (*Service Operation*) e Melhoria Continua (*Continual Service Improvement*) do Serviço.

O *Enhanced Telecommunications Operations Map* (eTOM) [TMForum] é muito semelhante ao ITIL, sendo mais aplicável no ramo das telecomunicações. Fornece um *framework* que descreve como a empresa deve organizar os seus departamentos e quais os processos executados por eles. "Divide" a empresa em três grandes grupos de processos: Gestão Empresarial (*Enterprise Management*), Operações (*Operations*) e Estratégia, Infra-estruturas e Produto (*Strategy, Infrastructure & Product*).

O grupo Bosch também tem definido para a sua sub-divisão *Car Multimedia* um grupo de processos transversais às diversas empresas espalhadas por diversos países, o UBK (*Unternehmensbereich Kraftfahrzeugausrüstung* – Divisão de Tecnologia Automóvel). Descreve um grupo de processos de referência que podem depois ser redefinidos tendo em conta aspectos específicos, como por exemplo da própria organização ou do país onde labora.

Integração de Aplicações Através de Enterprise Service Bus (ESB)

O conceito de integração de aplicações empresariais (*Enterprise Application Integration* – EAI) [Lee, Siau *et al.* 2003] surge a partir da necessidade de integração de sistemas com menores custos e menor programação. EAI engloba planos, métodos e ferramentas com o objectivo de modernizar, consolidar e coordenar todas as funcionalidades computacionais de uma empresa. Por norma, uma empresa possui aplicações de legado e bases de dados, e pretende continuar a utilizá-las enquanto adiciona ou executa migrações

para um conjunto de novas aplicações. Um projecto de EAI pode envolver uma mudança total na perspectiva do negócio da empresa e das duas aplicações, determinar como as aplicações existentes encaixam nessa nova perspectiva e delinear a uma forma de reutilizar eficientemente o que já existe enquanto novas aplicações e novos dados são adicionados.

Anteriormente, a integração de diferentes sistemas requeria reescrever os códigos dos sistemas de origem e de destino, o que representava grandes gastos de tempo e de dinheiro. O EAI utiliza um *middleware* que serve como uma ponte entre diferentes aplicações na integração de sistemas. Todas as aplicações podem comunicar livremente com outras através de uma camada de interface comum em vez de através de integração ponto-a-ponto, pelo que não é necessário uma programação extensa.

Um tipo de tecnologia *middleware* de EAI é o *Enterprise Service Bus* (ESB). A utilização de *hubs* e *brokers* no caso dos EAI dá lugar à utilização de barramentos no caso dos ESB (Figura 5). No primeiro caso a arquitectura é centralizada pois o *hub* ou o *broker* processa todas as trocas de mensagens. No segundo caso a arquitectura é distribuída dado que o ESB permite a implementação de várias funções fisicamente separadas. Na generalidade os produtos EAI são baseados em produtos proprietários (*e.g.* WebSphere Message Broker usa WebSphere MQ), enquanto que os ESB geralmente são baseados *open standards*, como *Java Message Service* (JMS) [Hapner, Burridge *et al.* 2002], XML ou *Web Services*.

Primeiramente deve ser considerado pela organização se é realmente necessário para o seu negócio a integração das suas aplicações. Se não existe essa necessidade, então implementar uma solução ESB torna-se desnecessária. Mas a realidade é que em muitas organizações existe mesmo essa necessidade, pois as soluções tecnológicas devem ser capazes de facilitar a que novos produtos possam ser lançados no mercado em menor tempo. Também a utilização de diferentes protocolos de comunicação, como JMS, SOAP, entre outros, dificulta a implementação da integração das aplicações, o que sugere a utilização de um ESB. Um produto ESB também permite a redução dos custos, dado que um ESB funciona como uma solução comum para toda a integração, facilitando a gestão e a manutenção.

Uma das principais funcionalidades de um ESB é a transparência da localização, pois fornece uma plataforma central que permite a comunicação com qualquer aplicação sem que tenha que as mensagens enviadas às mensagens recebidas. Tal como já foi referido anteriormente, um ESB permite integrar aplicações que utilizem diferentes protocolos, como JMS e SOAP. O ESB tem a capacidade de transformar as mensagens que estejam num determinado formato para um formato *open standard,* por exemplo o XPath. O ESB possui também funcionalidades de encaminhar mensagens, segurança e monitorização [Rademakers e Dirksen 2008].

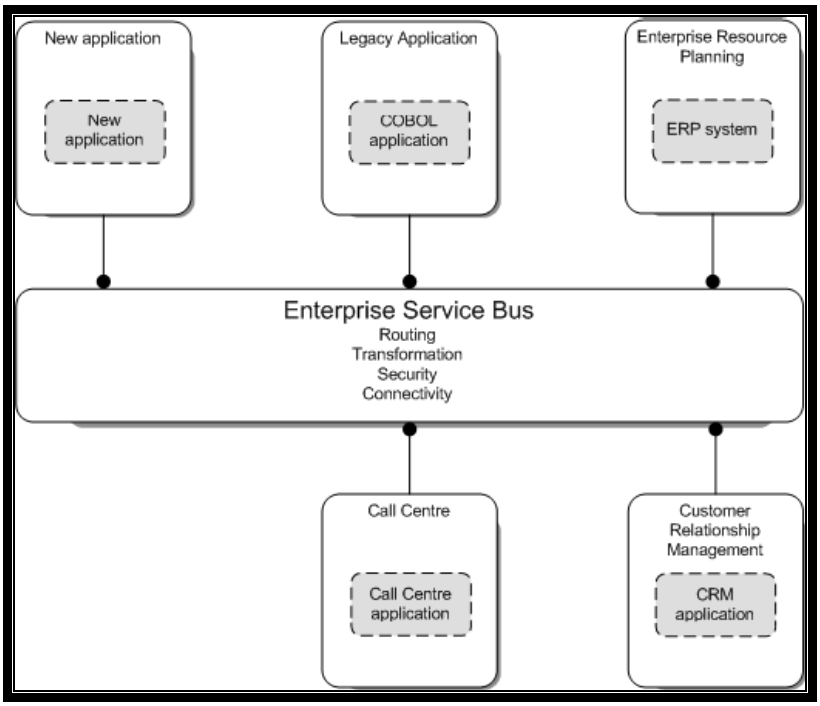

Figura 5. Exemplo de Integração Utilizando um ESB [Rademakers e Dirksen 2008]

Apache ServiceMix ESB

O *ServiceMix* é uma solução ESB aceite e utilizada por entidades respeitáveis no mercado de desenvolvimento de software, mas também é *open source* e baseado em *open standards*, o que traz benefícios na sua implementação, como baixos custos mas também assegura qualidade. O seu funcionamento tem como fundo o *Java Business Integration* (JBI) [Ten-Hove e Walker 2005]. O JBI define uma arquitectura que permite integração de sistemas a partir de componentes que são adicionados e operam uns com os outros através do método de troca de mensagens mediadas. Este método de troca de mensagens mediadas foi adoptado a partir da especificação do WSDL 2.0 de *Message Exchange Patterns* (MEP's) [Ten-Hove e Walker 2005]. Os MEP's definem um conjunto de tipos de troca de mensagens, podendo estas serem de apenas um sentido (*In-Only*), um sentido com confiança (*Robust In-Only*), pedido-resposta (*In-Out*) e de resposta opcional (*In Optional-Out*). O *ServiceMix* define dois tipos de componentes JBI: *Service Engines* (SE) e *Binding Components* (BC), tal como representado na Figura 6. Um SE fornece lógica de negócio e de processamento, *i.e.*, é o componente que processa um tipo de conjunto de dados de modo a executar um serviço Um BC fornece comunicação para o exterior, funcionando como uma "ponte" entre a entrada e saída de dados do (ou para o) ESB para o (ou para o) exterior.

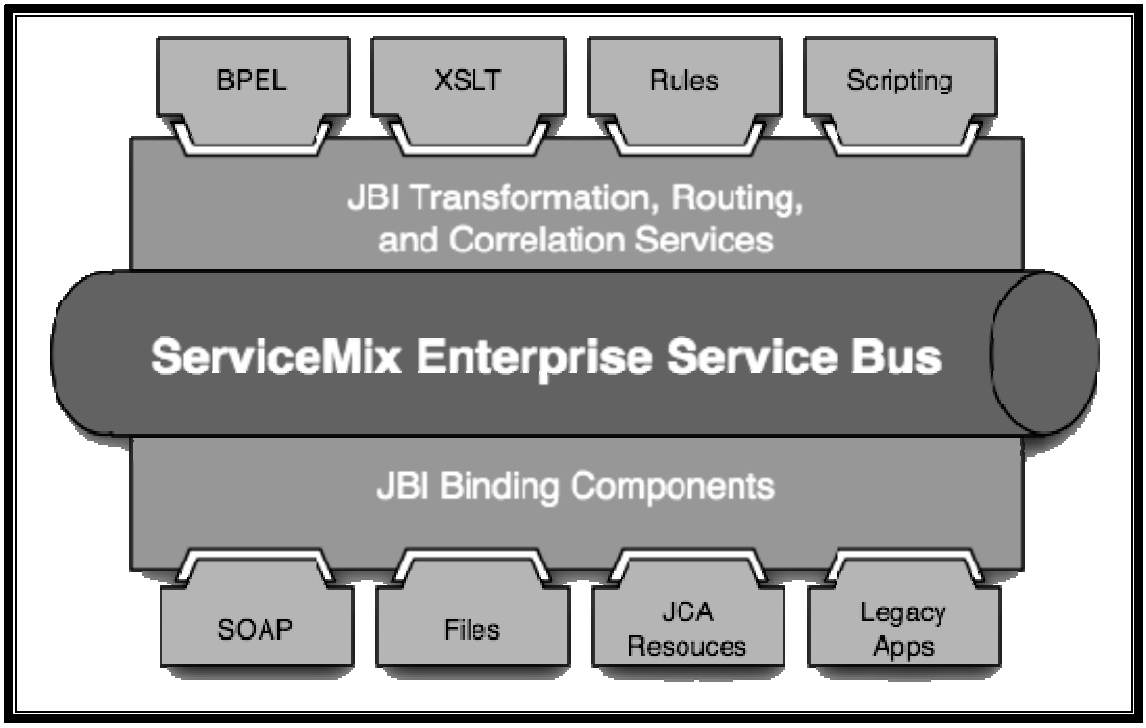

Figura 6. Framework do *Apache ServiceMix* [ServiceMix]

O *Apache ServiceMix* possui integração com uma série de outros projectos *Apache*. O *Apache ActiveMQ* é a base de troca de mensagens. E há também a integração com *Apache CXF*, *Apache ODE*, *Apache Camel* e *Apache Geronimo*. Uma grande série de componentes JBI foi desenvolvida para o *ServiceMix* que fornecem um conjunto de funcionalidades importantes já referidas anteriormente, como componentes JBI para suporte a protocolos como JMS e SOAP, e também componentes para implementar *Enterprise Integration Patterns* [Hohpe e Woolf 2004].

Na Figura 7 está representado o funcionamento do *ServiceMix* no que toca à troca de mensagens, dando uso às características JBI presentes neste ESB. Antes de mais, é de referir que os SE's e os BC's podem ter o papel de consumidores de serviços (*service consumers*), fornecedores de serviços (*service providers*) ou ambos. Estes papéis referem-se ao tipo de tratamento a que o serviço é sujeito, sendo que o fornecedor do serviço é o componente que envia uma mensagem de pedido de execução de um serviço, e o consumidor do serviço é o que executará o pedido e envia uma mensagem de resposta. A troca de mensagens entre o fornecedor e o consumidor tem que obedecer a normas próprias, que são baseadas em WSDL e XML. A mensagem deve estar num formato normalizado antes de entrar no barramento, sendo que volta ao seu formato "de origem" quando são enviadas para o exterior pelos BC's. As mensagens são normalizadas de modo a serem constituídas por três partes: *Payload*, *Message Properties* e *Message Attachments*. Estando num formato normalizado, todos os

componentes são capazes de interpretar as mensagens. Entre um SE e um BC, as mensagens são enviadas para um *Normalized Message Router* (NMR) [Ten-Hove e Walker 2005]. O NMR recebe e troca mensagens com os componentes JBI e reencaminha a mensagem para o componente pretendido para processamento, pelo que funcionam sempre como intermediários na troca das mensagens, pois os componentes não comunicam directamente entre eles. Para finalizar no que toca a mensagens, falta apenas referir os *Delivery Channels* (DC). Os DC's são os canais que fazem a comunicação do NMR com os SE's e BC's, por onde as mensagens (de pedido ou de resposta) circulam. O funcionamento dos DC's é similar ao funcionamento de um *socket* [Genender 2007].

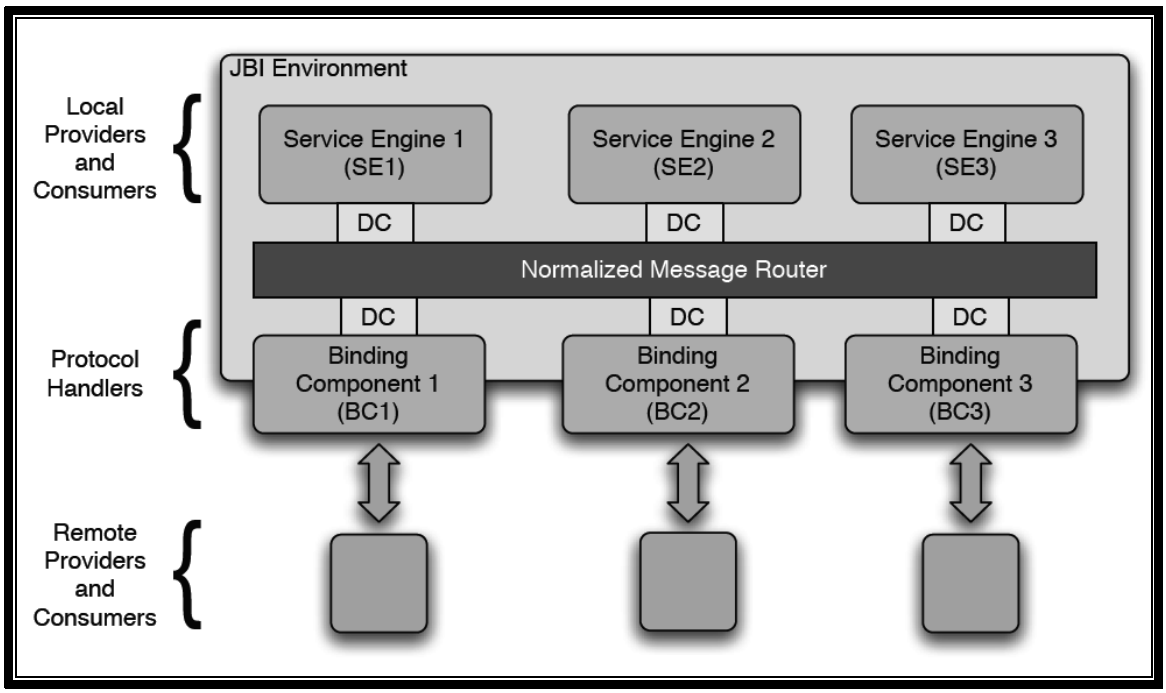

Figura 7. Troca de Mensagens em Ambiente JBI [Genender 2007]

Para implantar um componente no *ServiceMix* recorre-se à utilização de um *Service Unit* (SU) [Rademakers e Dirksen 2008] que irá fornecer instanciações do componente. O SU é um conjunto de ficheiros – XML neste caso –, cujo conteúdo é transmitido ao componente. A cada instanciação de um SE ou de um BC irá existir um SU que irá possuir as definições do componente instanciado. Para um SU poder ser utilizado no ESB, é necessário o uso de *Service Assemblies* (SA) [Rademakers e Dirksen 2008]. Um SA é uma colecção de SU's. Os componentes JBI são incapazes de interpretar SU's sem que estes estejam contidos num SA. Um SA é constituído por pelo menos um SU (Figura 8).

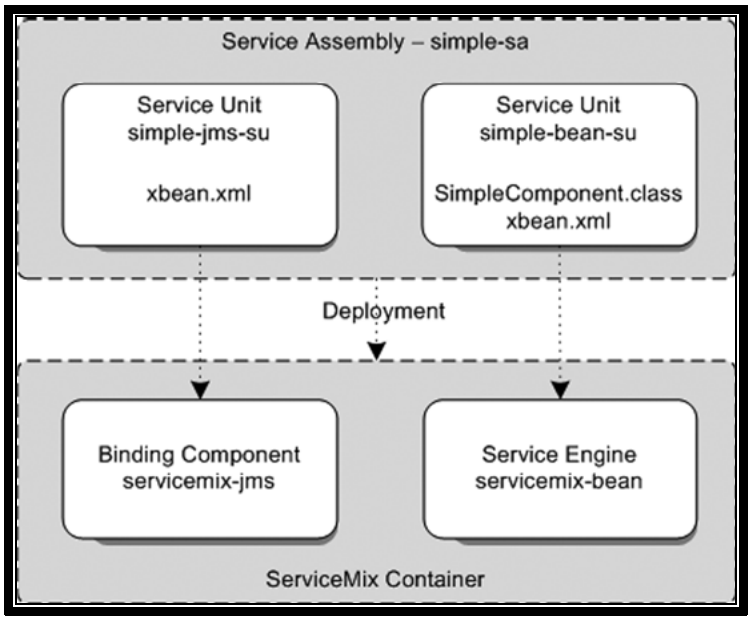

Figura 8. Composição do *Service Assembly* [Rademakers e Dirksen 2008]

No exemplo da Figura 8, retirado de [Rademakers e Dirksen 2008], o SA é constituído por dois SU's, um *JMS SU* e um *Bean SU*. O SA pode conter SU's que serão implantados como instanciações de diferentes componentes, *i.e.*, o mesmo SA pode conter SU's que irão dar origem a instanciações de BC's e a SE's. Assim que o SA é implantado no *ServiceMix*, são criadas instanciações de componentes dentro do *ServiceMix* (*ServiceMix Container*). O *JMS SU* é implantado no *ServiceMix* como um *JMS BC*, enquanto que o *Bean SU* é implantado como um *Bean SE*.

2.4 Model-Driven Development

Desde que a Engenharia de Software se preocupa em explicar *o quê* que a máquina deve fazer, ao invés de especificar *como* o irá fazer, que esta tenta aumentar o nível de abstracção da actividade que se pretende realizar. É neste âmbito que surge o desenvolvimento de sistemas de software baseado em modelos (*Model-driven Development* – MDD) [Atkinson e Kühne 2003]. Esta abordagem tem como característica o facto de recorrer a modelos para o desenvolvimento de sistemas de software, o que faz com que se eleve o nível de abstracção em relação ao desenvolvimento baseado na codificação. A utilização de modelos permite também que sistemas de software sejam desenvolvidos independentemente da plataforma em que estes irão ser implementados.

Um modelo é um conjunto de elementos que descrevem algo que irá ser desenvolvido. Deverá ser abstracto, compreensível, preciso, previsível e com menores custos

[Gasevic, Djuric *et al.* 2006]. Além do aumento do nível de abstracção, o MDD tem outra motivação que é o aumento da automação das tarefas a realizar. Isto é conseguido através da transformação de modelos e geração automática de código.

OMG's Model-driven Architecture (MDA)

Falando em *Model-Driven Development* (MDD), é comum associá-lo à iniciativa desenvolvida pelo *Object Management Group* (OMG) [OMG], o *Model-driven Architecture* (MDA) [OMG 2003]. Esta iniciativa surge de modo a que seja possível utilizar os principais standards da OMG (*The Unified Modeling Language* – UML: linguagem de modelação standard, é uma instância da linguagem MOF; *Meta Object Facility* – MOF: linguagem de meta-modelação; *XML Metadata Interchange* – XMI: intercâmbio de meta-dados; *Common Warehouse Metamodel* - CWM: linguagem de modelação para aplicações *Data warehouse*; *Object Constraint Language* – OCL: linguagem de expressões que estende o UML e o MOF neste aspecto; *Query/Views/Tranformations* – QVT: linguagem de definição de transformações, também utilizada para *queries* e *views* de modelos; e *Software Process Engineering Metamodel* – SPEM: perfil de meta-modelos e UML para descrever um processo de desenvolvimento de software concreto) [OMG] em qualquer plataforma, seja ela qual for, como por exemplo Web Services, .NET, CORBA, J2EE, entre outras (Figura 9).

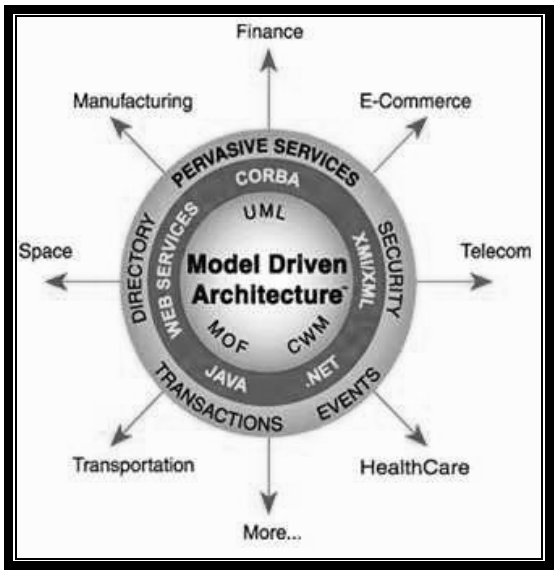

Figura 9. *OMG Model-Driven Architecture* [OMG]

A arquitectura define três níveis de abstracção para análise dos sistemas. Dados os níveis de abstracção, podem então ser definidos modelos para esse sistema. É definido o

Computer Independent Model (CIM), que foca no ambiente e nos requisitos do sistema; o *Platform Independent Model* (PIM), que foca nas operações do sistema; e o *Platform Specific Model* (PSM), que combina o PIM com os detalhes específicos da plataforma que o sistema irá utilizar.

Os Modelos CIM, PIM e PSM

O ponto de partida do MDA é o da separação entre as especificações das operações do sistema e os detalhes da forma como o sistema irá utilizar as capacidades da plataforma. Permitirá assim que: seja especificado o ambiente do sistema e os seus requisitos (CIM), o sistema seja especificado independentemente da plataforma que o suporta (PIM), sejam especificadas plataformas, seja escolhida uma plataforma para o sistema e transformar as especificações do sistema para especificações de uma determinada plataforma (PSM). Os três principais objectivos do MDA serão a portabilidade, interoperabilidade e reutilização em arquitecturas separadas [OMG 2003].

No CIM são especificados o ambiente e os requisitos do sistema, sendo que os detalhes da estrutura do sistema são ocultados, ou mesmo indeterminados. Por estas razões, pode também ser chamado de *domain model*. O utilizador do CIM não conhece os modelos utilizados para perceber a funcionalidade na qual os requisitos são articulados. O CIM serve então como uma importante ponte entre os especialistas do domínio e os seus requisitos e os especialistas em design e construção [OMG 2003].

O PIM é um modelo que a sua especificação não se altera de uma plataforma para outra. Possui um grau de independência da plataforma de forma a que o modelo possa ser executado em várias plataformas de diferentes tipos. O modelo deve ser desenvolvido respeitando as classes das diferentes plataformas onde o modelo poderá ser implementado. Uma técnica muito comum é o de implementar uma máquina virtual livre de tecnologia em várias e diferentes plataformas. A máquina virtual será utilizada como plataforma. O modelo deverá ser desenvolvido de modo a que especifique a plataforma. Mas, como a máquina virtual foi desenvolvida para ser implementada em diferentes plataformas, o modelo não irá especificar uma plataforma específica, garantindo a independência de plataformas.

O PSM resulta do processo de transformação do PIM para que as especificações do PIM sejam iguais às especificações da plataforma específica. O processo de transformação é efectuado combinando o PIM com os detalhes específicos da plataforma do sistema (Figura 10).

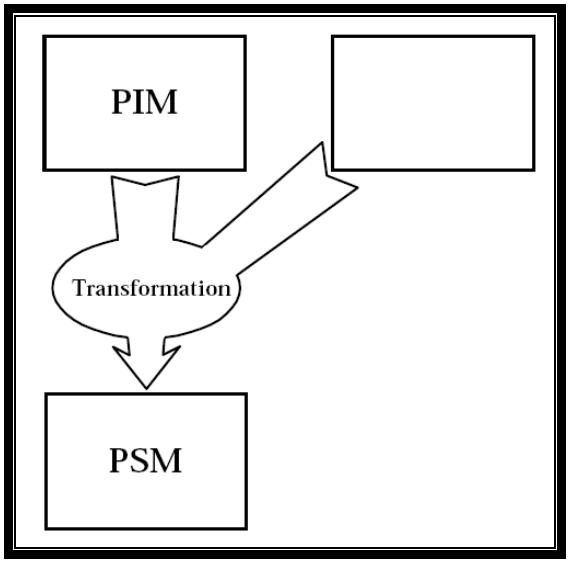

Figura 10. Transformação de um PIM para PSM [OMG 2003]

As 4 camadas

O MDA é baseado numa arquitectura de quatro camadas de meta-modelos, complementada pelos standards *Meta-Object Facility* (MOF), *Unified Modeling Language* (UML) e o *XML Metadata Interchange* (XMI). Os elementos pertencentes a uma camada são instanciados por elementos da camada abaixo (Figura 11).

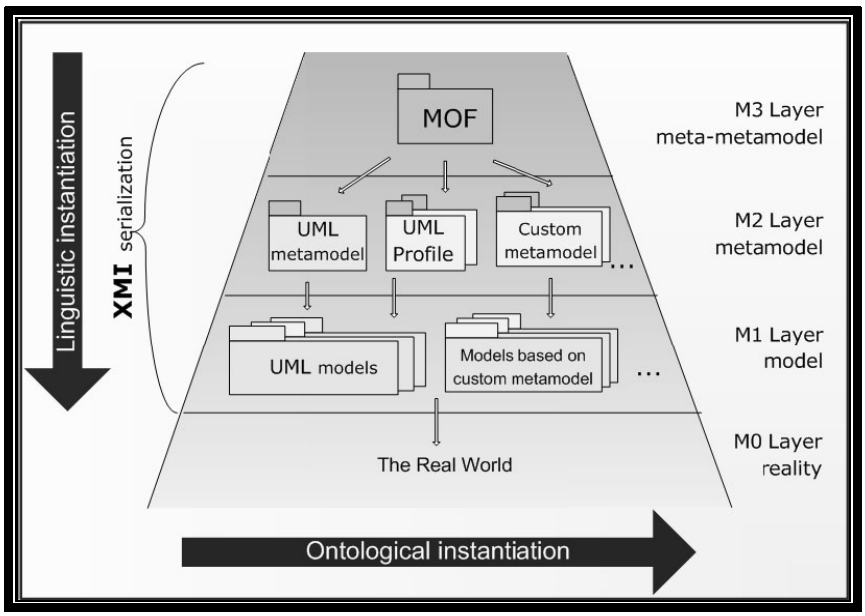

Figura 11. As 4 camadas MDA e as Relações de Instância, Linguística e Ontológica [OMG 2006]

A camada M3 representa o nível mais elevado de abstracção, o *meta-metamodel*, isto é, o MOF. O MOF, por si só, garante a abstracção e a definição da linguagem e o *framework* para especificar, construir e gerir tecnologicamente meta-modelos independentes.

Todos os meta-modelos *user-defined* definidos no MOF estão situados na camada M2. Um desses meta-modelos é o UML.

Os modelos do mundo real que estejam representados por conceitos pertencentes a um meta-modelo situado na camada M2 situam-se na camada M1.

Na camada M0 encontram-se elementos do mundo real, que se encontram modelados na camada M1. Os elementos podem ser coisas concretas do mundo real (como por exemplo, uma instância da classe *Pessoa* ou *Animal*), bem como objectos de linguagens de programação.

Meta-modelos são um modelo de especificação para uma classe de sistemas em estudo, onde cada sistema do conjunto é ele próprio um modelo válido expresso numa qualquer linguagem de modelação [Gasevic, Djuric *et al.* 2006]. Os meta-modelos ditam o que pode ser expresso para um modelo numa certa linguagem de modelação de modo a que o modelo seja considerado válido. Os meta-modelos são eles próprios modelos, logo também podem ser representados em linguagens de modelação. No caso do MDA, os meta-modelos estão situados na camada *meta-metamodeling* (M3), e a linguagem de modelação dos meta-modelos é o MOF.

O MOF é um conjunto mínimo de conceitos que podem definir outra linguagem de modelação. Fornece um framework para gestão de meta-dados e um conjunto de serviços de meta-dados para possibilitar o desenvolvimento de modelos e de sistemas baseados em meta-dados (como ferramentas de modelação e desenvolvimento, sistemas de data warehouse, repositórios de meta-dados, etc.). Na versão 2.0 [OMG 2005], existem dois tipos de meta-meta-modelos, o *Essential MOF* (EMOF) e o *Complete MOF* (CMOF). O EMOF favorece a simplicidade de implementação ao invés da expressividade enquanto que o CMOF é mais expressivo, mas mais difícil de implementar [Gasevic, Djuric *et al.* 2006].

Model-driven Engineering (MDE)

A abordagem do MDA está longe de ser uma abordagem de engenharia. [Bézivin 2004] constata que o *Model-driven Engineering* (MDE) possui uma visão mais alargada do que o MDA, porque combina a arquitectura com os processos e a análise. A abordagem de engenharia do MDE é "o foco nos modelos, ao invés do foco em objectos" [Bézivin 2004]. As ideias básicas do MDE são similares a outras abordagens, como a programação gerada, *domain specific languages* (DSL) [Deursen, Klint *et al.* 2000], computação com integração de modelos

(MIC), etc. O que se passa no MDA é que estes princípios de MDE são realizados com base nos standards da OMG (UML, MOF, etc).

O MDE é uma abordagem mais aberta e integrada, pois permite a utilização de vários conceitos de tecnologias, como linguagens de programação, ontologias, linguagens XML-based, sistemas de gestão de bases de dados (DBMS), MDA, etc.

Os conceitos de MDD descritos até agora são os conceitos-base para o desenvolvimento de software. No entanto, o foco deste trabalho é sobre uma área que abrange um tema que vai para além destes conceitos-base. Tal como o nome indica, estes serão a base do desenvolvimento do sistema, mas, para a demonstração da investigação, o principal tema ainda não foi desenvolvido, que é a transformação dos modelos, que se vão aplicar à linguagem de processos de negócio que for escolhida.

2.5 Transformação dos Modelos

Já vimos anteriormente que para passarmos do PIM para o PSM é necessário um processo de transformação do modelo. Se a principal motivação do MDD/MDA é o de permitir desenvolver software a um nível de abstracção mais elevado, não deixa de ser verdade que o principal desafio imposto é o de transformar o modelo que está no nível de abstracção mais alto num modelo a um nível de abstracção mais baixo, pois só é possível gerar o código automaticamente neste nível devido a já estar descrito para a plataforma específica em que vai ser implementado (PSM).

Esta transformação é comummente chamada de Instanciação, pois estamos a descer um nível de abstracção (M2 para M1, por exemplo). A transformação de modelos também é utilizada quando queremos transformar um modelo num meta-modelo, subindo um nível de abstracção (M2 para M3, por exemplo), que pode ser chamada de Promoção [Bragança e Machado 2008].

Para realizar o processo de transformação é necessário seguir uma estratégia de modo a assegurar que esta não é feita de qualquer maneira. Normalmente, o processo de transformação de modelos é realizado seguindo conceitos de mapeamento e transformação [Pastor e Molina 2007]. A OMG também sugere a utilização da técnica de marcação [OMG 2003], tal como representado na Figura 12. A marcação é feita indicando quais os elementos no PIM que serão transformados em PSM.

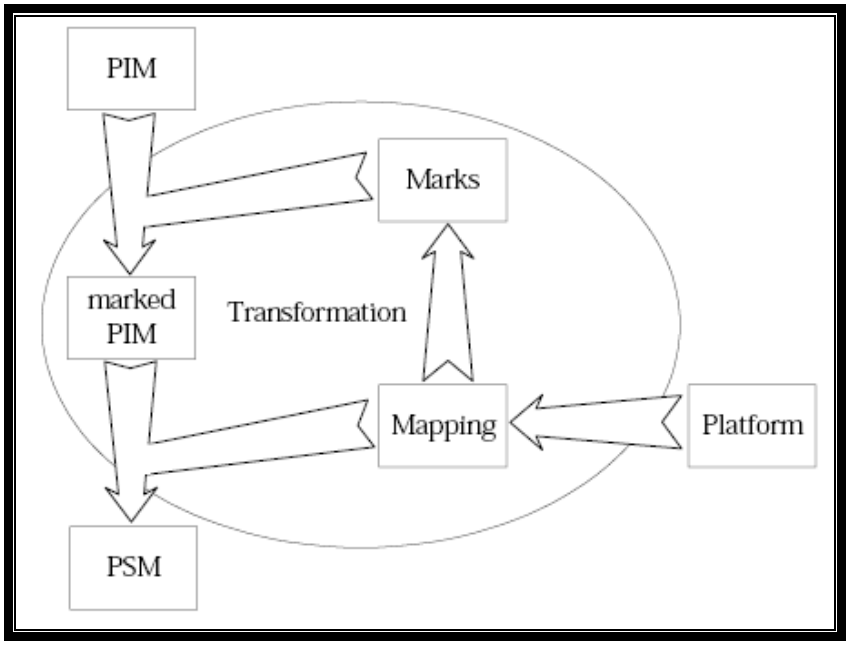

Figura 12. Transformação dos Modelos [OMG 2003]

No mapeamento são efectuadas relações entre os elementos do PIM e do PSM. Por exemplo, é possível criar um mapeamento que relacione uma classe do modelo a uma classe em Java [Armstrong, Ball *et al.* 2004]. Os mapeamentos devem ser baseados nos requisitos necessários à linguagem de programação ou ao modelo. É possível que existam relações entre um elemento do PIM e vários elementos do PSM, bem como um elemento do PSM pode se relacionar com vários elementos do PIM.

Efectuado o mapeamento, o processo de transformação realizará a geração de código, tendo por base o PIM, o PSM e o mapeamento efectuado, pois já terá toda a informação necessária para traduzir o modelo para código. Para ser possível este passo, é preciso que cada elemento do PSM possua informação necessária que possa ser traduzida em código. É exactamente para assegurar que essa informação se encontra no modelo que continuam a surgir novas abordagens para a transformação.

Para melhor compreender o processo de transformação de modelos, é necessário primeiramente introduzir o conceito de *Technological Spaces* (TS) [Bézivin, Dupé *et al.* 2003], que é usado para fins de análise e comparação. Um TS é um contexto de trabalho que possui um conjunto de conceitos associados, conhecimento, ferramentas, capacidades requeridas, entre outros… No contexto *model-driven*, são identificados os seguintes TS: sintaxes concretas e abstractas de linguagens de programação (*Syntax TS*), ontologias (*Ontology TS*), linguagens e ferramentas *XML-based* (*XML TS*), sistemas de gestão de bases de dados (DBMS TS) e *Model-driven Architecture* (*MDA TS*). Estes estão representados na Figura 13.

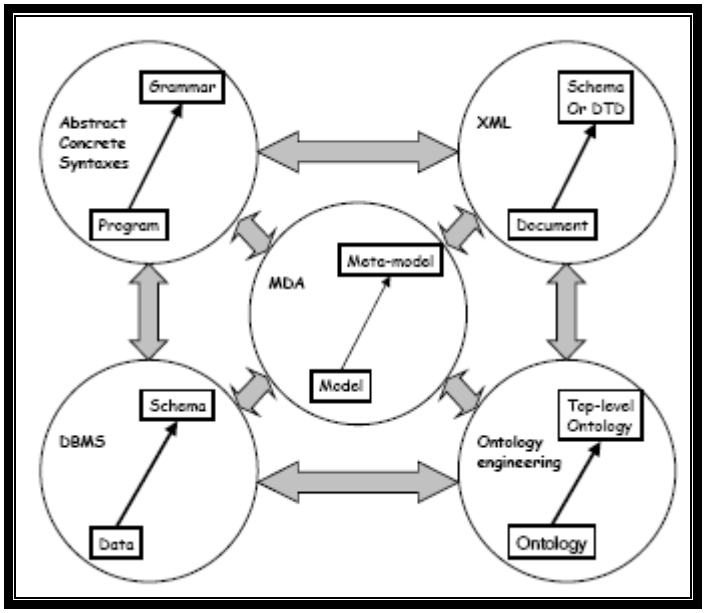

Figura 13. *Technological Spaces* e Relações Entre Eles [Bézivin, Dupé *et al.* 2003]

Também podemos notar na Figura que nenhuma das tecnologias é uma ilha, todas estão relacionadas com mais alguma(s) TS.

Na Figura 14 podemos ver um exemplo de transformação que pode ser entre o mesmo TS ou então entre TS diferentes. A migração de código Java para C# pode ser efectuada sem ter de sair do TS (*Syntax TS*, neste caso), tal como representado na operação D, pois existem ferramentas que efectuam a transformação automática de código. Alternativamente, podemos efectuar a mesma transformação utilizando TS diferentes, como passar do código Java (*Syntax TS*) para a correspondente modelação específica de UML (*MDA TS*) – operação R – esta ser transformada em modelação específica para C# e depois se gerado o código correspondente (*Syntax TS*) – operação F [Bézivin, Dupé *et al.* 2003].

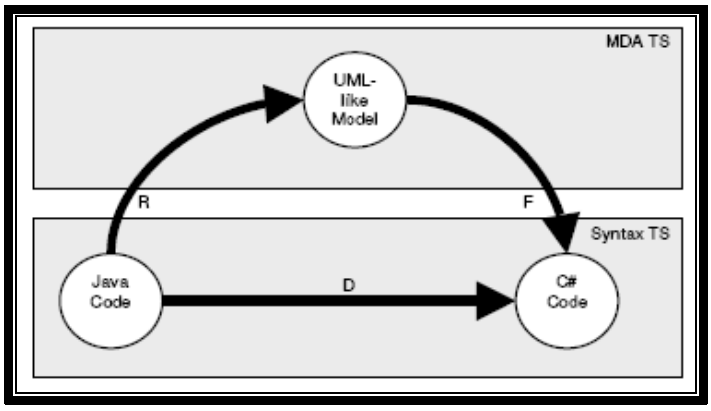

Figura 14. Transformação de Código Java em C# [Bézivin, Dupé *et al.* 2003]

4-Step Rule Set (4SRS)

O *4-Step Rule Set* (4SRS) é uma técnica proposta para transformar requisitos dos utilizadores em modelos arquitecturais que representam os requisitos do sistema. Associa, para cada objecto encontrado durante a fase da análise, uma determinada categoria: interface, dados e controlo. Cada uma destas categorias está relacionada a uma das três dimensões ortogonais em que a análise pode ser dividida (informação, comportamento e apresentação) [Machado, Fernandes *et al.* 2006]. O 4SRS é constituído por quatro passos: (i) criação de objectos; (ii) eliminação de objectos; (iii) empacotamento e agregação dos objectos e (iv) associação de objectos. O segundo passo é dividido em sete micro-passos.

A transformação é baseada no mapeamento de diagramas UML de casos de uso em diagramas UML de objectos. Outros esquemas podem ser usados, por exemplo diagramas UML de sequência, actividades e de estado [Machado, Fernandes *et al.* 2005]. Após a execução do 4SRS, obtemos a arquitectura lógica para o sistema que capta todos os seus requisitos funcionais e não funcionais. Adicionalmente, o 4SRS também pode ser utilizado para transformar um diagrama de objectos para que este mostre os serviços que o sistema necessita para o seu funcionamento. Isto é conseguido gerando um diagrama de objectos de serviços que corresponde à arquitectura lógica do serviço a ser especificado [Fernandes, Machado *et al.* 2006].

Query/View/Transformations (QVT)

O MOF *Query/View/Transformation* (QVT) [OMG 2005] é um standard da OMG para linguagens de transformações de modelos para o MDA. O objectivo do QVT é obter uma *domain specific language* (DSL) [Deursen, Klint *et al.* 2000] para realizar *queries, views* e transformações em modelos. O QVT opera ao nível dos modelos MOF (camada M3 do MDA), pelo que é utilizado num nível de abstracção maior e pretende-se diminuir esse nível de abstracção. Através do QVT é possível a transformação entre diferentes *techological spaces* (TS), MDA TS, Syntax TS, DBMS TS, etc. Para tal, já existem ferramentas específicas, como por exemplo o ATL [Bézivin, Dupé *et al.* 2003]. Outra das características é que as *queries* são expressões OCL, que podem retornar valores, elementos de modelos, etc.

O QVT tem na sua especificação uma parte declarativa e uma parte imperativa, sendo que a parte declarativa está dividida em duas camadas (*Relations* e *Core*) que formam o *framework* para a execução ao nível da semântica da parte imperativa (*Operational Mappings* e *Black-Box*), como se encontra representado na Figura 15.

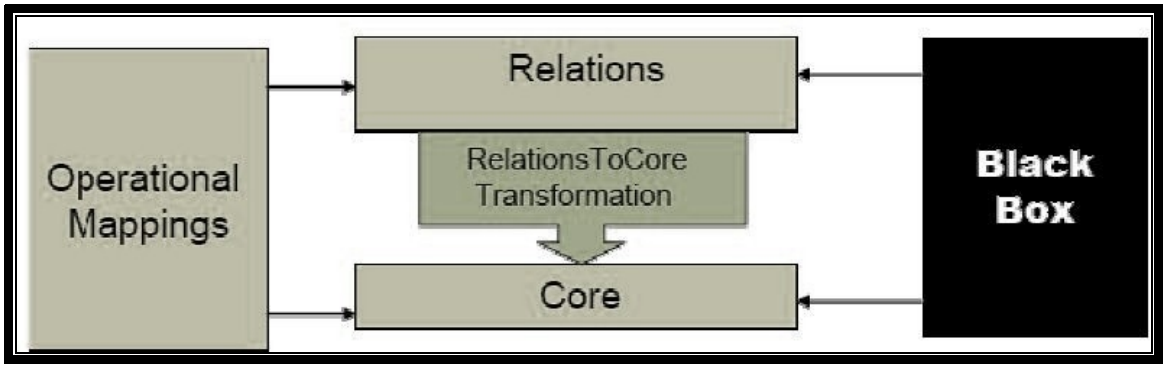

Figura 15. Relações entre Metamodelos QVT [OMG 2005]

A QVT *Relations* [OMG 2005] é uma linguagem que irá especificar as relações entre os modelos MOF. A QVT *Relations* consegue suportar padrões de correspondência complexos de objectos, criando implicitamente classes de rastreio e instâncias que irão guardar todas as ocorrências durante a transformação.

A QVT *Core* [OMG 2005] é uma linguagem/modelo que especifica os padrões de correspondência segundo um conjunto de variáveis avaliando as condições dessas variáveis num conjunto de modelos. Esta linguagem trata todos os elementos do modelo-fonte, modelo-final e modelo de rastreio por igual. É tão poderosa como a *Relations*, mas a sua semântica é mais simples, apesar de ser mais descritiva

A QVT *Operational Mappings* [OMG 2005] permite a realização de transformações utilizando uma abordagem imperativa (transformação operacional), ou complementa as relações utilizando operações imperativas implementando as relações (transformação híbrida). Fornece extensões OCL [OMG 2006] que permitem uma abordagem procedimental e baseada em sintaxes imperativas. A *Operational* serve também para ajudar a implementar relações, que sejam muito complicadas de executar apenas de forma declarativa.

2.6 Conclusões

Foram apresentadas características de algumas das principais linguagens de representação de processos de negócio (BPML, BPEL, XPDL, YAWL, EPC, CPN). Para desenvolver o software iremos utilizar a abordagem de MDD. Os standards MDA apresentados são largamente reconhecidos para a obtenção de software. Neste documento foram descritas os conceitos relevantes no desenvolvimento da investigação. Para a transformação do PIM para o PSM, serão utilizadas técnicas de mapeamento, marcação e posterior transformação.

Feita a revisão bibliográfica necessária, todos os conceitos serão absorvidos para a realização de casos de demonstração de modo a validar a investigação. Este trabalho vai ser realizado utilizando como caso de demonstração processos de negócio da empresa *Bosch Car Multimedia Portugal, Lda*. Visto que a empresa não utiliza uma linguagem de modelação de processos para os definir, tem que se proceder à escolha da mais apropriada.

O próximo passo irá ser modelar vários processos modelados na linguagem escolhida anteriormente. Estes processos vão ser modelados de forma a corresponderem ao PIM. Os processos modelados terão um modelo de referência como base, sendo que depois serão demonstrados exemplos de processos para suportar o caso de demonstração. Irá ser criado um repositório de diversos PIMs que depois poderão ser implementados num *Enterprise Service Bus*.

3. Processos de Negócio em Modelo PIM

Este capítulo pretende mostrar a primeira parte do trabalho no que toca a utilizar modelos (MDA) na construção de processos de negócio. O que se pretende demonstrar é uma primeira modelação dos processos considerando a utilização de normas de boas-práticas, tal como sugerido pela metodologia que é apresentada, e qual o valor acrescentado para a empresa que esta abordagem fornece. No entanto, existe um conjunto de trabalho de base que ainda necessita de ser realizado. A linguagem para modelar os processos ainda não está definida, pelo que ainda é necessário se proceder a uma escolha. A linguagem não pode ser escolhida de uma maneira "egoísta", dado que a análise propõe a toda uma organização a utilização futura da linguagem.

3.1 Introdução

É neste capítulo que se dá inicio ao caso de demonstração. Para tal, a empresa em que foram realizadas as demonstrações foi a *Bosch Car Multimedia Portugal, Lda.* A criação da *Bosch Car Multimedia Portugal, Lda.* esteve, desde o início, orientada por sólidos valores no âmbito da Qualidade, Inovação e Desenvolvimento. As suas práticas sempre se pautaram pela Melhoria Contínua ao nível dos processos e pelo posicionamento de vendas no ranking mundial. A *Bosch Car Multimedia Portugal, Lda.* é a maior fábrica de produção de auto-rádios da Europa. Lidera as vendas no Mercado Europeu e contribui activamente para a economia do país, encontrando-se entre os 10 maiores exportadores nacionais. O valor das compras ascende a 33 Milhões de Euros. O impacto a nível regional é substancial, empregando actualmente cerca de 2000 colaboradores. Entre estes encontra-se um grupo de Engenheiros e Técnicos (cujo número quadruplicou desde 1999) que desenvolvem novos produtos e sistemas, bem como métodos de produção inovadores. O seu trabalho é dirigido também ao melhoramento contínuo de produtos já existentes.

Este capítulo documenta as bases que tiveram de ser feitas na execução do caso de demonstração. A primeira tarefa a ser realizada prende-se com um dos objectivos propostos para este trabalho, que é o de escolher uma linguagem de representação de processos de negócio mais adequada para a empresa. Visto a investigação estar inserida num ambiente organizacional, a linguagem de execução de processos de negócio deve ser cuidadosamente escolhida, pois a escolha deve ser devidamente fundamentada para que as pessoas da organização se sintam convencidas de que devem passar a utilizar a linguagem nos seus projectos. A estratégia organizacional passa por três aspectos: estratégia de negócio, estratégia organizacional e estratégia de sistemas de informação [Pearlson e Saunders 2004]. Os padrões de *workflow* definidos por [Aalst, Hofstede *et al.* 2003] são largamente aceites na

comparação entre linguagens. Mas, no entanto, no ambiente organizacional, apenas uma das estratégias está a ser tida em conta (estratégia de sistemas de informação). Realizar uma análise no âmbito do negócio e no âmbito da própria organização torna portanto a comparação mais completa. Neste capítulo são comparadas as cinco linguagens de execução de processos de negócio apresentadas no capítulo anterior segundo os três tipos de estratégia, sendo escolhida a linguagem mais apropriada segundo os três aspectos. Ao nível tecnológico, as linguagens foram alvo de comparação quanto à sua capacidade de modelação de processos, utilizando como base padrões de fluxo de dados. Ao nível das pessoas, foram tidas em conta as características de cada colaborador da empresa. Ao nível da organização, foram tidas em conta factores importantes para qualquer empresa que lide com Tecnologias de Informação e também factores específicos da *Bosch Car Multimedia Portugal, Lda.* A linguagem escolhida é depois "apresentada", através de uma descrição dos conceitos necessários para modelar e executar os processos.

Na segunda parte do capítulo é apresentada uma metodologia de implementação de processos numa arquitectura organizacional, a *Business Implementation Methodology* (BIM) [Duarte, Machado *et al.* 2009], que apresenta uma proposta de transformação de processos de negócio em sistemas de software e que é a metodologia seguida no desenvolvimento do trabalho. A metodologia propõe um conjunto de fases para obter software com base em processos de negócio e sugere que uma possível forma de obter o software será a utilização do *Model-Driven Development* (MDD) [Atkinson e Kühne 2003], através da transformação de modelos PIM em modelos PSM [OMG 2003]. A descrição da metodologia efectuada neste capítulo pretende servir como base para o primeiro modelo de processos descrito no capítulo, bem como guia para o trabalho efectuado no capítulo seguinte. O modelo de processos inicial representa processos genéricos e devem reflectir utilização de normas de boas-práticas, através do uso de modelos de referência de processos.

3.2 As Linguagens Existentes

O trabalho de investigação não pode começar logo com modelação de processos. Nesta secção é apresentado o trabalho de base necessário para se proceder à modelação dos processos de negócio. Um estudo das linguagens de execução de processos de negócio existentes deve ser a primeira das tarefas do trabalho. Propõe-se que a escolha seja baseada em três âmbitos: tecnológico, organizacional e de negócio.

Para a comparação das linguagens apenas é realizada uma análise das características de cada uma. No entanto, escolhida a linguagem, logicamente o conhecimento destas é

insuficiente para representar os processos, pelo que antes de modelar os processos é apresentada a estrutura de um processo modelado na linguagem escolhida.

Escolha da Linguagem

Na secção 2.2 estão descritas as principais características sobre o funcionamento das principais linguagens de representação de processos de negócio, assim como o que as diferencia umas das outras. No sentido de escolher a linguagem mais adequada para modelar os processos de negócio na organização, as linguagens foram objecto de uma análise das vantagens de cada uma ao nível da sua utilização, do tipo de utilizadores e de aspectos inerentes à sua implementação e a este trabalho.

Para analisar as linguagens quanto à sua utilização, o objectivo passa por verificar qual a linguagem que melhor facilita a modelação dos processos. O que se verifica é que as linguagens não são iguais quanto a certos aspectos na sua modelação. Na modelação dos processos, nem todo o tipo de operações é possível de executar. A linguagem que mais operações permitir modelar é, portanto, a linguagem que traz mais vantagens quanto à sua modelação.

A linguagem mais apropriada para o tipo de utilizadores da *Bosch Car Multimedia Portugal, Lda.* também é tida em conta na escolha, pois estes serão os usufruidores da linguagem que for escolhida. Apesar de os utilizadores não serem todos iguais, é possível descrever perfis dos utilizadores tendo em conta os conhecimentos e apetências que possuem. A linguagem que melhor se encaixar nesses perfis é também vantajosa na sua implementação na organização, pois é a linguagem "à imagem" dos utilizadores.

Por fim, existem ainda outros aspectos inerentes às linguagens que são tidos em conta, nomeadamente aspectos referentes à implementação da linguagem na organização, pois a linguagem pode ser boa do ponto de vista funcional e "agradar" aos utilizadores, mas no fim de contas o mais importante é a organização, e é nela que se deve centrar.

Comparação com Base nos Padrões de Workflow

Para se determinar a linguagem mais completa no aspecto da sua funcionalidade, recorreu-se à comparação das linguagens analisando vinte padrões de controlo de fluxo, definidas por van der Aalst, Hofstede, Kiepuszewski e Barros em [Aalst, Hofstede *et al.* 2003]. A abordagem dos vinte padrões de *workflow* traz vantagens na medida em que analisa exemplos de controlo de fluxo que podem ocorrer durante um processo e é possível verificar se é possível na modelação do processo numa determinada linguagem. Esta abordagem é citada

em vários trabalhos tanto dos mesmos autores desta, como [Aalst, Dumas *et al.* 2002], [Aalst e Hofstede 2002] e [Aalst 2003], bem como em trabalhos de outros autores como [Mendling, Moser *et al*. 2006], [Mulyar 2005] e [Takecian 2008]. As definições referentes aos vinte padrões encontram-se em anexo (Anexo B) devido à sua extensibilidade e dado que o importante é a quantidade de padrões suportados pela linguagem e não os padrões em si, pelo que as definições são consideradas leitura adicional.

A Tabela 1 foi preenchida com base em estudos já referidos anteriormente, nomeadamente [Aalst, Dumas *et al.* 2002], [Aalst e Hofstede 2002] e [Aalst 2003] e [Mendling, Moser *et al.* 2006]. Se a linguagem suporta directamente o padrão está designado com um '+', se suporta indirectamente está designado com um '+/-', e se não suporta o padrão está designado com um '-'.

Tabela 1. Comparação das linguagens baseada nos padrões Workflow

Nº	Padrões de Controlo de Fluxo	BPML	BPEL	XPDL	YAWL	CPN
1	Sequência	+	+	+	+	+
2	Divisão paralela	+	+	+	+	+
3	Sincronização	+	+	+	+	+
4	Escolha exclusiva	+	+	+	+	+
5	Junção simples	+	+	+	+	+
6	Escolha múltipla	-	+	+	+	+
7	Junção sincronizada	-	+	+	+	-
8	Junção múltipla	+/-	-	-	+	+
9	Discriminador	-	-	+	+	-
10	Ciclo arbitrário	-	-	+	+	+
11	Terminação implícita	+	+	+	-	-
12	Múltipla Instância sem sincronização	+	+	+	+	+
13	Múltipla Instância com conhecimento prévio projecto	+	+	-	+	+
14	Múltipla Instância com conhecimento prévio execução	-	-	-	+	-
15	Múltipla Instância sem conhecimento prévio execução	-	-	-	+	-
16	Escolha em Diferido	+	+	-	+	+
17	Roteamento paralelo entrelaçado	-	+/-	-	+	+
18	Marco	-	-	-	+	+
19	Actividade cancelável	+	+	-	+	+/-
20	Caso cancelável	+	+	-	+	-

Ao visualizar a tabela, existem logo alguns dos dados que saltam à vista. O YAWL é, de longe, a linguagem que maior número de padrões suporta directamente, pelo que apenas não suporta directamente o padrão 11. As CPN e o BPEL não suportam ambos seis padrões. Também salta à vista que BPEL e BPML são muito semelhantes, com o BPML a suportar menor

número de padrões. O XPDL surge como a linguagem que menos suporta, apenas onze padrões.

É possível então dizer que o YAWL é a linguagem mais adequada do ponto de vista funcional. No entanto, este foi apenas o primeiro de três pontos de comparação das linguagens.

Comparação com base nos conhecimentos dos utilizadores

Este ponto de comparação é justificado com o facto de os utilizadores, neste caso os colaboradores do Departamento de Informática da *Bosch Car Multimedia Portugal, Lda.* responsáveis por desenvolvimento de sistemas de informação, serem as pessoas que irão utilizar e modelar os processos de negócio, sendo então necessário efectuar um estudo para concluir qual a linguagem de representação de processos que melhor se identifica com as características e competências destes.

A forma encontrada para se chegar a tal conclusão foi a de os utilizadores responderem a um questionário (Anexo C). O questionário foi estruturado de forma a que os inquiridos respondessem a questões sobre características presentes nas linguagens em comparação, como conhecimentos acerca de *Web Services*, linguagem XML, SOA, entre outros, e também conhecimentos de BPM de modo a analisar a sua maturidade em compreender processos de negócio e como estes devem ser modelados. A análise ao questionário permite também relacionar uma das linguagens de entre as estudadas com o nível de maturidade atribuído aos inquiridos.

Ao longo do questionário, algumas perguntas estão directamente relacionadas com uma linguagem de processos em particular. Existem também perguntas cujas respostas direccionam indirectamente para uma ou mais linguagens. As relações entre as respostas e as linguagens estão representadas na

Tabela 2. É através das respostas dadas que se vai determinar a linguagem mais adequada.

A tabela de respostas recebidas dos nove questionários está reproduzida no anexo D. Todas as respostas foram recolhidas e efectuou-se o cálculo da pontuação que cada um dos nove inquiridos deu a cada linguagem, apresentada na Tabela 3. A pontuação obtida por cada linguagem é dada por um valor relativo que tem em conta o número de respostas dadas pelos inquiridos, mas também a quantidade de respostas possíveis para cada linguagem. Isto justifica-se com o facto de, em algumas perguntas com respostas que se associam

indirectamente com a linguagem, a quantidade destas não ser equilibrada para todas as linguagens, pelo que este cálculo é mais justo do que uma simples soma da pontuação.

Tabela 2. Respostas que apontam para linguagens

Pergunta	BPML	BPEL	XPDL	YAWL	CPN
7 – "Sim"		x	x	x	x
8 – b, c, d, e		x		x	x
9 – "Sim"		x			
10 – c, d	x	x	x	x	
11 – UML b, c, d, e		x	x		
11 – BPMN b, c, d, e	x	x	x		
11 – EPC b, c, d, e				x	
12 – a, c, d, e, f, h, i, o, p, q	x				
12 – a, b, d, g, l, r, s		x			
12 – d, l, m, n, t			x		
12 – k, u				x	
12 - j					x
13 – "Sim"	x	x		x	x
14 – c, d, e	x	x		x	x
15 – c, d ,e	x	x	x	x	x
16 – a, b, c, d, e, f, g, j, l, m, n, o, p	x	x		x	x

Tabela 3. Pontuação dos inquiridos a cada linguagem

Linguagem	#1	#2	#3	#4	#5	#6	#7	#8	#9	**Total**
BPML	0.76	0.70	0.64	0.59	0.62	0.67	0.15	0.34	0.78	5.26
BPEL	0.76	0.80	0.74	0.60	0.72	0.88	0.38	0.53	0.79	6.20
XPDL	0.76	0.79	0.79	0.79	0.79	0.87	0.35	0.59	0.92	6.65
YAWL	0.65	0.94	0.83	0.57	0.70	0.90	0.34	0.49	0.69	6.11
CPN	0.84	0.99	0.78	0.59	0.76	0.95	0.30	0.48	0.75	6.43

Comparação segundo conceitos relevantes para a organização

Finalmente, foi realizado um levantamento de outros aspectos de comparação das linguagens relacionados com determinadas características relevantes para a implementação e utilização na organização e posterior análise e discussão com o objectivo de melhor fundamentar a escolha. Algumas das características definidas foram baseadas em características que são definidas para comparação de ferramentas das linguagens [Helkiö, Seppälä *et al.* 2006]. As restantes são da responsabilidade do autor, baseado em conceitos gerais de interesses das organizações de tecnologias (por exemplo, a maturidade), em conceitos relevantes para o problema em questão (a tradução e transformação das linguagens), bem como até sugestões de pessoas na empresa (por exemplo, o custo da implementação) ou conhecimentos do autor acerca do funcionamento da empresa (por exemplo, a integração da linguagem no SAP).

A análise quanto à maturidade da linguagem tem em conta o tempo em que a linguagem surge, as diferentes versões e actualizações e a data da última versão. O BPML tem como única versão a 1.0, que surge no ano de 2002 [Arkin 2002]. A primeira versão do BPEL surge em 2005 [Curbera, Goland *et al.* 2005] e a última versão, a 2.0, em 2007 [Oasis 2007]. BPML e BPEL possuem, portanto, um nível grande de maturidade. O XPDL e o YAWL surgem em 2002 e as suas versões mais recentes, a 2.1 no caso do XPDL [WfMC 2008], e a 2.0 BETA no caso do YAWL [YAWL System], são publicadas em 2008, pelo que possuem um nível de maturidade menor do que as linguagens anteriores. As CPN surgem em 1981 [Jensen 1992] e a ferramenta principal, o CPN Tools [CPNGroup], surge em 2001 e a versão mais recente data de 2006. Esta é, sem dúvida, a linguagem mais madura.

A análise quanto à usabilidade refere a facilidade na modelação das linguagens. Neste caso, as linguagens baseadas em fluxogramas, BPML, BPEL e XPDL, possuem maior grau de usabilidade em relação às linguagens baseadas em grafos, YAWL e CPN, pois um fluxograma permite melhor compreensão na leitura e escrita dos processos.

No que respeita à análise da quantidade de ferramentas existentes, obviamente que não é possível poder dizer com certezas absolutas a quantidade de ferramentas que suportam cada linguagem, pelo que foi realizada uma análise à quantidade suficiente para poder dizer se uma linguagem é suportada por mais ou menos ferramentas do que outra. O mesmo se passa na análise da quantidade de tutoriais on-line de cada linguagem.

Para se comparar a capacidade de tradução da linguagem, foi tido em conta se a linguagem pode ou não ser traduzida em qualquer uma das outras linguagens, i.e., migração de linguagem utilizando o mesmo *Technological Space* (secção 2.5, exemplo da Figura 14).

O esforço de aprendizagem refere-se ao esforço necessário (horas necessárias, por exemplo) para domínio da linguagem. Novamente, é mais fácil e requerem menos esforços as linguagens baseadas em fluxogramas, BPML, BPEL e XPDL.

A capacidade da linguagem se transformar é a característica mais relevante para este trabalho. Refere-se à capacidade de transformar a linguagem em código orientado a objectos, pois é esse o paradigma utilizado no desenvolvimento de software. Para esta análise, serviu como válida a utilização de diferentes *Technological Spaces* na transformação em código. Este ponto é o mais importante no âmbito deste trabalho, dado que é o tema-chave para a concretização de um dos objectivos propostos. O BPEL é a linguagem cujas características melhor permite a transformação para código orientado a objectos, pois é possível relacionar os construtores BPEL com os artefactos Java [Kloppmann, König *et al.* 2004], e já existem efectivamente ferramentas para a transformação (BPEL2JAVA) [BPEL2Java]. É também possível a transformação do BPEL em Diagramas de Actividade UML 2 (UML-AD)

[Reiter, Kapsammer *et al.* 2006], que podem posteriormente ser transformados em código usando o MOF QVT [OMG 2005]. O XPDL também tem relações com o UML2-AD [Guelfi e Mammar 2006], mas como só é possível a transformação de UML2-AD em XPDL e não o contrário, é possível estabelecer relações entre eles, mas mais limitada, pois analisando os padrões de workflow suportados entre UML e XPDL [Aalst 2003] conclui-se que existem muitos padrões que um suporta e o outro não. Quanto às linguagens BPML, YAWL e CPN, é possível transformá-las em UML através de ferramentas como o ATL [Bézivin, Dupé *et al.* 2003], mas este é um processo ainda mais limitado do que no caso do XPDL.

Como característica relevante para uma organização, é impossível não realizar uma análise quanto a custos. Felizmente, a maior parte das ferramentas que suportam as linguagens são de livre acesso, logo não acarretam custos para a empresa. Existem ferramentas livres para todas as linguagens, mas no entanto a comparação foi feita com base nas ferramentas mais cotadas. Neste contexto, as principais ferramentas para BPML, BPEL e YAWL são todas livres, o CPN Tools (ferramenta de referência para as CPN's) requer uma licença antes de a podemos utilizar, e o Enhydra JaWe (para o XPDL) é uma ferramenta paga.

O conceito de portabilidade é definido como "capacidade de pegar num artefacto criado num ambiente de um vendedor e utilizá-lo num ambiente de outro vendedor" [Oasis 2007]. Assim, o YAWL surge em vantagem dado que existe apenas um vendedor, logo a portabilidade é sempre assegurada. As CPN's, tal como o BPEL, também têm como característica funcionarem de igual forma independentemente da ferramenta onde correm. No caso do BPML, permite exportação de modelos de ficheiros XML válidos em qualquer ferramenta [Silver 2009]. O XPDL possui 3 *Mode Portability Conformance Classes*, o *Simple*, o *Standard* e o *Complete* [Silver 2009].

O conceito de interoperabilidade é definido como "capacidade de múltiplos componentes interagirem utilizando mensagens e protocolos bem-definidos. Inclui combinação de componentes de diferentes vendedores a executarem de forma semelhante" [Oasis 2007]. Neste caso, o BPML, o BPEL e o YAWL asseguram total interoperabilidade através de troca de mensagens entre serviços em WSDL. O XPDL assegura interoperabilidade através da troca de mensagens XML, mas não é totalmente automático, necessita de configuração. O mesmo se passa nas CPN's, mas neste caso ainda é mais limitado, pois a troca de mensagens pode ter de se feita em linguagens diferentes, o que requer uma configuração das invocações ainda mais difícil.

A segurança é outro aspecto que a organização não pode descurar, pelo que a linguagem escolhida deve garantir uma troca de dados segura e sem perda de informação. O BPML e o BPEL possuem nas suas especificações o *WS-Security*, que é o protocolo que garante

a segurança na troca de mensagens entre *Web Services*. O YAWL por sua vez garante a segurança dos dados através do *YAWL Manager*.

A eficiência traduz os aspectos do tempo de execução de um processo e dos recursos utilizados. O BPEL utiliza o protocolo *WS-Coordination* para orquestrar os *Web Services*, o YAWL utiliza o *YAWL Manager*, enquanto que as CPN's, devido às características de uma Rede de Petri, asseguram a eficiência através de fórmulas matemáticas. O BPML necessita de se relacionar com o WSCI (*Web Service Choreography Interface*) [Aalst, Dumas *et al.* 2002] para garantir a eficiência.

No que consiste à gestão dos dados, é importante que os dados, nomeadamente variáveis ou os seus valores, estejam sempre perfeitamente identificados quanto à sua localização. Neste aspecto, o XPDL e as CPN's possuem vantagem dado que o XPDL apenas trata dados em XML e nas CPN's os valores dos dados estão sempre guardados nos *tokens*, enquanto que no BPML, BPEL e YAWL, os dados são tratados em XML e WSDL.

O aspecto da integração da linguagem no funcionamento do SAP é relevante, dado que é o ERP utilizado para a organização gerir os seus processos. As linguagens BPML e BPEL podem ser integradas directamente, enquanto que o YAWL deve primeiramente ser traduzido em yEPC (*YAWL Event-driven Process Chain*).

Na Tabela 4 está representada a avaliação das linguagens segundo estes critérios, onde foi utilizada uma notação de 1 a 5, sendo que 1 e 2 correspondem a uma avaliação negativa, 3 é razoável mas já corresponde a avaliação positiva e 4 e 5 avaliação positiva e muito positiva, respectivamente.

Somando as pontuações obtidas pelas cinco linguagens, é possível verificar que o BPEL é a que maior número de pontos obteve, num total de 66 pontos. O BPEL é então a linguagem que melhor serve os interesses do ponto de vista organizativos. Quanto às restantes linguagens, o BPML obteve 59 pontos, seguido das CPN com 55, o YAWL com 53 e finalmente o XPDL, com apenas 47 pontos.

Tabela 4. Comparação no âmbito de uma organização

Nº	Características	BPML	BPEL	XPDL	YAWL	CPN
1	Maturidade	4	4	4	3	5
2	Usabilidade	4	4	4	3	3
3	Ferramentas existentes	4	5	3	2	5
4	Tutoriais on-line	5	5	3	3	5
5	Traduzível	5	5	3	4	4
6	Esforço Aprendizagem	4	4	4	3	3
7	Transformável em código OO	2	5	3	2	2
8	Custos de Implementação	5	5	3	5	4
9	Portabilidade	3	5	3	5	5
10	Interoperabilidade	5	5	4	5	3
11	Segurança	5	5	3	5	3

12	Eficiência	4	5	2	5	5
13	Gestão dos Dados	4	4	5	4	5
14	Integração no ERP SAP	5	5	3	4	3

Resultado das análises

Finalizada então a análise às linguagens segundo prismas de funcionalidade, utilizadores e de organização, é tempo então de realizar um "balanço". Não é possível dizer se alguma das três análises efectuadas é mais importante na escolha de uma linguagem do que outra, pelo que as três são consideradas com igual importância. Na Tabela 5 estão os resultados obtidos pelas linguagens nas três análises, tendo as análises todas o mesmo peso na obtenção do resultado final.

Tabela 5. Resultados finais das linguagens

Análise	**BPML**	**BPEL**	**XPDL**	**YAWL**	**CPN**
Funcionalidade	4º	2º	5º	1º	2º
Utilizadores	5º	3º	1º	4º	2º
Organização	2º	1º	5º	4º	3º
Final	4º	**1º**	4º	3º	2º

Como se pode ver na tabela, não há grandes dúvidas de que o **BPEL** obteve os melhores resultados, sendo então considerada a linguagem de representação de processos de negócio mais adequada para utilização na *Bosch Car Multimedia Portugal, Lda.* e na realização deste trabalho. O BPEL obteve os melhores resultados em relação às restantes linguagens nas análises, à custa de ter obtido a melhor pontuação segundo conceitos relevantes para a organização, mas também por ter obtido a segunda e terceira melhor pontuação nas restantes comparações, respectivamente.

O facto de ordenar as linguagens por classificação ao invés de apenas mencionar a linguagem com melhor resultado prende-se com a possibilidade de procurar alternativas à primeira opção, dado que a comparação é efectuada com o objectivo único de indicar uma linguagem com base num estudo científico, não sendo a empresa "obrigada" a aceitar ou implementar a linguagem escolhida.

Business Process Execution Language (BPEL)

A arquitectura e as principais características da linguagem BPEL já foram descritas na secção 2.2. O modo como o BPEL executa os processos modelados foi o ponto de análise na escolha das linguagens propostas neste documento. Agora que a linguagem já está escolhida, é

necessário no entanto efectuar uma descrição do BPEL ao nível da estrutura e modelação de um processo BPEL para melhor se compreender a construção dos processos.

Um processo BPEL é constituído por uma série de actividades realizadas sequencialmente, representando o fluxo dos dados. Como actividades básicas em BPEL temos *invoke, receive, reply, assign*, entre outros (Tabela 6). É também constituído por estruturas algorítmicas no caso de processos um pouco mais complexos, tais como *if, while, repeatUntil, flow,* entre outros (Tabela 7). As *Variables* e os *PartnerLinks* são os elementos tratados nas actividades e nas estruturas algorítmicas. Um *PartnerLink* faz a ligação com o interface de um serviço que interage com o processo, que se encontra num ficheiro WSDL (*Web Services Description Language*) separado. As *Variables* representam as mensagens que são trocadas durante o processo. Para a definição das estruturas algorítmicas que expressam condições e *Variables* através de expressões, podem ser utilizadas expressões XPath 1.0 que estão disponíveis em [W3C 1999].

Tabela 6. Actividades Básicas BPEL

Atividade Básica	**Descrição**
invoke	Faz uma chamada a algum *Web Service* disponibilizado
receive	Espera por uma mensagem de um agente externo
reply	Efectua a resposta referente a uma requisição previamente aceite
wait	Aguarda por um determinado tempo
assign	Copia dados entre variáveis ou insere-os segundo expressões
throw	Sinaliza uma falha interna na execução
compensate e *compensateScope*	Inicia a compensação de actividades que já foram concluídas com sucesso
exit	Finaliza imediatamente a instância do processo
empty	Não faz nenhuma acção. Pode ser usada como ponto de sincronização
rethrow	Lança novamente falhas que foram sinalizadas
validate	Assegura que os valores armazenados nas variáveis sejam válidos
extensionActivity	Define novas actividades

Tabela 7. Estruturas Algorítmicas BPEL

Estruturas algorítmicas	Descrição
sequence	Contem uma ou mais actividades executadas sequencialmente
flow	Contem um conjunto de actividades executadas concorrentemente
pick	Contem um conjunto de pares de actividades, executando a actividade associada a um evento ocorrido
while	Repete a execução de uma actividade enquanto uma condição for verdadeira
repeatUntil	Repete a execução de uma actividade até que uma condição de torne verdadeira
scope	Agrupa actividades em blocos
if	Especifica um comportamento condicional dentro do processo
forEach	Repete a execução de uma actividade por um declarado número de vezes
wait	Determina um prazo de espera para a execução de uma actividade

Um ficheiro BPEL é constituído por quatro partes. A primeira parte refere-se aos *namespaces* dos ficheiros XML necessários na execução do processo.

```
[1]  <process name="main"
[2]   targetNamespace= ...
[3]  xmlns:bpel="http://docs.oasisopen.org/wsbpel/2.0/process/executab
     le"
[4]  xmlns:tns=  ...
[5]  >
```

Figura 16. Exemplo de Código BPEL Referente a *Namespaces*

A segunda parte refere-se à definição do interface do *PartnerLink* do processo, bem como a localização do respectivo ficheiro WSDL, e de qualquer outro utilizado no processo.

```
[1]  <import location="main.wsdl" namespace=  …
[2]  importType="http://schemas.xmlsoap.org/wsdl/" />
[3]  <partnerLinks>
[4]   <partnerLink name="client"
[5]                partnerLinkType="tns:main"
[6]                myRole="mainProvider"/>
[7]  </partnerLinks>
```

Figura 17. Exemplo de Código BPEL Referente aos *Partnerlinks*

A terceira parte refere-se à definição das variáveis utilizadas, *i.e.*, o tipo de mensagens trocadas durante o processo.

```
[1]  <variables>
[2]   <variable name=  ...
[3]   messageType=     ...
[4]  />
[5]  </variables>
```

Figura 18. Exemplo de Código BPEL Referente às Variáveis

A quarta e última parte do ficheiro BPEL refere-se à sequência de actividades do processo.

```
[1]   <sequence name="main">
[2]   <receive name=    ...   partnerLink="client"
[3]   portType=   ...   operation=  ...
[4]   variable="input" createInstance="yes"/>
[5]   <assign name=     ...   >
[6]   <copy>
[7]   <from ...   />
[8]   <to   ...   />
[9]   </copy>
[10]  </assign>
[11]  <reply name=      ...   partnerLink="client"
[12]  portType=   ...    operation= ...
[13]  variable="output"/>
[14]  </sequence>
[15]  </process>
```

Figura 19. Exemplo de Código BPEL Referente à Sequência de Actividades

A cada processo BPEL existe sempre um ficheiro WSDL correspondente que contém as definições do interface do processo. Um ficheiro WSDL é constituído por cinco partes. A primeira parte contém os *namespaces* utilizados.

```
[1]  <?xml version="1.0"?>
[2]  <definitions name=     ...
[3]          targetNamespace=     ...
[4]          xmlns:tns=     ...
[5]          xmlns:plnk="http://docs.oasis-
     open.org/wsbpel/2.0/plnktype"
[6]          xmlns="http://schemas.xmlsoap.org/wsdl/"
[7]          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
```

Figura 20. Exemplo de Código WSDL Referente aos *Namespaces*

A segunda parte do ficheiro WSDL contém a definição do tipo de elementos das mensagens e *XML Schema* (XSD) [W3C].

```
[1]  <types>
[2]  <schema attributeFormDefault="unqualified"
[3]  elementFormDefault="qualified"
[4]  targetNamespace=  ...
[5]  xmlns="http://www.w3.org/2001/XMLSchema">

[6]  <element name=    ...
[7]  </element>

[8]  </schema>
[9]  </types>
```

Figura 21. Exemplo de Código WSDL Referente aos Tipos de Elementos

A terceira parte refere-se à definição do tipo de mensagens e das respectivas partes existentes nas mensagens definidas anteriormente no *schema*.

```
[1]  <message name=    ...   >
[2]  <part name= ...   element=    ...   />
[3]  </message>
```

Figura 22. Exemplo de Código WSDL Referente aos Tipos de Mensagens

A quarta parte refere-se à definição das portas utilizadas e especifica qual(ais) a(s) operação(ões) que ocorrem na respectiva porta e qual o tipo de mensagem *input* e *output*.

```
[1]  <portType name=   ...   >
[2]  <operation name=  ...   >
[3]  <input  message=  ...   />
[4]  <output message=  ...   />
[5]  </operation>
[6]  </portType>
```

Figura 23. Exemplo de Código WSDL Referente às Portas e Operações

A quinta e última parte contém a definição do papel (*role*) do *PartnerLink* referente a este interface, bem como as definições *Simple Object Access Protocol* (*SOAP)* [W3C 2007] e as respectivas ligações.

```
[1]   <plnk:partnerLinkType name=   ...   >
[2]   <plnk:role name=  ...    portType=  ...   />
[3]   </plnk:partnerLinkType>

[4]   <binding name=    ...    type=      ...   >
[5]   <soap:binding style="document"
[6]         transport="http://schemas.xmlsoap.org/soap/http" />
[7]   <operation name=  ...   >
[8]   <soap:operation
[9]         soapAction= ...    />
[10]              <input>
                        ...
[11]              </input>
[12]              <output>
                        ...
[13]              </output>
[14]  </operation>
[15]  </binding>
[16]  <service name=    ...   >
[17]  <port name= ...    binding=   ...   >
[18]  <soap:address location= ...    />
[19]  </port>
[20]  </service>
[21]  </definitions>
```

Figura 24. Exemplo de Código WSDL Referente às Fefinições do *Partnerlink*

Exemplo de um processo BPEL

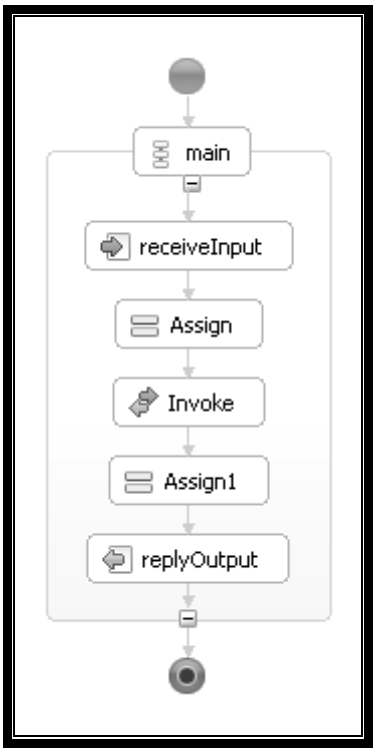

Figura 25. Exemplo de um Processo BPEL

3.3 Modelação de Processos de Negócio em PIM

A metodologia em que este trabalho é baseado propõe a utilização de normas de boas-práticas na definição dos processos de negócio que funcionam numa organização. A metodologia é detalhada de seguida de forma a apresentar os conceitos presentes e para uma melhor compreensão do trabalho que é realizado. Também a utilização da abordagem a modelos é justificada como base para a transformação dos processos em software, apresentando para já a modelação de processos de negócio em PIM. A modelação inicial dos processos utiliza os modelos de referência de processos, que garante a utilização de boas-práticas na empresa.

Relação entre a Metodologia BIM e Utilização de MDA

A *Business Implementation Methodology* (BIM) [Duarte, Machado *et al.* 2009] descreve a forma como software deve ser implementado de modo a suportar processos de negócios em empresas. Esta metodologia permite a implementação de software utilizando um conjunto de actividades faseadas, tendo por base o estado em que os processos de negócio estão definidos. Os estados para definição dos processos são *Generic* – obtidos através de uma definição dos processos ao nível de abstracção mais elevado –, *Instantiated* – instanciação dos processos *Generic*, no qual já se propõe uma "customização" do processo de modo a este melhor se aplicar na empresa no âmbito do negócio –, *Runnable* – processos que já se integram na arquitectura de Sistemas de Informação organizacional – e o *Software Implemented* – processo já totalmente suportado na empresa por um sistema de software. Tal como já foi referido, as fases da BIM são baseadas nos estados dos processos, *i.e.*, a cada fase da BIM corresponde a um estado dos processos. A BIM é constituída pelas fases de *Selection*, *Definition*, *Concretisation* e *Implementation.* Só é possível passar de uma fase para a fase seguinte quando o estado dos processos também se altera para o estado correspondente à fase seguinte. A cada estado da arquitectura dos processos (*Process Framework* – PF) corresponde então cada uma das fases da metodologia seguida.

Analisando as características desta metodologia, é possível estabelecer algumas relações entre a BIM e o *Model-driven Development/Architecture* (MDD/MDA). As características de um processo de negócio que estejam no estado *Generic*, *Instantiated* e *Runnable* são semelhantes às características de um modelo *Platform Independent Model* (PIM) do MDA, dado que nestes estados a representação dos processos não inclui ainda qualquer dependência em tecnologia (plataforma). Durante as três primeiras fases (*Selection*, *Definition*

e *Concretisation*) o processo não funciona com suporte de qualquer software, pois até ao fim da terceira fase ainda nem se tomou a decisão se o processo vai funcionar em software ou não. Aliás, no fim da fase da Concretização, é sugerido que exista um processo que seja posteriormente implementado conforme a fase seguinte da metodologia, mas que sejam propostas alternativas, entre elas pelo menos uma em que o processo não necessite de implementação de software, *i.e.*, que o seu modelo no estado final seja o obtido no estado *Runnable*. O estado de *Software Implemented* corresponde então ao modelo de *Platform Specific Model* (PSM), dado que o processo já funcionará em software, tendo para isso de obedecer às especificações da arquitectura tecnológica onde vai funcionar (Figura 26).

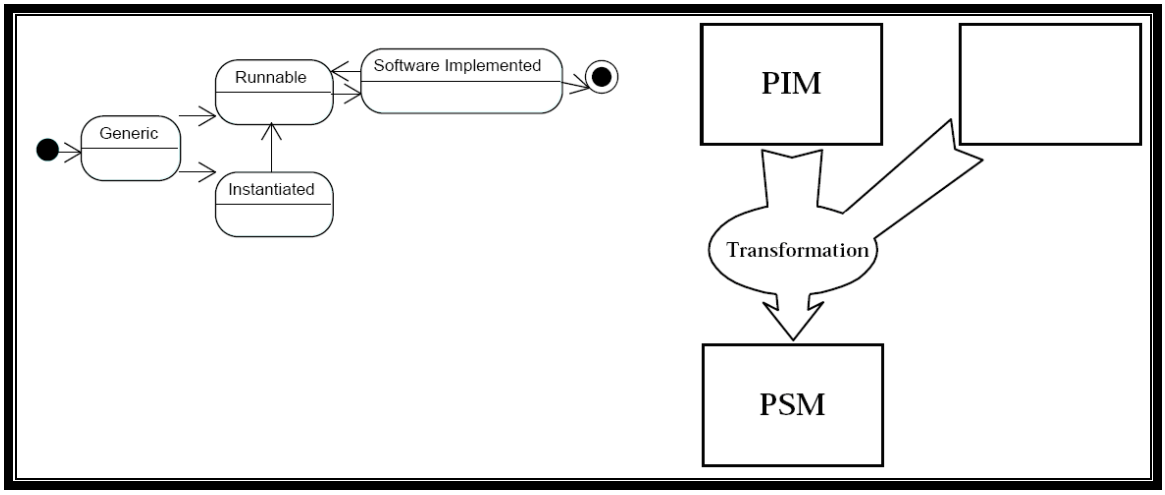

Figura 26. Estados dos PF's na BIM [Duarte, Machado *et al.* 2009] e Transformação de um PIM para PSM [OMG 2003]

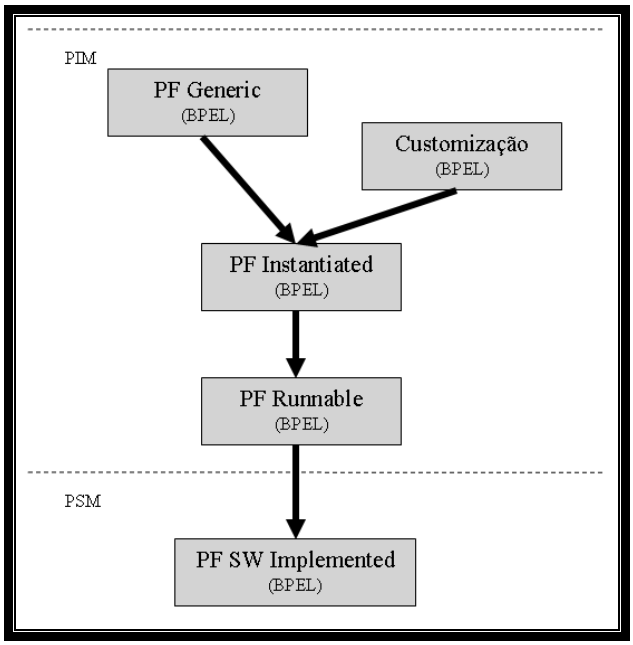

Figura 27. Esquema de Transformação Adoptado

Estabelecidas as relações entre os processos de negócio na BIM e os modelos MDA, o esquema que descreve as transformações que os modelos dos processos vão sofrer é obtido (Figura 27). Os processos irão ser modelados em BPEL até existirem dados suficientes para a implementação destes processos em software. O modelo CIM não foi considerado para este caso, mas, se tivesse sido, o processo seria definido no estado *Generic*, por exemplo através da linguagem BPMN, antes de se obter o mesmo processo no mesmo estado mas em BPEL.

Características de um Processo PIM

Para a modelação dos processos, tem-se em conta primeiramente as funcionalidades destes, ainda sem qualquer especificação sobre procedimentos de ordem técnica, apenas do ponto de vista do negócio. Este nível de modelação corresponde ao modelo CIM (*Computation Independent Model*) do *Model-Driven Architecture* (MDA) descrito na secção 2.4. Tal como descrito nessa secção, o CIM sofre transformações de modo a tornar-se um PIM. Ao nível do CIM não se justifica a utilização do BPEL para modelar o processo, pelo que uma linguagem apenas de notação, como BPMN, EPC ou UML será suficiente. Em certos casos, pode ser aconselhável a modelação do processo primeiramente ao nível do CIM antes de o modelar em PIM, ficando no entanto ao critério do analista do processo a necessidade ou não de modelar o processo primeiramente em CIM e depois o transformar em PIM, ou modelar o processo directamente em PIM. Após a transformação para o nível PIM, o processo modelado já irá conter especificações da implementação do processo. Mas, devido a ser um modelo de processo de negócio independente de plataforma, ainda não contém qualquer especificação sobre em que plataforma será implementado. Essas transformações são tratadas mais adiante.

Um modelo de processos ao nível do PIM modelado em BPEL deve ter um conjunto de requisitos a que o processo deve obedecer para garantir o fluxo correcto dos dados. Deve conter especificações computacionais ao nível da linguagem para execução dos processos, tal como descreve o MDA aquando a obtenção de um PIM a partir de um CIM. Daqui se retira a necessidade de modelar o processo a partir de uma linguagem que permita a execução do processo, como o BPEL, ao invés de utilizar linguagens que apenas permitem representar o processo utilizando notações, como BPMN, EPC ou UML. Linguagens como o BPMN devem ser utilizadas então a um nível mais abstracto da modelação do processo, quando o designer do processo e o analista do negócio pretendem apenas definir o controlo do fluxo dos dados, bem como especificar quem são os participantes no processo e quais os artefactos ligados a ele. Ao nível do PIM, o modelo do processo em BPEL deverá então já descrever o processo de negócio segundo especificações de serviços bem como integração e ligações entre diversas aplicações,

de modo a que responda aos requisitos funcionais e não-funcionais do próprio processo e permita maior nível possível de ligações automáticas entre serviços.

A modelação em processos BPEL é normalmente feita tendo por objectivo a integração das várias aplicações que uma organização pode utilizar. Ao nível do PIM a integração não é o "principal" objectivo, pelo que deverá ser apenas o de definir as ligações que permitam o fornecimento de um serviço qualquer que satisfaça os requisitos do processo. Se, por norma, utilizamos um ESB (*Enterprise Service Bus*) para que os vários componentes da arquitectura do sistema de informação de uma organização possam trocar mensagens de modo a viabilizar todo e qualquer processo de negócio, no modelo PIM apenas é considerado o componente "BPEL", como se este se encontrasse "sozinho" na arquitectura. Isto é, o modelo deve definir ligações entre serviços, mas não especificar o modo como a ligação é feita. Por exemplo, o modelo de processos pode definir ligações a um dado ERP, mas não são descritas as especificidades próprias que o ERP necessita para se interligar ao processo.

Outra característica do processo é que este é descrito como funciona no tempo presente ("*as-is*") e não como funcionará no tempo futuro ("*to-be*"), pois necessitaria de ter definidas especificações do tipo de ligações que iria ter com os serviços e softwares constituintes do processo, bem como também necessitaria de ser consideradas especificações no que respeita o desenvolvimento de sistemas de software que iriam suportar o funcionamento dos processos.

Como já referido anteriormente, para o nível do PIM deve-se considerar apenas o componente "BPEL", como se este estivesse "sozinho". Então, o modelo é constituído por um processo que terá as actividades básicas BPEL (*invoke, receive, reply, wait, assign, throw, compensate, compensateScope, exit, empty, rethrow, validate, extensionActivity*) e as estruturas algorítmicas que a linguagem BPEL suporta (*sequence, flow, pick, while, repeatUntil, scope, if, forEach, wait*), não contendo quaisquer descrições mais detalhadas acerca de como o processo irá ser executado.

"Best-practices"

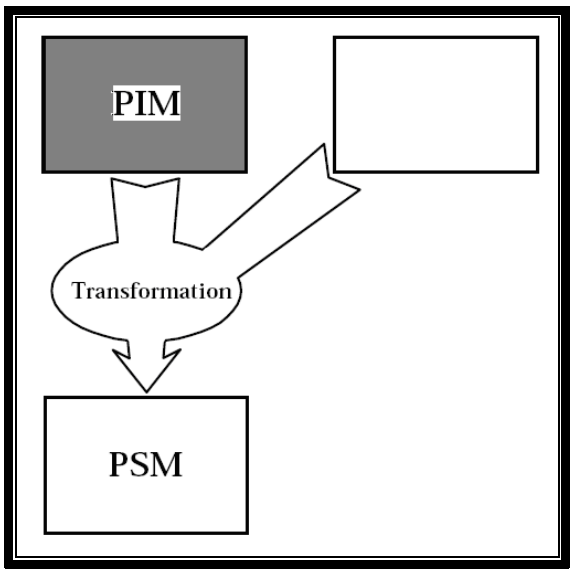

Figura 28. Fase da Transformação em que se Encontra o modelo PIM

Processos de boas-prácticas permitem às organizações melhorar os níveis de eficiência operacional. Deste modo, torna-se fundamental que as organizações possuam um repositório de processos de boas-prácticas, com processos que garantam a eficiência da empresa. Na elaboração das boas-prácticas deve-se ter em conta aspectos variados para além da eficiência do ponto de vista puro do negócio ou financeiro, como por exemplo, aspectos específicos do país de funcionamento da empresa ou questões de gestão ambiental, entre outros.

A utilização de boas práticas garante também à organização a obtenção de certificações de qualidade, como ISSO [ifSMF 2007], EFQM [EFQM 2001], entre outros. Estas certificações, para além de garantir que a empresa funciona de um modo eficiente, trazem benefícios em termos de imagem junto de clientes, fornecedores e colaboradores, o que também é positivo do ponto de vista do negócio.

No caso do Grupo Bosch, mais concretamente na sub-divisão *Car Multimedia* (CM), o modelo de referência para os seus processos de negócio é o UBK-RM (*Unternehmensbereich Kraftfahrzeugausrüstung - Reference Model*), que representa a Divisão de Tecnologia Automóvel. O primeiro nível dos processos é composto por três categorias de processos de negócio: de gestão, chave e de suporte. Os processos de gestão têm a função de suportar as tomadas de decisões por parte da gestão de alto nível. Os processos-chave são os que descrevem o funcionamento do negócio por parte da organização. Os processos de suporte não interferem no desenvolvimento do "produto", mas são igualmente necessários, pois, por norma, são eles que tratam da relação da empresa com os seus *stakeholders*. A este nível,

ainda não é possível estabelecer relações exactas entre os processos, pelo que a modelação em BPEL não se realiza. A modelação em BPEL só se justifica aquando da representação do processo ao nível das suas actividades, correspondente ao nível 3 do UBK-RM, que pode também ser comparado ao nível 3 sugerido pelo mLEARN [Coelho 2005] e SCOR [SCC 2008].

Para a modelação em BPEL do modelo PIM pretendido, o nível 3 do UBK-RM foi transformado em BPEL, obtendo-se assim um repositório de processos que representam as boas-prácticas. O exemplo dado neste capítulo é o processo de "Produção", pertencente ao processo-chave de "*Fulfillment*" (Figura 29).

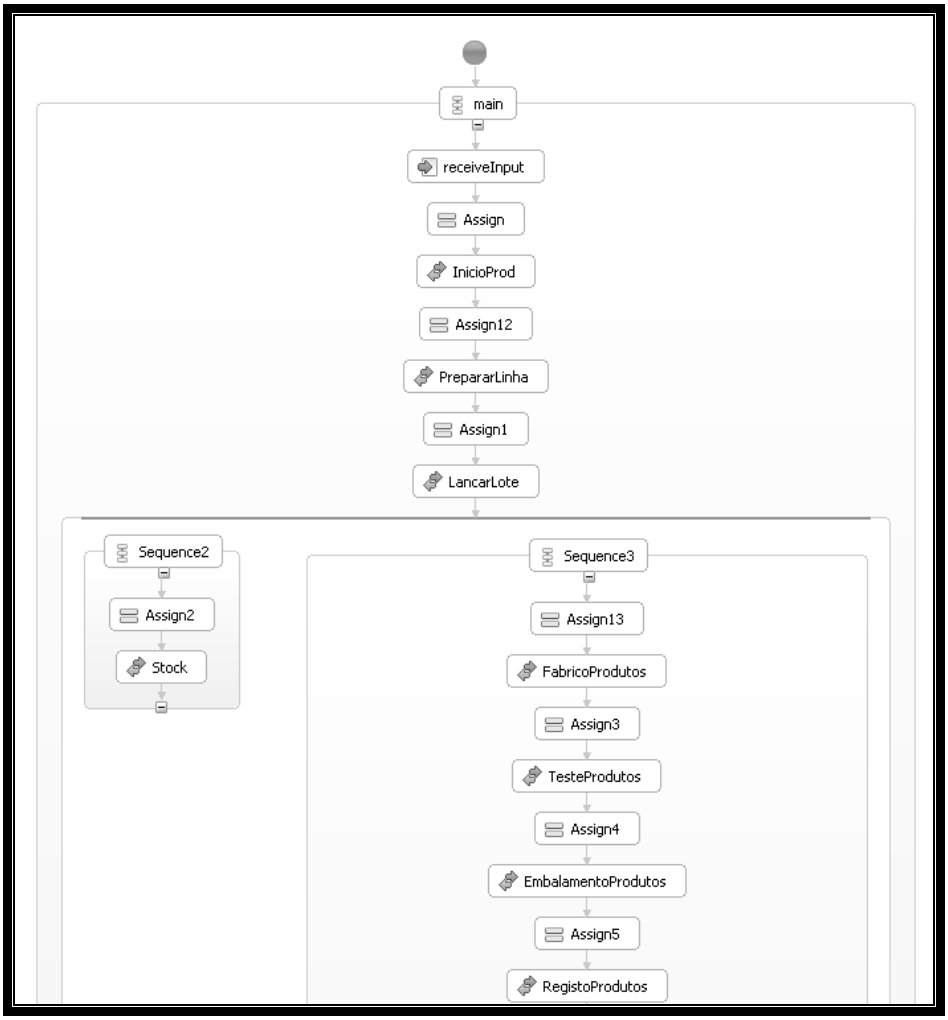

Figura 29. Excerto de Processo BPEL "Produção"

Na figura, pode-se ver um excerto do processo produtivo, que vai desde a preparação da linha, passando pelo consumo dos materiais, embalamento e – não representado na imagem – o armazenamento dos produtos. O início do processo é efectuado depois de realizado o planeamento do material. No processo de produção os materiais saem do stock, são consumidos e o produto – ou conjunto de produtos – é enviado para a área de expedição.

Acabado o processo, seguem-se os controlos de qualidade antes de irem parar aos clientes. Em resumo, o processo tem um processo antecessor e dois processos sucessores.

Generic PF de RegistoProdutos

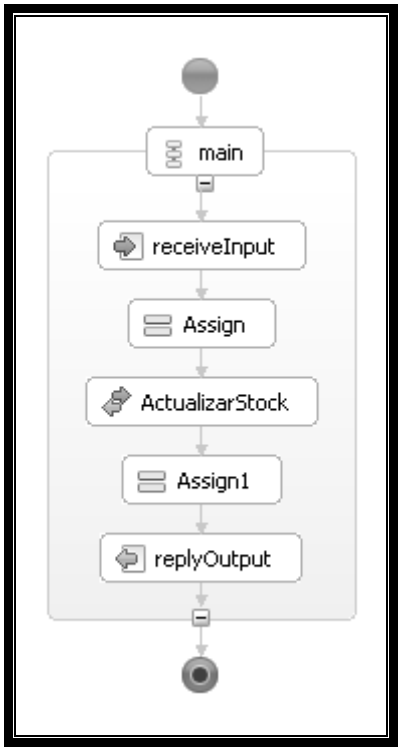

Figura 30. *Generic PF* "RegistoProdutos"

O "RegistaProduto" é um sub-processo do processo de Produção. Na Figura 30 está o modelo BPEL correspondente. O chefe de linha regista o produto / conjunto, após o consumo dos materiais, fabrico, teste e embalagem. Esta actividade resulta na actualização dos stocks. Visto que apenas é necessário uma actividade para este sub-processo, é necessário então apenas representar uma actividade *invoke* no modelo BPEL, neste caso denominado de "ActualizarStock".

3.4 Conclusões

Neste capítulo um dos objectivos propostos já foi cumprido. A linguagem de execução de processos de negócio a utilizar na *Bosch Car Multimedia Portugal, Lda*. foi escolhida tendo em consideração três tipos de estratégias: organizacionais, de negócio e de sistemas de informação. A análise de dois aspectos adicionais, ao invés de apenas considerar os vinte

padrões de *workflow* torna a escolha mais completa e fundamentada, pois assim a escolha não é puramente tecnológica. Visto que a análise pretende a escolha de uma linguagem para utilização de uma organização e não utilização individual, a análise à linguagem mais apropriada aos colaboradores da empresa, bem como a análise à linguagem mais apropriada ao negócio da organização, torna a escolha da linguagem mais abrangente e sincronizada com os reais interesses da empresa. Após muita investigação, ao nível de vários estudos dos padrões de *workflow*, percepções das pessoas e adequação ao nível do negócio, variados autores e variados aspectos relevantes e irrelevantes, distintos e não distintos, não sobraram dúvidas de que o WS-BPEL é mais adequado para a empresa em causa do que as restantes linguagens identificadas. Muitos autores definem o WS-BPEL como linguagem standard na execução de processos de negócio, mas essas afirmações não foram sequer tidas em conta durante a comparação, pois o alvo da comparação é apenas e só a empresa.

A BIM surge como uma metodologia interessante a estudar, pois sugere a utilização de modelos de referência na definição dos processos, em vez de simplesmente transformar em software os requisitos dos clientes. Neste capítulo ainda só a fase de Selecção é efectuada. Foram apresentadas quais as características que os processos de negócio possuem nesta fase, até estarem num estado em que já são executáveis. Numa fase inicial, para descrever os processos PIM, apenas é considerada a representação gráfica dos processos. É apresentado um exemplo de um modelo de processos que vai sofrer as transformações do seu estado ao longo da metodologia. O processo de exemplo, dado que pertence a um modelo de referência, servirá como base para as transformações seguintes propostas pela metodologia, até se atingir uma transformação que permita obter software.

4. Transformação Automatizada dos Processos

Neste capítulo, os processos de negócio em PIM vão sendo transformados até estarem devidamente prontos para serem transformados para PSM. Seguindo a BIM, os modelos dos processos são instanciados e alterados até se encontrarem executáveis, *i.e.*, são descritas as fases 2 e 3. Na segunda parte do capítulo, os conceitos da plataforma onde o software é executado são descritos. Estes conceitos são importantes na tarefa da transformação e da obtenção do PSM. As transformações são efectuadas seguindo os conceitos propostos pelo MDA, nomeadamente descrição da plataforma, mapeamento e marcação. Só após as três actividades se pode passar então à transformação em si. Finalmente, é apresentado um modelo PSM para demonstrar que a transformação resulta efectivamente numa implementação em software do processo de negócio.

4.1 Introdução

Ainda apenas a primeira fase da BIM, a Selecção, foi executada. O ponto de situação é que o processo está em PIM, mas, segundo a BIM, não está ainda pronto para ser transformado em software.

O modelo de processos está no seu estado genérico, pelo que apenas reflecte as propostas dos modelos de referência. O cliente pode agora expor a sua visão do negócio, pois tem como objectivo destacar-se e distinguir-se das organizações concorrentes. A BIM permite que três cenários: que seja o processo pedido pelo cliente a ser tratado na próxima fase, que seja o processo já obtido na fase anterior a ser tratado na próxima fase, não sofrendo instanciação, e por último, que seja um processo baseado no processo genérico mas que responda aos requisitos do cliente a ser tratado na próxima fase. Assim que se assegure que o processo está devidamente definido, passa-se então à fase de concretização no Sistema de Informação.

A última fase da BIM que ainda considera modelos de processos PIM, a de Concretização, também é tratada neste trabalho. É necessário assegurar que o processo está pronto para a transformação no final desta fase. Para tal, o modelo de processos deve estar totalmente concretizado no Sistema de Informação. Deve ser então definido a forma como o processo se "encaixa" na arquitectura da organização. No fim desta fase, o processo já é executável e está preparado para ser implementado em software.

A abordagem MDA é utilizada na transformação do modelo em software, considerando o modelo de processos obtido um PIM e o modelo de software a obter um PSM. Para ter o processo a executar em software, este será implementado num *Enterprise Service Bus* (ESB). O *Apache ServiceMix* [ServiceMix] é um ESB baseado em componentes JBI [Ten-Hove e Walker 2005].

A tarefa de transformação é executada seguindo os passos propostos pelo MDA. Este é um passo muito importante no trabalho, pois demonstra como um processo de negócio pode ser transformado em software de uma forma automatizada e assim, para além de resultar num desenvolvimento eficiente devido à utilização da BIM, resulta num desenvolvimento eficaz.

4.2 Transformação dos Processos PIM

Para se concluir as fases de Definição e de Concretização da BIM deve-se assegurar que os modelos dos processos são instanciados do modelo genérico (este passo não é obrigatório, o modelo pode passar para a fase seguinte sem ser instanciado desde que seja comprovado que os processos estão devidamente definidos e prontos a serem tratados na próxima fase) e que são executáveis. Estes modelos ainda são PIM e, assim que estão no estado de executáveis, estão na última fase antes de serem transformados em PSM.

Para melhor acompanhar a transformação dos processos, é sempre feito um ponto de situação da transformação tanto no âmbito da metodologia como no âmbito de MDA, pois os conceitos de MDA serão muito importantes na "preparação" e na transformação do modelo em sistemas de software, permitindo ao modelo de processos serem implementados em software.

Proposta de "Customização" do cliente

O passo seguinte para transformar os processos é efectuar o levantamento dos requisitos do cliente para este processo, para posteriormente efectuar a devida análise das transformações que o processo "*best-practice*" (PIM) irá sofrer de modo a satisfazer o cliente. De acordo com a BIM, esta fase corresponde a passar de um modelo *Generic* para um modelo *Instantiated* (Fase 1 para Fase 2). Para demonstrar a transformação, o exemplo escolhido foi o sub-processo de "Registo do Produto" já apresentado no capítulo 3. A escolha recaiu neste sub-processo pois não é um processo complexo – além de receber e responder ao pedido, apenas realiza uma actividade adicional – e, como é demonstrado mais adiante, a arquitectura em que este se integra é de grande importância.

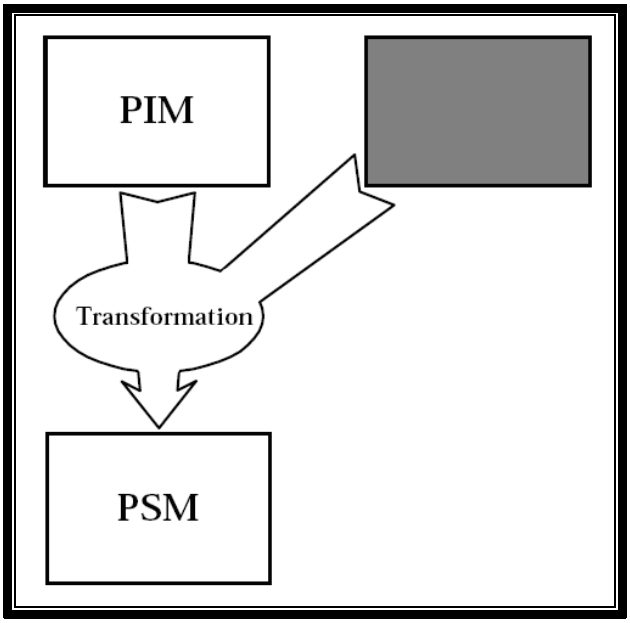

Figura 31. Fase da Transformação em que se Encontram os Requisitos

A proposta do cliente tem como objectivo acrescentar valor ao processo. No âmbito do negócio, o cliente tem como proposta no registo dos produtos/conjuntos "finais" (nesta altura do processo produtivo, os materiais já foram consumidos, embalados e testados, pelo que são considerados "produtos acabados") que seja adicionada uma entrada nos registos do novo "produto/conjunto acabado" e que seja feita uma actualização da quantidade de materiais em armazém para gestão do stock (Figura 32). A proposta também sugere que estas operações devem ser executadas pelo chefe de linha, dado que este é o responsável pelo supervisionamento do processo de produção do produto/conjunto em questão.

A actualização automática dos stocks traz valor à organização dado que permite, por exemplo, que a organização tenha conhecimento, em tempo real, que a quantidade de stock de um determinado material atingiu o seu valor de segurança, pelo que é aconselhável efectuar um pedido de encomenda ao fornecedor desse material. O registo de novo produto permitirá à organização que os dados do produto sejam persistentes, *i.e.*, que a informação seja registada e guardada para que possa ser acedida sempre que alguém da organização assim o entender.

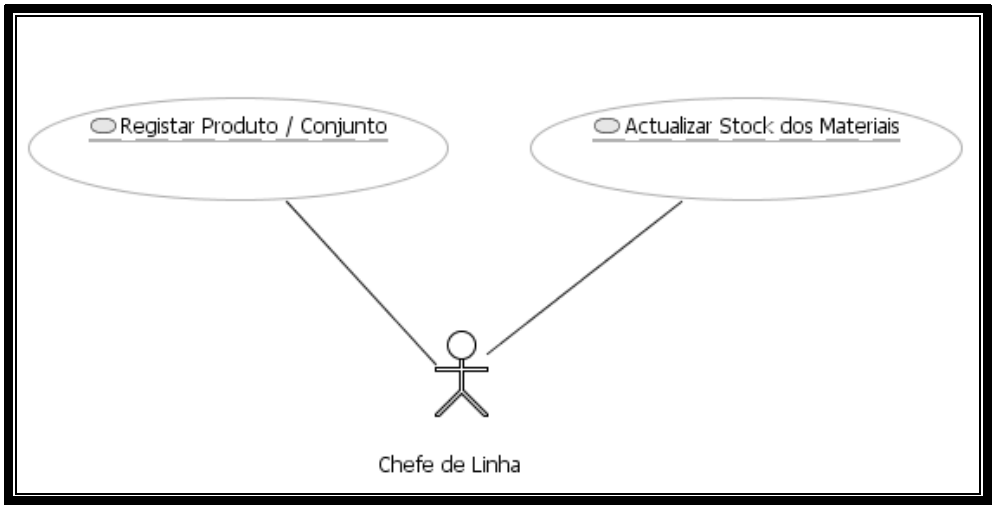

Figura 32. Operações Propostas pelo Cliente

Instantiated PF "Lançamento de Quantidades"

A segunda fase da BIM [Duarte, Machado *et al.* 2009], a Definição, descreve que o modelo genérico escolhido na fase anterior deve servir de base para se chegar a uma instanciação do modelo de processos. A metodologia descreve também que o modelo instanciado deve incluir a visão do cliente e a sua estratégia para o negócio.

Já foi visto anteriormente que uma proposta para acrescentar valor no registo dos produtos / conjuntos – o exemplo escolhido para a demonstração – é a de que este processo deve incluir duas actividades, adicionar registo de novo produto e actualizar quantidade do material consumido em stock. Na modelação em BPEL realizada anteriormente, em vez de apenas uma actividade de invocação, propõe-se que o processo deve ter então duas invocações (Figura 33).

O cliente refere que necessita de dar baixa dos materiais no stock, mas também de guardar os valores para consulta dos dados no futuro. Facilmente se identifica a necessidade de substituir a actividade proveniente do modelo genérico por duas actividades que satisfaçam os requisitos pedidos pelo cliente. Justifica-se uma invocação em paralelo pois os dados que serão introduzidos para cada uma das invocações são os mesmos. Sendo que as actividades se realizam em paralelo, e não sequencialmente, também faz com que o processo só termine se ambas as actividades forem executadas com sucesso – o que não aconteceria se fossem sequenciais –, o que é necessário para melhorar o negócio da organização.

De referir que o processo descrito anteriormente e representado na Figura 34 trata-se de uma proposta, pelo que podem ser desenvolvidas alternativas ao modelo instanciado. A BIM é uma metodologia flexível no que toca à quantidade de modelos desenvolvidos, pois

permite a alteração fácil do modelo definido em qualquer uma das suas fases e também a independência de tecnologias.

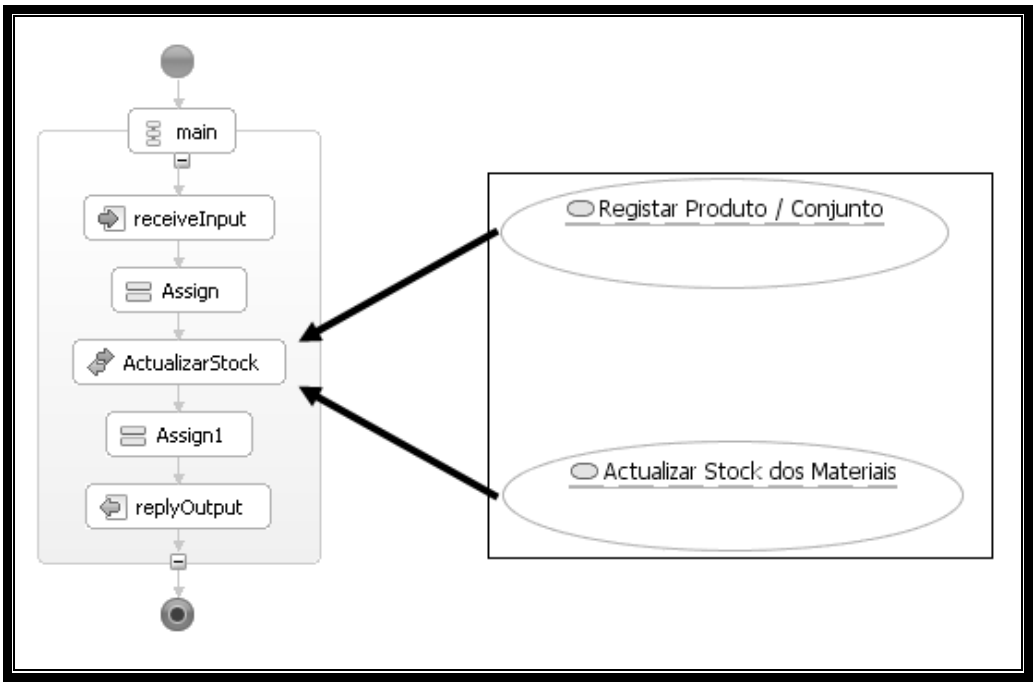

Figura 33. Processo de BPEL de "Registar Produto" e Actividades de Customização

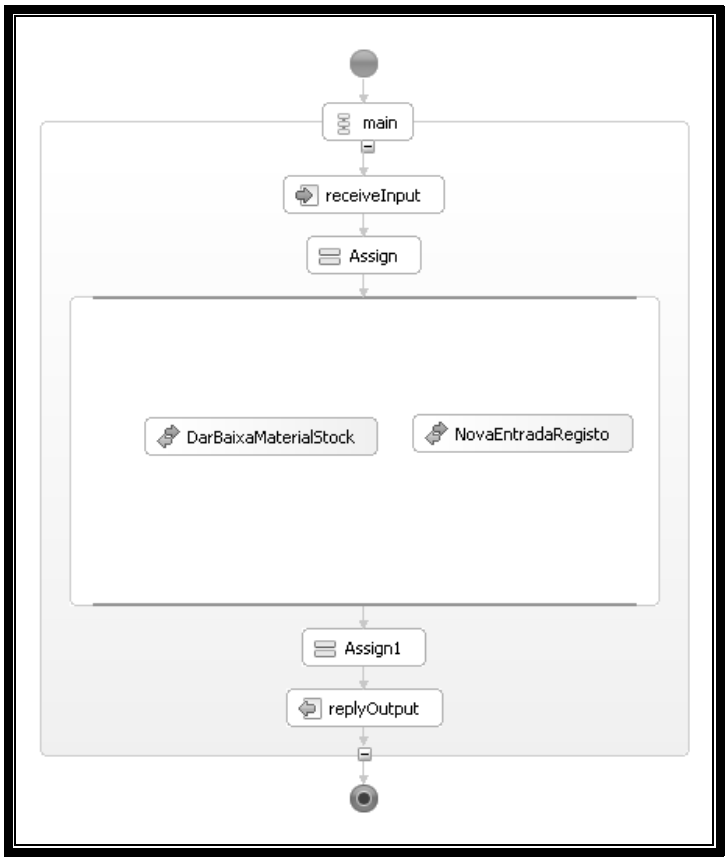

Figura 34. *Instantiated PF* "Lançamento de Quantidades"

Runnable PF "Lançamento de Quantidades"

É necessário agora concretizar o modelo no sistema de informação. A primeira tarefa a realizar será analisar que tipos de tecnologias podem ser utilizados para suportar a componente automática do processo. Uma proposta será utilizar um sistema ERP, muito utilizado no suporte a processos na larga maioria das empresas, entre as quais também se insere a *Bosch Car Multimedia Portugal, Lda.* Aliás, nesta empresa, o ERP é mesmo o principal sistema de suporte aos processos de negócio. O sistema de ERP utilizado é o sistema da SAP. No entanto, utilizar um sistema de gestão de bases de dados para inserir novos registos directamente na base de dados pode ser uma solução mais adequada.

Logo, além de actualizar o stock introduzindo os dados num ERP, é necessário guardar os valores numa base de dados. Uma proposta será então uma invocação, em paralelo, de um sistema de ERP e outra de um sistema de Base de Dados. Analisando a arquitectura deste, verifica-se que o processo é iniciado a partir de um software instalado num terminal local ou num terminal móvel (PDA). Como sistema de gestão de bases de dados estão em implementados na organização dois sistemas, Oracle e MySQL, pelo que qualquer um deles pode assegurar o registo do novo produto.

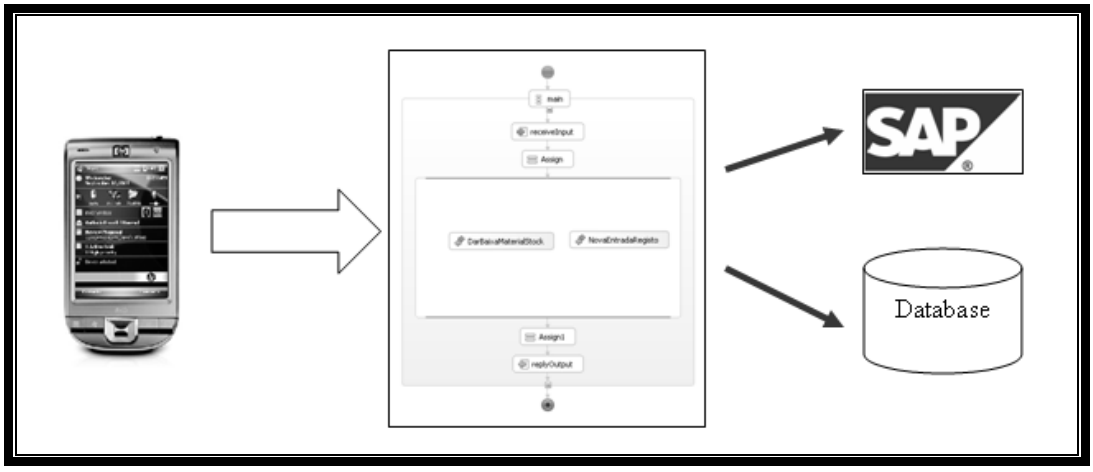

Figura 35. Representação das Relações dos Softwares que Executam o Processo

No que toca à transformação do modelo BPEL para a fase de Concretização, é bastante simples. Sendo que na fase anterior já se identificaram duas actividades que são executadas em paralelo – registar novo produto e dar baixa no stock – e, através da análise à arquitectura, já se definiu que serão os sistemas de ERP e de Base de Dados a suportarem as actividades, o modelo BPEL não irá sofrer transformações significativas de modo a estar no estado de *Runnable*. As actividades BPEL *invoke* definidas no modelo anterior são substituídas por

actividades *invoke* que invocam os sistemas de ERP e Base de Dados. De resto, na sua disposição, não existem mais quaisquer alterações nas actividades do modelo BPEL (Figura 36).

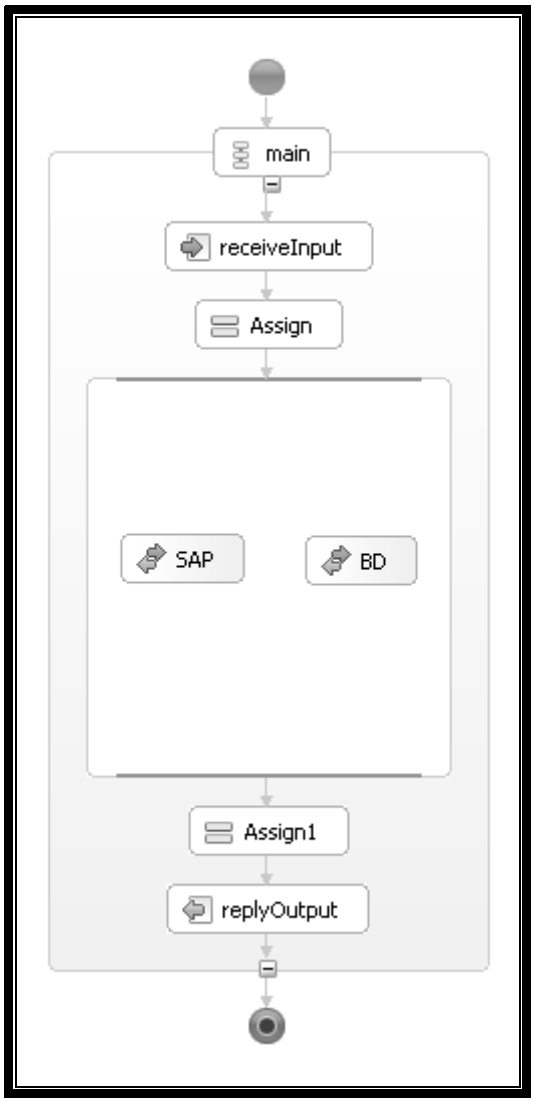

Figura 36. *Runnable PF* de "Lançamento de Quantidades"

Para completar a fase de Concretização da BIM é necessário que o PF esteja devidamente concretizado no Sistema de Informação. Para assegurar que tal acontece, já é necessário descrever devidamente o processo, pelo que a sua representação gráfica já não é suficiente. De seguida, são especificadas quais as localizações de cada actividade do processo, bem como o tipo de mensagens que são trocadas. Na Figura 37 está representado parte do código BPEL, nomeadamente a parte das importações (linhas 1 a 4) e a parte da sequência da actividade de invocação dos serviços externos em paralelo (linhas 5 a 14). A razão de representar estas partes prende-se com as ligações aos serviços realizadas, nomeadamente a utilização dos *namespaces*, de modo a que o orquestrador consiga identificar a localização

onde o serviço está alocado. Estes dados irão ser considerados na tarefa de transformação para PSM, visto que é provável que a transformação aborde estes tipos de elementos.

```
      ...

[1]   <bpel:import namespace="http://Sap" location="SapArtifacts.wsdl"
[2]   importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
[3]   <bpel:import namespace="http://Bd" location="BdArtifacts.wsdl"
[4]   importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>

      ...

[5]   <bpel:flow name="Flow">
[6]   <bpel:invoke name="SAP" partnerLink="sap" operation="process"
[7]   inputVariable="sapRequest" outputVariable="sapResponse"
[8]   portType="ns3:Sap">
[9]   </bpel:invoke>
[10]  <bpel:invoke name="BD" partnerLink="bd" operation="process"
[11]  inputVariable="bdRequest" outputVariable="bdResponse"
[12]  portType="ns1:Bd">
[13]  </bpel:invoke>
[14]  </bpel:flow>

      ...
```

Figura 37. Excerto do Código BPEL Final

Visto que o PF se encontra no estado *Runnable*, já deve ser considerado para o processo estar em funcionamento, não apenas o BPEL, mas também o WSDL associado ao processo (Figura 38). Já foi explicado anteriormente na secção 2.2 e 3.2 que cada processo BPEL é composto por um ficheiro BPEL que possui um interface correspondente num ficheiro no formato WSDL.

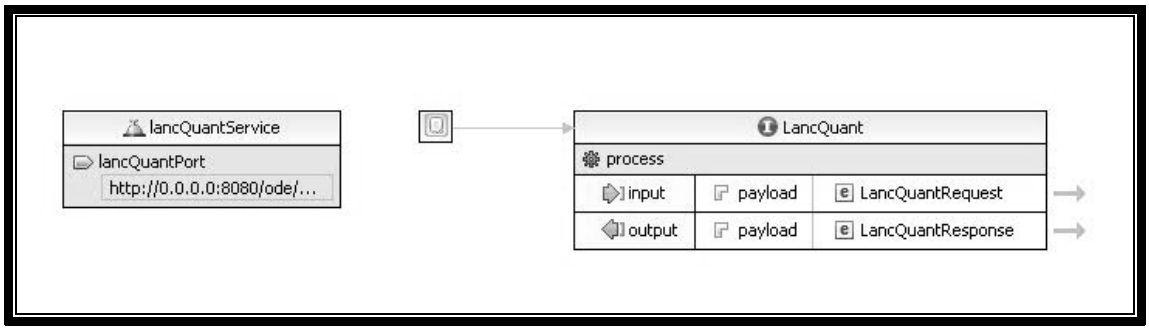

Figura 38. Representação do WDSL de "Lançamento de Quantidades"

O que importa ressalvar do WSDL são os dados referentes ao serviço, ligação, tipo de porta, operações e elementos de entrada e saída. O processo ficou definido como fornecedor de um serviço de "*lancQuantService*", em que as ligações são efectuadas através de um

"*LancQuantPort*", localizado em "*http://0.0.0.0:8080/ode/processes/LancQuantService*". O tipo de porta ficou definido como "*LancQuant*", que fornece apenas a operação "*process*". A operação "*process*" corresponde neste caso ao processo principal, tal como já tinha sido definido no código BPEL (nas actividades de "*receive*" e "*reply*") e tem como elementos de entrada e saída "*LancQuantRequest*" e "*LancQuantResponse*", respectivamente. O código-fonte do processo BPEL e do seu respectivo interface WSDL encontram-se em anexo E.

Pode-se pensar que, após obter o processo no estado executável e assegura a concretização no Sistema de Informação, esta fase está completa. Analisando com mais pormenor a fase de Concretização, nomeadamente as tarefas definidas para o responsável pelo processo de negócio, verifica-se que estas são a de decidir se o processo deve ser implementado ou não em software e de definir processos alternativos a conceber. Esta última é altamente aconselhável, no âmbito da gestão de risco da organização. No caso do processo apresentado, o processo vai ser implementado em software (e descrito na próxima secção). Podem ser apresentadas alternativas de processos que avançam para a fase seguinte da BIM mas também é adequado que sejam apresentadas propostas para processos cujos modelos já se encontram nos seus estados finais, *i.e.*, não avançam para a próxima fase e serem implementados em software. Serão portanto processos de negócio puramente "manuais".

Mais uma vez se realça que este modelo não deve ser o único desenvolvido nesta fase da BIM. A conveniência de existir uma alternativa ao modelo de processo implementado em software justifica-se pelo facto de que qualquer um dos componentes que participam no processo de negócio está sujeito a falhas, pode passar por fases de baixa performance, necessidade de actualizações ou sobrecarga de tarefas, entre outras possíveis razões. Possuir um modelo alternativo diminui os riscos para a organização, mas não os elimina completamente. Possuindo mais alternativas, é possível afirmar que o risco diminui exponencialmente, mas, como é óbvio, nunca é possível eliminar o risco por completo.

Tal como referido anteriormente, é adequado que seja apresentada pelo menos uma alternativa em que o modelo represente um processo que não seja implementado em software. Nesta fase da BIM, o modelo no estado executável deve ser obtido a partir do modelo no estado instanciado ou genérico, mesmo que, na elaboração inicial desses modelos, o propósito da elaboração do modelo fosse que este percorresse todas as fases da BIM. A partir do exemplo do modelo instanciado apresentado anteriormente, a proposta para alternativa para um processo executável e concretizado na organização será a de executar as actividades já definidas mas sem recorrer a qualquer tipo de tecnologia, como por exemplo realizar os registos em papel. Seriam então efectuados os registos relativos à saída de material, bem como os registos da entrada de novos produtos acabados para a zona de expedição.

4.3 Modelo de Processos em PSM

O modelo de processos de negócio já percorreu todos os seus estados antes de ser implementado em software. Nesta fase, o modelo deixará de ser PIM para ser transformado em PSM. Os processos serão executados num *Enterprise Service Bus* pois permite a integração das aplicações de uma forma standard. Para melhor se perceber a transformação, os conceitos da arquitectura tecnológica são apresentados de uma forma generalizada, devido à quantidade e complexidade dos conceitos, para não sair dos aspectos essenciais.

A tarefa de transformação demonstra a forma como os elementos do processo PIM irão ser inseridos nos componentes da plataforma. No fim de definir os componentes com as transformações, o processo de negócio está então implementado em software, tal como definido na BIM. Sendo esta fase completada, o projecto terá então percorrido toda a metodologia com sucesso.

Transformação do modelo em PSM

No final da secção 4.2, o PIM está modelado e os requisitos do cliente para o processo estão levantados. Estão então reunidos os conteúdos necessários para se especificar as transformações necessárias para a plataforma onde o processo vai ser implementado (Figura 39). É nesta fase que todos os detalhes necessários para a transformação final do modelo PIM para um modelo PSM são definidos. Esta tarefa é tão ou mais crucial do que as tarefas realizadas anteriormente e a tarefa da própria obtenção do PSM, dado que os detalhes da transformação são toda a base para a obtenção do PSM, pelo que os detalhes devem ser analisados ao pormenor.

Seguindo o modelo proposto pela OMG para o MDA (Figura 40), para efectuar a fase de transformação do modelo é necessário descrever a plataforma, seguindo-se o mapeamento e a marcação dos elementos que irão ser transformados. Seguindo as especificações da OMG, a tarefa de transformação é feita de forma automatizada e, dado o nível de detalhe de cada uma das descrições efectuadas, o risco de erros na transformação do modelo é então reduzido. Só depois é que já é possível realizar-se o processo de transformação do modelo.

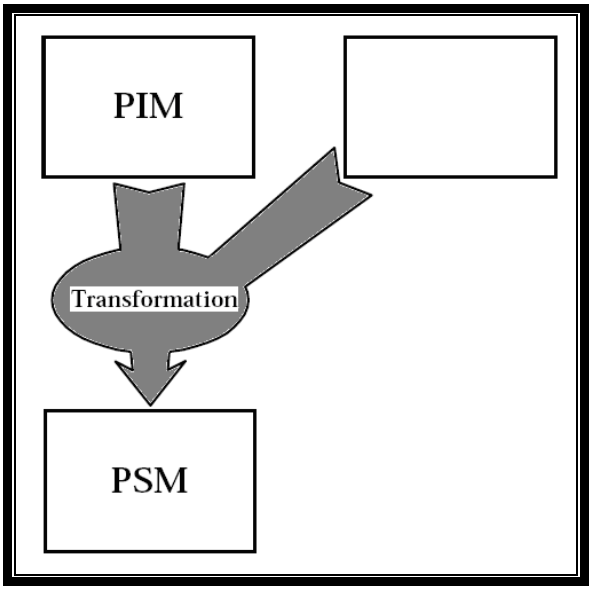

Figura 39. Fase da Transformação em que se Encontra a Transformação do Modelo

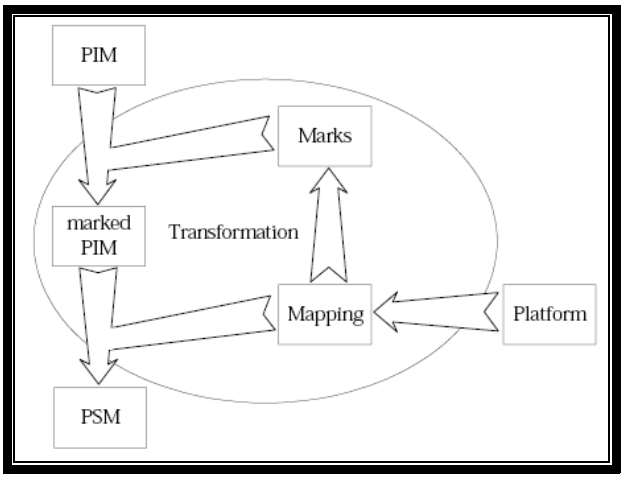

Figura 40. Transformação dos Modelos [OMG 2003]

Plataforma

Os processos suportados por software vão ser executados "dentro" de um *Enterprise Service Bus* (ESB). A partir da arquitectura do Sistema de Informação em que o processo é executado, definido na secção anterior para se chegar ao estado executável (*Runnable*) na fase de Concretização da BIM, é identificado um dos artefactos mais importantes da BIM, os *Orchestrated Business Objects* (OBO) [Duarte, Machado *et al.* 2009]. Para a identificação dos OBO's, deve-se iniciar por identificar quais as tecnologias presentes na arquitectura representada na Figura 35. É necessário também ter em atenção as especificidades tecnológicas que o ESB que vai suportar o processo, o *ServiceMix*.

Na arquitectura identificam-se quatro tecnologias diferenciadas: o software-cliente, que dá ordem de inicio ao processo através da sua invocação; o BPEL, com a função de orquestrador do processo que define a ordem como os serviços serão invocados; o ERP, que executa a actividade de saída de material do stock; e a Base de Dados, que executa a actividade de registo de novo produto para expedição. Analisando as características do funcionamento do *ServiceMix* quanto a execuções e ligações externas, é possível identificar a necessidade de três *Binding Components* (BC's), de modo a possuir ligações para o ERP e a Base de Dados e outra ligação de pedido e resposta ao cliente. Um tipo de BC adequado para a tarefa é um que interprete *Web Services*, dado que as características do protocolo SOAP são das mais apropriadas para enviar pedidos e respostas para um dado cliente. No que toca às ligações com o ERP e a Base de Dados, esta opção baseia-se no facto de ainda não existir um *Service Engine* (SE), para funções, neste caso, no ERP SAP, e também dado que assegura ligações *Java Database Connectivity* (JDBC) [Sun]. Para estes dois últimos, a execução do *Web Service* é feita através de SE's, ou seja, um *SAP SE* e de um *BD SE.* Necessita também de um *BPEL SE* (na Apache denomina-se de *Orchestration Director Engine* – ODE) [ODE], para executar o *workflow* do processo e orquestrar os serviços.

Resumidamente, executar o processo de negócio no ESB necessitará de seis OBO's, correspondentes a seis componentes JBI: um BPEL SE para executar a orquestração do *workflow*; dois CXF SE's, um para SAP e outro para a BD, para inserção dos dados através de *Web Services*; e três CXF BC's, dois que efectuarão a ligação dos Web Services definidos nos correspondentes CXF SE's ao sistema externo respectivo, e outro que efectuará a ligação do pedido do cliente ao processo e a resposta do processo ao cliente (Figura 41).

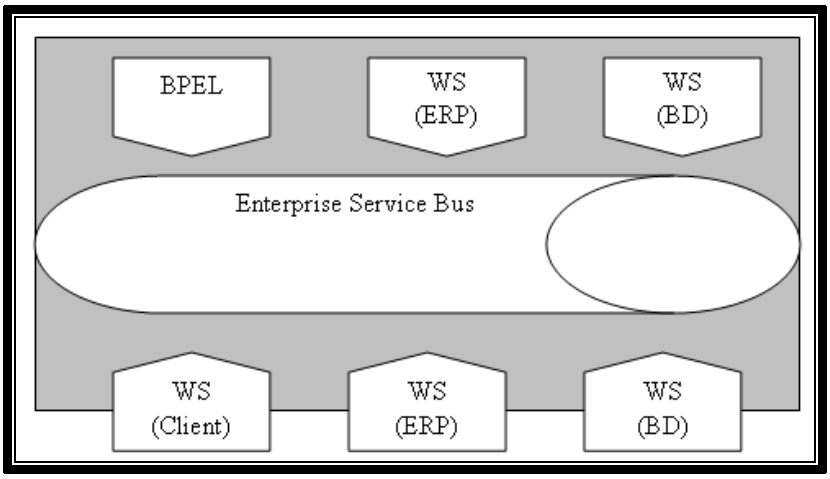

Figura 41. Componentes da Plataforma Criados Para Executar o Processo

Mapeamento

Tabela 8. Relações do Mapeamento PIM-PSM

Elemento PIM	→	**Elemento PSM**
WSDL – Tipos dos Elementos do tipo *Request*	→	Parâmetros de entrada da classe do POJO do SU (CXF SE's)
WSDL – Nome do Serviço ("*LancQuantService*", *"SapService*" e "*BdService*")	→	Nome do *Web Service* no ficheiro JAVA pertencente ao SU (CXF SE's)
		targetService do SU (CXF BC's)
		Nome do serviço do *partnerLink* respectivo no SU (BPEL SE "ODE")
WSDL – Tipo de Porta do Serviço ("*LancQuant*", "*Sap*" e "*Bd*")	→	*targetInterface* do SU (CXF BC's)
WSDL – Nome da Porta do Serviço ("*LancQuantPort*", "*SapPort*" e "*BdPort*")	→	*targetEndpoint* do SU (CXF BC's)
		Nome da porta do serviço do *partnerLink* respectivo no SU (BPEL SE "ODE")
BPEL – *namespaces* dos serviços importados	→	*namespaces* nos ficheiros *xbean.xml* e *targetNamespace* no ficheiro JAVA do SU (CXF SE's)
		namespaces dos ficheiros WSDL que são *PartnerLinks* no ficheiro BPEL
BPEL – Ficheiros WSDL importados	→	*wsdl* do SU (CXF BC), gerados pelo CXF SE
BPEL – Variável "input" e WSDL – Elemento *Request*	→	Parâmetros de entrada do *Web Service* do cliente
		Tipo de elemento "input" do *Request* do WSDL
		Variável "input" do BPEL
BPEL – Actividade *Invoke* "SAP"	→	Actividade BPEL *Invoke "*SAP SE"
BPEL – Ficheiro WSDL que é *PartnerLink* "Sap"	→	Ficheiro WSDL gerado pelo CXF SE
BPEL – Actividade *Invoke "BD*"	→	Actividade BPEL *Invoke "*BD SE"
BPEL – Ficheiro WSDL que é *PartnerLink* "BD"	→	Ficheiro WSDL gerado pelo CXF SE
BPEL – Variável "output" e WSDL – Elemento *Response*	→	Parâmetros de retorno do *Web Service* do cliente
		Tipo de elemento "output" do *Response* do WSDL
		Variável "output" do BPEL

O mapeamento (Tabela 8) que descreve as relações entre os artefactos dos modelos de PIM e de PSM reflectem um grupo de regras que a transformação irá seguir. A tarefa de mapeamento não apresenta um grau de dificuldade elevado, dado que, neste caso, o processo modelado em BPEL apresenta apenas seis actividades.

As actividades de *invoke* pretendem que um determinado serviço lhes envie uma resposta a um determinado pedido, pelo que o componente do ESB que corresponde a estas características é o *Service Engine.* Assim, o mapeamento é directo, pelo que as actividades de *invoke* irão invocar o OBO respectivo, *i.e.*, a invocação "SAP" está relacionada com o OBO "*SAP SE*", enquanto que a invocação "BD" está relacionada com o OBO "*BD SE*". As duas actividades de *assign* não sofrem alterações para o novo modelo, pois são igualmente necessárias no novo modelo e o seu conteúdo não é modificado, pois os valores que este atribui às variáveis também se mantêm inalterados. Os *namespaces*, bem como os dados do serviço do WSDL – *portType, service* e *port* –, correspondem aos dados de identificação dos SU's que vão ser criados. O ficheiro WSDL é, aliás, o ficheiro que "circula" no NMR.

O ficheiro WSDL é o mais importante na transformação, dado que apenas a definição das mensagens (partes e tipos) não são consideradas na transformação, pelo que o programador define conforme entender. No que toca ao ficheiro BPEL, a definição dos *PartnerLinks* e das Variáveis não são abrangidas na transformação, pois a definição do *PartnerLinkType* fica igual, e o nome e o *role* não são considerados na transformação e ficam ao critério do programador.

Marcação

Tal como a tarefa de mapeamento, também a de marcação é simples, dado a baixa complexidade do modelo de processos. A marcação dos elementos é feita a partir das relações identificadas no mapeamento. Consultando a coluna dos elementos PIM da tabela 8, estão então os elementos a serem marcados, respectivamente listados na tabela 9 e representados na Figura 42. A marcação é útil no sentido em que, tal como pode ser observado na tabela 8, um elemento no PIM pode dar origem a mais do que um elemento no PSM. Para estes casos, a técnica da marcação é utilizada para que não se corra o risco de algum elemento ficar "esquecido" na transformação, o que resultaria num modelo PSM incompleto. Em alguns casos, é aconselhável listar cada elemento a marcar, e não generalizar os elementos pelo seu tipo, como nos casos em que a listagem refere o mesmo tipo de elementos, mas aplicáveis a "Sap" e a "Bd".

Mais fácil do que consultando ambas as tabelas anteriores, na Figura 42 é possível ver uma representação do mapeamento e da marcação, onde, do lado esquerdo da figura, estão os elementos do PIM mapeados e que vão ser marcados e, do lado direito, está o respectivo elemento PSM resultante da transformação a ser realizada.

Tabela 9. Elementos para Marcação

Elemento PIM
Actividade BPEL *Invoke* "SAP"
Actividade BPEL *Invoke* "BD"
Namespace do "*SapService*" importado no BPEL
Namespace do "*BdService*" importado no BPEL
Ficheiro *SapArtifacts.wsdl* importado no BPEL
Ficheiro *BdArtifacts.wsdl* importado no BPEL
PortType "*Sap*" do serviço no WSDL
PortType "*Bd*" do serviço no WSDL
ServiceName "*SapService*" do serviço no WSDL
ServiceName "*BdService*" do serviço no WSDL
PortName "*SapPort*" do serviço no WSDL
PortName "*BdPort*" do serviço no WSDL
Elementos/Variável do tipo "input" no WSDL/BPEL
Elementos/Variável do tipo "output" no WSDL/BPEL

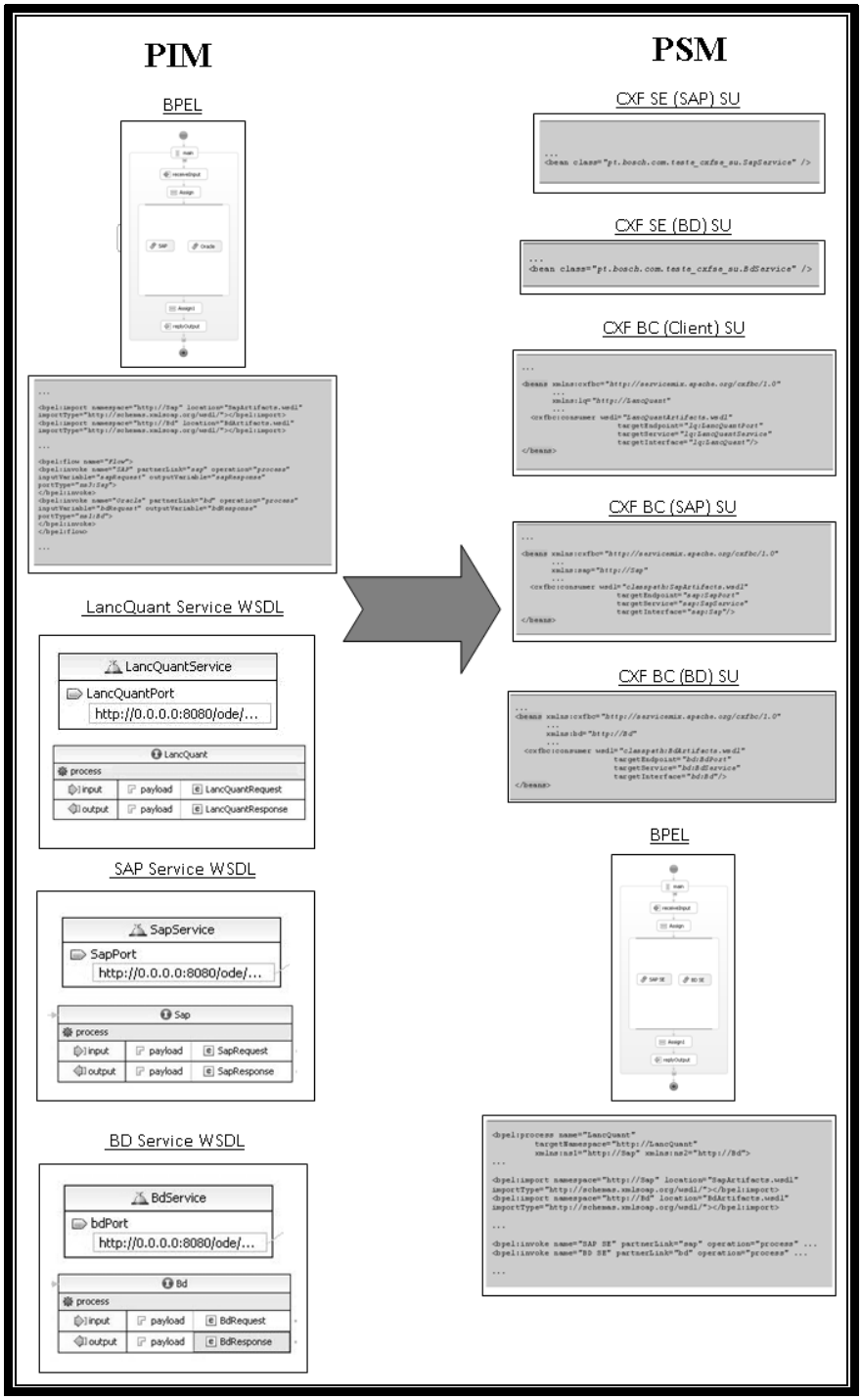

Figura 42. Mapeamento Entre os Componentes PIM e PSM

As tarefas representadas na Figura 40, de modo a definir a transformação, nesta fase já foram efectuadas. As descrições sobre a plataforma e dos elementos que vão ser transformados e que vão dar origem a outros elementos para o modelo final permitem que a transformação do modelo PIM do processo de negócio para o modelo PSM seja efectuada de modo automatizado. As tarefas de mapeamento e de marcação podem ser feitas de maneira

automática utilizando o MOF QVT [OMG 2005]. Introduzindo quais os elementos a transformar e os elementos que resultam da transformação, o código final é obtido. Neste caso, sendo que o processo pode ser considerado simples, o uso do QVT não é essencial, pois, como já foi visto, as alterações são poucas, pelo que a utilização deste standard da OMG fica à consideração de cada um. O QVT por norma é útil nos casos em que os modelos e as transformações são mais complexos.

Exemplo de Processo PSM

O processo de transformação a que o MDA se refere constitui o último passo na obtenção de um modelo PSM (Figura 43). O modelo PSM obtido representa o processo de negócio totalmente implementado em software, com as devidas especificidades da plataforma tecnológica onde o processo é executado.

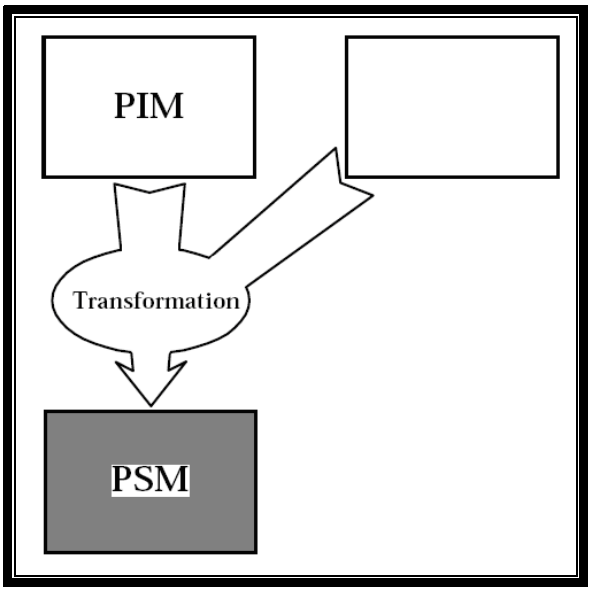

Figura 43. Fase da Transformação em que se Encontra o PSM

Tal como já visto anteriormente, as especificações para a transformação para a plataforma em causa passam pela descrição da plataforma (*Apache ServiceMix*), mapeamento das relações e marcação dos elementos. Estas três descrições constituem a base para a devida transformação para PSM. A partir da consulta da descrição da plataforma verifica-se que existe um conjunto de propriedades que é necessário definir. Revisitando a descrição da plataforma, conclui-se que o ESB utilizado requer a definição de *Service Engines* (SE), *Binding Components* (BC), *Service Assemblies* (SA) e *Service Units* (SU).

Seguidamente, irá ser feita uma demonstração da criação e da definição dos SE's, BC's e SU's (contidos posteriormente no respectivo SA). De modo a dar seguimento ao raciocínio apresentado ao longo deste capítulo e do anterior, o processo de negócio a implementar no ESB *ServiceMix* será o exemplo de processo que tem vindo a ser demonstrado ao longo da metodologia. A definição dos SU's, tal como os restantes componentes JBI, é feita criando um projecto *Maven 2* [Maven]. O Maven 2 é um software *open source* para gestão e automatização de projectos em Java, desenvolvido pela *Apache Software Foundation* [Apache]. O método baseia-se na construção de um *Project Object Model* (POM), onde se descreve todo o processo de construção de um projecto, ou seja, as suas directorias, dependências (*jars*) e a sua sequência de construção. O *Maven 2* fornece uns modelos ("*archetypes*") com código que pode ser reutilizado. Os modelos fornecem todos os dados necessários para a posterior compilação dos SU's.

Service Engines

O primeiro passo é definir SU's que, depois de efectuado o *deploy* no *ServiceMix*, irão criar SE's. O tipo de SE pretendido é do tipo CXF. Mais uma vez se refere que esta opção se baseia no facto de ainda não existir um SE para funções no ERP SAP. O funcionamento do SU é descrito num ficheiro *xbean.xml*, gerado a partir de um modelo *Maven* específico para *ServiceMix*. A definição do tipo é feita logo na primeira linha de código, onde é definido o *namespace* ("xmlns:cxfse=…"), enquanto que o resto do código descreve elementos necessários para a constituição do SU. Este ficheiro é bastante simples pelo facto de que apenas indica onde está localizado a classe do *POJO* (*Plain Old Java Object*) [Johnson, Hoeller et al. 2005] que é relevante para o *Service Engine*.

Neste caso, é à classe "*SapService*", que se refere o *Web Service* exposto pelo *CXF SE*. O código que antecede diz respeito à localização definida aquando da definição dos *groupId* e *artifactId* para identificação do ficheiro e projecto referente. O SU contém também um ficheiro Java [Armstrong, Ball *et al.* 2004], neste caso *SapService.java*, que é o *Web Service* que é executado pelo *CXF SE*. Este ficheiro Java utiliza o *SAP Java Connector* (JCo) [Schuessler 2001] para efectuar inserção e recepção de dados entre o *Web Service* e o ERP SAP, obedecendo assim correctamente aos requisitos de uma ligação ao ERP. Após compilação via *Maven 2*, a classe Java *"SapService*" vai criar um novo ficheiro WSDL (Figura 44), que neste caso foi renomeado para "*SapArtifacts*" por uma questão de simplicidade no mapeamento dado ser esse o nome dado pela ferramenta de design de BPEL do Eclipse para os WSDL's. É este

ficheiro WSDL que o BPEL final (ainda a descrever posteriormente) irá invocar. Dentro desse ficheiro, o *namespace* a definir para o ficheiro é o que advém do mapeamento do PIM.

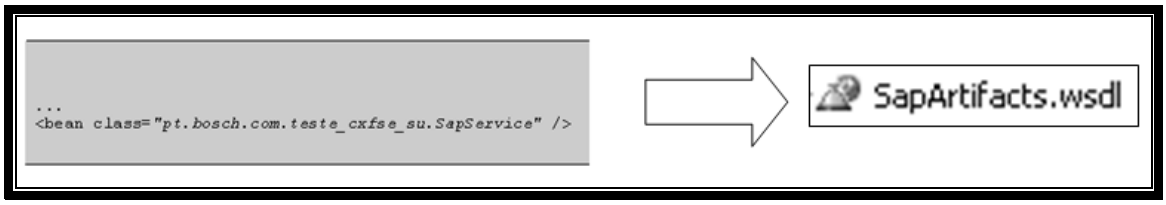

Figura 44. Ficheiro de Serviço WSDL Criado a Partir da Classe Java

Tal como o CXF SE que executa a introdução dos valores no SAP, a definição do SU para o CXF SE para a BD é semelhante. A utilização de um *CXF SE* prende-se com o facto de este suportar ligações JDBC. A diferença da definição deste SU para o SU anterior é apenas a da definição da classe do *POJO*, que será referente à classe "*BdService*", o *Web Service* exposto pelo *CXF SE*. Este ficheiro, "*BdService.java*", conterá no seu código a ligação JDBC para assim comunicar com a Base de Dados. Também neste caso um WSDL é gerado após compilação via *Maven 2*, tal como no caso anterior.

Binding Components

Já foi explicado como o SU que dará lugar a um SE depois de efectuado o *deploy* no *ServiceMix* deve ser configurado. A tarefa de definir um SU que define um futuro BC é semelhante (Figuras 45 e 46). Tal como anteriormente, o tipo de componente é definido através da primeira linha de código ("xmlns:cfxbc=…"). Neste caso, Os dados dos elementos do que definem o BC devem ser preenchidos com os dados do BPEL e do WSDL do modelo PIM tal como sugere a tabela 8. Tal permitirá uma correcta identificação da localização dos componentes pretendidos pelo ESB, que assim pode direccionar correctamente os dados (linhas de código 3 a 6). Dado que o pedido de inicio do processo é feito pela invocação de um *Web Service* por parte do cliente, então é necessário que a comunicação entre o cliente e o processo em BPEL seja feita através de um *CXF BC* e que este seja configurado da forma como está sugerido na Figura 45.

No código BPEL está definido que o processo invoca dois processos externos, o que indica a necessidade de mais dois componentes adicionais. O tipo de componente necessário é um *CXF BC* que comunicará os dados ao CXF SE definido na Figura 46, enviando e recebendo os valores de um *Web Service*.

```
     ...

[1]  <beans xmlns:cxfbc="http://servicemix.apache.org/cxfbc/1.0"
            ...
[2]         xmlns:lq="http://LancQuant"
            ...
[3]    <cxfbc:consumer wsdl="LancQuantArtifacts.wsdl"
[4]                     targetEndpoint="lq:LancQuantPort"
[5]                     targetService="lq:LancQuantService"
[6]                     targetInterface="lq:LancQuant"/>
[7]  </beans>
```

Figura 45. Código *xbean.xml* do SU do tipo *CXF BC* para Executar o Processo

```
...

<beans xmlns:cxfbc="http://servicemix.apache.org/cxfbc/1.0"
       ...
       xmlns:sap="http://Sap"
       ...
  <cxfbc:consumer wsdl="classpath:SapArtifacts.wsdl"
                  targetEndpoint="sap:SapPort"
                  targetService="sap:SapService"
                  targetInterface="sap:Sap"/>
</beans>
```

Figura 46. Código *xbean.xml* do SU do tipo *CXF BC* para Executar no SAP

Para inserir os dados na base de dados, a tarefa será igual ao caso do ERP, *i.e.*, os valores são inseridos através de um *Web Service*. Assim, o *Binding Component* para comunicar com o ESB será igualmente do tipo CXF BC (Figura 46).

```
...
<beans xmlns:cxfbc="http://servicemix.apache.org/cxfbc/1.0"
       ...
       xmlns:bd="http://Bd"
       ...
  <cxfbc:consumer wsdl="classpath:BdArtifacts.wsdl"
                  targetEndpoint="bd:BdPort"
                  targetService="bd:BdService"
                  targetInterface="bd:Bd"/>
</beans>
```

Figura 47. Código *xbean.xml* do SU do tipo *CXF BC* para Executar na BD

O modelo PSM fica completo com a representação do BPEL. Tal como se pode ver na Figura 48, em termos de notação visual este não sofre alterações, dado que o mapeamento ("SAP" para "SAP SE" e "BD" para "BD SE") não requer adição ou remoção de actividades BPEL. Na Figura 49 também é possível visualizar parte do código BPEL.

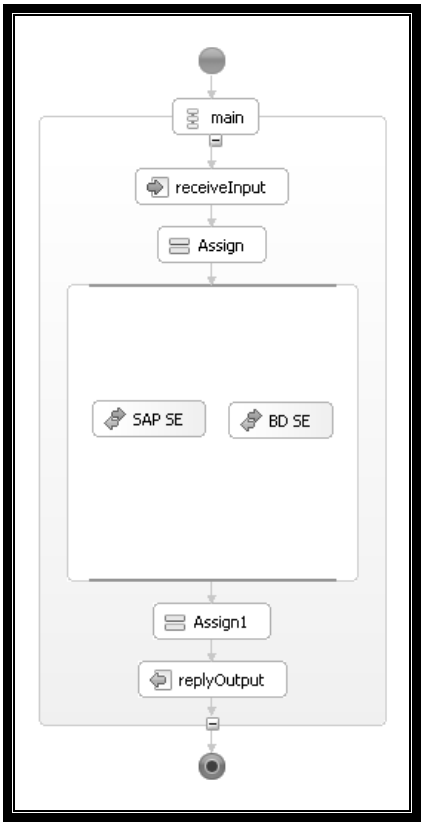

Figura 48. Processo BPEL Final

```
...
<bpel:import namespace="http://Sap" location="SapArtifacts.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
<bpel:import namespace="http://Bd" location="BdArtifacts.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
...
<bpel:flow name="Flow">
<bpel:invoke name="SAP BC" partnerLink="sap" operation="process"
inputVariable="sapRequest" outputVariable="sapResponse"
portType="ns3:Sap">
</bpel:invoke>
<bpel:invoke name="BD BC" partnerLink="bd" operation="process"
inputVariable="bdRequest" outputVariable="bdResponse"
portType="ns1:Bd">
</bpel:invoke>
</bpel:flow>
...
```

Figura 49. Excerto do Código BPEL Final

O processo em BPEL, de modo a ser interpretado pelo ODE (*Service Engine* da Apache de execução de processos BPEL) [ODE], necessita de um ficheiro XML que descreve as ligações aos *PartnerLinks* do processo para que o processo possa ser executado depois de realizado o *deploy*. Ao contrário do que acontece nos restantes SU's, em que o ficheiro a configurar é o *xbean.xml*, no ODE a descrição do seu funcionamento é feita num ficheiro *deploy.xml*. Após o ficheiro estar devidamente definido, o SU está criado e pronto a ser implementado na arquitectura. Quando todos os SU's estiverem criados, estão então prontos para serem agrupados num SA, como está representado mais adiante.

No que toca a SU's, a tarefa está terminada, pois já não existem mais OBO's para definir. Já foi descrito anteriormente que o *ServiceMix* é incapaz de interpretar os SU's, a não ser que estes estejam contidos num SA. Para definir um SA, é necessário definir o(s) SU(s) contido(s). Primeiramente, os SU's devem ser compilados, sendo que cada SU dá origem a um ficheiro compilado. A definição dos SU's que o SA vai conter (Figura 50) é feita num ficheiro *pom.xml – Project Object Model*. O ficheiro POM, gerado a partir do modelo *Maven 2*, contem as definições gerais do projecto, como nome e localização, bem como a identificação dos componentes que serão utilizados no ESB. Na parte relativa às dependências do projecto, com base no POM, o *Maven 2* fica encarregue de descarregar para a máquina do programador todas as bibliotecas que estão relacionadas entre si, e que se encontram num repositório central.

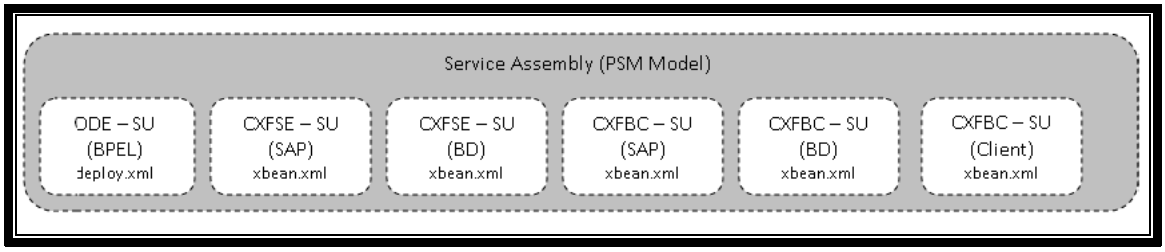

Figura 50. SA dos SU's criados

Após o *deploy* no *ServiceMix* os SU's contidos no SA irão dar lugar aos componentes JBI (Figura 51). Dadas as definições contidas nos respectivos ficheiros *xbean.xml*, cada SU resulta num SE ou num BC, consoante o tipo definido. O *ServiceMix* é capaz de identificar, através do conteúdo dos ficheiros *XBean*, qual o tipo de componente definido no SU. Efectuado o *deploy*, o *ODE SU* torna-se num *ODE SE*, os *CXFSE SU's* tornam-se em *CXF SE* e os *CXFBC SU's* tornam-se em *CXF BC*. O ESB com os devidos componentes constitui assim o PSM (Figura 52).

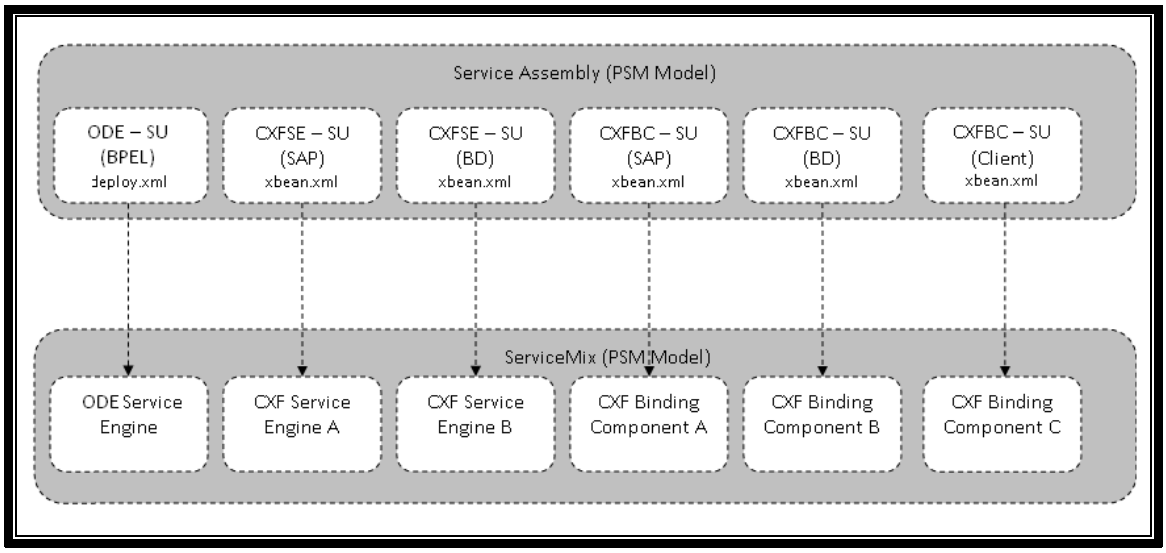

Figura 51. BC e SE depois do *Deploy* do SA (adaptado de [Rademakers e Dirksen 2008])

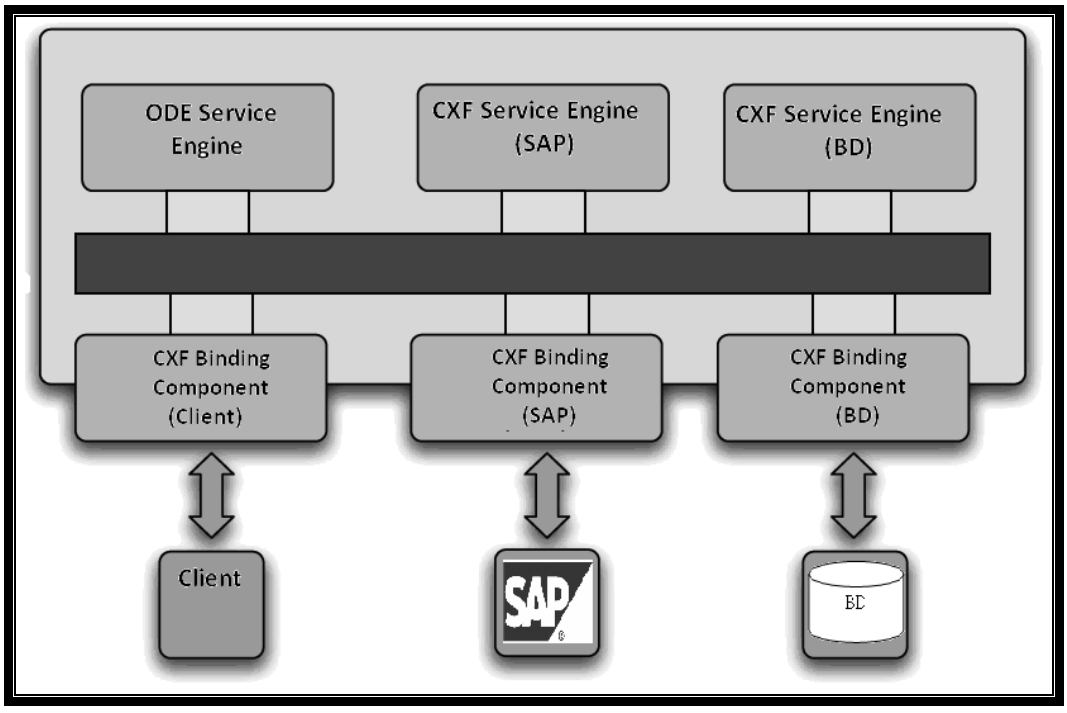

Figura 52. *ServiceMix* do Modelo PSM Final (adaptado de [Genender 2007])

4.4 Conclusões

Neste capítulo o modelo de processos genérico que tinha sido obtido na primeira fase da BIM sofreu as alterações até ter percorrido todas as fases da metodologia. Numa primeira fase, o modelo de processos foi sendo transformado até estar executável. As transformações permitiram obter um processo de negócio que reflecte a visão de negócio do cliente e garante

o uso de boas-práticas organizacionais. No final da terceira fase da BIM, o processo de negócio já está devidamente concretizado na organização. O modelo de processos, depois de modelado de base durante a primeira fase, foi sofrendo algumas transformações durante as segunda e terceira fases. No final de cada fase, o modelo de processos possui um determinado estado, de modo a assegurar que a fase está concluída. Os estados dos modelos permitem que os processos sejam transformados de forma segura e eficaz. O processo exemplificado no capítulo foi sofrendo transformações simples e rápidas de efectuar. Assim que finalizada a terceira fase da metodologia, o processo estava no seu estado final PIM antes de ser transformado em PSM.

Para passar à última fase da BIM, a de implementação em software, propôs-se utilizar o *Model-Driven Architecture* para transformar os modelos de processos em modelos de software. O MDA propõe que a transformação para um modelo PSM seja efectuada realizando tarefas de descrição da plataforma, mapeamento, marcação e transformação. As três primeiras tarefas são a base para a tarefa final de transformação do modelo. Realizadas as três tarefas, já se possui todos os dados requeridos e a implementação em software é simplificada.

A implementação dos modelos de processos num *Enterprise Service Bus* surge como uma proposta que permite integração do processo e das aplicações de uma forma standard. O *ServiceMix* foi o ESB escolhido pois utiliza ambiente JBI e assim a integração é efectuada com o uso de *open standards*. Os dados da integração referentes ao processo já advêm dos dados da transformação, demonstrando assim a utilidade do MDA. Na definição dos *Service Engines* e *Binding Components*, bem como dos *Service Units* e do *Service Assembly* que os vai conter, necessários na execução dos processos definidos, os dados são obtidos eficazmente a partir da transformação, sendo os dados referentes ao restante código parte dos requisitos da definição dos próprios componentes no ESB.

Foi apresentado um exemplo de implementação dos processos num sistema ESB, neste caso no *Apache ServiceMix*. Este foi um *open ESB* escolhido por razões mencionadas no início da secção 4.3, mas em [Rademakers e Dirksen 2008] são disponibilizados outros exemplos de *open ESBs* e utilizar ESBs corporativos também pode ser a opção seleccionada. Também é descrito como configurar o *ServiceMix*, mas existem *online* tutoriais gratuitos sobre configurar este ESB.

5. Conclusão

Este último capítulo apresenta resumidamente todo o trabalho realizado e os resultados alcançados. De uma forma sintetizada, esclarece-se se os objectivos inicialmente propostos foram efectivamente atingidos. Para finalizar, são apresentadas algumas propostas para investigação futura relacionada com esta temática.

A *Business Implementation Methodology* (BIM), descrita resumidamente neste documento, surge com o objectivo de auxiliar no desenvolvimento de sistemas de informação que assegurem que os requisitos dos processos de negócio dos clientes sejam satisfeitos. A característica principal da BIM é a de, na tentativa de assegurar que os processos descritos acrescentem valor ao cliente, utilizar modelos de referência de processos que forneçam as melhores práticas organizacionais definidas entre as organizações de referência do sector, e finalizar a implementação num sistema de software que responda afirmativamente aos requisitos dos processos do cliente.

Um cliente pretende obviamente que a sua organização seja única. Logo, os seus processos de negócio, especialmente os processos críticos, devem ser únicos, o que dificulta o desenvolvimento de software no que toca ao tempo e recursos dispendidos nestes projectos, dado que têm dificuldade em criar bases de conhecimentos que podem ser reaproveitadas em projectos futuros. Nestes casos, a BIM fornece vantagens no sentido em que a implementação é baseada em transformações faseadas e sugere a utilização de modelos de referência e de outros aspectos organizacionais como base de todo o trabalho de implementação, o que permite o reaproveitamento de conhecimentos adquiridos anteriormente.

Primeiramente foi realizado um estudo aprofundado de cinco das mais significativas linguagens de modelação de processos de negócio existentes no mercado. Através de três critérios, foi demonstrada qual a linguagem de descrição de processos de negócio mais adequada ao contexto da *Bosch Car Multimedia Portugal, Lda*. O WS-BPEL (*Web Services Business Process Execution Language*) demonstrou ser mais adequada na globalidade dos aspectos considerados: tecnológicos, específicos da organização e específicos do negócio. A análise efectuada provou o BPEL como mais vantajosa no suporte aos padrões de *workflow* adoptados para comparação das linguagens de processos; nas respostas dos colaboradores a questões sobre características presentes nas linguagens em comparação, como conhecimentos acerca de *Web Services*, linguagem XML, SOA, entre outros, e também conhecimentos de BPM de modo a analisar a sua maturidade em compreender processos de negócio e como estes

devem ser modelados; e por fim em aspectos relevantes no âmbito do negócio, como a maturidade, usabilidade, esforço de aprendizagem, custos das licenças, entre outros.

A demonstração da utilização dos conceitos de MDA para alcançar a ultima fase da BIM foi conseguida. O modelo PIM foi sendo descrito durante a execução das três primeiras fases da BIM – Selecção, Definição e Concretização –, pelo que apenas no fim da terceira fase o modelo PIM estava pronto a ser transformado. Para obter o modelo de processos implementado em software, equivalente à última fase da BIM – Implementação –, foi utilizada a técnica do MDA para obter o modelo PSM. O PSM foi então obtido automatizadamente utilizando as técnicas de descrição da plataforma, mapeamento, marcação e transformação no novo modelo. O modelo PSM obtido responde aos requisitos pedidos pela metodologia para que a ultima fase fosse dada como finalizada.

Segundo a BIM, a selecção e definição do modelo de processos de negócio é sempre feita pelo cliente, o que não assegura que os melhores aspectos do ponto de vista da organização sejam os escolhidos. A BIM propõe apenas alguns dos possíveis modelos, bem como alguns aspectos que podem estar contidos no modelo, que podem ser implementados para serem concretizados no sistema de informação. No entanto, o modelo de processos definido antes de ser concretizado no SI está sempre dependente da opinião do cliente, que pode ter ideias fixas para o negócio e simplesmente ignorar boas-práticas sugeridas pela BIM.

Outro ponto relevante da metodologia é a utilização de modelos de referência de processos. No caso de demonstração deste documento é apresentado e utilizado como modelo genérico o modelo UBK-RM. Podiam no entanto ser utilizados como modelo genérico qualquer outro modelo de referência como o SCOR, DCOR, ITIL ou eTOM.

Juntamente com a selecção de aspectos organizacionais que forneçam qualidade aos processos, como por exemplo o EFQM, é obtido um modelo de processos de negócio que proporcione à organização a execução de boas-práticas que irão trazer valor acrescentado para o seu negócio.

A utilização destes modelos genéricos como base para posteriores transformações garante que o modelo final, implementado em software ou não, além de utilizar modelos de referência, serve como um *template*, em que não é necessário modelar processos a partir do zero cada vez que se inicia um projecto. Neste documento só foi utilizado um modelo de referência, mas pode-se perfeitamente construir um repositório de modelos de referência, bem como uma série de outros aspectos organizacionais, de modo a proporcionar poupança de trabalho no início de cada projecto.

Nas segunda e terceira fases da BIM mostra-se uma forma de realizar transformações pontuais ao modelo de processos genérico, de forma automatizada, para que estes respondam aos requisitos da estratégia de sistemas de informação do cliente. Pretende-se que os processos de negócio, além de proporcionar boas-práticas organizacionais, reflictam a visão e a estratégia do cliente para o negócio.

Dado que ainda não existirem componentes próprios para tratar dados para ERP's (SAP ou outro qualquer) nem para Bases de Dados, foi proposto que as ligações para esses sistemas fossem realizadas através de *Web Services*. Assim, o serviço seria executado num *CXF Service Engine* e a ligação do serviço ao sistema externo seria efectuado utilizando um *CXF Binding Component*. Posteriormente à definição de um SE e um BC para tratar os dados para o ERP, e outra definição semelhante para a Base de Dados, propôs-se que os restantes componentes seriam um BPEL SE para orquestrar o processo e um *CXF BC* que efectue a ligação do cliente com o ESB. Dada esta arquitectura do ESB e os seis componentes definidos, foi proposto um mapeamento entre um processo definido apenas num nível abstracto – um PIM – e um exemplo de um processo concebido à base de *Web Services* – o PSM. Tal como representado na tabela 8, propôs-se um mapeamento que descreve a relação entre um BPEL e os SE's e BC's propostos para executar o processo no ESB.

A criação dos *Service Units* e do *Service Assembly* não representa o aspecto relevante no que toca à transformação para o modelo final PSM, tal apenas se aplica aos componentes *Service Engines* e *Binding Component*. Como foi demonstrado, após a análise à plataforma, mapeamento e marcação, a tarefa de transformação é praticamente directa. As três tarefas antecedentes são realmente as mais trabalhosas, pois representam o "grosso" da actividade global de transformação dos modelos. No entanto, a forma como as tarefas são automatizadas permite que a probabilidade de ocorrência de erros na transformação seja reduzida. A necessidade de criar *Service Units* e *Service Assembly* é representada no documento como a implementação do modelo PSM num sistema de software ESB, que é o passo seguinte para obter o modelo completo implementado em software. Não é uma descrição pormenorizada da implementação do ESB para não cair no erro de realizar uma descrição muito extensa, complexa e de difícil interpretação. Apenas a maneira como os dados são tratados, desde os que são provenientes do modelo PIM, que passam pela transformação, até estarem em *Service Engines*, *Binding Components*, *Service Units* e *Service Assemblies*, é representada no documento.

5.1 Trabalho Futuro

A linguagem escolhida para modelar os processos neste trabalho foi o BPEL. Foi representado no BPEL que o processo foi sendo transformado e possuía um determinado estado consoante a fase da BIM. Uma limitação encontrada é que o BPEL permite apenas modelar processos totalmente automáticos, *i.e.*, processos de negócio em que é ao nível computacional que se desenrola toda a actividade. Conhecendo a realidade organizacional, verifica-se que muitos dos processos, e até alguns dos mais importantes, não são totalmente automáticos, no sentido em que necessitam de intervenção humana de modo a seguirem o seu fluxo normal. Uma norma recente, e com muito pouco suporte de ferramentas, é o *BPEL4People* (B4P) [Oasis 2007]. O B4P permite representar actividades humanas num processo BPEL juntamente com as actividades já presentes no WS-BPEL. Seria interessante verificar se o B4P também suporta transformações em diversos estados da BIM, bem como a implementação dos processos num ESB.

Durante a escolha das linguagens, foi verificado que existem conceitos para avaliar competências de programadores no que toca a desenvolver software, tais como o PSP, TSP e o SWEBoK. Não existem no entanto diferenciações no que toca a conceber processos de negócio. Seria um trabalho interessante, dado que nem todo o conhecimento em BPM é igual, pois muitos conceitos de diferentes áreas são abrangidos, como tecnologias, gestão e sistemas de informação, e assim alguns níveis de maturidade podem ser definidos.

É comum num ambiente organizacional que os processos de negócio sejam automáticos ao máximo, com o objectivo de melhorar níveis de eficácia dos seus processos. Foram apresentados neste documento exemplos de processos de negócio cuja execução é baseada na utilização de *Web Services*. A introdução de semântica nos *Web Services* (*Semantic Web Services*) [McIlraith, Son *et al.* 2001] permitiria aumentar a execução automática dos processos ao nível computacional, pois estes seriam interpretáveis uns para os outros. Consiste em agregar aos *Web Services* meta-dados sobre as propriedades, capacidades, interfaces, pré-requisitos e consequências de uso desses serviços, codificados de forma que possam ser interpretados pela máquina. É com este objectivo que surge a *DARPA Agent Markup Language for Services* (DAML-S) [Martin 2002]. A DAML-S é baseada em marcações (*markups*) que permitirão que uma aplicação Web saiba qual o *input* necessário para que posteriormente outra aplicação Web (potencialmente um *Web Service*) possa fornecer o serviço, que informação será retornada em seguida e como executar e interagir automaticamente com essa informação.

A BIM é independente de notações ou aplicações, pelo que utilizar notação BPEL ou BPMN em algumas fases, por exemplo, na fase de Implementação, é suportado pela metodologia. No entanto, utilizar conceitos de DAML-S bem como de *Web Ontology Language for Services* (OWL-S) [W3C 2004] para descrever processos de negócio implicaria algumas alterações no comportamento destes, pois DAML-S e OWL-S são diferentes de BPEL e WSDL. Utilizando os conceitos de semântica, a arquitectura de *Web Services* iria sofrer profunda remodelação. Já existem casos de implementação B2B, em grelha, ubíquos e móveis, mas este é um conceito cuja implementação em ambientes organizacionais tem ainda um caminho longo a percorrer. Dadas as diferenças no paradigma do BPEL e do DAML-S, é difícil depreender se a utilização do MDA para obter um modelo implementado em software pode ser realizada da mesma forma como foi apresentada neste trabalho. Daí que utilizar estes conceitos de *Web Services* semânticos poderá, ou não, originar trabalhos bem diferentes do trabalho aqui apresentado.

Bibliografia

Aalst, W. M. P. v. d. (2003). "Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language." QUT Technical report FIT-TR-2003-06.

Aalst, W. M. P. v. d., L. Aldred, *et al.* (2004). Design and implementation of the YAWL system. 16 th International Conference on Advanced Information Systems Engineering (CAiSE 04), Riga, Latvia, Spring Verlag.

Aalst, W. M. P. v. d., M. Dumas, *et al.* (2006). From BPMN Process Models to BPEL Web Services. IEEE International Conference on Web Services (ICWS'06). I.-C. Society.

Aalst, W. M. P. v. d., M. Dumas, *et al.* (2002). "Pattern Based Analysis of BPML (and WSCI)." FIT Technical Report FIT-TR-2002-05.

Aalst, W. M. P. v. d. e A. H. M. t. Hofstede (2002). "YAWL: Yet Another Workflow Language." QUT Technical report

Aalst, W. M. P. v. d., A. H. M. t. Hofstede, *et al.* (2003). "Workflow Patterns " Distributed and Parallel Databases **14**.

Abran, A., P. Bourque, *et al.* (2004). Guide to the Software Engineering Body of Knowledge - SWEBOK, IEEE Press.

Andrew, K.R. (1987). “The Concept of Corporate Strategy”. Irwin, Homewood, IL.

Apache. "Apache Software Foundation." from http://www.apache.org/.

Arkin, A. (2002). "Business Process Modeling Language." BPMI.org.

Armstrong, E., J. Ball, *et al.* (2004). the J2EE 1.4 Tutorial, Addison Wesley.

Atkinson, C. e T. Kühne (2003). "Model-Driven Development: A Metamodeling Foundation." Software, IEEE.

Bauer, B., J. P. Muller, *et al.* (2004). A Model-Driven Approach to Designing Cross-Enterprise Business Processes.

Berndtsson, M., J. Hansson, *et al.* (2007). Thesis Projects: A Guide for Students in Computer Science and Information Systems, Springer-Verlag New York, Inc.

Bézivin, J. (2004). "In Search of a Basic Principle for Model Driven Engineering." Upgrade **Vol. 5 No.2**.

Bézivin, J., G. Dupé, *et al.* (2003). First experiments with the ATL model transformation language: Transforming XSLT into XQuery. 2nd OOPSLA Workshop on Generative Techniques in the context of MDA, Anaheim, CA, USA.

Bézivin, J., S. Hammoudi, *et al.* (2004). Applying MDA Approach to B2B Applications: A Road Map. Workshop on Model Driven Development (WMDD 2004) at ECOOP 2004, SpringerVerlag.

BPEL2Java. "BPEL to Java (B2J) Subproject." from http://www.eclipse.org/stp/b2j/.

BPMI (2001). Release of BPML Specification Paves Way for Implementations. BMPI.org.

BPMI.org "Business Process Management Initiative." http://www.bpmi.org.

Bragança, A. e R. J. Machado (2008). Transformation Patterns for Multi-staged Model Driven Software Development. 12th International Software Product Line Conference.

CMMI (2006). CMMI for Development, Version 1.2, Software Engineering Institute (SEI).

Coelho, J. S. (2005). Arquitectura da Empresa Centrada nos Processos: O Factor Determinante para o Alinhamento Estratégico dos SI. Sistemas de Informação Organizacionais. L. Amaral, R. Magalhães, C. C. Morais, A. Serrano e C. Zorrinho, Edições Sílabo.

Couloris, G., J. Dollimore, *et al.* (2001). Distributed Systems: Concepts and Design, Addison Wesley.

CPNGroup. "CPN Tools: Computer Tool for Coloured Petri Nets." from http://wiki.daimi.au.dk/cpntools/cpntools.wiki.

Curbera, F., Y. Goland, *et al.* (2005). BPEL4WS Version 1.0 specification, IBM Corporation.

Deursen, A. v., P. Klint, *et al.* (2000). "Domain-specific languages: an annotated bibliography." ACM SIGPLAN Notices **35**: 26-36.

Duarte, F. J., J. M. Fernandes, *et al.* (2006). Business Modeling in Process-Oriented Organizations for RUP-based Software Development. **Reference Modeling for Business Systems Analysis**

Duarte, F. J., R. J. Machado, *et al.* (2007). Automated Information Systems Generation for Process-Oriented Organizations. QUATIC '07: Proceedings of the 6th International Conference on Quality of Information and Communications Technology, IEEE Computer Society.

Duarte, F. J., R. J. Machado, *et al.* (2009). A Methodology to Transform Business Processes into Software Systems, Submitted to Publication to International Journal of Business Process Integration and Management, Special issue on Dynamic and Declarative Business Processes.

EFQM (2001). EFQM Excellence Model - Large Companies, Operational and Business Units version.

Fernandes, J. M., R. J. Machado, *et al.* (2006). A Demonstration Case on the Transformation of Software Architectures for Service Specification. Proceedings of the 5th IFIP Working Conference on Distributed and Parallel Embedded Systems - DIPES 2006, Braga, Portugal.

Fernandez, A. (2005) "L'essentiel du *Tableau de Bord*". Éditions d'Organisation. Paris.

Gasevic, D., D. Djuric, *et al.* (2006). Model Driven Architecture and Ontology Development.

Genender, J. (2007). Building a Service Oriented Architecture with ServiceMix Greater Quebec Software Symposium.

Guelfi, N. e A. Mammar (2006). A formal framework to generate XPDL specifications from UML activity diagrams. SAC '06: Proceedings of the 2006 ACM symposium on Applied computing, ACM.

Hammer, M. (1990) "Reengineering Work: Don't Automate, Obliterate" Harvard Business Review.

Hapner, M., R. Burridge, *et al.* (2002). *Java Message Service*, *Sun Microsystems, Inc.*

Harmon, P. (2004). "XML BP Languages: An Update." Business Process Trends **Volume 2, Number 4**.

Helkiö, P., A. Seppälä, *et al.* (2006). Evaluation of Intalio BPM Tool T-86.5161 Special Course in Information System Integration.

Hohpe, G. e B. Woolf (2004). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley.

Humphrey, W. S. (1994). “The Personal Process in Software Engineering”. Third International Conference on the Software Process.

Humphrey, W. S. (1999). Introduction to the Team Software Process, Software Engineering Institute (SEI).

ifSMF (2007). An Introductory Overview of ITIL® V3.

Jensen, K. (1992). Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Springer.

Jeston, J. e J. Nelis (2006). “Business Process Management: Practical Guidelines to Successful Implementations”. Elvesier Ltd.

Johnson, R., J. Hoeller, *et al.* (2005). Professional Java Development with the Spring Framework, Wrox Press Ltd.

Junior, P. K. (2005). "Shark: um *engine* de *workflow* estensível baseado na especificação WfMC."

Júnior, P. S. S., J. P. A. Almeida, *et al.* (2008). Construção de um Metamodelo para o ARIS Method: Escavação, Refatoração e Análise. Third Workshop on Ontologies and Metamodeling in Software and Data Engineering - WOMSDE 2008.

Kaplan, R. e D. Norton (1996). “Linking the Balanced Scorecard to strategy”.

Karjalainen, S. (2004). “Business Process Modelling Language Report”. Computer Science. Helsinki, Faculty of Science - Department of Computer Science - University of Helsinki.

Kloppmann, M., D. König, *et al.* (2004). "Business process choreography in WebSphere." IBM Systems Journal **2**.

Koehler, J., R. Hauser, *et al.* (2005). "Declarative techniques for model-driven business process integration." IBM Systems Journal **44**: 47-65.

Lee, J., K. Siau, *et al.* (2003). "Enterprise integration with ERP and EAI." Commun. ACM **46**: 54-60.

Lezoche, M., M. Missikoff, *et al.* (2008). Business Process Evolution: a Rule-based Approach. 9th Workshop on Business Process Modeling, Development, and Support (BPMDS'08).

Machado, R. J., J. M. Fernandes, *et al.* (2005). Transformation of UML Models for Service-Oriented Software Architectures. Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS’05).

Machado, R. J., J. M. Fernandes, *et al.* (2006). Refinement of Software Architectures by Recursive Model Transformations. PROFES 2006 : proceedings of the International Conference on Product-Focused Software Process Improvement.

Maldaner, L. A. e E. S. Pasqual (2006). "Uma Análise de Linguagens de Composição de Serviços: A Utilização de BPEL e YAWL."

Martin, D. (2002). DAML-S: Bringing Services to the Semantic Web, *DAML-S Coalition*.

Martins, V. M. M. (2005). Integração de sistemas de informação : perspectivas, normas e abordagens. Escola de Engenharia. Guimaraes, Universidade do Minho.

Maven. "Apache Maven." from http://maven.apache.org.

McIlraith, S. A., T. C. Son, *et al.* (2001). "Semantic Web Services." **Ieee Intelligent Systems 16**: 46–53.

Mendling, J., M. Moser, *et al.* (2006). Transformation of yEPC business process models to YAWL. Proceedings of the 2006 ACM symposium on Applied computing, Dijon, France.

Michalewicz Z., M. Schmidt, *et al.* (2006). "Adaptive Business Intelligence". Springer.

Milner, R., J. Parrow, *et al.* (1989). "A Calculus of Mobile Processes, Part I." I and II. Information and Computation **100**.

Mitre (2004). "Guide to the (Evolving) Enterprise Architecture Body of Knowledge (EABoK)". MITRE Corporation.

Mulyar, N. A. (2005). Pattern-based Evaluation of Oracle-BPEL (v.10.1.2).

Nüttgens, M., T. Feld, *et al.* (1998). Business Process Modeling with EPC and UML: Transformation or Integration? The Unified Modeling Language - Technical Aspects and Applications. Heidelberg.

Oasis (2007). Web Services Business Process Execution Language 2.0 (WS-BPEL).

Oasis (2007). WS-BPEL Extension for People (BPEL4People) Version 1.0.

ODE. "Apache ODE." from http://ode.apache.org.

Oliveira, S. A. R. d. (2006). Colored Petri Nets in the Animation of UML Models for Requirements Validation. Departamento de Sistemas de Informação. Guimarães, Escola de Engenharia - Universidade do Minho. **Dissertação de Mestrado**.

OMG. "The Object Management Group." from http://www.omg.org/.

OMG (2003). MDA Guide Version 1.0.1, Object Management Group.

OMG (2005). MOF QVT Final Adopted Specification, Object Management Group.

OMG (2006). "Meta Object Facility (MOF) Core Specification Version 2.0." OMG.

OMG (2006). Object Constraint Language (OCL) OMG Available Specification Version 2.0, Object Management Group.

OMG (2009). OMG Unified Modeling Language (OMG UML),Infrastructure version 2.2, Object Management Group.

OMG (2009). OMG Unified Modeling Language (OMG UML),Superstructure version 2.2, Object Management Group.

Pastor, O. e J. C. Molina (2007). Model-Driven Architecture in Practice - A Software Production Environment Based on Conceptual Modeling.

Pearlson, K. E. e C. S. Saunders (2004). Managing and Using Information Systems: A Strategic Approach, John Wiley & Sons, Inc.

Petri, C. A. (1962). Kommunikation mit Automaten. Bonn, Institute fur Instrumentelle Mathematik. **PhD Thesis**.

Pomeroy-Huff, M., J. L. Mullaney, *et al.* (2005). The Personal Software Process (PSP) Body of Knowledge, Version 1.0. S. R. CMU/SEI-2005-SR-003, Software Engineering Institute (SEI).

Porter, M. (1980). “Competitive Strategy: Techniques for Analyzing Industries and Competitors”. Free Press.

Rademakers, T. e J. Dirksen (2008). Open-Source ESBs in Action, Manning Publications.

Reiter, T., E. Kapsammer, *et al.* (2006). "A Generator Framework for DomainSpecific Model Transformation Languages." ICEIS.

Rodriguez, A., E. Fernandez-Medina, *et al.* (2007). "Towards CIM to PIM Transformation: From Secure Business Processes Defined in BPMN to Use-Cases." BPM.

Rungworawut, W. e T. Senivongse (2006). Using Ontology Search in the Design of Class Diagram from Business Process Model. Transactions on Engineering, Computing and Technology.

Sandoe, K., G. Corbitt, *et al.* (2001). Enterprise Integration, New York.

Santos, M. Y. e Ramos, I. (2009). “Business Intelligence – Tecnologias de Informação na Gestão de Conhecimento”. FCA – Editora de Informática.

SCC (2006). Dcor overview - design-chain operations reference-model 1.0, Supply-Chain Council

SCC (2008). Scor overview - supply-chain operations reference-model 9.0, Supply-Chain Council

Schuessler, T. (2001). Applications With the Sap Java Connector (JCO), AraSoft.

ServiceMix. "Apache ServiceMix." from http://servicemix.apache.org.

SEI. “Software Engineering Institute | Carnegie Mellon”. http://www.sei.cmu.edu/

Silver, B. (2009). "Model Portability in BPMN 2.0 " BPMS Watch: Bruce Silver’s Blog on Business Process Management Retrieved 23/03/09, from http://www.brsilver.com/wordpress/2009/03/04/model-portability-in-bpmn-20-2/.

Smith, H. e P. Fingar (2002). Business Process Management - The Third Wave, Meghan-Kiffer Press.

Sun. "Java Database Connectivity (JDBC)." from http://java.sun.com/javase/technologies/database.

Takecian, P. L. (2008). ACP e LOTOS: um estudo comparativo baseado em conceitos de BPEL e padrões de controle de fluxo. Instituto de Matemática e Estatística. São Paulo, Universidade de São Paulo. **Dissertação de Mestrado**.

Ten-Hove, R. e P. Walker (2005). Java Business Integration (JBI) 1.0, Sun Microsystems, Inc.

TMForum. "Business Process Framework - *eTOM*." from http://www.tmforum.org/BusinessProcessFramework/1647/home.html.

Vanhatalo, J. (2006). BPEL Repository User Guide, IBM Corporation.

W3C. "XML Schema." from http://www.w3.org/XML/Schema.

W3C. (1999). "XML Path Language (XPath) Version 1.0." from http://www.w3.org/TR/xpath.

W3C. (2004). "OWL-S: Semantic Markup for Web Services." from http://www.w3.org/Submission/OWL-S.

W3C. (2007). "SOAP Version 1.2." from http://www.w3.org/TR/soap12/.

W3C. (2007). "Web Services Descprition Language (WSDL) version 2.0." from http://www.w3.org/TR/wsdl20.

WfMC (2008). Workflow Management Coalition: Final XPDL 2.1 Specification. WFMC-TC-1025-Oct-10-08-A.

WfMC.org "Workflow Management Coalition." http://www.wfmc.org.

White, S. A. (2004). "Business Process Modeling Notation (BPMN) Version 1.0."

YAWL-System. "YAWL Yet Another Workflow Language." http://www.yawl-system.com/

Zhao, W., R. Hauser, *et al.* (2006). "Compiling business processes: untangling unstructured loops in irreducible flow graphs." Int. J. Web Grid Serv. **2**: 68-91.

Anexo A – *Draft* de um *Artigo* Científico Relacionado com a Dissertação

Transcrição da versão “draft” do artigo a ser futuramente submetido a conferência.

Using MDA Transformations in Order to Obtain Information Systems

Nuno Santos [1]
pg13315@alunos.uminho.pt

Ricardo Machado [1]
rmac@dsi.uminho.pt

Francisco Duarte [1,2]
francisco.duarte@pt.bosch.com
francisco.duarte@dsi.uminho.pt

[1] *Dept. de Sistemas de Informação, Universidade do Minho, Guimarães*
[2] *Bosch Car Multimedia Portugal, Lda., Braga*

Abstract

In the development of an Information System, the definition of business processes with maximum level of automation allows process-oriented organizations to be able to fulfil clients' requirements in less time. In order to ensure quality in the business processes definition, a methodology with well-defined tasks should be followed. A proposed methodology, the Business Implementation Methodology (BIM), defines automated transformations of business processes in a way that software implementation in organization's information system are less effort consuming. Like BIM, this paper proposes the use of OMG's Model-Driven Architecture (MDA) for obtaining the Information System. The transformation of a process framework that is a Platform Independent Model (PIM) to a Platform Specific Model (PSM) will allow obtaining software implementation. This paper proposes an automated transformation of a business process model in a software executing model, by suggesting mapping between a BPEL process and a set of JBI components who will execute in an Enterprise Service Bus (ESB).

Keywords*: BPM; Information System; Business Implementation Methodology; MDA Transformation; PIM; PSM; ESB.*

1. Introduction

The concept of Business Process Management (BPM) [1] is already largely implemented by companies. Since organizations already possess information systems BPM-oriented, clients requirements are based in business processes, therefore the models to use in software development that fulfil those requirements must be business process oriented [2] [3].

In organizational environments the time spent in the business process definitions in order to fulfilling clients' requirements must be smaller and smaller, as well as the associated costs. Integrating enterprise applications and not just those in short time and costs isn't an easy task, due to the heterogeneity of the applications.

The methodology used in this article, the Business Implementation Methodology (BIM) [4], has as its core the usage of process reference models to provide best practices in the business section and the implementation of software systems that ensures fulfilment of clients business process requirements. The core motivation of this article arises from the last one. To implement the business process model in software a model-driven approach may be gainful in terms of time spent and complexity. Usage of the Model-Driven Architecture (MDA) [5] suggests the design of two models. The Platform Independent Model (PIM) represents

the business process model, and the Platform Specific Model (PSM) the execution in software of the previous model [6]. It only lefts to know if, after the business model is defined correctly, software obtaining from MDA concepts will allow highly automated tasks.

For that manner, it is experimented the business process transformations using PIM and PSM concepts from MDA. This is an important step in business implementation because it results in an effective development of software, besides the efficiency provided from BIM. To have the business process executing in software, it will be implemented in an Enterprise Service Bus (ESB), the Apache ServiceMix [8]. Maximizing the automated level of the business process implementation ensures a safe and efficient implementation, providing a minimum amount of errors during all phases of an implementation project, which in turn will result in a reduction of time and resources for the company.

Chapter 2 introduces related work to this article. Chapter 3 contains the core concepts which this work was based on, namely BIM, MDA and ESB. In chapter 4 it is established a relation between the 4 states that business processes are defined in BIM and PIM and PSM states defined in MDA. In chapter 5 MDA transformations are applied to the PIM obtained after the third phase of BIM. In chapter 6 the PSM is presented, correctly implemented in an ESB software system. Finally, in chapter 7, conclusions of the realized work are obtained, as well as a proposal for future work.

2. Related Work

[9] transforms PIM business processes in UML2 Activity Diagrams [20] [21] into a PSM in BPEL [31] through REL (Regular Expression Language) method, as in [10] which makes the same transformation using ATL (ATLAS Transformation Language) [11], or else [12] which transforms the model using OCL (Object Constraint Language) [13] rules. Another kind of approach is the one made by [14] e [15], which starts to design a CIM business process model in EPC [16], transforms the CIM in a PIM business process model in BPMN [17] and finally obtains the PSM business process model in BPEL [31]. Another different approach is presented in [18] and [19], which makes transformation of a CIM in BPMN in a PIM in UML, respectively Use Case and Class Diagrams.

One methodology for business implementation is mLEARN [22]. mLEARN is known by “organizational competencies methodology”, which provides techniques for providing business process representations, straight aligned with the strategic goals. It also clarifies the responsibilities inside the organization. In order to add software implementation to business processes, techniques like 4-Step Rule Set (4SRS) [23] or the MOF QVT [24] standard from OMG can be used. 4SRS is a technique for transforming user requirements in architectural models that represent system requirements. QTV allows relations between models, based in transformation rules.

The Spring architecture [25] provides a platform for executing business processes. It is constituted by three layers: handling user interface (*e.g.* Web Browsers or EJB), implementing business rules and entities (*e.g.* Application Server), and persistent data (*e.g.* Database Server).

In the scope of enterprise application integration, [26] defines a set of Enterprise Integration Patterns, where requirements to obey

application integration are defined, as well some techniques and solutions.

In order to implement application integration there are some open Enterprise Service Bus (open ESBs). Open ESBs bring gains due the fact they are open source and use open standards. Some examples of most known tools are Mule, Apache ServiceMix, OpenESB, Apache Synapse, JBoss ESB, Apache Tuscany, Fuse ESB, OW2 PEtALS and OpenAdapter [27].

3. Core Concepts

3.1 Business Implementation Methodology (BIM)

BIM [4] is a methodology for implementing business process of an organization in software. This methodology proposes use of best-practices in the business domain and allows customisation of the process in a way that enterprise's needs are attended by the process in question. It is a flexible methodology, which allows various technologies independent from each other, as well as vendor independency.

BIM is composed by four phases: Selection of adequate generic business processes; Definition of the processes to use; Concretisation of the business process in the organization information system which it will execute in; and Implementation in software of the various entities that compose the process (Figure 1).

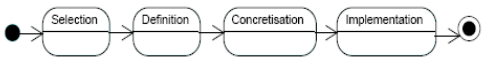

Figure 1. The 4 phases of BIM [4]

The particularity that defines which phase of the methodology the business process is presently is his framework state, in a way it responds to certain requirements of the methodology. Once the necessary requirements to complete each phase are defined, it is attributed a state to the process framework – PF. The states defined in the methodology are Generic, Instantiated, Runnable and Software Implemented. To each state of the process framework corresponds one of the BIM phases.

3.2 PIM and PSM

The starting point of MDA is the separation between system's operations specifications and description of how the system will use platform capabilities. So it will allow that: system's environment and requirements are specified (CIM), the system is specified independently from the platform that supports it (PIM), the platform is specified, a platform is chosen for the system and system's specifications are transformed into specifications of a platform (PSM).

The PIM is a model where its specification doesn't alter from one platform to another. It has a level of platform independency in a way that the model may be executed in various platforms of different types. The model must be developed respecting classes from different platforms where the model may be implemented in the future.

PSM results from the PIM transformation process so the PIM specifications are equal to the specifications of the platform. The PIM transformation process is made by combining PIM details with specific details of the platform (Figure 2).

Normally, model transformation is executed using mappings and transformations [28]. OMG also proposes marking [5], as represented in Figure 3. Marking is made by indicating

PIM elements that will be transformed into PSM.

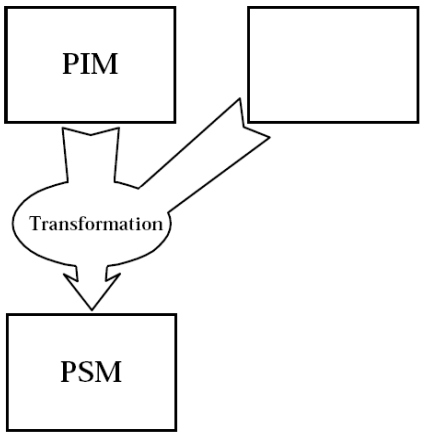

Figure 2. Transformation from PIM to PSM [5]

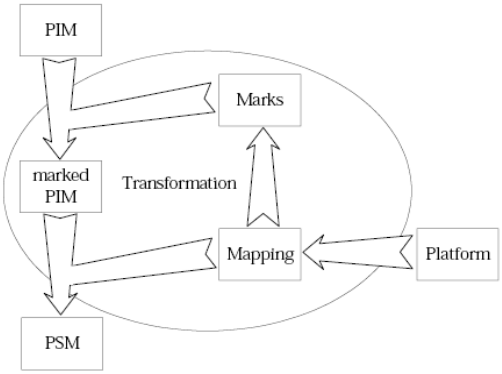

Figure 3. Model Transformation [5]

In mapping, relationship between PIM and PSM elements is made. For example, it is possible to create a mapping that shows relations between model's classes and Java classes. Mappings must be based in necessary requirements to the coding language or to the model. It is possible to exist relations between one PIM element and several PSM elements, as well as a PSM element can be related with several PIM elements. Once mapping is done, transformation will result in code generation, based in PIM, PSM and the realized mapping, because it has all the information it needs to transform the model into code. So this last step is possible, it is required that the PSM element possesses the necessary information that can be translated into code.

3.3 Apache ServiceMix

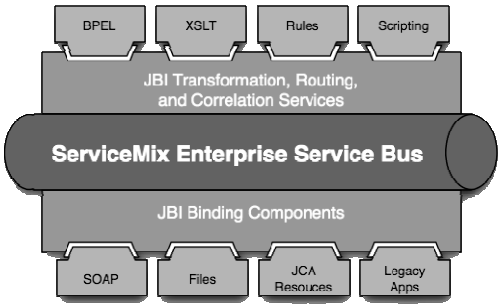

Figure 4. Apache ServiceMix [8]

The implemented ESB is Apache ServiceMix [8]. ServiceMix is an ESB solution accepted and used by respectable entities in the software development market, but it also is open source and open standard based, which brings benefits in its implementation, like low costs but also ensures quality. Its behaviour is based in Java Business Integration (JBI) [29]. JBI defines a framework that allows application integration based in added components that interoperate through a method of normalized messages exchange. This method of message exchange was adopted from WSDL 2.0 specification's Message Exchange Patterns (MEP's) [30]. MEP's define a set of message exchange types, being possible to be In-Only, Robust In-Only, In-Out and In Optional-Out. JBI defines two types of components: Service Engines (SE) and Binding Components (BC), as represented in Figure 4. SE provides business and processing logic, *i.e.*, it is the component that processes a set of data in order to execute a service. BC provides communication with the exterior, working as a "bridge" between in and out data from (or to) the ESB to (or from) the exterior.

In order to deploy a component into ServiceMix, a Service Unit (SU) [27] that will provide component instantiation is used. The SU is a set of XML files whose content is transmitted to the component. Each SE or BC instantiation requires a SU which has the instantiated component definition. In order to a SU can be executed in the ESB, it is necessary the use of Service Assemblies (SA) [27]. SA is a collection of SU's. JBI components are unable to interpret SU's without these being inside a SA. A SA is constituted by at least one SU.

4. Relation Between BIM and MDA

Analysing the characteristics of this methodology, it is possible to establish relationship between BIM and MDA. The characteristics of a business process that are in the Generic, Instantiated and Runnable state are similar to the characteristics of a Platform Independent Model (PIM) from MDA, due to business process representations in these states don't include any platform dependency. During the first three BIM phases (Selection, Definition and Concretisation) the process doesn't require any kind of software, because until the end of the third phase it isn't yet decided if the process will execute in software or not. In fact, BIM suggests that, in the end of the Concretisation phase, a process is defined to be implemented in the next phase of the methodology, but also existing alternatives, and at least one of them that doesn't require software implementation, *i.e.*, his final process framework be obtained in the Runnable state. The Software Implemented state corresponds to the Platform Specific Model (PSM), due to the fact that the process is executed using software, so it needs to obey specifications of the technologic platform.

5. Transformation of the model into PSM

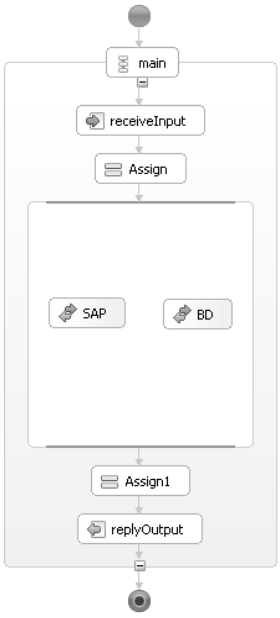

Figure 5. Runnable PF of "Quantity Insertion"

In order to complete the BIM Concretisation phase it is necessary that the PF is correctly concretised in the Information System. Business process was designed in WS-BPEL [31], as represented in Figure 5. This business process is in PIM and already possesses the characteristics described in BIM to be transformed in PSM.

Because the PF is in the Runnable state, it must be considered to the execution of the process not only the BPEL code, but also its WSDL [30] code (Figure 6).

What matters to save from the WSDL code are the data referring to the service, the port, the port type, operations and in and out elements. The process is defined as a service provider as "*lancQuantService*", with a port defined as "*LancQuantPort*". The port type is defined as "*LancQuant*".

The "*process*" operation corresponds in this case to the main process and has as in and out elements "*LancQuantRequest*" and "*LancQuantResponse*", respectively.

Figure 6. WSDL Representation of "Quantity Insertion"

Platform

Following the model proposed by OMG for MDA, in order to make the model transformation phase, it is necessary to describe the platform, following the mapping [28] and the marking of the elements that will be transformed. Only after that it is possible to execute the model transformation.

In the framework four OBO's [4] can be identified: the client-software, which gives the process initiate order to the process through its invocation; the BPEL, with the process orchestration role, which defines the order the services will be invocated; the ERP [32], which executes the material consume in stock; and a DataBase (DB), which executes the new product registry to shelf. Analysing the characteristics of the ServiceMix behaviour it is possible to identify the need of three Binding Components (BC's), in order to have connections to the ERP, the DB and the request and response to the client. A BC adequate to the task is one that interprets Web Services [33], due the characteristics of the SOAP protocol are more appropriated to send requests and responses to a client. In what concerns the connections to the ERP and the DB, this choice is based on the fact that it doesn't exist yet a Service Engine (SE) with those functions, in this case, ERP SAP [16], and also due the fact that it ensures Java Database Connectivity (JDBC) [34]. For these last two, the execution of the Web Service is made through SE's, *i.e.*, one SAP SE and one DB SE. it requires also a BPEL SE (in Apache it is called an Orchestration Director Engine – ODE) [35], to execute the process workflow and orchestrate the services (Figure 7).

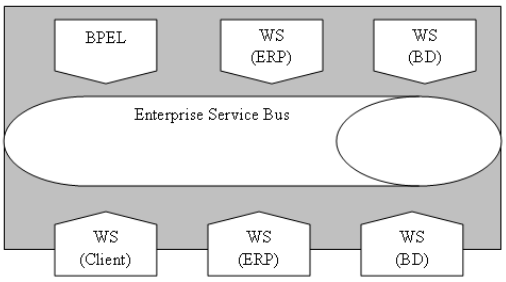

Figura 7. Platform Components Created to Execute the Process

Mapping

The invoke activities pretend that a certain service send them a response to a certain request, therefore the corresponding ESB component with this characteristic is the Service Engine. So, mapping is direct. Invoke activities will invoke the respective OBO, *i.e.*, "*SAP*" invocation is related with the "*SAP SE*" OBO, while "*BD*" invocation is related with "*BD SE*" OBO.

Namespaces, as well as the service data from the WSDL code – *portType, service* e *port* –, correspond to the identification data of the created SU's. The WSDL is, in fact, the file that "circulates" in the Normalized Message Router. For better understanding, the relations are shown and described in Table 1.

Table 1. Relations of the Mapping PIM-PSM

PIM Elements	→	PSM Elements
WSDL – Elements from the *Request* type	→	Entry parameters of the POJO class of the SU (CXF SE's)
WSDL – Service name ("*LancQuantService*", *"SapService*" e "*BdService*")	→	Name of the Web Service in the JAVA file belonging to the SU (CXF SE's)
		targetService of the SU (CXF BC's)
		Name of the service of the respective *partnerLink* in the SU (BPEL SE "ODE")
WSDL – Port type of the service ("*LancQuant*", "*Sap*" e "*Bd*")	→	*targetInterface* of the SU (CXF BC's)
WSDL – Port name of the service ("*LancQuantPort*", "*SapPort*" e "*BdPort*")	→	*targetEndpoint* of the SU (CXF BC's)
		Port name of the service of the respective *partnerLink* in the SU (BPEL SE "ODE")
BPEL – *namespaces* of the imported services	→	*namespaces* in the *xbean.xml* files and *targetNamespace* in the JAVA file of the SU (CXF SE's)
		namespaces of the WSDL files which are *PartnerLinks* in the BPEL file
BPEL – Imported WSDL files	→	*wsdl* of the SU (CXF BC), generated by the CXF SE
BPEL – "*input*" variable and WSDL – *Request* element	→	Entry parameters of the client Web Service
		"*input*" element from the *Request* of the WSDL
		"*input*" variable of the BPEL
BPEL – Invoke activity "*SAP"*	→	BPEL invoke activity "SAP SE"
BPEL – WSDL file which is "Sap" *PartnerLink*	→	Generated WSDL file from the CXF SE
BPEL – Invoke activity "*BD"*	→	BPEL invoke activity "BD SE"
BPEL – WSDL file which is "Bd" *PartnerLink*	→	Generated WSDL file from the CXF SE
BPEL – "*output*" variable and WSDL – *Response* element	→	Return parameters of the client Web Service
		"*output*" element from the *Response* of the WSDL
		"*output*" variable of the BPEL

Marking

Just as in mapping, marking is also a simple task, due to the low complexity of the process framework. The marking of the elements is made from the relationships identified in the mapping. In Figure 8 it is possible to see a representation of the mapping and the marking, where, in the left side of the figure, the PIM elements are marked and, in the right side, it is the resulting PSM element of the transformation.

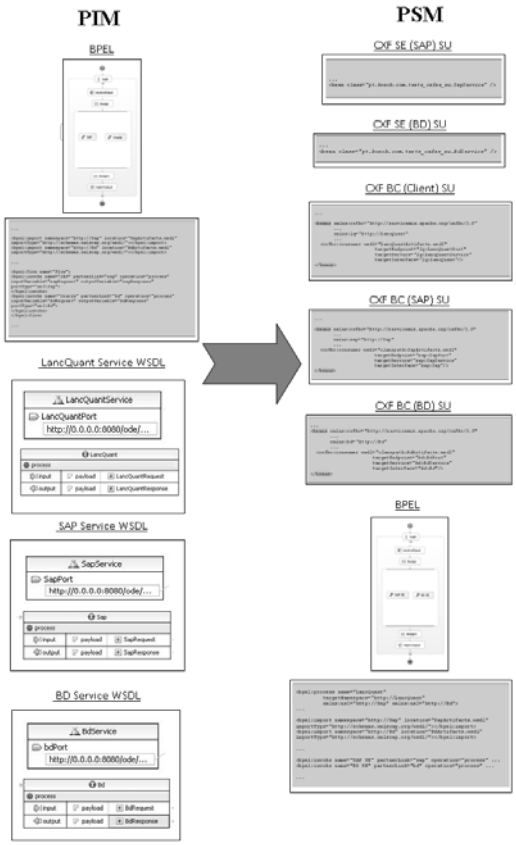

Figure 8. Mapping Between the PIM and PSM Components

6. The PSM

Service Engines

The first step is to define SU's who, after being deployed in ServiceMix, will create SE's. The wanted SE type is a CXF kind. It's important to refer once again that this choice is based on the fact that there doesn't exist yet a SE to ERP SAP functions. The behaviour of the SU is described in the *xbean.xml* file, generated from a *Maven* [36] archetype specific for ServiceMix. The definition of the SU as a SE component is made in the first line of the coding, where the namespace is defined ("xmlns:cxfse=…"), while the rest of the code describes the necessary elements to the body of the SU. This file is quite simple because it

just indicates the location of the POJO class (*Plain Old Java Object*) [25] which is relevant for the Service Engine.

In this case, it is the class "*SapService*" the CXF SE exposed Web Service refers to. The code part that precedes it refers to the *groupId* and *artifactId.* The SU also contains a Java [37] file, in this case *SapService.java*, which is the Web Service executed by the *CXF SE*. This Java file uses SAP Java Connector (JCo) [38] to do the insertion and reception of data between the Web Service and the ERP SAP, obeying correctly the requirements of a connection to an ERP. After building via *Maven 2*, the Java class *"SapService*" will create a new WSDL file, in this case renamed to "*SapArtifacts*" for a question of simplicity in the mapping due to the fact this is the name automatically given by the BPEL Design tool from Eclipse to WSDL files. This WSDL file will be the one who will be invoked in the future by the BPEL final file (described ahead). Inside that file, the chosen namespace will be the same as provided by the mapping.

Just like in the CXF SE with a Web Service that executes the introduction of the values in SAP, the definition of the SU for the CXF for the DB is similar. The usage of a CXF SE is justified with the fact it supports JDBC connections. The main difference of this SU to the previous one is just the definition of the POJO class, which will refer to the "*BdService*" class, the Web Service exposed by the *CXF SE*. This file, "*BdService.java*", will contain the cod for the JDBC connection so it is able communicate with the DataBase. Also in this case the WSDL file is generated after a *Maven 2* build.

Binding Components

It has already been explained how a SU that will originate one SE after being deployed in ServiceMix must be configured. The task to configure a SU that originates a BC is similar. Just like before, the type of component is defined in the first line of the code ("xmlns:cfxbc=…"). In this case, the component is a CXF BC which will communicate data to the CXF SE, sending and receiving values of a Web Service. The element data that defines the BC must be filled with the data from the PIM BPEL and WSDL files as suggested by Table 1. Such will allow a correct identification of the component endpoints required by the ESB, which then can route correctly the data.

The PSM is completed with the BPEL representation. As shown in Figure 9, in terms of visual notation, it doesn't suffer modifications, due to the fact that the mapping ("SAP" to "SAP SE" and "BD" to "BD SE") doesn't require any addition or removal of BPEL activities.

The business process in BPEL, so it can be interpreted by ODE, requires a XML file that describes connections to process *PartnerLinks* so the process can be executable after deploy. Contrarily to what happens with the rest of the SU's, in which the *xbean.xml* file defines its execution, the behaviour of ODE is configured by a *deploy.xml* file. After being correctly defined, the SU is ready to be implemented in the framework. When all SU's are created, they are ready to be grouped in a SA.

As in for SU's, the task is completed. It was already described that, in order to ServiceMix being capable to interpret SU's, they must be contained in a SA. First step, SU's must be compiled, which each SU originates a compiled file. The definition of the

SU's which the SA will contain is made in a *pom.xml* file – Project Object Model.

Figure 9. Final Process in BPEL

The POM file, generated by a *Maven 2* archetype, contains the general information of the project, like name and location, as well as the identification of the components which will be used in the ESB. After being deployed in ServiceMix, the SU's contained in the SA will give place to JBI components (Figure 10). Given the respectively *xbean.xml* file, each SU originates one SE or a BC. ServiceMix is capable of identifying, through the XBean files, what kind of component is defined in the SU.

So, after deploy, ODE SU becomes ODE SE, CXFSE SU becomes CXF SE and CXFBC SU becomes CXF BC.

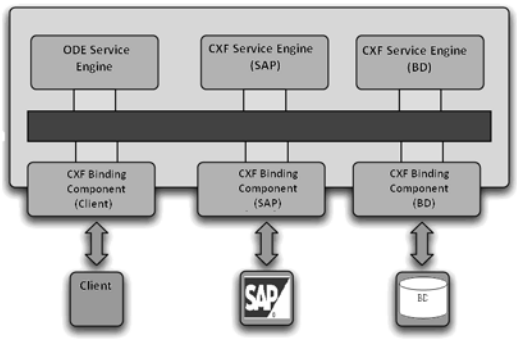

Figure 10. PSM Final Model in ServiceMix

7. Conclusions and Future Work

In order to pass to the last phase of BIM this article proposes the use of Model-Driven Architecture to transform business process models into software. MDA proposes the transformation is made by realizing tasks of platform description, mapping, marking and transforming. The first three are the ground for the final task of transforming the model. After realizing the first three tasks, all required data is collected and the software implementation is simplified.

The implementation of business process models in an Enterprise Service Bus arises as a proposal that allows an integration of the process and applications in a standard way. In the definition of the Service Engines and Binding Components the data is obtained effectively from the model transformation.

A found limitation is that BPEL only allows designing fully automatic business processes, *i.e.*, business processes where all activity is executed in the computing domain. By knowing enterprise reality, most of business processes, some of them core ones, aren't fully automatic, in a way that they need human intervention for following their normal flow. A recent standard, and with very little tool support, is BPEL4People (B4P) [39]. B4P allows representation of human tasks in a BPEL process together with the known activities of WS-BPEL.

References

[1] Smith H. and P. Fingar (2002). "Business Process Management – The Third Wave". Meghan-Kiffer Press.

[2] Fernandes J.M. and F.J. Duarte (2005). "A Reference Framework for Process-Oriented Software Development Organizations",

Software and Systems Modeling (SoSyM), Springer-Verlag.
[3] Fernandes J.M. and F.J. Duarte (2004). "Using RUP for Process-Oriented Organisations". Bomarius, F., Iida, H. (eds.), 5th Int. Conf. on Product Focused Software Process Improvement (PROFES 2004), Lecture Notes in Computer Science 3009, Springer-Verlag.
[4] Duarte F.J., R.J. Machado and J.M. Fernandes (2009). "A Methodology to Transform Business Processes into Software Systems". Submitted to Publication to International Journal of Business Process Integration and Management, Special issue on Dynamic and Declarative Business Processes.
[5] OMG (2003). "MDA Guide Version 1.0.1." Object Management Group.
[6] Duarte F.J., R.J. Machado and J.M. Fernandes (2007). "Automated Information Systems Generation for Process-Oriented Organizations". Sixth International Conference on the Quality of Information and Communications Technology.
[7] Martins V. M. M. (2005). "Integração de Sistemas de Informação: Perspectivas, normas e abordagens". Departamento de Sistemas de Informação. Guimarães, Escola de Engenharia, Universidade do Minho. Dissertação de Mestrado.
[8] Apache ServiceMix. http://servicemix.apache.org/home.html
[9] Zhao W., R. Hauser, K. Battacharya, B.R. Bryant and F. Cao (2006). "Compiling business processes: untangling unstructured loops in irreducible flow graphs". Int. J. Web and Grid Services, Vol.2, No. 1. Inderscience Enterprises Ltd.
[10] Bezivin J., S. Hammoudi, D. Lopes and F. Jouault (2004). "Applying MDA Approach to B2B Applications: A Road Map". Workshop on Model Driven Development (WMDD 2004), ECOOP 2004, Oslo, Norway.
[11] Bezivin J., G. Dupé, F. Jouault, G. Pitette and J.E. Rougui(2003). "First experiments with the ATL model transformation language: Transforming XSLT into XQuery". 2nd OOPSLA Workshop on Generative Techniques in the context of MDA, Anaheim, CA, USA.
[12] Koehler J., R. Hauser, S. Sendall and M. Wahler (2005). "Declarative techniques for model-driven business process integration". IBM Systems Journal, Vol.44, No.1.
[13] OMG (2006). "Object Constraint Language. OMG Available Specification Version 2.0". Object Maganement Group.
[14] B. Bauer, J. P. Müller and S. Roser (2004). "A Model-driven Approach to Designing Cross-Enterprise Business Processes". MIOS Workshop in OTM Conference, Springer.
[15] Lezoche M., M. Missikoff and L. Tininini (2008). "Business Process Evolution: a Rule-based Approach". Proceedings of the 9th Workshop on Business Process Modeling, Development and Support (BPMDS'08), France.
[16] Nüttgens, M., T. Feld, *et al*. (1998). "Business Process Modeling with EPC and UML: Transformation or Integration?". The Unified Modeling Language - Technical Aspects and Applications. Heidelberg.
[17] White S.A. (2004). "Business Process Modeling Notation (BPMN)": Business Process Management Initiative.
[18] Rodríguez A., E. Fernández-Medina and M. Piattini (2007). "Towards CIM to PIM transformation: From secure business processes defined in BPMN to use-cases. In: G. Alonso, P. Dadam, and M. Rosemann (2007). "Business Process Management". Proceedings of 5th International Conference, BPM 2007, Brisbane, Australia. LectureNotes in Computer Science. Springer.
[19] Rungworawut W. and T. Senivongse (2006). "Using Ontology Search in the Design of Class Diagram from Business Process Model". Proceeding of World Academy of Science, Engineering and Technology Vol.12.
[20] OMG (2009). "OMG Unified Modeling Language (OMG UML) Infrastructure Version 2.2". Object Management Group.
[21] OMG (2009). "OMG Unified Modeling Language (OMG UML) Superstructure Version 2.2". Object Management Group.
[22] Coelho, J. (2005). "Sistemas de Informação Organizacionais", Edições Sílabo, Lisboa.
[23] Machado, R. J., J. M. Fernandes, P. Monteiro and H. Rodrigues (2005). "Transformation of UML Models for Service-Oriented Software Architectures". Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05).
[24] OMG (2005). "MOF QVT Final Adopted Specification " OMG.

[25] Johnson R., J. Hoeller, A. Arendsen, T. Risberg and C. Sampaleanu (2005). "Professional Java™ Development using the Spring Framework". Wiley Publishing.
[26] Hohpe, G. and B. Woolf (2004). "Enterprise Integration Patterns". Addison Wesley.
[27] Rademakers T. and J. Dirksen (2008). "Open Source EBSs in Action". Manning Publication.
[28] Pastor, O. and J. C. Molina (2007). "Model-Driven Architecture in Practice - A Software Production Environment Based on Conceptual Modeling". Springer.
[29] Ten-Hove, R. and P. Walker (2005), "Java™ Business Integration (JBI) 1.0". Sun Microsystems, Inc. http://jcp.org/aboutJava/communityprocess/final/jsr208/index.html
[30] W3C, Web Service Description Language (WSDL). http://www.w3.org/TR/wsdl
[31] OASIS (2007). "Web Services Business Process Execution Language Version 2.0 ".
[32] Sandoe, K., G.Corbitt and R. Boykin (2001). "Enterprise Integration". New York.
[33] Couloris G., J. Dollimore and T. Kindberg (2001). "Distributed Systems: Concepts and Design". Addison Wesley.
[34] Sun, Java Database Connectivity (JDBC). http://java.sun.com/javase/technologies/database
[35] Apache ODE. http://ode.apache.org/
[36] Apache Maven. http://maven.apache.org/
[37] Bodoff, S., E. Amstrong, J. Ball, D.B. Carson, I. Evans, D. Green, K. Haase and E. Jendock (2004). "The J2EE™ Tutorial". Second Edition. Addison-Wesley.
[38] Schuessler, T. (2001). "Developing applications with the SAP java connector (JCO)". AraSoft. http://ARAsoft.de/
[39] OASIS (2007). "WS-BPEL Extension for People (BPEL4People), Version 1.0"

Anexo B – Definições dos Padrões de Workflow

[Aalst, Hofstede *et al*. 2002]

Padrões básicos de controlo de fluxo

Padrão 1: Sequência

Definição. Série ordenada de actividades, em que uma actividade é habilitada para executar somente após a execução da actividade anterior.

Padrão 2: Divisão Paralela

Definição. Um ponto no processo em que uma linha de execução única é dividida em duas ou mais linhas de execução, de forma que duas ou mais actividades possam começar simultaneamente ou em qualquer ordem. Dessa forma, actividades podem ser executadas concorrentemente, em vez de seriadas.

Padrão 3: Sincronização

Definição. Um ponto no processo em que múltiplas actividades paralelas convergem para uma única linha de execução. Essas actividades são sincronizadas nesse ponto, pois todas elas precisam estar completas para que o processo possa continuar.

Padrão 4: Escolha Exclusiva

Definição. Um ponto no processo em que, baseado em uma decisão interna ao sistema ou nos dados de controlo do workflow, um de entre os diversos ramos é escolhido. Assim, o caminho divide-se em alternativas exclusivas.

Padrão 5: Junção Simples

Definição. Um ponto no processo em que dois ou mais ramos alternativos se juntam, sem ser necessária a sincronização. Observação: assume-se, nesse padrão, que não ocorre execução em paralelo nos diversos caminhos alternativos.

Padrões avançados de ramificação e sincronização

Padrão 6: Escolha Múltipla

Definição. Um ponto no processo em que, baseado em uma decisão ou nos dados de controlo do workflow, um ou mais ramos são escolhidos. A diferença para a *escolha exclusiva* é justamente a quantidade de caminhos que, neste padrão, se pode escolher na hora da execução (desde um até todos os caminhos).

Padrão 7: Junção Sincronizada

Definição. Um ponto no processo em que múltiplos ramos convergem para um único ramo. Se mais de um ramo estiver com actividades activas, é preciso esperar o término de todas elas, sincronizando-as, para continuar. Se apenas um ramo for escolhido, o processo deve convergir sem sincronização.

Padrão 8: Junção Múltipla

Definição. Um ponto no processo em que dois ou mais ramos convergem sem sincronização. Se mais de um ramo foi escolhido anteriormente para executar, a actividade seguinte à junção será iniciada várias vezes, com a chegada de cada uma das actividades provenientes dos diversos ramos.

Padrão 9: Discriminador

Definição. Um ponto no processo que espera pelo término de um dos seus ramos de entrada antes de habilitar a actividade subsequente. A partir daí, ele espera o fim da execução dos outros ramos activos, ignorando-os. Depois disso, ele se reinicia. É importante notar que, ao contrário da *junção múltipla*, o número de vezes que a actividade subsequente é executada não depende do número de ramos activos de entrada do discriminador.

Padrões estruturais

Padrão 10: Ciclo Arbitrário

Definição. Um ponto no processo em que uma ou mais actividades (partes do processo) podem ser executadas repetidamente, formando um laço. Este padrão permite laços não estruturados. Assim, o laço pode ter mais de um ponto de entrada ou ponto de saída.

Padrão 11: Terminação Implícita

Definição. Uma instância de um processo (ou subprocesso) deve ser finalizada quando não há mais nada a ser feito, isto é, quando já não há actividades activas no (sub)processo.

Padrões envolvendo múltiplas instâncias

Padrão 12: Múltiplas Instâncias Sem Sincronização

Definição. Ponto do processo em que, dentro do contexto de uma mesma instância do processo, várias instâncias de uma actividade podem ser criadas por meio de novas linhas de execução (threads), independentes umas das outras. Não existe a necessidade de sincronizar essas linhas de execução. A criação dessas instâncias não depende de regras.

Padrão 13: Múltiplas Instâncias Com Conhecimento Prévio em Tempo de Projecto

Definição. Um ponto do processo em que, dentro do contexto de uma mesma instância do processo, uma actividade é habilitada várias vezes. O número de instâncias de uma dada actividade em uma dada instância de um processo é previamente conhecido em tempo de projecto do processo. Após o término de todas as instâncias da actividade, uma outra actividade fica habilitada para execução.

Padrão 14: Múltiplas Instâncias Com Conhecimento Prévio em Tempo de Execução

Definição. Um ponto do processo em que, dentro do contexto de uma mesma instância do processo, uma actividade é habilitada várias vezes. O número de instâncias dessa actividade pode variar dependendo da instância do processo, podendo depender de características da instância ou ainda da disponibilidade de recursos. Este número é conhecido em algum momento em tempo de execução, antes que as instâncias dessa actividade sejam criadas. Após o término de todas as instâncias da actividade, uma outra actividade fica habilitada para execução.

Padrão 15: Múltiplas Instâncias Sem Conhecimento Prévio em Tempo de Execução

Definição. Um ponto do processo em que, dentro do contexto de uma mesma instância do processo, uma actividade é habilitada várias vezes. O número de instâncias de uma dada actividade não é conhecido nem durante o projecto do processo (como no Padrão 13) nem em tempo de execução antes da criação das instâncias (como no Padrão 14). Mesmo enquanto algumas das instâncias da actividade são executadas ou até mesmo depois de completadas,

outras instâncias podem ser criadas. Após o término de todas as instâncias da actividade, uma outra actividade fica habilitada para execução.

Padrões baseados em estados

Padrão 16: Escolha em Diferido

Definição. Um ponto no processo em que um entre vários ramos é escolhido. A escolha não é feita baseada em dados existentes ou regras (como ocorre no padrão Escolha Exclusiva), mas sim baseada em uma escolha oferecida ao ambiente. Apenas uma alternativa é escolhida para execução.

Padrão 17: Roteamento Paralelo Entrelaçado

Definição. Conjunto de actividades que é executado em uma ordem arbitrária. A ordem é definida em tempo de execução e, para uma mesma instância de workflow, duas actividades não podem ser executadas no mesmo momento. Todas as actividades do conjunto serão executadas.

Padrão 18: Marco

Definição. Um ponto no processo em que a execução de uma actividade depende de uma instância de um processo estar em um estado específico. A actividade é habilitada somente se um certo marco foi atingido, mas ainda não tiver expirado. Este padrão será exemplificado por meio de dois casos distintos, que reflectem diferentes situações.

Padrões de cancelamento

Padrão 19: Actividade Cancelável

Definição. Ocorre quando uma actividade habilitada é cancelada antes do começo da sua execução. Quando a actividade é cancelada, é necessário remover a linha de execução que controla a actividade. Caso a execução da actividade já tenha começado, ela é cancelada e, quando possível, a instância em execução é paralisada e removida.

Padrão 20: Caso Cancelável

Definição. Ocorre quando uma instância do processo é completamente removida. Esta remoção inclui todas as actividades que estão executando no momento, aquelas que ainda poderiam ser executadas e todos os subprocessos. A instância do processo é armazenada como completada de modo mal-sucedido.

Anexo C – Questionário

O presente questionário tem como objectivo avaliar os conhecimentos de cada um na área de BPM e de modelação de processos de negócio. Pretendo para tal chegar a uma conclusão sobre a linguagem de modelação de processos mais adequada para o perfil dos colaboradores da empresa.

As perguntas irão recair sobretudo no grau de familiarização acerca de alguns conceitos. Em alguns casos, pretende-se saber se “está” ou “não está” familiarizado com o conceito. Noutros, pretende-se saber ser mais específico, com o objectivo se está “muito” ou “pouco” familiarizado.

A identificação não é relevante para a interpretação dos resultados.

Obrigado!!!

I. Conceitos BPM

1. Dos seguintes conceitos da temática de BPM, indique os que conhece (definição de "conhecer": saber o que são, o que fazem, para que servem… não requer necessariamente já ter utilizado qualquer ferramenta de implementação dos conceitos). Assinale com um "X" a seguir ao nome (pode assinalar **as opções que entender**):

   a. BPM (Business Process Management) ___
   b. BPMS (Business Process Management Systems) ___
   c. BAM (Business Activity Monitoring) ___
   d. EAI (Enterprise Application Integration) ___
   e. ERP (Enterprise Resource Planning) ___
   f. CRM (Customer Relashionship Management) ___
   g. SCM (Supply Chain Management) ___

2. Na Gestão da Qualidade dos processos de negócio, indique quais as estratégias que conhece. Assinale com um "X" a seguir ao nome (pode assinalar **as opções que entender**):

   a. TQM (Total Quality Management) ___
   b. Six Sigma ___
   c. Kaizen (melhoria contínua) ___
   d. BSC (Balanced ScoreCard) ___
   e. Outra(s) ___
      Qual? _____

3. Já participou em algum projecto BPM (reestruturação/melhoria, modelação e monitorização dos processos de negócio de uma empresa)? Assinale com um "X" **uma opção**:
   Sim: _____ Não: _____

4. Dos seguintes conceitos de projectos BPM, mesmo que tenha escolhido "Não" em 3., indique os que conhece (o que são, para que servem…). Assinale com um "X" a seguir ao nome (pode assinalar **as opções que entender**):

    a. Objectivos Operacionais (Curto-Prazo) ___
    b. Critérios de desempenho ___
    c. Arquitectura de Processos:
        i. Processos 1º nível (ou macro-processos) ___
        ii. Sub-processos ___
        iii. Actividades ___
        iv. Tarefas ___
        v. Operações ___
    d. Necessidades da informação ("o quê") ___
    e. Requisitos da informação ("como") ___
    f. Sistemas aplicacionais ___

II. Conceitos de Linguagens de Modelação de Processos de Negócio

5. Sabe o que são diagramas de workflow? (Assinale com um "X **uma opção**) Sim: _____ Não: _____

6. Indique o seu grau de familiaridade em relação a operações algébricas – AND, OR, NAND, NOR, XOR: (Assinale com um "X" **uma opção**)

    a. Não possuo conhecimentos acerca de nenhuma operação ___
    b. Conheço algumas operações ___
    c. Conheço todas as operações ___
    d. Conheço todas as operações, mas só estou à vontade para aplicar algumas delas ___
    e. Conheço todas as operações, e sei perfeitamente aplicá-las todas ___

7. Sabe o que são diagramas em grafos? (Assinale com um "X" **uma opção**) Sim: _____ Não: _____

8. Indique o seu grau de familiaridade em relação à modelação de Redes de Petri: (Assinale com um "X" **uma opção**)

   a. Não possuo conhecimentos ___
   b. Possuo conhecimentos mínimos ___
   c. Possuo conhecimentos medianos ___
   d. Possuo conhecimentos bons ___
   e. Possuo conhecimentos muito bons ___

9. Conhece a técnica do Cálculo de π (pi)? (Assinale com um "X" **uma opção**) Sim: _____ Não: _____

10. Indique o seu grau de familiaridade em relação à linguagem de programação XML: (Assinale com um "X" **uma opção**)

    a. Não possuo conhecimentos acerca da linguagem ___
    b. Sei apenas que é uma linguagem de programação para a Web ___
    c. Possuo conhecimentos medianos acerca da sua programação ___
    d. Estou perfeitamente à vontade para programar ___

11. Das seguintes linguagens de modelação, indique, **para cada uma**, a opção correcta que reflita o seu conhecimento quanto à modelação da linguagem: (Assinale com um "X")

	Não sei modelar	Sei modelar pouco	Sei modelar suficiente	Sei modelar bem	Sei modelar muito bem
UML 2.0 Casos de Uso Actividades Estado					
BPMN					
EPC (ARIS)					

12. Das seguintes linguagens (algumas de modelação, outras de coreografia, outras de execução, etc.), indique as que conhece (o que são, o que fazem, para que servem…). Assinale com um "X" a seguir ao nome (pode assinalar **as opções que entender**):

a. BPDM (Business Process Definition Metamodel) ___
b. BPEL (Web Services Business Process Execution Language) ___
c. BPML (Business Process Modelling Language) ___
d. BPMN (Business Process Modelling Notation) ___
e. BPRI (Business Process Runtime Interface) ___
f. BPSM (Business Process Semantic Model) ___
g. BPSS (Business Process Schema Specification) ___
h. BPQL (Business Process Query Language) ___
i. BPXL (Business Process Extension Layer) ___
j. CPN (Colored Petri-Nets) ___
k. EPC (Event-driven Process Chain) ___
l. UML 2.0 (Unified Modelling Language 2.0 ___
m. WAPI (Workflow API) ___
n. Wf-XML (Workflow-XML) ___
o. WS-CDL (Web Services Choreography Description Language) ___
p. WSCI (Web Service Choreography Interface) ___
q. WSCL (Web Services Conversation Language) ___
r. WSFL (Web Services Flow Language) ___
s. XLANG (eXtensible Language) ___
t. XPDL (XML Process Definition Language) ___
u. YAWL (Yet Another Workflow Language) ___

III. Conceitos SOA (Service-Oriented Architecture)

13. Conhece a norma SOA (Service-Oriented Architecture)? (Assinale com um "X" **uma opção**) Sim: _____ Não: _____

14. Classifica os seus conhecimentos em Web Services como: (Assinale com um "X" **uma opção**)

a. Não possuo conhecimentos ___
b. Possuo conhecimentos mínimos ___
c. Possuo conhecimentos medianos ___
d. Possuo conhecimentos bons ___
e. Possuo conhecimentos muito bons ___

15. Classifica os seus conhecimentos em concorrência (threading) como: (Assinale com um "X" **uma opção**)

f. Não possuo conhecimentos ___
g. Possuo conhecimentos mínimos ___
h. Possuo conhecimentos medianos ___
i. Possuo conhecimentos bons ___
j. Possuo conhecimentos muito bons ___

16. Dos seguintes conceitos da norma SOA, mesmo que tenha escolhido "não" em 14., indique as que conhece (o que são, o que fazem, para que servem…). Assinale com um "X" a seguir ao nome (pode assinalar **as opções que entender**):

a. B2B (Business-to-Business) ___
b. B2C (Business-to-Consumer) ___
c. B2E (Business-to-Employee) ___
d. B2G (Business-to-Government) ___
e. CORBA (Common Object Request Broker Architecture) ___
f. EDI (Electronic Data Interchange) ___
g. ESB (Enterprise Service Bus) ___
h. FTP (File Transfer Protocol) ___
i. HTTP (HiperText Transfer Protocol) ___
j. RPC (Remote Procedure Call) ___
k. SMTP (Simple Mail Transfer Protocol) ___
l. SOAP (Simple Object Access Protocol) ___
m. SOMF (Service-Oriented Modelling Framework) ___
n. UDDI (Universal Description, Discovery and Integration) ___
o. Web Services ___
p. WSDL (Web Services Description Language) ___

Obrigado!

Anexo D – Tabela de respostas

Questão	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
1									
a.	X	X		X		X	X	X	X
b.	X								X
c.	X					X			
d.	X	X							X
e.	X	X	X	X	X	X	X	X	X
f.	X	X	X			X		X	X
g.	X	X	X			X		X	X
2									
a.	X	X	X			X		X	X
b.	X	X	X			X	X		
c.	X	X	X				X		
d.	X	X	X			X		X	
e.							X		
3	Não	Sim	Não	Não	Sim	Sim	Não	Não	Sim
4									
a.	X	X	X	X				X	
b.	X	X	X		X			X	
c.									
i.	X	X	X			X		X	X
ii.	X	X	X			X		X	X
iii.	X	X	X	X		X		X	X
iv.	X	X	X	X		X		X	X
v.	X	X	X			X		X	
d.	X	X				X		X	X
e.	X	X				X		X	X
f.	X	X						X	
5	Sim	Sim	Não	Sim	Sim	Sim	Sim	Sim	Sim
6	e.	e.	d.	e.	e.	e.	d.	d.	d.
7	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
8	b.	b.	b.	a.	b.	d.	b.	b.	b.
9	Não	Não	Não	Não	Não	Sim	Não	Não	Não
10	b.	c.	c.	d.	c.	d.	c.	c.	c.
11									
UML CU	d.	e.	d.	e.	d.	d.	c.	c.	c.
UML A	d.	e.	d.	e.	d.	d.	c.	c.	c.
UML E	c.	e.	d.	e.	d.	d.	c.	c.	c.
BPMN	c.	a.	a.	a.	a.	a.		a.	b.
EPC	c.	c.	b.	a.	a.	b.		a.	a.
12									
a.									X
b.	X	X				X			X
c.	X								
d.	X					X			X
e.									
f.									
g.									
h.	X								
i.									

j.	X	X				X			
k.	X	X				X			
l.	X	X	X	X	X	X		X	X
m.									
n.	X								
o.	X								
p.	X								
q.									
r.	X								X
s.	X								
t.	X					X			X
u.	X								
13	Sim	Sim	Sim	Sim	Sim	Sim	Não	Sim	Sim
14	c.	c.	c.	d.	d.	d.	b.	b.	c.
15	b.	d.	c.	d.	d.	d.	b.	b.	c.
16									
a.	X	X				X	X		X
b.	X	X				X			
c.		X							
d.	X	X							
e.	X	X	X			X			
f.	X	X	X					X	
g.	X	X							
h.	X	X	X	X	X	X		X	X
i.	X	X	X	X	X	X	X	X	X
j.	X	X	X		X	X		X	
k.	X	X	X			X		X	X
l.	X	X	X	X	X	X		X	X
m.									
n.	X	X				X			
o.	X	X	X		X	X		X	
p.	X	X	X	X	X	X		X	X

Anexo E – Código do Processo em *Runnable*

Ficheiro BPEL:

```
<bpel:process name="LancQuant"
         targetNamespace="http://LancQuant"
         suppressJoinFailure="yes"
         xmlns:tns="http://LancQuant"
         xmlns:bpel="http://docs.oasis-
open.org/wsbpel/2.0/process/executable"
         xmlns:ns1="http://Bd " xmlns:ns3="http://Sap">

    <!-- Import the client WSDL -->
    <bpel:import namespace="http://Sap" location="SapArtifacts.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
    <bpel:import namespace="http://Bd" location="BdArtifacts.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
    <bpel:import location="LancQuantArtifacts.wsdl"
namespace="http://LancQuant"
              importType="http://schemas.xmlsoap.org/wsdl/" />

    <!--
================================================================= -->
    <!-- PARTNERLINKS
-->
    <!-- List of services participating in this BPEL process
-->
    <!--
================================================================= -->
    <bpel:partnerLinks>
        <!-- The 'client' role represents the requester of this
service. -->
        <bpel:partnerLink name="client"
                     partnerLinkType="tns:LancQuant"
                     myRole="LancQuantProvider"
                     />
        <bpel:partnerLink name="bd" partnerLinkType="ns1:Bd"
partnerRole="BdProvider"></bpel:partnerLink>

        <bpel:partnerLink name="sap" partnerLinkType="ns3:Sap"
partnerRole="SapProvider"></bpel:partnerLink>
    </bpel:partnerLinks>

    <!--
================================================================= -->
    <!-- VARIABLES
-->
    <!-- List of messages and XML documents used within this BPEL
process  -->
    <!--
================================================================= -->
    <bpel:variables>
        <!-- Reference to the message passed as input during
initiation -->
        <bpel:variable name="input"
                  messageType="tns:LancQuantRequestMessage"/>

        <!--
```

```
          Reference to the message that will be returned to the
requester
          -->
        <bpel:variable name="output"
                  messageType="tns:LancQuantResponseMessage"/>
        <bpel:variable name="bdResponse"
messageType="ns1:BdResponseMessage"></bpel:variable>
        <bpel:variable name="bdRequest"
messageType="ns1:BdRequestMessage"></bpel:variable>


        <bpel:variable name="sapResponse"
messageType="ns3:SapResponseMessage"></bpel:variable>
        <bpel:variable name="sapRequest"
messageType="ns3:SapRequestMessage"></bpel:variable>
    </bpel:variables>

    <!--
================================================================= -->
    <!-- ORCHESTRATION LOGIC
-->
    <!-- Set of activities coordinating the flow of messages across
the    -->
    <!-- services integrated within this business process
-->
    <!--
================================================================= -->
    <bpel:sequence name="main">

        <!-- Receive input from requester.
             Note: This maps to operation defined in LancQuant.wsdl
             -->
        <bpel:receive name="receiveInput" partnerLink="client"
                 portType="tns:LancQuant"
                 operation="process" variable="input"
                 createInstance="yes"/>

        <!-- Generate reply to synchronous request -->
        <bpel:assign validate="no" name="Assign">
            <bpel:copy>
                <bpel:from>
                    <bpel:literal xml:space="preserve"><tns:SapRequest
xmlns:tns="http://Sap" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <tns:data></tns:data>
  <tns:turno></tns:turno>
  <tns:linha></tns:linha>
  <tns:id_produto></tns:id_produto>
  <tns:qtd_produzida></tns:qtd_produzida>
  <tns:qtd_refugo></tns:qtd_refugo>
  <tns:qtd_erros_vacuum></tns:qtd_erros_vacuum>
  <tns:qtd_erros_id></tns:qtd_erros_id>
  <tns:qtd_placas_mas></tns:qtd_placas_mas>
  <tns:qtd_maus></tns:qtd_maus>
</tns:SapRequest>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="sapRequest"
part="payload"></bpel:to>
            </bpel:copy>
            <bpel:copy>
```

```
                <bpel:from>
                    <bpel:literal xml:space="preserve"><tns:BdRequest
xmlns:tns="http://Bd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <tns:data></tns:data>
  <tns:turno></tns:turno>
  <tns:linha></tns:linha>
  <tns:id_produto></tns:id_produto>
  <tns:qtd_produzida></tns:qtd_produzida>
  <tns:qtd_refugo></tns:qtd_refugo>
  <tns:qtd_erros_vacuum></tns:qtd_erros_vacuum>
  <tns:qtd_erros_id></tns:qtd_erros_id>
  <tns:qtd_placas_mas></tns:qtd_placas_mas>
  <tns:qtd_maus></tns:qtd_maus>
</tns:BdRequest>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="bdRequest"
part="payload"></bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:data]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="bdRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:data]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:turno]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="bdRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:turno]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:linha]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="bdRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:linha]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
                        <![CDATA[tns:id_produto]]>
```

```
                    </bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="bdRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:id_produto]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_produzida]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="bdRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:qtd_produzida]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_refugo]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="bdRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:qtd_refugo]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_erros_vacuum]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="bdRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:qtd_erros_vacuum]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_placas_mas]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="bdRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:qtd_placas_mas]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_maus]]></bpel:query>
```

```
                </bpel:from>
                <bpel:to part="payload" variable="bdRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:qtd_maus]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:data]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns3:data]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:turno]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns3:turno]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:linha]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns3:linha]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:id_produto]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns3:id_produto]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_produzida]]></bpel:query>
                </bpel:from>
```

```
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns3:qtd_produzida]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_refugo]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns3:qtd_refugo]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_erros_vacuum]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns3:qtd_erros_vacuum]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_erros_id]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns3:qtd_erros_id]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_placas_mas]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns3:qtd_placas_mas]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_maus]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
```

```
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns3:qtd_maus]]></bpel:query>
                </bpel:to>
            </bpel:copy>
        </bpel:assign>
        <bpel:flow name="Flow"><bpel:invoke name="SAP"
partnerLink="sap" operation="process" inputVariable="sapRequest"
outputVariable="sapResponse"
portType="ns3:Sap"></bpel:invoke><bpel:invoke name="Bd"
partnerLink="bd" operation="process" inputVariable="bdRequest"
outputVariable="bdResponse"
portType="ns1:Bd"></bpel:invoke></bpel:flow>
        <bpel:assign validate="no" name="Assign1">
            <bpel:copy>
                <bpel:from>
                    <bpel:literal
xml:space="preserve"><tns:LancQuantResponse
xmlns:tns="http://localhost:8080/LancQuant"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <tns:sap></tns:sap>
  <tns:bd></tns:bd>
</tns:LancQuantResponse>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="output" part="payload"></bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="bdResponse">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:result]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="output">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:bd]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="sapResponse">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns3:result]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="output">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:sap]]></bpel:query>
                </bpel:to>
            </bpel:copy>
        </bpel:assign>
        <bpel:reply name="replyOutput"
               partnerLink="client"
               portType="tns:LancQuant"
               operation="process"
               variable="output"
               />
    </bpel:sequence>
</bpel:process>
```

Ficheiro WSDL:

```
<?xml version="1.0"?>
<definitions name="LancQuant"
        targetNamespace="http:/LancQuant"
        xmlns:tns="http://LancQuant"
        xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     TYPE DEFINITION - List of types participating in this BPEL
process
     The BPEL Designer will generate default request and response
types
     but you can define or import any XML Schema type and use them as
part
     of the message types.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <types>
        <schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
                targetNamespace="http://localhost:8080/LancQuant"
                xmlns="http://www.w3.org/2001/XMLSchema">

            <element name="LancQuantRequest">
                <complexType>
                    <sequence>
                        <element name="data" type="int"/>
                        <element name="turno" type="string"/>
                        <element name="linha" type="int"/>
                        <element name="id_produto" type="int"/>
                        <element name="qtd_produzida" type="int"/>
                        <element name="qtd_refugo" type="int"/>
                        <element name="qtd_erros_vacuum" type="int"/>
                        <element name="qtd_erros_id" type="int"/>
                        <element name="qtd_placas_mas" type="int"/>
                        <element name="qtd_maus" type="int"/>
                    </sequence>
                </complexType>
            </element>
            <element name="LancQuantResponse">
                <complexType>
                    <sequence>
                        <element name="sap" type="boolean"/>
                        <element name="bd" type="boolean"/>
                    </sequence>
                </complexType>
            </element>
        </schema>
    </types>


<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     MESSAGE TYPE DEFINITION - Definition of the message types used as
     part of the port type defintions
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <message name="LancQuantRequestMessage">
        <part name="payload" element="tns:LancQuantRequest"/>
    </message>
    <message name="LancQuantResponseMessage">
        <part name="payload" element="tns:LancQuantResponse"/>
    </message>

<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PORT TYPE DEFINITION - A port type groups a set of operations
into
     a logical service unit.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->

    <!-- portType implemented by the LancQuant BPEL process -->
    <portType name="LancQuant">
        <operation name="process">
            <input  message="tns:LancQuantRequestMessage" />
            <output message="tns:LancQuantResponseMessage"/>
        </operation>
    </portType>

<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PARTNER LINK TYPE DEFINITION

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="LancQuant">
        <plnk:role name="LancQuantProvider" portType="tns:LancQuant"/>
    </plnk:partnerLinkType>

    <binding name="LancQuantSoapBinding" type="LancQuant">
      <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http" />
      <operation name="process">
            <soap:operation
                  soapAction="http://localhost:8080/LancQuant/process"
/>
            <input>
                  <soap:body use="literal" />
            </input>
            <output>
                  <soap:body use="literal" />
            </output>
      </operation>
    </binding>
    <service name="lancQuantService">
      <port name="lancQuantPort" binding="LancQuantSoapBinding">
            <soap:address
location="http://0.0.0.0:8080/ode/processes/LancQuantService" />
      </port>
    </service>
</definitions>
```

Anexo F – Código dos Componentes do Processo em *Software Implemented*

Ficheiro xbean.xml do CXF SE (para SAP):

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:cxfse="http://servicemix.apache.org/cxfse/1.0"
       xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://servicemix.apache.org/cxfse/1.0
http://servicemix.apache.org/schema/servicemix-cxfse-3.2.3.xsd
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

    <cxfse:endpoint>
        <cxfse:pojo>
          <bean class="pt.bosch.com.teste_cxfse_su.SapService" />
        </cxfse:pojo>
    </cxfse:endpoint>

</beans>
```

Ficheiro xbean.xml do CXF SE (para BD):

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:cxfse="http://servicemix.apache.org/cxfse/1.0"
       xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://servicemix.apache.org/cxfse/1.0
http://servicemix.apache.org/schema/servicemix-cxfse-3.2.3.xsd
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

    <cxfse:endpoint>
        <cxfse:pojo>
          <bean class="pt.bosch.com.teste_cxfse_su.BdService" />
        </cxfse:pojo>
    </cxfse:endpoint>

</beans>
```

Ficheiro xbean.xml do CXF BC (para SAP):

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:cxfbc="http://servicemix.apache.org/cxfbc/1.0"
       xmlns:xsi="http://http://www.w3.org/2001/XMLSchema-instance"
       xmlns:sap="http://Sap"
       xsi:schemaLocation="http://servicemix.apache.org/cxfbc/1.0
http://servicemix.apache.org/schema/servicemix-cxfbc-3.2.3.xsd
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
```

```
  <cxfbc:consumer wsdl="classpath:SapArtifacts.wsdl"
                      targetEndpoint="sap:SapPort"
                      targetService="sap:SapService"
                      targetInterface="sap:Sap"/>

</beans>
```

Ficheiro xbean.xml *do* CXF BC *(para BD):*

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:cxfbc="http://servicemix.apache.org/cxfbc/1.0"
       xmlns:xsi="http://http://www.w3.org/2001/XMLSchema-instance"
       xmlns:bd="http://Bd"
       xsi:schemaLocation="http://servicemix.apache.org/cxfbc/1.0
http://servicemix.apache.org/schema/servicemix-cxfbc-3.2.3.xsd
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

  <cxfbc:consumer wsdl="classpath:BdArtifacts.wsdl"
                      targetEndpoint="bd:BdPort"
                      targetService="bd:BdService"
                      targetInterface="bd:Bd"/>

</beans>
```

Ficheiro BPEL do Processo:

```
<bpel:process name="LancQuant"
         targetNamespace="http://LancQuant"
         suppressJoinFailure="yes"
         xmlns:tns="http://LancQuant"
         xmlns:bpel="http://docs.oasis-
open.org/wsbpel/2.0/process/executable"
         xmlns:ns1="http://Bd" xmlns:ns2="http://Sap">

    <!-- Import the client WSDL -->
    <bpel:import namespace="http://Sap" location="SapArtifacts.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
    <bpel:import namespace="http://Bd" location="BdArtifacts.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
    <bpel:import location="LancQuantArtifacts.wsdl"
namespace="http://LancQuant"
              importType="http://schemas.xmlsoap.org/wsdl/" />

    <!--
================================================================= -->
    <!-- PARTNERLINKS
-->
    <!-- List of services participating in this BPEL process
-->
    <!--
================================================================= -->
    <bpel:partnerLinks>
        <!-- The 'client' role represents the requester of this
service. -->
        <bpel:partnerLink name="client"
                      partnerLinkType="tns:LancQuant"
```

```
                    myRole="LancQuantProvider"
                    />
        <bpel:partnerLink name="bd" partnerLinkType="ns1:Bd"
partnerRole="BdProvider"></bpel:partnerLink>
        <bpel:partnerLink name="sap" partnerLinkType="ns2:Sap"
partnerRole="SapProvider"></bpel:partnerLink>
    </bpel:partnerLinks>

    <!--
================================================================= -->
    <!-- VARIABLES
-->
    <!-- List of messages and XML documents used within this BPEL
process  -->
    <!--
================================================================= -->
    <bpel:variables>
        <!-- Reference to the message passed as input during
initiation -->
        <bpel:variable name="input"
                  messageType="tns:LancQuantRequestMessage"/>

        <!--
          Reference to the message that will be returned to the
requester
          -->
        <bpel:variable name="output"
                  messageType="tns:LancQuantResponseMessage"/>
        <bpel:variable name="sapResponse"
messageType="ns2:SapResponseMessage"></bpel:variable>
        <bpel:variable name="sapRequest"
messageType="ns2:SapRequestMessage"></bpel:variable>
        <bpel:variable name="bdResponse"
messageType="ns1:BdResponseMessage"></bpel:variable>
        <bpel:variable name="bdRequest"
messageType="ns1:BdRequestMessage"></bpel:variable>
    </bpel:variables>

    <!--
================================================================= -->
    <!-- ORCHESTRATION LOGIC
-->
    <!-- Set of activities coordinating the flow of messages across
the    -->
    <!-- services integrated within this business process
-->
    <!--
================================================================= -->
    <bpel:sequence name="main">

        <!-- Receive input from requester.
             Note: This maps to operation defined in LancQuant.wsdl
             -->
        <bpel:receive name="receiveInput" partnerLink="client"
                 portType="tns:LancQuant"
                 operation="process" variable="input"
                 createInstance="yes"/>

        <!-- Generate reply to synchronous request -->
        <bpel:assign validate="no" name="Assign">
```

```
            <bpel:copy>
                <bpel:from>
                    <bpel:literal xml:space="preserve"><tns:BdRequest
xmlns:tns="http://Bd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <tns:smd></tns:smd>
  <tns:linha></tns:linha>
  <tns:qtd_produzida></tns:qtd_produzida>
  <tns:defeitos></tns:defeitos>
  <tns:oee></tns:oee>
</tns:BdRequest>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="bdRequest"
part="payload"></bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from>
                    <bpel:literal xml:space="preserve"><tns:SapRequest
xmlns:tns="http://Sap" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <tns:data></tns:data>
  <tns:turno></tns:turno>
  <tns:linha></tns:linha>
  <tns:id_produto></tns:id_produto>
  <tns:qtd_produzida></tns:qtd_produzida>
  <tns:qtd_refugo></tns:qtd_refugo>
  <tns:qtd_erros_vacuum></tns:qtd_erros_vacuum>
  <tns:qtd_erros_id></tns:qtd_erros_id>
  <tns:qtd_placas_mas></tns:qtd_placas_mas>
  <tns:qtd_maus></tns:qtd_maus>
</tns:SapRequest>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="sapRequest"
part="payload"></bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:data]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns2:data]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:turno]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns2:turno]]></bpel:query>
                </bpel:to>
            </bpel:copy>
```

```
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:linha]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns2:linha]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:id_produto]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns2:id_produto]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_produzida]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns2:qtd_produzida]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_refugo]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns2:qtd_refugo]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_erros_vacuum]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns2:qtd_erros_vacuum]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
```

```
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_erros_id]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns2:qtd_erros_id]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_placas_mas]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns2:qtd_placas_mas]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_maus]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="sapRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns2:qtd_maus]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
                        <![CDATA[tns:linha]]>
                    </bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="bdRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:linha]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_produzida]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="bdRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:qtd_produzida]]></bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
```

```
                <bpel:from part="payload" variable="input">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:qtd_maus]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="bdRequest">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:defeitos]]></bpel:query>
                </bpel:to>
            </bpel:copy>
        </bpel:assign>
        <bpel:flow name="Flow"><bpel:invoke name="SAP SE"
partnerLink="sap" operation="process" inputVariable="sapRequest"
outputVariable="sapResponse"></bpel:invoke><bpel:invoke name="BD SE"
partnerLink="bd" operation="process" inputVariable="bdRequest"
outputVariable="bdResponse"></bpel:invoke></bpel:flow>
        <bpel:assign validate="no" name="Assign1">
            <bpel:copy>
                <bpel:from>
                    <bpel:literal
xml:space="preserve"><tns:LancQuantResponse
xmlns:tns="http://LancQuant"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <tns:sap></tns:sap>
  <tns:bd></tns:bd>
</tns:LancQuantResponse>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="output" part="payload"></bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="bdResponse">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns1:result]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="output">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:bd]]></bpel:query>
                </bpel:to>
            </bpel:copy>

            <bpel:copy>
                <bpel:from part="payload" variable="sapResponse">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[ns2:result]]></bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="output">
                    <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDAT
A[tns:sap]]></bpel:query>
                </bpel:to>
            </bpel:copy>
        </bpel:assign>
        <bpel:reply name="replyOutput"
               partnerLink="client"
               portType="tns:LancQuant"
               operation="process"
```

```
                variable="output"
                />
    </bpel:sequence>
</bpel:process>
```

Ficheiro WSDL do Processo:

```
<?xml version="1.0"?>
<definitions name="LancQuant"
        targetNamespace="http://LancQuant"
        xmlns:tns="http://LancQuant"
        xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     TYPE DEFINITION - List of types participating in this BPEL
process
     The BPEL Designer will generate default request and response
types
     but you can define or import any XML Schema type and use them as
part
     of the message types.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <types>
        <schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
                targetNamespace="http://LancQuant"
                xmlns="http://www.w3.org/2001/XMLSchema">

            <element name="LancQuantRequest">
                <complexType>
                    <sequence>
                        <element name="data" type="int"/>
                        <element name="turno" type="string"/>
                        <element name="linha" type="int"/>
                        <element name="id_produto" type="int"/>
                        <element name="qtd_produzida" type="int"/>
                        <element name="qtd_refugo" type="int"/>
                        <element name="qtd_erros_vacuum" type="int"/>
                        <element name="qtd_erros_id" type="int"/>
                        <element name="qtd_placas_mas" type="int"/>
                        <element name="qtd_maus" type="int"/>
                    </sequence>
                </complexType>
            </element>

            <element name="LancQuantResponse">
                <complexType>
                    <sequence>
                        <element name="sap" type="boolean"/>
                        <element name="bd" type="boolean"/>>
                    </sequence>
                </complexType>
            </element>
        </schema>
    </types>
```

```
<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     MESSAGE TYPE DEFINITION - Definition of the message types used as
     part of the port type defintions

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <message name="LancQuantRequestMessage">
        <part name="payload" element="tns:LancQuantRequest"/>
    </message>
    <message name="LancQuantResponseMessage">
        <part name="payload" element="tns:LancQuantResponse"/>
    </message>

<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PORT TYPE DEFINITION - A port type groups a set of operations
into
     a logical service unit.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->

    <!-- portType implemented by the LancQuant BPEL process -->
    <portType name="LancQuant">
        <operation name="process">
            <input  message="tns:LancQuantRequestMessage" />
            <output message="tns:LancQuantResponseMessage"/>
        </operation>
    </portType>


<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PARTNER LINK TYPE DEFINITION

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="LancQuant">
        <plnk:role name="LancQuantProvider" portType="tns:LancQuant"/>
    </plnk:partnerLinkType>

    <binding name="LancQuantSoapBinding" type="tns:LancQuant">
      <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http" />
      <operation name="process">
            <soap:operation soapAction="http://LancQuant/process" />
            <input>
                  <soap:body use="literal" />
            </input>
            <output>
                  <soap:body use="literal" />
            </output>
      </operation>
    </binding>
    <service name="LancQuantService">
      <port name="LancQuantPort" binding="tns:LancQuantSoapBinding">
            <soap:address
location="http://0.0.0.0:8080/ode/processes/LancQuant" />
```

```
      </port>
    </service>
</definitions>
```

Ficheiro deploy.xml do ODE:

```
<?xml version="1.0" encoding="UTF-8"?>
<deploy xmlns="http://www.apache.org/ode/schemas/dd/2007/03"
xmlns:Bd="http://Bd" xmlns:LancQuant="http://LancQuant"
xmlns:Sap="http://Sap">
  <process name="LancQuant:LancQuant">
    <process-events generate="all"/>
    <provide partnerLink="client">
      <service name="LancQuant:LancQuantService"
port="LancQuantPort"/>
    </provide>
    <invoke partnerLink="bd">
      <service name="Bd:BdService" port="bdPort"/>
    </invoke>
    <invoke partnerLink="sap">
      <service name="Sap:SapService" port="SapPort"/>
    </invoke>
  </process>
</deploy>
```

Ficheiro xbean.xml do CXF BC ("Client"):

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:cxfbc="http://servicemix.apache.org/cxfbc/1.0"
       xmlns:xsi="http://http://www.w3.org/2001/XMLSchema-instance"
       xmlns:lq="http://LancQuant"
       xsi:schemaLocation="http://servicemix.apache.org/cxfbc/1.0
http://servicemix.apache.org/schema/servicemix-cxfbc-3.2.3.xsd
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

  <cxfbc:consumer wsdl="classpath:LancQuantArtifacts.wsdl"
                      targetEndpoint="lq:LancQuantPort"
                      targetService="lq:LancQuantService"
                      targetInterface="lq:LancQuant"/>

</beans>
```

Ficheiro pom.xml do SA:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>fsi12.pt.bosch.com</groupId>
  <artifactId>psm-lancquant-sa</artifactId>
  <packaging>jbi-service-assembly</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>SA for LancQuant</name>
  <url>http://www.myorganization.org</url>
  <repositories>
    <repository>
      <id>apache</id>
      <name>Apache Repository</name>
      <url>http://people.apache.org/repo/m2-ibiblio-rsync-
repository</url>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
      <releases>
        <enabled>true</enabled>
      </releases>
    </repository>
    <repository>
      <id>apache.snapshots</id>
      <name>Apache Snapshots Repository</name>
      <url>http://people.apache.org/repo/m2-snapshot-repository</url>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
      <releases>
        <enabled>false</enabled>
      </releases>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>apache</id>
      <name>Apache Repository</name>
      <url>http://people.apache.org/repo/m2-ibiblio-rsync-
repository</url>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
      <releases>
        <enabled>true</enabled>
      </releases>
      </pluginRepository>
      <pluginRepository>
        <id>apache.snapshots</id>
        <name>Apache Snapshots Repository</name>
        <url>http://people.apache.org/repo/m2-snapshot-
repository</url>
        <snapshots>
          <enabled>true</enabled>
        </snapshots>
```

```
        <releases>
          <enabled>false</enabled>
        </releases>
      </pluginRepository>
  </pluginRepositories>
  <properties>
    <servicemix-version>3.2.3</servicemix-version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>fsi12.pt.bosch.com</groupId>
      <artifactId>bd-cxfbc-su</artifactId>
      <version>0.0.1-SNAPSHOT</version>
    </dependency>
    <dependency>
      <groupId>fsi12.pt.bosch.com</groupId>
      <artifactId>bd-cxfse-su</artifactId>
      <version>0.0.1-SNAPSHOT</version>
    </dependency>
    <dependency>
      <groupId>fsi12.pt.bosch.com</groupId>
      <artifactId>lancquant-cxfbc.su</artifactId>
      <version>0.0.1-SNAPSHOT</version>
    </dependency>
    <dependency>
      <groupId>fsi12.pt.bosch.com</groupId>
      <artifactId>sap-cxfbc-su</artifactId>
      <version>0.0.1-SNAPSHOT</version>
    </dependency>
    <dependency>
      <groupId>fsi12.pt.bosch.com</groupId>
      <artifactId>sap.cxfse-su</artifactId>
      <version>0.0.1-SNAPSHOT</version>
    </dependency>
    <dependency>
      <groupId>fsi12.pt.bosch.com</groupId>
      <artifactId>teste-ode-su</artifactId>
      <version>{version}</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.servicemix.tooling</groupId>
        <artifactId>jbi-maven-plugin</artifactId>
        <version>${servicemix-version}</version>
        <extensions>true</extensions>
        <configuration>
          <type>service-assembly</type>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```