

Estudo de um caso de Localização de um Software ERP de Código Livre

Luís Alfredo Harriss Maranesi

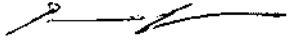
Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Luís Alfredo Harriss Maranesi e aprovada pela Banca Examinadora.

Campinas, 11 de Novembro de 2011.

Hans Liesenberg (Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Na folha i e ii - Onde se lê: Luís Alfredo Harriss Maranesi Leia-se: Luís Alfredo Harriss Maranesi


Prof. Dr. Paulo Lício de Geus
Coordenador de Pós Graduação
Instituto de Computação UNICAMP
Matr.103268

FICHA CATALOGRÁFICA ELABORADA POR
MARIA FABIANA BEZERRA MÜLLER - CRB8/6162
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

Unidade PPG
T/UNICAMP 111.928
Cutter _____
V. _____ Ed. _____
Tombo BC 95051
Proc. 16-100-15
C _____ P 8
Preço 41,80
Data 8/05/12
Cod. tit. 899399

M325e

Maranesi, Luís Alfredo Harriss, 1985-

Estudo de um caso de localização de um software
ERP de código livre / Luís Alfredo Harriss Maranesi. -
Campinas, SP : [s.n.], 2011.

Orientador: Hans Kurt Edmund Liesenberg.
Dissertação (mestrado) – Universidade Estadual de
Campinas, Instituto de Computação.

1. Sistemas de informação gerencial - Administração.
2. Engenharia de software auxiliada por computador -
Estudo de casos . 3. Software livre. 4. Apache
(Programa de computador). 5. Gestão de negócios.
- I. Liesenberg, Hans Kurt Edmund, 1953-.
- II. Universidade Estadual de Campinas. Instituto de
Computação. III. Título.

Informações para Biblioteca Digital

Título em inglês: Open source ERP localization case study

Palavras-chave em inglês:

Management information systems
Computer-aided software engineering - Case studies
Free software
Apache (Computer file)
Negotiorum gestio

Área de concentração: Ciência da computação

Titulação: Mestre em Ciência da Computação

Banca examinadora:

Hans Kurt Edmund Liesenberg [Orientador]
Regina Lúcia de Oliveira Moraes
Maria Cecília Calani Baranauskas

Data da defesa: 11-11-2011

Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO


Dissertação Defendida e Aprovada em 11 de novembro de 2011, pela
Banca examinadora composta pelos Professores Doutores:




Prof. Dr. Regina Lúcia de Oliveira Moraes
FT / UNICAMP



Profª. Drª. Maria Cecília Calani Baranauskas
IC / UNICAMP



Prof. Dr. Hans Kurt Edmund Liesenberg
IC / UNICAMP



Estudo de um caso de Localização de um Software ERP de Código Livre

Luís Alfredo Harriss Maranesi

Novembro de 2011

Banca Examinadora:

- Hans Liesenberg (Orientador)
- Fábio de Nogueira Lucena
Instituto de Informática - Universidade Federal de Goiás
- Cecília Baranauskas
Instituto de Computação - UNICAMP
- Regina Moraes
Faculdade de Tecnologia - UNICAMP
- Ariadne Maria B. Rizzoni Carvalho
Instituto de Computação - UNICAMP

Resumo

Soluções de Software de Gestão Empresarial (ERP - *Enterprise Resource Planning*) no Brasil são normalmente de código proprietário, caras de adquirir e implantar. No mercado brasileiro micro e pequenas empresas poderiam se beneficiar muito com a existência de soluções de ERP mais acessíveis. Uma possível solução seria o uso de programas de código livre para atender a essa demanda, tal como o projeto *Apache Open For Business*, um conjunto de aplicativos e um *framework* voltado para soluções de gestão empresarial. Neste estudo espera-se investigar a localização (processo de adaptação de um sistema para uma determinada cultura). Não apenas no que diz respeito à mera tradução dele, mas a aspectos legais, fiscais e contábeis, buscando aumentar sua usabilidade e viabilidade para empresários brasileiros.

Abstract

In Brazil, Enterprise Resource Planning (ERP) software are usually proprietary, expensive to acquire and deploy. Micro and small businesses could benefit greatly from the existence of more affordable ERP solutions. One possible solution would be to use open source software to meet this demand. Of relevance in this scenario there is the project Apache Open For Business, a suite of applications and a *framework* aimed at business management solutions. The main purpose of this study is to investigate the localization (i.e. adapting computer software to different cultural contexts) of this software, not only regarding to mere translation, but also the legal, tax and accounting aspects, seeking to increase its usability and feasibility for Brazilian businessmen.

Agradecimentos

Em primeiro lugar gostaria de agradecer minha família, que sempre me apoiou e orientou a buscar conhecimento e entender as questões que me causavam curiosidade.

Agradeço também à CAPES pelo apoio financeiro que possibilitou o desenvolvimento do presente estudo. E, principalmente, ao Professor Hans que me orientou sempre indicando o caminho a ser seguido quando as dúvidas surgiam.

Sumário

Resumo	vii
Abstract	ix
Agradecimentos	xi
1 Introdução	1
1.1 Objetivo	1
1.2 Motivação	1
1.3 Organização do Trabalho	2
2 Globalização, Internacionalização e Localização	5
2.1 Fatores a se levar em conta durante a internacionalização	8
2.1.1 Desacoplamento entre texto, código e layout	8
2.1.2 Formatos numéricos e de data	11
2.1.3 Imagens, símbolos e cores	12
2.1.4 Funcionalidades	13
2.1.5 Interfaces estendidas	14
2.2 Planejamento da localização de um software	15
2.3 Processos para a localização de software	16
2.4 Considerações finais	20
3 Usabilidade	21
3.1 Métodos de Inspeção de Usabilidade	23
3.2 Métodos de inspeção	23
3.3 Métodos de Teste	24
3.4 Considerações finais	26
4 Sistemas de Gestão Empresarial	27
4.1 Módulo de Vendas e Marketing	29
4.2 Gestão do relacionamento com o cliente	30

4.3	Gestão de materiais e da produção	31
4.4	Contabilidade e Financeiro	32
4.5	Recursos Humanos	33
4.6	Gestão da Cadeia de Suprimentos	34
4.7	Comércio Eletrônico	34
4.8	Business Intelligence	35
4.9	Considerações finais	35
5	Apache Open For Business	37
5.1	Arquitetura - Camada de Persistência de Dados	38
5.2	Arquitetura - Camada de lógica de negócios	40
5.3	Arquitetura - Camada de apresentação	42
5.4	Comparação com outros paradigmas arquiteturais	45
5.5	Processo recomendado para o desenvolvimento	47
5.6	Funções implementadas pelo OFBiz	50
5.7	Adequação a padrões	52
5.8	Considerações finais	53
6	Sistema Tributário Nacional	55
6.1	Obrigações Acessórias	58
6.1.1	Nota fiscal modelo 1 e 1A	59
6.1.2	Livros Fiscais	62
6.2	Sistema Público de Escrituração Digital - SPED	63
6.2.1	SPED - Contábil - ECD	65
6.2.2	SPED - Fiscal - EFD	65
6.2.3	NF-e - Ambiente Nacional	66
6.2.4	Certificação Digital utilizada no projeto SPED	68
6.3	Considerações finais	69
7	Análise para Localização	71
7.1	Análise da internacionalização do OFBiz	72
7.1.1	Análise do desacoplamento entre texto, código e layout	72
7.1.2	Análise de formatos numéricos e de data	73
7.1.3	Análise do uso de imagens, símbolos e cores	75
7.1.4	Análise das interfaces estendidas	76
7.2	Análise de Funcionalidades para Localização	77
7.3	Dimensões dos arquivos a serem traduzidos	80
7.4	Localização em outras línguas	81
7.5	Considerações finais	83

8	Localização de Componentes	85
8.1	Preparação do código	85
8.2	Elaboração do glossário	86
8.3	Tradução dos rótulos	87
8.4	Localização de funcionalidades	88
8.4.1	Prova de conceito para o projeto SPED	89
8.4.2	Adição de campos	90
8.4.3	Geração do XML da NF-e	94
8.4.4	Cálculo de impostos	95
8.5	Localização de Interfaces Estendidas	97
8.6	Considerações finais	98
9	Resultados e Conclusão	99
A	Heurísticas de Usabilidade	103
B	Dicionário de Termos Utilizados na localização	105
C	Dimensões para tradução do diretório applications do OFBiz	113
D	Dimensões para tradução do diretório SpecialPurpose do OFBiz	117
E	Dimensões para tradução do diretório Framework do OFBiz	121
F	Alteração necessária para adição de campos	125
G	Ferramentas Utilizadas	127
	Bibliografia	128

Lista de Tabelas

2.1	Expansão do texto em cada língua com inglês como referência (100%) [1]	10
6.1	Significado do primeiro algarismo no CFOP	61
6.2	Códigos de Situação Tributária - CST	61
6.3	Códigos de Situação Tributária - CST	62
7.1	Dimensões totais dos diretórios a serem traduzidos	81
7.2	Rótulos em cada locale	82
B.1	Dicionário construído para a localização	111
C.1	Arquivos e números de rótulos do módulo Accounting	113
C.2	Arquivos e números de rótulos do módulo commonext	113
C.3	Arquivos e números de rótulos do módulo content	114
C.4	Arquivos e números de rótulos do módulo de Recursos Humanos	114
C.5	Arquivos e números de rótulos do módulo Manufacturing	114
C.6	Arquivos e números de rótulos do módulo Marketing	114
C.7	Arquivos e números de rótulos do módulo Order	115
C.8	Arquivos e números de rótulos do módulo Party	115
C.9	Arquivos e números de rótulos do módulo Product	115
C.10	Arquivos e números de rótulos do módulo WorkEffort	115
D.1	Arquivos e números de rótulos usados no componente Assetmaint	117
D.2	Arquivos e números de rótulos usados no componente Ebay	117
D.3	Arquivos e números de rótulos usados no componente Ebaystore	117
D.4	Arquivos e números de rótulos usados no componente Ecommerce	118
D.5	Arquivos e números de rótulos usados no componente Googlebase	118
D.6	Arquivos e números de rótulos usados no componente Googlecheckout	118
D.7	Arquivos e números de rótulos usados no componente Myportal	118
D.8	Arquivos e números de rótulos usados no componente Oagis	118
D.9	Arquivos e números de rótulos usados no componente Ofbizwebsite	119
D.10	Arquivos e números de rótulos usados no componente POS	119

D.11	Arquivos e números de rótulos usados no componente Projectmgr	119
D.12	Arquivos e números de rótulos usados no componente Shark	119
D.13	Arquivos e números de rótulos usados no componente WebPOS	119
D.14	Arquivos e números de rótulos usados no componente Workflow	120
E.1	Arquivos e números de rótulos comuns a muitos componentes	121
E.2	Arquivos e números de rótulos usados no componente Webtools	121
E.3	Arquivos e números de rótulos usados no componente BI	122
E.4	Arquivos e números de rótulos usados no componente Base	122
E.5	Arquivos e números de rótulos usados no componente Birt	122
E.6	Arquivos e números de rótulos usados no componente Example	122
E.7	Arquivos e números de rótulos usados no componente Guiapp	122
E.8	Arquivos e números de rótulos usados no componente Minilang	123
E.9	Arquivos e números de rótulos usados no componente Security	123
E.10	Arquivos e números de rótulos usados no componente Service	123
E.11	Arquivos e números de rótulos usados no componente Webapp	123

Lista de Figuras

2.1	Diagrama da processo de criação de um produto global [1]	7
2.2	Diferentes representações gráficas de acordo com a cultura. (a) Representação típica no ocidente. (b) Representação típica da Tailândia [46] . .	12
2.3	Processo usual de um projeto de localização [2]	18
2.4	Processo proposto para incluir o desenvolvimento de funcionalidades relevantes à localização	19
3.1	ISO13407 - Processo de projeto centrado no usuário	22
4.1	Evolução dos ERPs desde os primeiros MRPs	27
4.2	Interação do módulo de vendas com outros módulos dos ERP [49]	30
5.1	À esquerda, herança múltipla que poderia ser utilizada para implementar o <i>Delegator Pattern</i> . À direita, a abordagem possível e mais adequada em JAVA.	39
5.2	Diagrama simplificado do funcionamento da camada de serviços	41
5.3	Diagrama simplificado do funcionamento da camada de apresentação, baseada no <i>Controller Servlet</i>	44
5.4	Amostra do arquivo <i>PartyEntityLabels.xml</i> pertencente ao componente <i>Party</i> com rótulos da interface para diversas línguas	45
5.5	Esboço da arquitetura típica de sistemas baseados em JAVA	46
5.6	Esboço da arquitetura típica de sistemas baseados em linguagens de script	46
5.7	Esboço da arquitetura do projeto Open For Business	47
5.8	Processo recomendado para o desenvolvimento de componentes para o OFBiz	48
6.1	Modelo 1 da Nota Fiscal	60
6.2	Integração que o SPED promete alcançar	64
6.3	Transmissão de dados do EFD para a o SPED. Depois de transmitidos para o SPED são distribuídos para as SEFAZ de cada estado pela chamada Rede de informações (RIS)	66

6.4	Interação entre contribuinte e SEFAZ e o uso do DANFE para a circulação de mercadorias	67
8.1	Tela de Edição de Produtos alterada, já com os campos para NFe	93

Capítulo 1

Introdução

O presente trabalho trata de um estudo de localização de um Sistema de Gestão Empresarial de código livre ao contexto brasileiro. Neste Capítulo são delineados, em linhas gerais, os principais aspectos deste trabalho, preparando assim as bases para a posterior apresentação mais aprofundada do trabalho completo.

1.1 Objetivo

O trabalho aqui apresentado consiste no estudo de localização do software Apache Open For Business (OFBiz) [9]. Pretendem-se investigar os métodos e práticas utilizados para internacionalizar e localizar um software de grande porte.

Entende-se pela internacionalização e localização o processo de preparar e adaptar um software para o uso em países ou regiões de língua ou cultura diversa daquela original.

Além disso, espera-se estudar alguns aspectos da internacionalização que vão além daqueles mais comumente citados nos trabalhos da área, como aspectos legais, fiscais e contábeis que afetam um software da natureza do OFBiz. A proposta é apurar, em particular, como fazer a integração com o Sistema Público de Escrituração Digital (SPED) [14] de forma elegante e eficiente, sem alterações muito radicais nas aplicações já existentes, visto serem críticos os aspectos tributários específicos para um sistema dessa natureza.

A par disso, espera-se contribuir com a comunidade do projeto com algum material já localizado. Ao menos os principais componentes, aqueles mais importantes e imprescindíveis para o uso básico do *software* pelos usuários, serão entregues traduzidos.

1.2 Motivação

Sistemas para gestão empresarial são fundamentais para empresas poderem crescer e aperfeiçoar suas atividades. No Brasil as soluções disponíveis no mercado são, em sua

maioria, se não todas, baseadas em código proprietário e normalmente implicam em altos gastos para sua aquisição e implantação. Ao se estudar a localização do Projeto *Apache Open For Business*, acredita-se que seja possível dar início a uma introdução de uma alternativa de código aberto aos produtos disponíveis no mercado brasileiro.

Segundo alguns especialistas na área de Sistemas de Gestão Empresarial (ou em inglês *Enterprise Resource Planning* - ERP) [21], a maioria desses sistemas são bem parecidos e com a mesma gama de funcionalidades. O que os diferencia são a usabilidade, o suporte, a facilidade de implantação, a estabilidade, a flexibilidade e evolução de novas versões.

Dessa maneira, a localização do OFBiz melhoraria a usabilidade do produto no mercado nacional. Já o suporte, a estabilidade, a flexibilidade e a evolução de novas versões poderiam se basear na comunidade que respalda o projeto *Apache Open For Business*. E por fim, o suporte e a facilidade de implantação poderiam ficar a cargo de empresas nacionais interessadas em prestar tais serviços.

1.3 Organização do Trabalho

Nas próximas seções, o Estudo da Localização do Apache Open For Business é apresentado, o mais fielmente possível, segundo os passos para sua realização na ordem em que foram dados. Partiu-se de um estudo buscando a compreensão dos conceitos envolvidos no âmbito da localização, passando pelos aspectos que mais poderiam interferir no domínio do sistema até chegar a provas de conceito que pudessem completar o estudo, proporcionando dados e impressões para uma avaliação.

No Capítulo 2 são apresentados os achados de uma revisão bibliográfica na área de internacionalização e localização. Logo em seguida, são apresentados os principais fatores apontados na literatura a serem levados em consideração nos processos de localização. Com base nisso, é apresentado um diagrama que retrata o que normalmente os profissionais da área consideram um processo adequado para a execução de tais projetos. Nessa etapa, já pensando-se em *software* de domínios específicos, é proposto um outro diagrama que inclui a localização de funcionalidades.

Como normalmente o foco principal de projetos de localização de um software é a interface de usuário, uma das etapas do processo proposto é a avaliação da qualidade do trabalho, é necessário ter uma base para compreender como isso pode ser feito. A melhor forma para tal é entender outro ramo da área de interfaces que trata da usabilidade. O Capítulo 3 define o que é usabilidade e aponta os principais métodos para se avaliar e testar a usabilidade de um software, assim como os aspectos necessários a serem considerados.

O Capítulo 4 parte para um domínio um pouco menos teórico que aqueles abordados nos capítulos 2 e 3, apresentando a classe de *software* chamada de Sistemas de Gestão Empresarial, também conhecidos pela sua sigla em inglês ERP. Nela é apresentada um pouco

da história desses sistemas assim como sua importância para a organização e administração de empresas. São apresentados também os módulos mais comuns, de um ponto de vista mais geral, daquelas funções normalmente presentes nas principais soluções disponíveis no mercado.

Partindo da base com o estudo dos ERP, o Capítulo 5 apresenta o projeto Apache Open For Business (OFBiz), o objeto particular do presente estudo. Nele, primeiramente, é descrita a história do sistema. Em seguida, suas características arquiteturais são apresentadas para que se possa compreender a comparação com outros paradigmas arquiteturais existentes. Com uma compreensão mais clara da arquitetura, o processo de desenvolvimento recomendado pela comunidade responsável pelo projeto é exposto, baseado em uma espécie de diagrama dos passos necessários. Este Capítulo é encerrado com uma lista resumida das funcionalidades implementadas no sistema.

Reconhecendo a necessidade de levar em conta as normas fiscais vigentes para o objeto de estudo, o Capítulo 6 aborda algumas características do sistema tributário nacional, considerando as principais obrigações dos contribuintes brasileiros. A discussão é mantida em um nível mais superficial do que seria por pessoas da área contábil ou tributária por meramente pretender entender quais aspectos da realidade brasileira podem interferir na localização do OFBiz. Mas para tal é de fundamental importância que o Sistema Público de Escrituração Digital (SPED) também seja apresentado. Trata-se de novas normas impostas pela Receita Federal que visam a modernização do fisco e têm uma influência direta em projetos de Sistemas de Gestão Empresarial.

Esses cinco capítulos formam a base necessária para o estudo específico propriamente dito. Assim, o Capítulo 7 apresenta uma análise do projeto OFBiz segundo todos os aspectos relevantes para a localização, apresentados no Capítulo 2. Também são analisadas as dimensões de todos os arquivos a serem traduzidos e o estágio da tradução para outras línguas.

Em seguida o Capítulo 8 descreve mais detalhadamente todas as etapas para o estudo da localização do OFBiz, desde a tradução dos componentes até o estudo da localização de funcionalidades, com foco no Sistema Público de Escrituração Digital. Completando a descrição das etapas de localização, o Capítulo 8.5 trata das interfaces estendidas, ou seja, os materiais de apoio ao usuário.

Por fim, o capítulo 9 finaliza o trabalho ao apresentar os resultados e tecer algumas conclusões.

Capítulo 2

Globalização, Internacionalização e Localização

Com o processo de globalização e o enfraquecimento de fronteiras entre países devido ao desenvolvimento dos meios de comunicação, cada vez mais há semelhanças entre problemas que diferentes culturas enfrentam. Dessa forma, faz-se necessário que o conhecimento e as soluções desenvolvidas em determinados locais possam ser reutilizadas e adotadas em outros com as devidas adaptações.

Para a sociedade como um todo, essa possibilidade significa uma distribuição mais rápida e igualitária de conhecimento e tecnologia. Para empresas de software, significa um maior alcance a novos mercados, maximizando as receitas. E, para empresas tradicionais, significa ter a capacidade de atuar em mais mercados sem perder controle sobre suas atividades, integrando e coordenando todas as suas filiais.

Para que isso seja possível, é necessário que, ao se iniciar qualquer projeto, seja levada em conta a questão da distribuição dele em outros contextos culturais. Muitas empresas, notadamente aquelas que atuam em âmbito internacional, levam isso muito a sério. Na realidade, já existe uma indústria ao redor dessas necessidades.

Empresas, que constituem essa indústria, oferecem os chamados serviços linguísticos. Vão desde assessoria na criação de estratégias globais para a distribuição de produtos, orientação de redatores técnicos na elaboração de documentos até a tradução propriamente dita desses documentos ou software. Dessa forma, auxiliam empresas de todos os ramos na atuação para além de suas fronteiras nacionais.

Esse mercado de serviços linguísticos compreende e se confunde um pouco com o ramo da computação no que tange as interfaces de usuário. Trata, em grande parte, da adaptação de software para que possam ser utilizados em outras culturas diferentes daquela em que foi criado.

Grosso modo, em um primeiro momento, pode-se dizer que essa adaptação é alcançada pela internacionalização e localização, duas técnicas utilizadas no desenvolvimento de soft-

ware a ser utilizado em escala global.

Internacionalização é a prática de isolar elementos específicos de uma determinada cultura do restante do programa para que se possa adaptar tais elementos a outras culturas - como textos, formatos de número e convenções de símbolos - sem exigir, idealmente, alterações e adaptações mais incisivas no código da aplicação [46].

Exemplos de técnicas de internacionalização incluem a separação de variáveis de texto do código do programa, preparação para o uso de diferentes moedas, diferentes formatos de números e datas, formatos de telefones e endereços, unidades de medidas, codificação de caracteres e direções diferentes de escrita, como acontece com línguas orientais.

Por sua vez, a localização é o complemento da internacionalização, inserindo um programa previamente internacionalizado em um novo contexto cultural como se tivesse sido produzido localmente [46]. Graças à localização, o produto pode ser utilizado em diversas partes do mundo e permite inclusive que pessoas de várias localidades possam trabalhar em conjunto.

Especialistas do meio acadêmico e do mercado afirmam que essas práticas devem ser levadas em conta desde o começo do projeto para que o produto tenha chances reais de sucesso internacional [27][1].

De fato, já foi demonstrado que projetos de software livre, que adotam tal postura, têm tido mais sucesso que os demais [51]. Alguns software com tal preocupação, como o servidor Http Apache, chegam a ocupar nichos muito maiores que seus concorrentes de código proprietário. Por outro lado, alguns projetos, apesar de resolverem problemas complexos de forma eficiente e elegante, não são bem aceitos pela simples falta de internacionalização e facilidades de localização.

Porém, no âmbito da indústria de serviços linguísticos, os profissionais consideram, além da internacionalização e localização, a globalização e tradução como passos constituintes da preparação de um software para o mercado global.

A globalização é um conceito mais amplo que envolve não apenas aspectos referentes ao desenvolvimento técnico mas, sim, às estratégias de marketing global do produto. Consiste na conceitualização do produto de forma que possa ser vendido em todos os lugares do mundo. Grandes empresas como Microsoft, Apple e Adobe levam essa prática muito a sério, e dessa forma, quase não encontram barreiras internacionais para seus produtos.

Já a tradução, está na outra ponta do processo de criação do produto, sendo ela uma tarefa básica. É de fato a simples passagem das palavras da língua original para aquela do local de destino. Apesar de parecer a mais simples de todas, é um engano menosprezá-la. Ela depende do domínio do software. É indicado que os responsáveis pela tradução tenham conhecimentos não só da língua original e da língua alvo, mas também noções linguísticas e do jargão da área que o software atende.

Assim, o processo de desenvolvimento de um produto para o mercado global é constituído das etapas de globalização, internacionalização, localização e tradução. Alguns

profissionais da área representam esse processo pelo uso de um diagrama em formato de alvo [1], em que a camada mais externa é a globalização e, a mais interna, a tradução como esboçada na Figura 2.1.



Figura 2.1: Diagrama da processo de criação de um produto global [1]

Em tal figura fica claro a dependência que há do foco na globalização dos projetos para que a internacionalização seja feita adequadamente, assim como a localização e, por fim, a tradução. Um projeto com a globalização em mente é pré-requisito para as demais etapas. Os círculos mais externos são pré-requisitos para os internos.

No caso da localização de um software, todas essas etapas, se levadas em conta desde o início do desenvolvimento, podem facilitar a distribuição em qualquer língua desejada. Notadamente a etapa de internacionalização, quanto mais bem feita mais facilita os processos de localização e tradução. E isso não é verdade apenas no que tange à programação, mas, sim, também para a documentação, a chamada interface estendida do sistema.

No entanto, há na literatura também algumas argumentações de que todo esse processo de globalização não se limita apenas à interface do *software* mas, em alguns casos, também ao seu núcleo e algumas funcionalidades. É difícil encontrar autores que discutam com mais profundidade este tema, mas há alguns poucos bons exemplos.

O primeiro deles é de Gregory E. Kersten que, juntamente com outros pesquisadores, argumenta que o funcionamento dos *software* pode espelhar a cultura de seus desenvolvedores e usuários [34][33]. Por isso, a localização pode ter que incluir outros passos e outras preocupações além da simples tradução da interface. Segundo ele é necessário perceber que os sistemas de informação na realidade encapsulam inúmeros aspectos da cultura de onde são desenvolvidos, como o conhecimento, métodos e a filosofia de seu povo.

Outro exemplo vem de José Leopoldo Nhampossa que, em seu artigo [39], relata o

desafio de localizar um Sistema de Informações de Saúde (*Health Informational Systems - HIS*), de código livre, criado na África do Sul para o uso em Moçambique. O autor ressalta, em primeiro lugar, a diferença existente entre *software* de uso em domínios específicos de negócios e outros de uso geral. Segundo ele, o primeiro tipo, que é justamente o caso do HIS, é mais difícil de localizar por incluir aspectos relacionados à cultura e à organização dos países onde é usado.

Depois segue sua argumentação mostrando várias dificuldades que surgiram durante o processo de localização, das quais a mais representativa do problema da localização em domínios específicos é a questão da diferença da organização do sistema de saúde entre os dois países.

Mas há ainda outros tipos de sistemas mais específicos que podem significar desafios muito maiores para a localização. Trata-se do ramo da localização de *video games*, que não deixa de ser um tipo de *software* mas, diferentemente da maioria, seu principal objetivo é entreter o usuário. Há autores [37] que argumentam que essa área, por seu foco em entretenimento, exige dos envolvidos, além das habilidades normais, uma boa dose de criatividade. Algo imprescindível para que um jogo desenvolvido no Japão, por exemplo, possa ser jogado em países ocidentais com o mesmo grau de diversão.

2.1 Fatores a se levar em conta durante a internacionalização

Das etapas do desenvolvimento de um sistema, a mais crítica para a localização do software é a internacionalização. Segundo acadêmicos e profissionais da indústria de localização, os principais pontos a serem levados em conta são os apresentados nas subseções seguintes [46][1].

2.1.1 Desacoplamento entre texto, código e layout

Para facilitar a localização e a tradução da interface do software, todas as frases, sejam elas rótulos ou qualquer tipo de mensagem, devem ser desacopladas do código do programa durante o desenvolvimento. Uma das técnicas mais comuns, atualmente, é separá-las em arquivos que possuem apenas o texto da interface e, então, o próprio programa determinar através de chaves e de acordo com sua configuração, de qual arquivo obter o texto apropriado.

A vantagem dessa técnica é poder localizar os elementos textuais de um programa para quantos idiomas forem necessários, bastando apenas a adição dos arquivos correspondentes a cada língua.

Cada sistema operacional oferece formas diferentes de se fazer esse desacoplamento.

Programadores para Windows podem usar as “*Dynamic-link libraries*” (DLL ou, em português, bibliotecas de ligação dinâmica). Programadores de ambiente Linux podem usar, por exemplo, a ferramenta gettext, que prepara o código para ter suas cadeias de caracteres traduzidas. No ambiente desenvolvimento Java, usa-se normalmente os chamados “*Resources Bundles*” [43], mais precisamente do tipo *Property*, que armazena as cadeias de caracteres localizadas em arquivos de extensão “properties”.

Alguns desenvolvedores optam ainda por separar todas as cadeias de caracteres em arquivos xml consultados pelo programa. A vantagem dessa abordagem é a possibilidade de aglutinar vários idiomas em menos arquivos, sendo distinguidos pelas *tags* do arquivo *xml*.

Mas não basta separar todas cadeias de caracteres da lógica do programa, existem outras boas práticas que devem ser levadas em consideração para facilitar a tradução. Deve-se evitar a concatenação de frases no código. Algumas vezes programadores compõem uma frase com dois ou mais trechos de texto. Apesar de diminuir o número de cadeias de caracteres necessário, a mesma construção pode não fazer sentido em todas as línguas, causando problemas no momento da tradução.

Ainda com relação ao número de cadeias de caracteres, é considerada uma boa prática tentar reutilizá-las consistentemente durante o contexto do programa. Assim, os arquivos de localização serão menores e a tradução poderá ser feita em menos tempo.

Quanto ao layout da interface, deve-se levar em conta que alguns idiomas orientais são escritos da direita para esquerda. Dessa forma, exigem que programadores deixem portas abertas para representação do texto em todos os sentidos possíveis.

Outro fator bastante relevante é a diferença no conjunto de caracteres que cada língua utiliza. São necessários muito menos caracteres para representar uma frase na língua chinesa do que na inglesa, por exemplo (Isto porque, apesar de haver um número muito maior de ideogramas do que letras, um ideograma pode representar por si só uma palavra ou frase inteira através de seu significado).

Antes da invenção do Unicode isso era um dos principais problemas enfrentados durante a internacionalização e localização. Nenhum dos conjuntos de caracteres disponíveis acomodava todos os símbolos possíveis de todas as línguas, sendo necessário encontrar a representação correta para cada tradução.

Além disso, o mesmo texto em várias línguas pode ocupar espaços diferentes pois o número de palavras e caracteres necessários varia muito entre elas. De fato alguns pesquisadores já promoveram um estudo aproximado do comprimento de sentenças em várias línguas, tendo-se como referência o inglês [47]. Foi usada parte da Declaração Universal dos Direitos Humanos para se ter uma aproximação e o resultado está resumido na Tabela 2.1.

Pela Tabela 2.1 uma interface em inglês precisaria permitir uma expansão da ordem de 28% para poder ser traduzida para o holandês. Por outro lado deveria permitir uma

Expansão/contração de texto para cada idioma	
Idioma	diferença (%)
Árabe	104%
Chinês	61%
Tcheco	117%
Holandês	128%
Inglês	100%
Esperanto	92%
Persa	104%
Finlandês	103%
Francês	111%
Alemão	108%
Grego	128%
Hebraico	83%
Hindi	83%
Húngaro	113%
Italiano	109%
Japonês	115%
Coreano	123%
Português	110%
Russo	115%
Espanhol	117%
Suaíle	88%
Sueco	95%

Tabela 2.1: Expansão do texto em cada língua com inglês como referência (100%) [1]

redução de 39% quando traduzida para o chinês.

2.1.2 Formatos numéricos e de data

Praticamente no mundo inteiro são adotados os algarismos arábicos como padrão, porém o formato dos números pode variar em cada cultura. O primeiro exemplo disso são os separadores decimais. Em países como Inglaterra e Estados Unidos o ponto é usado como separador decimal e vírgula para separador dos milhares. Já em países como Brasil, Portugal e Espanha, por exemplo, o costume é contrário. Usa-se a vírgula para as casas decimais e o ponto para os milhares.

O número mil unidades e cinquenta décimos, por exemplo, seria escrito 1.000,50 em Portugal e 1,000.50 nos Estados Unidos. É necessário que a internacionalização das rotinas que aceitam e validam números no programa seja bem feita, caso contrário operações que dependam de números fornecidos pelo usuário poderão ser inconsistentes em diferentes países do mundo.

A representação de datas também é algo mais importante do que pode parecer. No Brasil, a ordem usada para representar datas consiste no dia, seguido pelo mês e por fim o ano. Já nos Estados Unidos, o mais comum é usar a sequência mês, dia, ano. Em outros países são usadas outras sequências e alguns usuários, independentemente de sua origem, ainda costumam alternar entre o uso de números e abreviações para representar os meses do ano.

Assim como a representação de datas, a representação das horas também difere entre diferentes culturas. Algumas optam por representar as horas do dia entre 0 e 24 horas, enquanto outras usam a representação entre 0 e 12 horas.

Novamente, para que haja uma consistência no funcionamento do software independentemente de onde for utilizado, é necessário que todas essas possibilidades sejam devidamente tratadas e validadas por programadores.

Por fim, outro aspecto bastante relevante para a internacionalização é que algumas culturas possuem calendários diferentes do calendário ocidental. Um exemplo é o calendário judaico que baseia-se em ciclos lunares e possui uma contagem totalmente diferente. O ano de 2010 do calendário ocidental corresponde ao ano 5770 do calendário hebraico.

Além do calendário judaico, há os calendários islâmico, chinês, indiano entre outros. Se a interface permitir o uso de tais calendários, uma possível abordagem pode ser implementar rotinas que façam uma conversão entre calendários para manter a coerência de resultados entre diferentes culturas para as quais o software possa ser localizado. O importante é que essas diferenças sejam consideradas, a fim de se manter os resultados coerentes.

2.1.3 Imagens, símbolos e cores

Boa parte das interfaces modernas se baseiam em ícones ou figuras para passar ao usuário alguns significados para facilitar o uso do programa. A ideia é boa e tem se provado eficaz desde o surgimento das interfaces gráficas, porém a localização deve levar em consideração que, para diferentes culturas, as mesmas imagens podem possuir significados diferentes e, mesmo que possuam significado semelhante, podem não transmitir o mesmo conceito para quem as vê.

Patricia Russo e Stephen Boor citam como exemplo disso o caso do ícone da “lixeira” em interfaces gráficas [46]. Afirmam que na Tailândia o símbolo de lixeira é diferente daquele que a maioria dos ocidentais estão acostumados, aquela lata de lixo cheia de frisos com uma tampa. Na cultura tailandesa representam-na por um cesto daqueles trançados com algumas moscas em cima, assim como demonstra a Figura 2.2. Por essa diferença cultural, se um sistema fosse localizado para ser usado na Tailândia talvez os usuários se confundissem até associar o significado do novo símbolo.

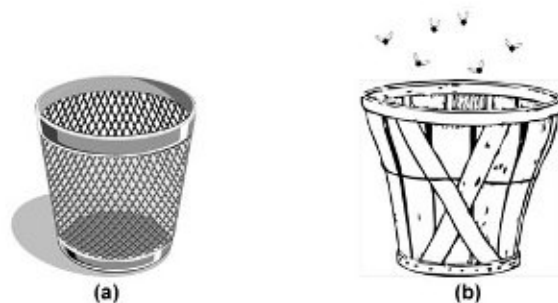


Figura 2.2: Diferentes representações gráficas de acordo com a cultura. **(a)** Representação típica no ocidente. **(b)** Representação típica da Tailândia [46]

Algo que talvez seja menos corriqueiro na localização de um software, mas ainda está relacionado ao assunto de interpretação cultural de imagens e símbolos, é a questão de fotos ou ilustrações que podem ser ofensivas ou não aceitáveis em algumas culturas. Um exemplo clássico, utilizado por alguns autores, refere-se a diferenças culturais e religiosas entre povos ocidentais e orientais, mais precisamente islâmicos.

Se, de alguma forma o software fizer uso de alguma ilustração que mostre uma mulher ocidental, trajada de acordo com os hábitos ocidentais, isso poderia vir a ofender um usuário islâmico. Algo que poderia ser ofensivo seria uma imagem de uma mulher com os cabelos ou as pernas à mostra.

Além de imagens, há as cores como fator relevante. Nem sempre as cores passam o mesmo significado para diferentes culturas. Outro exemplo clássico é o significado da cor vermelha para os ocidentais e para os chineses. No ocidente possui significado de perigo já na China traz a ideia de felicidade.

Imagine que a cor vermelha fosse usada em software que avisasse a um controlador, digamos de uma usina nuclear, a respeito do perigo de uma operação. No ocidente não haveria grandes problemas para os controladores seguirem o aviso. Por outro lado, na China, isso poderia até causar um acidente. Com certeza esse exemplo é um tanto exagerado, mas a aceitação no mercado de um produto pode depender de questões como essa.

Por isso, se o software fizer uso de cores ou qualquer tipo de imagem, ilustração ou ícone, é necessário que os responsáveis pela localização investiguem se existe alguma possibilidade de um mal entendido acontecer. E se for o caso, é preciso localizar além do texto também esses elementos. Dessa forma, é imprescindível que a internacionalização leve em conta a possibilidade de isso ocorrer e crie possibilidades para a fácil alteração desses elementos.

Outro aspecto a se pensar é a necessidade que pode surgir para se localizar também os textos que acompanham figuras, diagramas e gráficos. As próprias figuras devem ser armazenadas em algum formato que separe camadas de texto de figuras para, assim, facilitar a localização.

2.1.4 Funcionalidades

Da mesma forma que alguns itens do software, como imagens e ícones, podem não servir para todas culturas, há a possibilidade de algumas das funcionalidades do programa não serem úteis para alguns públicos alvo e até faltarem funcionalidades que atendam demandas específicas.

No presente trabalho há um exemplo típico disso. Os sistemas de gestão empresarial desenvolvidos em outros países, quando trazidos para o Brasil, necessitam passar por adaptações que vão muito além da simples tradução. Até grandes produtos de software proprietários, líderes de mercado, sofrem com esse problema e, por isso, abrem portas para desenvolvedores locais produzirem módulos complementares. São os chamados módulos satélites e o problema mais comum, que precisam resolver, é fazer com que os sistemas de gestão empresarial estrangeiros atendam às exigências legais e fiscais do governo brasileiro.

Dessa forma, durante o desenvolvimento, a questão da globalização deve ser levada em conta pensando-se que, para poder distribuir o produto globalmente, ele deve permitir a localização mesmo em aspectos de implementação. Os desenvolvedores devem ter consciência de que suas soluções podem não ser universais e devem facilitar a complementação de seu trabalho por outros.

Com certeza há inúmeras soluções de arquitetura e engenharia de software para isso, desde a organização do desenvolvimento em módulos até a criação de API ou *frameworks*. Há autores que argumentam inclusive no sentido de que o desenvolvimento com vistas a um mercado global deve fazer uso de uma arquitetura o mais modular possível, de forma que aspectos muito dependentes da cultura do usuário possam ser levados em conta por módulos ou componentes extras [34][33].

O fato é que, apesar de não serem tão evidentes quanto os aspectos tais como texto e imagens, aspectos referentes a funcionalidades do software podem ter um grande impacto na etapa de localização.

2.1.5 Interfaces estendidas

A produção de documentos técnicos, como manuais e materiais para suporte de clientes e usuários, também devem considerar as estratégias de globalização. Os responsáveis pela elaboração dessas interfaces estendidas devem igualmente levar em conta diretrizes de internacionalização. Isso garante que a localização e tradução do material possa ser feita em tempo menor e pelo menor custo.

Empresas e profissionais da área [1] apontam vários elementos na escrita que são cruciais para a etapa de localização e tradução, indo desde o espaço em branco que se deve deixar em cada página até as fontes utilizadas para formatar o texto. Os principais elementos são listados a seguir:

- **O layout do texto deve acomodar expansões e contrações** - Assim, como os desenvolvedores das interfaces devem levar em conta as diferenças de espaço ocupado por diferentes idiomas, os escritores de documentação devem ter em mente que, quando seus textos forem traduzidos, provavelmente vão ocupar um espaço diferente do original. O problema disso fica evidente quando se considera o trabalho dos profissionais de suporte ao cliente, que muitas vezes necessitam fazer referência a páginas específicas dos manuais para orientar pessoas no uso do produto. Se o texto for elaborado deixando de 20 a 30% de espaço para acomodar eventuais expansões, é possível, inclusive, facilitar o treinamento do pessoal de suporte;
- **Separação entre textos e gráficos** - Se a documentação incluir imagens, gráficos ou ilustrações, provavelmente possuirá legendas ou menções a tais elementos. Essas legendas ou menções devem ser elaboradas de forma mais desacoplada possível das imagens. Se isso não ocorrer, os responsáveis pela tradução terão que, além de traduzir o texto, editar figuras. Isso faz com que o processo seja mais longo e se torne mais caro. Além disso, muitos manuais de software incluem impressões da tela do programa e elas devem ser desacopladas do texto para que possam ser facilmente ser trocadas pelas impressões da interface já localizada;
- **Escolha adequada de fontes** - Nem todos os pacotes de fontes possuem caracteres adequados para representar todas as línguas. Quando considerados os idiomas asiáticos, isso fica evidente, mas mesmo entre línguas ocidentais pode haver esse tipo de problema. Algumas possuem caracteres acentuados e outras não. Por isso, deve-se escolher fontes que acomodem o maior número de idiomas possível. Mais uma vez,

trata-se de uma diretriz para minimizar o trabalho dos responsáveis pela localização e manter uma coerência entre as versões em todos os idiomas;

- **Criação de um glossário direcionado aos tradutores** - Uma técnica utilizada para se baratear o processo de tradução é a criação de glossários pelos redatores técnicos. Tendo em mente que eles são as pessoas que mais entendem do assunto sobre o qual estão escrevendo, se elaborarem também um glossário vão eliminar boa parte do trabalho dos tradutores em pesquisar e entender o jargão usado na documentação;
- **Criação de uma lista de acrônimos** - A par do glossário, deve ser elaborada uma lista de todos os acrônimos utilizados no texto. Isso é de extrema importância pois a tradução de acrônimos não é tão simples quanto a de palavras isoladas.

2.2 Planejamento da localização de um software

Para se localizar um software, é necessário que se faça um planejamento adequado. Com base no trabalho de internacionalização, que deve ter sido efetuado previamente, define-se o que é necessário ser localizado e para qual público esse trabalho será direcionado. Basicamente, no que diz respeito à globalização e internacionalização, tudo depende da qualidade do desenvolvimento do software e do público para o qual se destina.

Os especialistas do mercado recomendam que os seguintes pontos sejam levados em consideração [1]:

- **Componentes a serem localizados** - Nem sempre todos os componentes de um software necessitam ser localizados ou, então, não necessitam ser localizados em um primeiro momento. Boa parte dessa decisão depende do público para o qual o software será distribuído;
- **Mercados e línguas alvo** - É preciso considerar para quais mercados a localização será focada, o público que vai usar o sistema e a linguagem ou o dialeto para o qual será traduzido. Em projetos de grandes dimensões é preciso levar em conta, inclusive, as diferenças que há entre a mesma língua falada em diversos locais, como acontece entre Brasil e Portugal, França e Canadá ou Estados Unidos e Reino Unido;
- **Requisitos legais, regulatórios e comerciais** - De acordo com o mercado alvo do sistema é necessário levar em conta requisitos legais impostos pelo governo ou agências regulatórias do local onde esse mercado se encontra;
- **O prazo a ser cumprido** - Dependendo do prazo imposto para o projeto, algumas decisões acerca dos componentes a serem localizados, ou partes deles, devem ser tomadas a tempo;

- **Frequência de atualizações** - Dependendo do tipo de sistema, é necessário levar em conta que suas atualizações também deverão ser localizadas;
- **Problemas conhecidos no sistema** - Problemas ou defeitos que serão corrigidos em próximas versões devem ser conhecidos e, caso influenciem em algum aspecto da localização, precisam ser considerados, principalmente no momento da localização de interfaces estendidas;
- **Compatibilidade do sistema com a língua alvo** - É necessário verificar como a etapa de internacionalização foi feita e se o sistema é compatível com a língua para a qual se quer localizá-lo. Caso essa etapa não tenha garantido tal compatibilidade, talvez sejam necessários procedimentos mais incisivos no código do programa;
- **Planejamento de testes** - Depois de localizado e traduzido, o sistema deve ser testado, tanto do ponto de vista funcional quanto de usabilidade para os usuários alvo. É imprescindível que seja garantido o funcionamento e que a tradução seja adequada quanto ao jargão utilizado pelo usuários.

2.3 Processos para a localização de software

Ao se analisar qualquer indústria, ficará clara a importância de processos para empresas do ramo. Processos permitem que se alcancem padrões de qualidade, custos sejam reduzidos, prazos cumpridos e resultados reproduzíveis.

Na indústria de desenvolvimento de software existem vários processos para garantir a qualidade dos serviços prestados pelas empresas desenvolvedoras. Alguns exemplos são os modelos de desenvolvimento em cascata, espiral, iterativo incremental, dentre outros. Todos eles descrevem etapas que devem ser seguidas desde a obtenção das especificações do cliente até as etapas de desenvolvimento, testes e controle de qualidade.

No ramo da localização não é diferente. Apesar de não ser comum encontrar em textos acadêmicos a descrição ou menção a processos a serem seguidos para a localização de software, empresas que prestam esse tipo de serviço descrevem os processos, que adotam a fim de garantir um padrão de qualidade para seus clientes. Consultando os processos de algumas empresas [1][28][2], é possível discriminar as seguintes etapas:

- **Planejamento** - Assim como descrito no Capítulo 2.2, o planejamento inclui a avaliação da qualidade do software e seu grau de internacionalização, o mercado e a língua para os quais será localizado, os requisitos legais e comerciais que podem impactar a localização, estudar os problemas já conhecidos do sistema e o planejamento dos testes que avaliarão a qualidade do resultado;

- **Determinação dos custos e prazos** - Com base nos aspectos levantados pelo planejamento, os custos do projeto e os prazos a serem cumpridos são estimados, de acordo com as necessidades do cliente e as possibilidades da empresa de executar o trabalho;
- **Preparação do código** - Nesta etapa define-se qual versão do sistema utilizar para a localização;
- **Elaboração de um glossário** - Este trabalho deve ser feito de preferência por linguistas e pessoas que conheçam o domínio do qual o software faz parte. Consiste do levantamento dos principais termos do programa, no estudo do jargão utilizado e na tradução dos termos contidos no glossário a ser adotado no processo de localização. Com esse passo garante-se que a tradução será feita com qualidade e consistência entre todas as porções do sistema;
- **Tradução do software** - Nesta etapa é feita, de fato, a tradução da interface do software, usando-se para tal o glossário criado na etapa anterior;
- **Adaptação da interface** - Dependendo da qualidade da internacionalização do software pode ser necessário fazer adaptações na interface. Se ocorrer uma expansão ou contração que ocorrerem em textos, por exemplo, pode ser necessário ajustar os tamanhos dos elementos que compõem a interface. Questões como ícones ou imagens que necessitem ser localizadas serão consideradas nesta etapa;
- **Localização de interfaces estendidas** - Tradução de manuais, documentações online e qualquer outro tipo de documento ou mídia de apoio a usuários;
- **Controle de qualidade** - Etapa em que linguistas e, possivelmente, profissionais que conhecem o domínio do sistema revisam todos os materiais traduzidos para determinar se há a necessidade ou não de correções;
- **Testes de Funcionalidade** - Etapa na qual se verifica se alguns dos passos da localização afetaram as funcionalidades do software;
- **Aprovação por parte do cliente** - Os resultados são repassados pelo cliente e, se for o caso, por equipes locais do mercado alvo da localização.

A Figura 2.3 mostra um fluxograma de um processo que se assemelha aos normalmente utilizados por empresas que oferecem serviços de localização.

Porém, apesar de a literatura disponível e os profissionais da área apontarem como passo da localização a análise de funcionalidades a serem retiradas ou adicionadas, é difícil encontrar referências que sugiram diretrizes ou soluções para essa questão. Por isso, ao

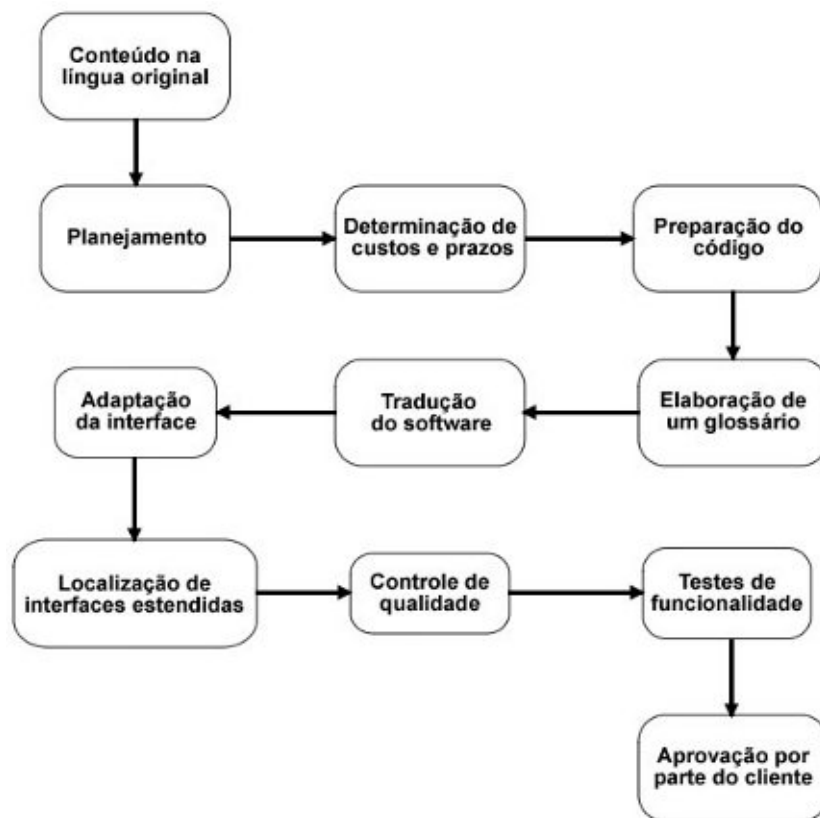


Figura 2.3: Processo usual de um projeto de localização [2]

se deparar com um projeto em que haja a necessidade de se remover ou adicionar uma funcionalidade talvez seja necessário usar uma variação desse processo.

É necessário que hajam programadores disponíveis para estudar a arquitetura do software e propor uma solução que cause o menor impacto possível no funcionamento das demais funcionalidades. Essa solução deve ser integrada ao restante do programa e, então, todas as funcionalidades do programa, tanto as originais quanto as adicionais, devem ser testadas para garantir que o impacto da localização tenha sido exclusivamente de adição, substituição ou remoção de funcionalidades e não adição de defeitos.

Pensando nas possíveis adições ou remoções de funcionalidades ao software original, para torná-lo de fato utilizável no ambiente de destino, o processo sugerido na Figura 2.3 poderia ser alterado para incluir uma etapa de desenvolvimento e outra de integração. Porém, para tal, as fases de planejamento, determinação de custos e prazos, elaboração de glossário, controle de qualidade e testes de funcionalidade devem ser executadas levando-se em conta a adição ou remoção de funcionalidades.

O planejamento e a determinação de custos e prazos devem considerar o tempo e os valores necessários para alocar programadores, assim como a integração de novas funciona-

lidades às originais ou adequação das últimas. A elaboração do glossário deve incluir novos vocabulários relacionados às alterações efetuadas. Por sua vez, o controle de qualidade e testes de funcionalidade se tornam mais complexos por incluírem preocupações com o novo código integrado ao projeto original.

Essas duas etapas podem, em princípio, ser executadas em paralelo com algumas das outras fases por se tratarem de trabalho mais voltado a desenvolvedores. Dessa forma, uma possível abordagem seria executá-las após a etapa de elaboração de glossário e antes da adaptação da interface. Portanto, ao mesmo tempo em que a tradução do software é feita, uma outra equipe pode cuidar do desenvolvimento das funcionalidades específicas para o software localizado. Uma vez consolidadas a tradução e o desenvolvimento, parte-se para a adaptação da interface. A Figura 2.4 ilustra essa proposta.



Figura 2.4: Processo proposto para incluir o desenvolvimento de funcionalidades relevantes à localização

2.4 Considerações finais

Neste capítulo foram apresentados e relacionados os conceitos de internacionalização e localização. Foi destacada a importância que ambos possuem no sucesso de um software ao redor do mundo.

Para que um projeto tenha sucesso, do ponto de vista de distribuição e adoção, é necessário que seus projetistas foquem sempre nos aspectos apresentados como básicos para o processo de internacionalização.

É interessante também observar a relação que esses conceitos possuem com a administração de grandes empresas. Grandes corporações ao redor do mundo, aquelas que conseguem transpor barreiras culturais e geográficas, mesmo que não diretamente ligadas ao ramo da computação, normalmente adotam técnicas consoantes com os conceitos apresentados. Graças a essas práticas, podem distribuir seus produtos com mais facilidade e conquistar consumidores ao redor do mundo.

Uma vez que o foco do presente trabalho é na interface do usuário, o próximo capítulo procura discutir o conceito de usabilidade de um software. Conceito este, intimamente ligado à localização, uma vez que esse processo não deve impactar na usabilidade do sistema para usuários de diferentes cenários culturais, mantendo sua interface sempre adequada e de fácil uso.

Capítulo 3

Usabilidade

Segundo Jakob Nielsen, usabilidade é um atributo de qualidade que determina quão fácil de usar uma interface é [50][42]. Ainda, segundo Nielsen, definir usabilidade depende dos seguintes aspectos:

- **Facilidade de Aprender** (*Learnability*), referindo-se à facilidade que os usuários têm para aprender e adaptar-se ao uso do software;
- **Eficiência** (*Efficiency*), que é avaliada levando-se em conta quanto tempo os usuários levam para executar tarefas, uma vez que tenham aprendido a usar o software;
- **Facilidade de memorização** (*Memorability*), que é avaliada levando-se em conta o tempo que um usuário afastado do sistema, por um período mais longo, leva para se readaptar e voltar a usar o software de forma eficiente;
- **Número de erros** (*Errors*), que mede quantos erros os usuários cometem, quão graves eles são e quão facilmente os usuários se recuperam deles;
- **Satisfação** (*Satisfaction*), que avalia a satisfação do usuário ao usar o sistema.

O mesmo autor define uma série de dez heurísticas, ou em outras palavras diretrizes[41], que podem ser utilizadas para o desenvolvimento de interfaces com boa usabilidade. Todas elas são reproduzidas no Apêndice A para que possam ser referenciadas em outras partes do presente trabalho.

Investir em usabilidade, segundo especialistas e profissionais da área, deve ser algo a se pensar desde o início de um projeto. Dessa forma, as chances de o software obter um grande número de usuários aumenta consideravelmente. Segundo a Associação de Profissionais de Usabilidade (UPA - *Usability Professionals' Association*), o investimento em usabilidade pode significar as seguintes vantagens para empresas [29]:

- Produtividade aumentada;

- Custos com suporte e treinamento reduzidos;
- Vendas e lucros aumentados;
- Tempo e custos de desenvolvimento reduzidos;
- Custos de manutenção reduzidos;
- Satisfação do cliente aumentada.

Para que o investimento seja eficiente e presente desde o início de um projeto, foi proposto um padrão internacional, o ISO 13407: *Human-centered design process*, ou em português, Processo de projeto centrado no usuário. O diagrama da Figura 3.1 ilustra o processo proposto pela ISO 13407.

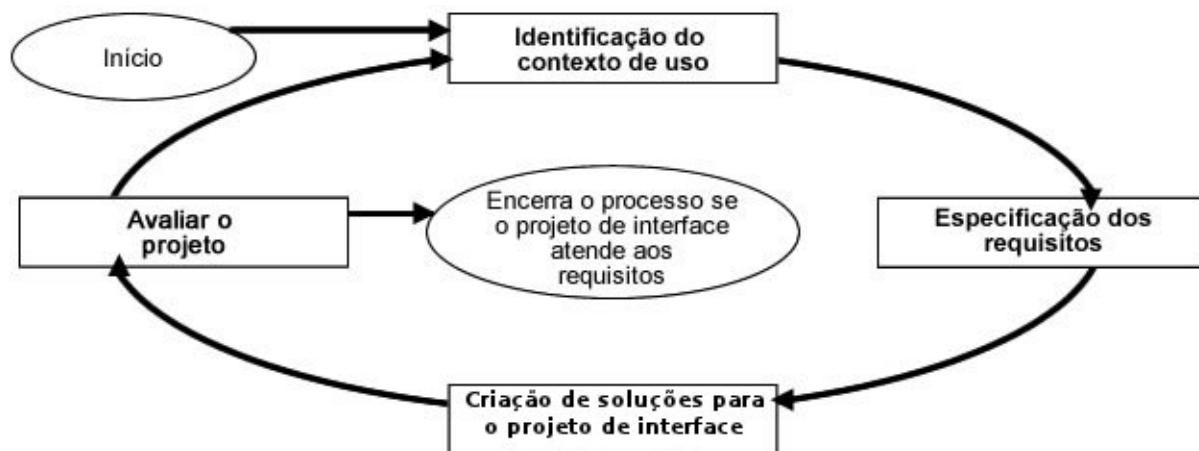


Figura 3.1: ISO13407 - Processo de projeto centrado no usuário

O processo é dividido em quatro estágios, executando um ciclo iterativo até que o produto alcance as metas de usabilidade desejadas [30]:

- Identificação do contexto de uso - consiste em caracterizar o usuário do sistema, a finalidade para qual o usará e sob quais condições;
- Especificação dos requisitos - consiste em identificar as funcionalidades que o software deve possuir;
- Criação de soluções para o projeto da interface - que consiste em traduzir uma concepção em um projeto concreto;
- Avaliar o projeto de interface - que consiste em utilizar métodos de inspeção aplicados por especialistas e teste de usabilidade com representantes do público alvo para avaliar se a usabilidade está de acordo com as metas do projeto.

Esta última fase é considerada a mais importante de todas. Para executá-la podem ser usados vários métodos de inspeção e teste, sendo necessários, em testes de usabilidade, a participação de usuários.

3.1 Métodos de Inspeção de Usabilidade

A fim de desenvolver software com melhor usabilidade são adotadas técnicas para garantir a qualidade do produto final. Uma delas é a avaliação da usabilidade, utilizada na quarta fase do processo proposto pela ISO 13407. Alguns especialistas dividem essas técnicas em dois grupos: Métodos de teste, que contam com a participação de usuários, e métodos de inspeção, que podem ser executados pelos próprios desenvolvedores ou especialistas em projetos de interfaces.

É importante ressaltar que a maioria dos métodos de avaliação de usabilidade não dependem do software pronto, podendo ser executados com protótipos ou versões ainda incompletas do sistema. Na realidade, a ideia do desenvolvimento centrado no usuário é justamente levar em conta as dificuldades e problemas dos usuários durante todo o processo de desenvolvimento.

3.2 Métodos de inspeção

Métodos de inspeção são normalmente executados por especialistas da área de usabilidade que avaliam características da interface segundo técnicas específicas [40][25]. Em geral, a literatura aponta os métodos de inspeção como sendo vantajosos por poderem ser usados desde o início do desenvolvimento e por aplicarem heurísticas e modelos de interação já reconhecidos e aceitos na engenharia de usabilidade. Por outro lado, são considerados falhos principalmente por não levarem diretamente em conta as dificuldades e opiniões de usuários reais.

Entre os métodos de inspeção mais citados na literatura encontram-se: a avaliação de heurísticas, o percurso cognitivo e a análise de ações.

A avaliação de heurística (*Heuristic Evaluation*) envolve quatro a cinco especialistas que avaliam se aspectos de ordem estrutural de um projeto de interface de usuário, como menus, diálogos e rótulos, seguem os padrões (ou heurísticas) recorrentes em projetos com alto grau de usabilidade [40]. Normalmente, os especialistas envolvidos na avaliação são mantidos sem poder se comunicar durante o processo de identificação de falhas de usabilidade para, assim, procurar garantir que as opiniões não interfiram umas sobre as outras.

O percurso cognitivo (*Cognitive Walkthrough*), por sua vez, baseia-se na análise detalhada de execução de tarefas representativas supostamente executadas pelos futuros

usuários [40], com o intuito de verificar possíveis falhas de usabilidade nos fluxos de execução de tais tarefas.

Já a análise de ações (*Action analysis*) é usada quando o desempenho é um fator crítico e dividida em duas modalidades. A primeira, mais exata e detalhada, é chamada de formal pelos autores [35] e consiste em descrever todas as subtarefas que formam tarefas típicas de usuários executadas com o apoio da interface. Essa descrição é bastante detalhada, sendo frequentemente considerada “em nível de aperto de teclas” (do inglês *keystroke-level*).

A ideia ao descrever tão detalhadamente as tarefas que compõem o uso da interface, é poder usar tabelas com médias de tempo para cada ação do usuário a fim de estimar o tempo médio para executar cada tarefa. De acordo com alguns autores é possível estimar esse tempo com uma margem de erro de até 20%.

A segunda modalidade é chamada em inglês de *Back-of-the-envelope* (verso do envelope), que é uma analogia a fazer um rascunho de cálculos aproximados, ou seja, algo mais simples que a análise formal de ações. Nessa modalidade, a descrição das tarefas é bem menos detalhada.

Simplificadamente, ambas modalidades buscam avaliar a facilidade e possíveis fontes de erros ao executar uma tarefa em particular através da interface, encaixando-se nas diretrizes do chamado método GOMS (*Goals, Operators, Methods, and Selection Rules*) [7] que busca observar e entender como se relacionam os objetivos, o usuário, os métodos e as escolhas que eles fazem.

Para tal, são levados em conta os passos mentais e físicos que constituem as subtarefas, analisando sua duração e complexidade. As principais conclusões possíveis em relação a problemas no uso são que as tarefas exigem subtarefas complexas demais ou, então, que levam muito tempo para serem executadas com sucesso.

3.3 Métodos de Teste

Métodos de teste de usabilidade, ao contrário dos de inspeção, contam necessariamente com a participação de usuários. São os métodos mais básicos e indispensáveis para a avaliação da usabilidade de um software [25]. Por contarem com a participação dos usuários, são os métodos com mais chance de trazer informações de como realmente os usuários interagem com o sistema e sobre suas dificuldades decorrentes em tais interações.

Os cinco principais métodos de teste de usabilidade são: o pensamento em voz alta, a codescoberta, a observação de campo, registro de dados e a avaliação por questionários.

O pensamento em voz alta (*Thinking Aloud* - THA) consiste em um usuário utilizar o sistema, executar tarefas pertinentes e falar em voz alta tudo o que pensa ao usar o software, bem como descrever aquilo que entende e associações que faz. Os testes devem ser executados apenas observando e ouvindo os usuários, sem interferir, de forma alguma,

nas suas atuações.

À medida em que os usuários verbalizam seus pensamentos, os especialistas podem entender como eles enxergam o sistema e identificar as falhas de usabilidade da interface [25], que podem ser desde o vocabulário inadequado até o tempo excessivo para executar tarefas que deveriam ser simples.

O método de codescoberta (*Co-discovery*) é feito utilizando-se sempre pares de usuários, que utilizam o sistema juntos conversando em voz alta à medida que usam o sistema. Apesar de não incluir a verbalização como no método do pensamento em voz alta, a ideia é quase a mesma. Procura-se ouvir e observar a interação entre os dois usuários para obter informações sobre falhas de usabilidade da interface do programa. A conversa deles, claro, deve ser pertinente à investigação da interface e a execução de tarefas típicas.

A desvantagem desse método é que, comparado aos demais métodos de teste, requer o dobro de usuários. Em compensação, espera-se que a interação entre dois usuários gere comentários e observações mais precisas do que apenas um usuário dizendo aquilo que pensa [25].

Dos testes de usabilidade, o mais simples é o da observação de campo (*Field Observation*). Este consiste simplesmente em observar o usuário utilizando o sistema em um ambiente no qual se sinta à vontade, idealmente em seu próprio local de trabalho. Acima de tudo, o usuário não deve sofrer nenhuma interferência da parte do observador. Uma técnica possível de registro é a filmagem das atividades do usuário para posterior análise. O foco dessa técnica é a descoberta de problemas gritantes de usabilidade, óbvios o suficiente para serem notadas pela simples observação.

Há também uma modalidade mais eletrônica que as demais, que talvez seja uma das poucas exceções no que diz respeito a depender de um software com algumas funcionalidade para poder ser avaliada. É o registro de dados (*Data logging*), que consiste em coletar dados, como o tempo de cada tarefa ou frequência com que cada funcionalidade é utilizada, durante a execução do programa, gerando informações que podem ser analisadas estatisticamente. Este método serve para obter dados quantitativos da usabilidade de sistemas, normalmente após a conclusão do desenvolvimento e consequente lançamento do produto.

O último método de testes a ser citado é o de questionários, que faz com que usuários respondam perguntas com suas impressões a respeito do sistema. É tido como um método indireto de se aferir a qualidade da usabilidade, pois não examina a interface em si, mas, sim, a satisfação dos usuários ao utilizá-la e a recordação de funcionalidades utilizadas. As dificuldades relacionadas a essa técnica é a necessária experiência para a elaboração dos questionários e o fato de que nem sempre a opinião do usuário reflete seu comportamento real.

Em compensação, critérios subjetivos, como a satisfação do usuário, podem ser medidos.

3.4 Considerações finais

Neste capítulo foi descrito o conceito de usabilidade de uma interface e as formas de avaliá-la e quantificá-la. A importância de se considerar a usabilidade em um trabalho de localização é garantir a qualidade do resultado entregue. A tradução e a adaptação de um software para determinada cultura não deve torná-lo mais difícil de usar.

Uma vez que o estudo proposto tem como objeto um sistema de Gestão Empresarial e os principais conceitos para um trabalho de localização foram apresentados, o próximo capítulo consiste em um estudo mais detalhado a respeito de sistemas dessa natureza.

Nele procura-se mostrar a evolução dos chamados ERP, suas principais funcionalidades e módulos e sua importância na estrutura das empresas.

Capítulo 4

Sistemas de Gestão Empresarial

Os sistemas de gestão empresarial, também conhecidos pela sigla ERP, do inglês *Enterprise Resource Planning*, são pacotes de software que buscam gerenciar todos os departamentos de uma empresa ao integrar dados e processos da organização em um único sistema. Essa integração de todos os departamentos de uma empresa possibilita a automação e dá agilidade a muitos de seus processos.

Os primeiros sistemas dessa natureza surgiram entre as décadas de 60 e 70 com objetivos mais modestos do que os ERPs atuais. Eram sistemas conhecidos pela sigla MRP, *Material Requirement Planning*, e serviam basicamente para o controle de estoque. Calculavam quais matérias-primas e em que quantidade deveriam ser compradas a partir de determinada demanda pelos produtos finais e o processo utilizado para produzi-los.

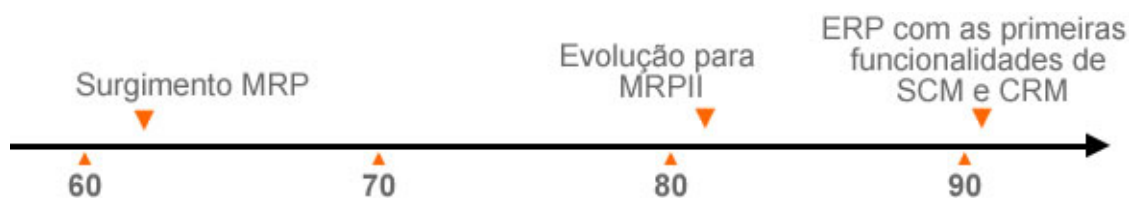


Figura 4.1: Evolução dos ERPs desde os primeiros MRPs

Por volta da década de 1980 começaram a surgir os sistemas identificados pela sigla MRP II, do inglês *Manufacturing Resource Planning*, que, apesar de possuírem a mesma sigla de seus predecessores, não tinham a mesma finalidade. Estes eram mais completos, buscando planejar toda a produção e todos os recursos necessários.

Dessa forma, os MRPs II planificavam toda a fábrica, levando em conta todas as máquinas utilizadas e todos os funcionários disponíveis. Lidavam com todos os problemas possíveis, como falta de funcionários, ferramentas defeituosas e mudanças repentinas do plano de venda, para manter a produção em curso de forma eficiente. Os MRPs II faziam exatamente isso: alocavam as operações minuto a minuto, máquina a máquina,

pessoa a pessoa, procurando otimizar todo o processo [21].

O primeiro desses sistemas era bem específico, controlando a compra de insumos. O segundo significou um passo adiante, à medida que passou a controlar muitas variáveis envolvidas no processo de produção. Mas, mesmo assim, havia ainda a necessidade de gerir outros departamentos das fábricas que necessitavam automatizar e ganhar eficiência em seus processos. Assim, as empresas produtoras de *software* perceberam a demanda pelo desenvolvimento de módulos de gestão de Recursos Humanos, Vendas, Finanças, Controladoria, entre outros.

Então, na década de 1990, aparecem os ERPs, uma nova sigla para descrever os sistemas que integravam os antigos MRPs I e II com os novos módulos que gerenciavam os demais departamentos e necessidades das empresas.

Atualmente, os ERPs continuam crescendo e recebendo novas funcionalidades. Algumas dessas funcionalidades merecem destaque, como por exemplo:

- CRM (*Costumer Relationship Management*), ou Gestão de Relacionamento com o Cliente, voltado para informações e atendimento aos clientes da empresa, incluindo funções de *e-commerce* e *call centers*;
- SCM (*Supply Chain Management*), ou Gestão da Cadeia de Suprimentos, que consiste em uma evolução de sistemas para controle de vendas, automatizando as comunicações entre empresas e seus fornecedores;
- BI (*Business Intelligence*), ou Inteligência nos Negócios, que dá apoio a decisões, coletando e ajudando a interpretar dados obtidos a partir dos processos da empresa.

Os sistemas de ERP podem trazer muitas vantagens para empresas [3], algumas delas são:

- Economia de custos, por eliminar interfaces manuais que, do contrário, demandariam mão-de-obra dedicada;
- Eficiência, por melhorar o fluxo de informação dentro da empresa, permitindo decisões mais acuradas e eficazes, a partir de dados mais bem utilizados por meio de relatórios e comparações;
- Diminuição de atividades duplicadas;
- Maior sinergia entre as diferentes plantas de grandes corporações.

Nas próximas subseções são apresentados os principais módulos de um ERP e suas funções.

4.1 Módulo de Vendas e Marketing

Apesar de vendas e marketing serem coisas diferentes, ambas possuem o mesmo objetivo: o de aumentar as vendas. Enquanto o papel do setor de vendas de uma empresa é, obviamente, vender os produtos, o papel do Marketing é melhorar o ambiente e as possibilidades para as vendas.

A melhor forma de se compreender o papel desses módulos é entender os processos por trás dos departamentos de Vendas e Marketing. Esses processos podem ser discriminados em processos operacionais e de controle administrativo. Os primeiros incluem funções corriqueiras como malas diretas, telemarketing, gestão de contatos e *prospecting* (a busca por novos clientes).

Os processos de controle administrativo são normalmente discriminados por: Gestão de vendas, Estimativa de vendas, Anúncios e Promoções e, por último, *Pricing*, o responsável pela definição de preços para os produtos.

O Processo de Gestão de Vendas diz respeito aos gestores de venda da empresa, que são responsáveis por delimitar territórios para os vendedores e alocá-los para maximizar os rendimentos. As decisões desses gestores dependem de informações de vendas passadas, comparando o desempenho de vendedores, produtos, territórios entre outros.

A previsão de vendas preocupa-se com as necessidades dos clientes ao buscar segmentar o mercado em grupos alvo para poder planejar a produção e desenvolvimento de produtos e serviços.

Já o processo de Anúncios e Promoções, diz respeito a escolher os canais e tipos de mídia ideais para promover os produtos. Esse tipo de decisão é tomada usando como base informações a respeito do público alvo do produto. Além disso, deve-se manter um monitoramento constante das campanhas de publicidade escolhidas. Assim, é possível adequar os investimentos em publicidade de acordo com os resultados obtidos e esperados.

Nos Sistemas de Gestão Empresarial atuais, ao contrário dos antigos software específicos para Vendas e Marketing, há uma integração com os demais módulos, como contabilidade e inventário, por exemplo. Isso possibilita análises melhores e mais eficiência. O destaque é a integração com os módulos de CRM (Customer Relationship Management - Gestão de Relacionamento com o Cliente), que torna ainda melhor as decisões que necessitam de dados referentes aos consumidores.

De certa forma, pode-se até dizer que os módulos de Vendas e Marketing nos ERP assumem um papel de grande importância, às vezes até central, segundo alguns autores [49]. Isso porque muitos módulos dependem de informações vindas do módulo de vendas para operar melhor. O planejamento de produção, por exemplo, pode basear-se em informações e estatísticas do módulo de vendas para dimensionar aquilo que precisa ser produzido. A análise de lucratividade (*Profitability Analysis*) é outro exemplo, onde estratégias podem ser delineadas a partir das informações dos pedidos registrados pelo módulo de vendas. O

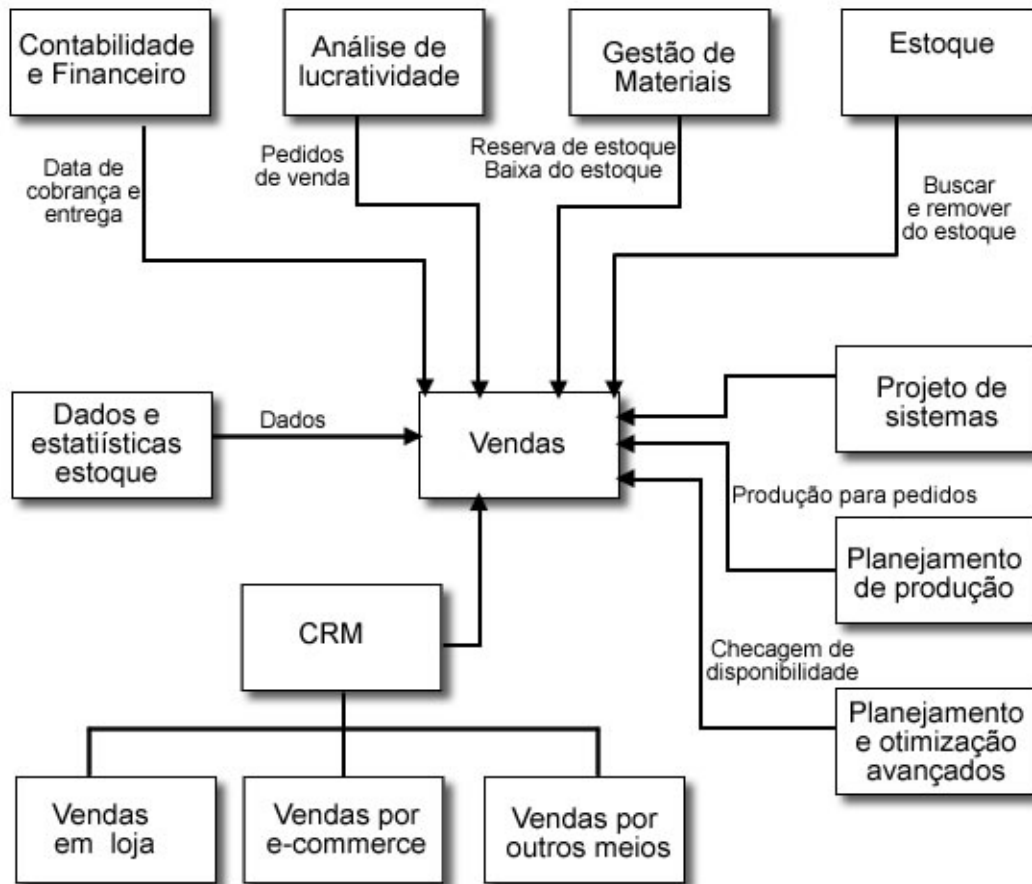


Figura 4.2: Interação do módulo de vendas com outros módulos dos ERP [49]

diagrama da Figura 4.2 [49] mostra esse papel quase que central.

4.2 Gestão do relacionamento com o cliente

Uma das estratégias de negócio das empresas é o foco nos clientes, baseada no entendimento e antecipação das suas necessidades. A função do CRM (*Costumer Relationship Management*), ou de um sistema de Gestão do Relacionamento com o Cliente, é justamente essa. O CRM usa dados obtidos de outros departamentos da empresa, notadamente dos de Vendas e Marketing, para manter um relacionamento.

A origem dos módulos CRM são os pacotes de software de automação de força de vendas. As principais funções deles são [49]:

- Gestão das atividades de venda - cuja função é guiar os representantes de vendas em todos os passos do processo de venda, desde o contato com clientes em potencial até o chamado *follow-up*, que é o acompanhamento do cliente após uma venda;

- Gestão de vendas e territórios - usado para que os gestores de vendas monitorarem o desempenho de seus vendedores e otimizarem a atuação da equipe;
- Gestão de contatos - que permite aos vendedores consultar os contatos de clientes segundo critérios que podem ser úteis para concretizar vendas. Com ele é possível, por exemplo, consultar quais clientes compraram um determinado produto; com essa informação pode-se fazer o *follow-up* ou oferecer produtos complementares e suprimentos;
- Gestão de Oportunidades Promissoras (*Lead management*) - para a análise de vendas em potencial direcionadas clientes com alta probabilidade de adquirir produtos. Dessa forma os gestores podem alocar seus representantes de venda para abordarem os clientes certos, de acordo com seu conhecimento de produtos e território de atuação;
- Gestão de configuração - atende às necessidades de empresas que configuram produtos de acordo com as necessidades dos clientes. Um exemplo são empresas que vendem hardware de computador e montam soluções para seus clientes. Esses sistemas permitem fazer a cotação de configurações específicas para os clientes, assim como concretizar vendas baseadas nessa configurações.

Dessa maneira, o sistemas de Gestão do Relacionamento com o Cliente permite, juntamente com os módulos de vendas e marketing, aumentar o sucesso das vendas da empresa com base no bom relacionamento com clientes.

4.3 Gestão de materiais e da produção

Essa área da gestão de empresas há muito já faz uso de sistemas de informação para exercer suas funções. Como citado anteriormente, os ERP de hoje tiveram sua origem nos MRP (*Material Requirement Planning*), responsáveis pela gestão do estoque de materiais, e os MRP II (*Manufacturing Resource Planning*), responsáveis por planificar todo o processo produtivo de acordo com as matérias primas, os processos utilizados, o maquinários e a mão-de-obra disponível.

Os ERP integram esses processos, ao acoplá-los a partir de seus dados, com os demais processos dos outros departamentos da empresa. Dessa maneira, outros setores da empresa podem cumprir suas funções com base em dados extraídos da produção e dos materiais utilizados. O setor de contabilidade, por exemplo, pode aferir com mais precisão os custos de matérias primas e dos bens produzidos, o que permite ao setor de vendas, por sua vez, gerir cotações mais precisas para os clientes em potencial.

Da mesma forma, dados do setor de venda e marketing podem ser utilizados para determinar a demanda por produtos e, então, planejar a compra de materiais e a produção propriamente dita.

De maneira geral, a produção de uma empresa baseia-se nas seguintes atividades:

- Definir o plano de produção;
- Adquirir a matéria-prima;
- Agendar o uso dos equipamentos, instalações e mão-de-obra para processar a matéria-prima;
- Projetar produtos e serviços;
- Produzir a quantidade certa com o nível de qualidade adequado dentro dos prazos impostos.

Pode-se dividir o planejamento da produção em processos operacionais e administrativos. Dentre os principais processos usados na administração pode-se citar: a compra e o recebimento de matéria-prima, o controle de qualidade, a contabilidade dos custos e a gestão do inventário.

Já dentre os processos administrativos estão:

- O planejamento do uso de matéria prima, que inclui a identificação do estoque requerido pelo plano de produção, a determinação dos prazos para os fornecedores abastecerem o estoque, o cálculo dos níveis seguros para o estoque e da quantidade mais vantajosa, do ponto de vista financeiro, a ser encomendada de cada material;
- O cálculo da capacidade de produção, que, a partir de dados como a mão-de-obra disponível, equipamentos, espaço físico e outras características específicas do processo de produção, determinarem se há condições adequadas e seguras para produzir de acordo com o plano de produção;
- O desenvolvimento de produtos;
- O agendamento da produção.

4.4 Contabilidade e Financeiro

Os setores financeiro e de contabilidade das empresas são responsáveis pela administração dos recursos monetários. Assim, esses setores estão presentes desde o pagamento de

fornecedores e funcionários até o pagamento de eventuais reembolsos a clientes. Desempenham papel importante também na avaliação e análise de possíveis investimentos e gastos, autorizando ou não a concretização de orçamentos.

Os sistemas de informação destinados a equipar esses setores normalmente auxiliam a contabilidade a elaborar o balanço da empresa, controlar ativos e passivos, controlar o inventário, controlar contas a pagar e receber, controlar a folha de pagamentos e ordens de compra.

Com a integração oferecida pelos ERP, todas essas funções podem ser executadas de forma mais eficiente, pois dados de outros departamentos e módulos podem ser processados automaticamente, simplificando passos que compõem essas tarefas. Além disso, funções de caráter administrativo, como planejar o orçamento, gerir o caixa e planejar os investimentos da empresa, podem ser executadas com mais precisão por possuírem informações de todos os demais departamentos da empresa.

4.5 Recursos Humanos

O setor de Recursos Humanos é responsável pela gestão de todos os aspectos referentes aos funcionários da empresa. Como nos demais setores, é possível classificar as tarefas em operacionais e administrativas.

As operacionais compreendem:

- gestão e armazenamento de informações de funcionários;
- alocação de funcionários;
- gestão de informações de desempenho de cada funcionário;
- relatórios para órgãos governamentais que regulam as atividades trabalhistas;
- informações da folha de pagamentos.

Já as administrativas compreendem a análise e o planejamento de cargos, a gestão de informações de recrutamento e o treinamento e desenvolvimento profissional.

Como nos demais módulos de um ERP, a vantagem do módulo de Recursos Humanos é permitir a integração desse departamento com os demais. Para gerenciar a folha de pagamentos, por exemplo, os funcionários desse setor se beneficiam e se tornam mais eficazes pela integração com o departamento de contabilidade. A alocação de funcionários, por sua vez, pode ter como base as informações dos módulos de produção para alocar a mão-de-obra.

Do ponto de vista administrativo, o módulo de Recursos Humanos torna as decisões dos gestores em relação à mão-de-obra mais bem fundamentadas e eficientes.

Acima de tudo, a maior vantagem para a empresa, gerada por essa integração, é que no passado, boa parte do tempo das áreas de recursos humanos era usada para atualizar registros de funcionários da empresa. Atualmente, em compensação, mais esforços podem ser direcionados para análise de necessidades da empresa e aprimoramento da mão-de-obra disponível.

4.6 Gestão da Cadeia de Suprimentos

A gestão da cadeia de suprimentos de uma empresa é fundamental para o planejamento de sua produção. Os antigos MRPII e os primeiros ERP possuíam funções de planejamento de produção limitados e necessitavam de ajustes no agendamento da produção. Já os sistemas de SCM (*Supply Chain Management*) trabalham com algoritmos que não necessitam desses ajustes e, dessa forma, permitem um planejamento em tempo real.

A integração dos ERP com os SCM ou mesmo os ERP, que incluem módulos de SCM, permitem, graças ao planejamento em tempo real, que o planejamento da produção reaja rapidamente a mudanças de fornecimento e demanda.

Especialistas afirmam que investimentos na gestão da cadeia de suprimentos podem trazer vários benefícios, dentre eles [23]: maiores lucros, aumento de produtividade, economias em custos operacionais, diminuição de estoques e redução do período de tempo entre a entrada e a conclusão de pedidos.

4.7 Comércio Eletrônico

O comércio eletrônico talvez seja a atividade que mais cresceu com a expansão da internet. Ao que tudo indica, o comércio eletrônico ameaça o papel de intermediários e atravessadores, facilitando o comércio e melhorando os preços para os consumidores.

Sistemas de Comércio Eletrônico surgiram separadamente dos sistemas de gestão empresarial, servindo apenas como uma forma de as empresas conseguirem vender seus produtos pela internet, seja para outras empresas (B2B, ou *Business to Business*) ou para consumidores (B2C, *Business to Consumers*), eliminando atravessadores e agilizando os processos. Além disso, diminuem os custos operacionais e favorecem a comunicação entre todas as pontas da negociação [4].

No entanto, conforme a modalidade do comércio eletrônico foi crescendo, a dificuldade de gerenciar dados de clientes, pedidos e catálogo cresceu também. Mais uma vez, os ERPs trouxeram a possibilidade de integração e o consequente aumento de eficiência. Com eles as empresas podem administrar melhor a cadeia de fornecedores e atender melhor seus consumidores finais.

Sistemas de comércio eletrônico por si só têm pouco a oferecer, tanto a gestores quanto funcionários, mas unidos a sistemas de gestão empresarial são ferramentas extremamente poderosas [4].

4.8 Business Intelligence

A principal meta dos ERP é, sem dúvida, integrar o melhor possível áreas e funcionários de uma empresa em prol de seu funcionamento mais eficiente. Qualquer empresa de porte razoável, com um número grande de pedidos e transações, gera muitos dados e precisa mantê-los de forma eficiente e, para isso, nem sempre basta que haja apenas a integração.

A integração, em geral, permite que departamentos e gestores realizem suas funções mais rapidamente por terem acesso a dados de outras áreas. Porém, para gestores de mais alto escalão, que necessitam projetar políticas e planejar a atuação da empresa a longo prazo, são necessários meios de analisar todos os dados disponíveis.

Alguns afirmam que muitas das empresas grandes, que investem em tecnologia da informação, são ricas em dados, mas pobres em informação [52]. Isso quer dizer que ainda lhes falta ferramentas analíticas para os dados que possuem.

Os módulos de BI (*Business Intelligence*) são desenvolvidos com vistas a essa necessidade de análise, se apoiam em informações e análises de negócios no contexto de processos chave para subsidiar a decisões e ações com vistas a um desempenho melhor [52].

Dessa forma, os módulos de BI captam dados de todos departamentos e os resumem em indicadores objetivos que apoiam gestores em decisões e ações críticas para a empresa. Os principais benefícios citados por especialistas na área são o aumento de vendas, redução de custos e aumento dos lucros.

4.9 Considerações finais

Neste capítulo foi apresentada um pouco da história dos Sistemas de Gestão Empresarial, mostrando sua evolução desde os primeiros MRP até os sistemas de grande porte existentes hoje em dia.

A principal característica desses sistemas a ser salientada é a ideia de integração entre vários setores de uma empresa e até entre diferentes empresas. Essa integração normalmente é alcançada a partir do tratamento adequado dos dados de vários setores.

A necessidade de tratar e interpretar dados da melhor forma possível, fica ainda mais clara nos módulos de *Business Intelligence*. Neles, a ideia principal é justamente aglutinar informações e permitir seu uso para tomada de decisões.

No próximo capítulo será apresentado o projeto Apache Open For Business, uma iniciativa de uma comunidade de software livre para a implementação de um Sistema de

Gestão Empresarial completo e bem estruturado.

Capítulo 5

Apache Open For Business

O projeto *Apache Open For Business*, também conhecido pela sigla OFBiz, é um sistema formado por um conjunto de aplicativos voltados às necessidades de empresas, formando um ERP propriamente dito. Foi criado em maio de 2001, por David E. Jones e Andy Zenesky, e cresceu, em termos de desenvolvimento e adoção, durante os 5 anos seguintes até que, em 2006, o projeto foi aceito como parte do *Apache Incubator Project Managing*. Atualmente continua crescendo e tem sido bastante adotado em razão de suas novas funcionalidades relacionadas a comércio eletrônico.

Os aplicativos presentes no conjunto distribuído pela comunidade visam facilitar a administração de todos os aspectos de uma empresa, desde administração de clientes e fornecedores, contabilidade, administração de recursos e ativos, até tendências mais recentes como a integração com CMS (*Content Managment Software*), *E-commerce* e BI (*Business Intelligence*). Todos eles foram desenvolvidos com base em estudos de outros sistemas de ERP e CRM (*Costumer Resource Management*) seguindo boas práticas de engenharia de software e modelos de estruturas de dados comprovadamente eficientes [32].

O projeto adota tecnologias bastante difundidas no mercado como a linguagem de programação JAVA e padrões, relativamente simples e populares, como XML, HTML e SOAP. No entanto, o grande diferencial do projeto é o uso dessas tecnologias baseado em padrões de projeto (*Design Patterns*) reconhecidamente eficientes em conjunto com novas tecnologias que permitem a implementação de mecanismos de metaprogramação, baseados normalmente em XML, e que buscam simplificar o desenvolvimento e extensão de funcionalidades.

A arquitetura do projeto *Apache Open For Business* segue o modelo conceitual de três camadas conhecido como MVC (*Model-view-controller*), possuindo, dessa forma, uma camada para persistência de dados, outra para a lógica de negócios e, por fim, uma de apresentação e interação com o usuário.

Além disso, todo o sistema é organizado em componentes, que são implementados com essa arquitetura e buscam isolar cada funcionalidade que um ERP deve oferecer. Mas,

mesmo assim, componentes podem, se necessário, usar funcionalidades oferecidas pelos demais componentes graças à aplicação de conceitos de Arquitetura Orientada a Serviços (SOA - *Service Oriented Architecture*).

E é desse mecanismo de acoplamento fraco que os diversos componentes, que formam o ERP, se comunicam e compartilham informações. Ainda é possível complementar o sistema através da implementação de componentes próprios, baseando-se, caso conveniente, nas funcionalidades já existentes dos demais componentes padrão.

Trata-se de um sistema de grande porte, composto e implementado com várias linguagens e tecnologias como será visto adiante. Para se dar uma ideia do tamanho do sistema e sua complexidade foi feita uma contagem apenas do número de linhas de arquivos XML e JAVA, as principais linguagens que compoem o sistema. O resultado foi de 643.616 linhas XML e 401.988 JAVA, totalizando 1.045.604 linhas de código.

A seguir são detalhadas as principais características do projeto.

5.1 Arquitetura - Camada de Persistência de Dados

A camada de persistência de dados foi concebida para ser uma abstração que pudesse encapsular qualquer modelo de representação das informações. Essa camada leva o nome de *Entity Engine* e é baseada no padrão de projeto de nome *Delegation* [26].

Trata-se de um padrão de projeto de software em que a responsabilidade pela execução de certa tarefa é delegada a outra parte ou componente do software [20]. Ou seja, um objeto apresenta certo comportamento para o mundo externo, mas, na realidade, a implementação dele é feita por outro objeto.

Esse comportamento pode ser implementado pelo uso de herança múltipla ou, no caso de JAVA, que não possui suporte a esse tipo de herança, com herança simples. Essa possibilidade é mostrada no diagrama da Figura 5.1.

No caso da camada de persistência de dados do OFBiz, isso significa que a escrita em um banco de dados é delegada a objetos implementados especialmente para lidar com bancos de dados específicos.

Dessa forma, o *Entity Engine* baseia-se nos *Generic Values* e no *Generic Delegator*, sendo o primeiro a representação genérica de valores e o segundo um mecanismo genérico para leitura e escrita.

Isso permite o uso de qualquer Sistema Gerenciador de Bancos de Dados Relacional (SGBDR) escolhido. Na realidade o projeto já dá suporte a mais de dez deles e, graças a essa abstração, outros podem ser adicionados de forma relativamente fácil. Alguns exemplos de bancos já operados por este padrão são: Derby, a configuração padrão, MySQL, PostgreSQL, Oracle SQL, Microsoft SQL Server, HyperSQL, Firebird, entre outros.

Assim como as demais camadas do OFBiz, a configuração e manipulação da camada

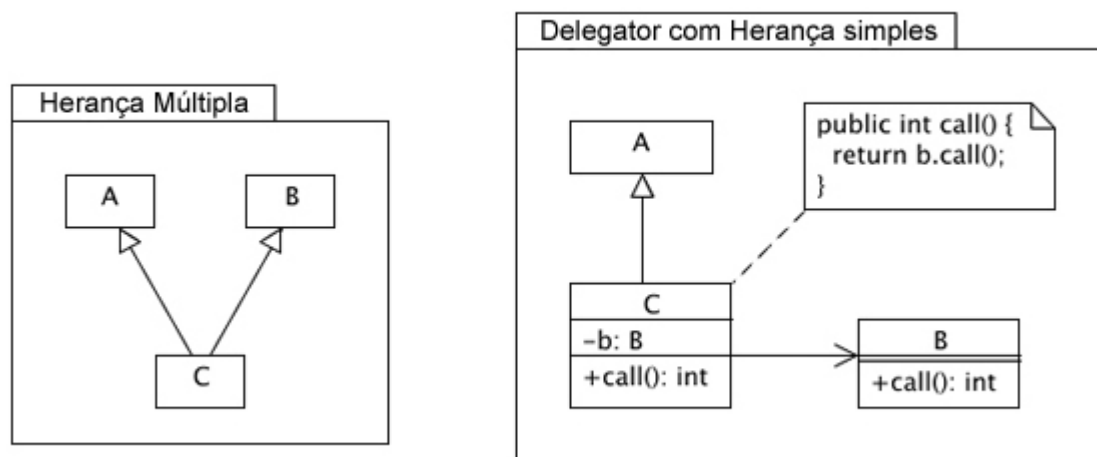


Figura 5.1: À esquerda, herança múltipla que poderia ser utilizada para implementar o *Delegator Pattern*. À direita, a abordagem possível e mais adequada em JAVA.

de persistência de dados, pode ser feita através de uma espécie de metalinguagem de programação que abstrai boa parte dos detalhes que seriam necessários para lidar com um SGBDR.

O que acontece na prática com o *Entity Engine*, do ponto de vista do programador responsável pela definição do banco de dados a ser utilizado, é que as tabelas, os campos e as relações existentes entre eles podem ser manipulados com o auxílio de apenas três arquivos: um responsável pela definição do SGBDR a ser usado, outro para a declaração das tabelas e um outro para incluir as tabelas na inicialização do sistema.

A intenção aqui não é entrar em detalhes da implementação, mas, sim, explicitar a simplicidade da abstração fornecida pelo *Entity Engine*. Cada um desses arquivos é melhor detalhado a seguir.

O sistema possui um arquivo de nome *entityengine.xml*, no qual é possível definir o banco de dados a ser usado e o *Delegator* apropriado. Para alterá-lo basta alterar parâmetros referentes ao SGBDR desejado assim como dados de endereço, porta etc.

Cada componente, por convenção, possui um diretório de nome *entitydef* que possui, dentro dele, um arquivo de nome *entitymodel.xml* no qual são definidas as tabelas, campos e relações usadas pelo componente. Nele é possível definir chaves primárias e chaves estrangeiras, por exemplo.

A título de ilustração, para a criação de *JOIN* entre tabelas essa metalinguagem oferece o que, no contexto do OFBiz, é chamado de *View-entity*. Grosso modo consiste em uma forma de consolidar dados de diversas tabelas em tempo de execução sem a necessidade de criar extensos comandos SQL para isso.

Por fim, as definições criadas no arquivo *entitymodel.xml* são incluídas no arquivo *com-*

ponent.xml do mesmo componente para que sejam carregadas pelo sistema ao ser inicializado.

A propósito, durante a inicialização do sistema todos os arquivos referentes às definições da *Entity Engine* são varridos e verificados para averiguar se há alguma alteração nas definições de tabelas ou campos. Isso significa que o próprio sistema encarrega-se de fazer a atualização e manter a consistência dos dados.

5.2 Arquitetura - Camada de lógica de negócios

A camada da lógica de negócios do projeto Open For Business é toda implementada baseada em uma arquitetura orientada a serviços (SOA).

A implementação orientada a serviços do OFBiz é alcançada graças ao uso do padrão de projeto de nome *Factory Pattern* [24]. Graças a ele, o sistema cria instâncias do chamado *Service Engine*, responsáveis por lidar com cada tipo de serviço. Além disso, há também instâncias do *Service Dispatcher*, que são os componentes responsáveis por repassar a chamada do serviço para a instância do *Service Engine* adequado, dependendo de sua implementação.

Diferentes tipos de *Service Engine* são necessários pois, nessa camada, responsável pela lógica por trás dos componentes que formam o OFBiz, os serviços podem ser implementados usando-se um leque bastante amplo de tecnologias como Java, BeanShell, Groovy, JPython, Mini-Language (uma linguagem própria do projeto), dentre outras.

O diagrama da Figura 5.2 [24] demonstra, de forma simplificada, o funcionamento da camada de serviços do sistema.

Quanto à quantidade de linguagens disponíveis para implementação, a razão para tantas delas é relativamente simples: oferecer várias opções adequadas a cada tipo de problema. Ou seja, algumas vezes a implementação baseada em um script como BeanShell ou Groovy pode ser mais rápida e simples que a criação de classes em Java.

Já a razão para a criação de uma linguagem própria para o projeto pode ser um pouco mais difícil de entender à primeira vista. Porém, a explicação dos autores para tal é o fato de que muitas das tarefas necessárias para implementação de sistemas corporativos incluem subtarefas que, muitas vezes, se tornam repetitivas e poderiam levar a duplicações desnecessárias de código [31].

Dessa forma, a solução encontrada foi a utilização do padrão de projeto conhecido como *Interpreter Pattern*, que consiste, basicamente, na criação de um interpretador de sentenças, em uma determinada linguagem, descritoras de operações [20].

Segundo os autores e a comunidade do projeto [31], esse padrão permitiu a criação de uma linguagem simples que possibilita a execução de tarefas mais complexas. Afirmam, inclusive, que o uso dessa linguagem simplificada, de nome Mini-Language, economiza 70

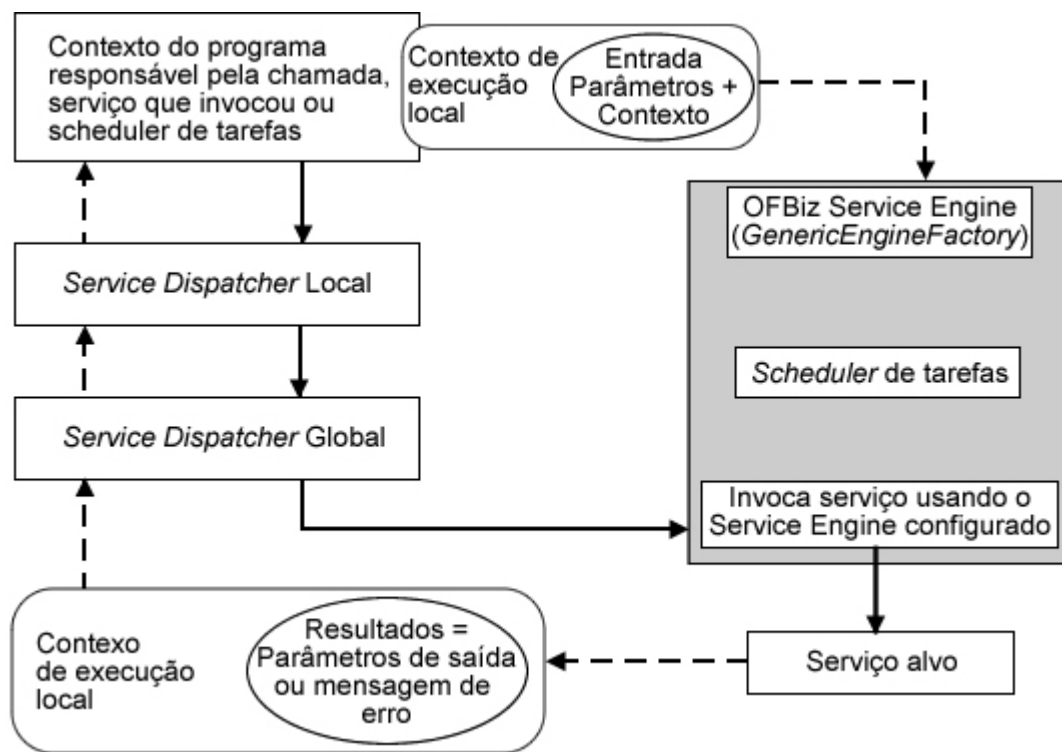


Figura 5.2: Diagrama simplificado do funcionamento da camada de serviços

a 90% do tempo do programador.

O mais interessante a respeito da camada de serviços é que serviços podem ser invocados pela requisição direta do usuário por meio da interface de usuário, pelos próprios serviços ou mesmo por partes remotas via SOAP, por exemplo.

Graças a essas características, principalmente a invocação de serviços pelos próprios serviços, permite uma alta taxa de reaproveitamento de código, diminuindo, em consequência, o tempo necessário para o desenvolvimento de funcionalidades.

Ainda graças à versatilidade com que os serviços podem ser invocados, é implementada uma espécie de arquitetura dirigida a eventos (EDA - *Event Driven Architecture*). No contexto do projeto OFBiz, são definidas as chamadas ECAs, do inglês, *Event Condition Action*, ou em português algo como Ação Condicionada a Evento, significando que ações podem ser disparadas mediante o acontecimento de eventos e determinadas condições.

Há três categorias de ECAs[31]:

- **SECA**, ou *Service Event Condition Action*, que é o disparo de uma ação mediante a invocação de um serviço;
- **EECA**, ou *Entity Event Condition Action*, que representa uma regra para invocação de um serviço a partir de um evento relacionado à manipulação de uma *Entity*, ou

seja, de algum conteúdo da camada de persistência de dados;

- **MECA**, ou *Mail Event Condition Action*, que são basicamente mecanismos para tratar automaticamente e-mails recebidos pelo sistema.

No âmbito de software ERP, essa possibilidade de execução é bastante útil. Uma vez que o espírito desses sistemas é a integração entre os sistemas responsáveis por cada departamento, a existência de serviços disparados por eventos e condições torna possível uma maior sinergia entre todos os componentes, assim como a consolidação de dados usados por eles.

A implementação desses eventos é muito similar à implementação das *entities*, pelo menos a grosso modo. Assim como as *entities*, eles são definidos usando arquivos XML específicos e, depois, seu carregamento é configurado para ser efetuado durante a inicialização do sistema, ficando disponíveis para serem invocados durante a execução.

Em cada componente há, por convenção, um diretório de nome *servicedef* que, novamente por motivos de padronização, contém no mínimo três arquivos: *Services.xml*, *secas.xml* e *groups.xml*.

Os serviços são definidos no arquivo *Services.xml* e, basicamente, são referências a outros arquivos que de fato possuem a implementação deles. Tais declarações são justamente para garantir que cada serviço seja invocado pelo devido *Service Engine*, dependendo da linguagem usada na implementação.

O mesmo vale para o arquivo *secas.xml* para serviços a serem invocados a partir de determinados eventos e condições. Por último, o arquivo *group.xml* é utilizado para agrupar determinados serviços para que seja possível executá-los todos de uma vez, se necessário.

5.3 Arquitetura - Camada de apresentação

De forma geral, pode-se dizer que o projeto Open For Business como um todo é baseado em *Tomcat* e outros *software* da Fundação *Apache* como *Catalina*, *Geronimo* etc. Mas o principal a se saber para se entender o funcionamento da camada de apresentação do projeto, é que usa-se o *Tomcat* como servidor de *servlets* para gerar as páginas que serão visualizadas em qualquer *browser* capaz de lidar com HTML.

Existem vários mecanismos possíveis para se criar as páginas ou, em outras palavras, a interface para os componentes do OFBiz, mas o padrão mais indicado pela comunidade é o chamado *Screen Widget*.

Trata-se, de forma geral, de mais um mecanismo baseado em uma metalinguagem, usando XML, que descreve os componentes que formam as telas. Nesse contexto, tais componentes são chamados de *widgets* que são, em outras palavras, pequenas porções da interface, que possuem funções específicas e juntas compõem a interface completa.

Na realidade o *Screen Widget* é o responsável por interpretar os arquivos XML escritos na referida metalinguagem e transformá-los em páginas web propriamente ditas, devidamente codificadas em HTML.

Essa é, por alto, a maneira de se criar as telas da interface. Porém, não é só isso. Pode-se citar, no mínimo, dentre outras, duas estruturas que podem compor as telas: menus e formulários. Para isso são definidos os *Menu Widget* e *Form Widget*.

O *Menu Widget* é usado para criar os menus da interface dos componentes. E o *Form Widget* é uma das formas que o usuário tem para fazer requisições a serviços passando parâmetros de entrada. As páginas geradas com a participação do *Form Widget* permitem ao usuário, por exemplo, gravar informações na camada de persistência de dados ou, então, fazer requisições à camada de serviço para consulta desses dados.

Todos esses *widgets* possuem opções para que o fluxo do carregamento das telas possam ser baseadas em condições. Isso permite que as telas apresentem opções ou comportamentos de acordo com a atuação do usuário ou, ainda, qualquer outra condição baseada na camada de serviços ou dados.

É importante, a essa altura, ressaltar como é feito o controle das páginas e serviços a serem invocados, de acordo com as requisições feitas ao sistema. Acontece que o OFBiz possui um *Controller Servlet* encarregado de receber as requisições de URLs HTTP ou HTTPS e, então, repassar os pedidos para o *engine* responsável por executar a tarefa.

De um ponto de vista prático, o *Controller Servlet* pode ser entendido como o mecanismo baseado nas informações presentes no arquivo *controller.xml* de cada componente, que nada mais é do que uma compilação de todas as URLs possíveis e seus respectivos serviços ou telas.

No caso dos serviços, as entradas no arquivo *controller.xml* basicamente apontam para serviços a serem invocados e a serem consultados no arquivo *service.xml* bem como as ações a serem tomadas de acordo com a resposta retornada por cada serviço.

Já no caso das telas, algumas vezes chamadas nesse contexto de *views*, as entradas no arquivo *controller.xml* apontam simplesmente para as suas respectivas definições. Após o mapeamento da URL para a tela em questão, o chamado *View Handler* prepara a visualização a partir dos arquivos que definem as telas e eventuais serviços ou dados necessários e, só então, retorna para o browser a página em HTML.

A Figura 5.3 apresenta o funcionamento da camada de apresentação desde uma requisição do navegador até a entrega da tela já processada. Vale a pena reparar que a própria camada de apresentação pode obter informações da camada de persistência de dados para compor a interface.

Outro ponto a ser citado aqui é quanto às mensagens e aos rótulos da interface. No início do projeto, o OFBiz recorria à mesma abordagem que a maioria dos sistemas JAVA utilizam, ou seja, arquivos *.properties* para separar esses textos da implementação da interface propriamente dita.

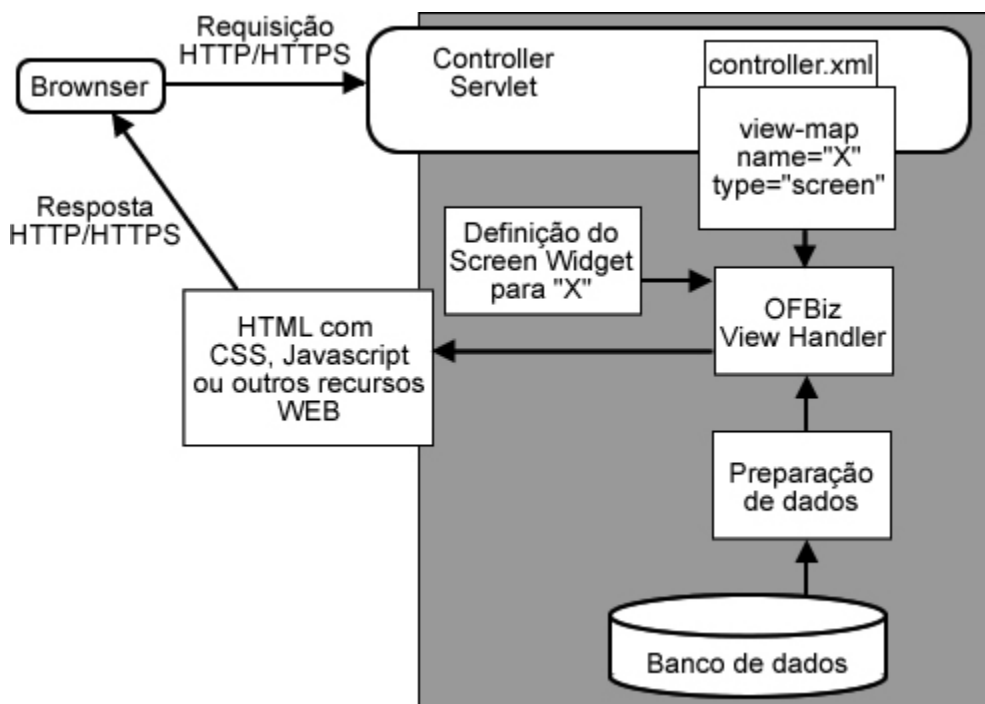


Figura 5.3: Diagrama simplificado do funcionamento da camada de apresentação, baseada no *Controller Servlet*

No entanto, com a evolução do projeto, os desenvolvedores optaram pela adoção de outro método. Como quase tudo no projeto, agora esse desacoplamento entre a implementação da interface e seu conteúdo sensível à linguagem do usuário é feito por meio de arquivos XML.

Por convenção, cada componente possui, em um diretório de nome *config*, arquivos com o sufixo **Labels.xml* que contém todas as mensagens que compõem a interface. A Figura 5.4 mostra o início do arquivo *PartyEntityLabels.xml* pertencente ao componente *Party*, que basicamente é responsável por lidar com todas as pessoas, clientes, empresas e qualquer outra parte relacionada com a empresa que o software atender.

Como é possível observar na Figura 5.4, as mensagens são agrupadas em *tags property* que são traduzidas pelas *tags value*. Na implementação da interface, o identificador (*key*) de cada mensagem pode ser usada para carregar a mensagem adequada de acordo com a linguagem do usuário.


```

<resource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <property key="AgreementItemType.description.AGREEMENT_COMMISSION">
    <value xml:lang="de">Kommission</value>
    <value xml:lang="en">Commission rate</value>
    <value xml:lang="es">Tasa de comisión</value>
    <value xml:lang="fr">Taux de commission</value>
    <value xml:lang="hi_IN">दलाली दर</value>
    <value xml:lang="it">Provvigione</value>
    <value xml:lang="pt_BR">Comissão</value>
    <value xml:lang="ro">Provizion</value>
    <value xml:lang="ru">Секция</value>
    <value xml:lang="th">ค่าคอมมิชชั่น</value>
    <value xml:lang="zh">节</value>
  </property>
  <property key="AgreementItemType.description.AGREEMENT_EXHIBIT">
    <value xml:lang="de">Exponat</value>
    <value xml:lang="en">Exhibit</value>
    <value xml:lang="es">Despliegue</value>
    <value xml:lang="fr">Exposition</value>
    <value xml:lang="hi_IN">अवर्तन करे</value>
    <value xml:lang="it">Esposizione</value>
    <value xml:lang="pt_BR">Exibição</value>
    <value xml:lang="ro">Expozitie</value>
    <value xml:lang="ru">Образец</value>
    <value xml:lang="th">การแสดงผล</value>
    <value xml:lang="zh">展览</value>
  </property>

```

Figura 5.4: Amostra do arquivo *PartyEntityLabels.xml* pertencente ao componente *Party* com rótulos da interface para diversas línguas

5.4 Comparação com outros paradigmas arquiteturais

Um documento interessante disponível na comunidade do Open For Business [8] compara a arquitetura do projeto com outras abordagens normalmente adotadas. Em especial, faz a comparação entre a estrutura adotada em grandes projetos JAVA, que visam uma boa estruturação, e projetos em scripts que visam rápido desenvolvimento.

Normalmente, o desenvolvimento em JAVA visa a separação em componentes e funcionalidades para que várias equipes possam trabalhar em conjunto. Cada uma delas construindo uma camada ou parte das camadas que formam o projeto. A Figura 5.5 representa a arquitetura típica de um sistema em JAVA.

Nos sistemas típicos em JAVA, ao menos os com características web como o OFBiz, a apresentação é feita por classes que geram a interface gráfica, chamadas de *Servlet*, responsáveis por gerar páginas em HTML. A camada de negócios é implementada por



Figura 5.5: Esboço da arquitetura típica de sistemas baseados em JAVA

classes que descrevem a lógica por trás do aplicativo. E por fim, a camada de persistência de dados normalmente é descrita por classes que podem ser serializadas e, então, armazenadas, normalmente em uma base de dados.

Sistemas web baseados em linguagens como PHP e Perl normalmente refletem o foco dos desenvolvedores em desenvolvimento rápido. Apesar de ambas possuírem opções para o desenvolvimento orientado a objetos, em geral o que acontece é o desenvolvimento sem a divisão em classes ou mesmo em camadas. Os dados são manipulados em estruturas como listas ou mapeamentos e persistidos diretamente no banco de dados.

Pode-se dizer que é muito comum que essa abordagem resulte em prazos curtos para entrega, porém com qualidade discutível por gerarem códigos difíceis de se manter. A Figura 5.6 representa a estrutura mais comum de projetos dessa natureza. Como é possível observar, em contraste com os sistemas em JAVA, consiste basicamente em apenas uma camada responsável por todas as operações.



Figura 5.6: Esboço da arquitetura típica de sistemas baseados em linguagens de script

A abordagem utilizada na arquitetura do Open For Business busca unir as qualidades da estruturação dos projetos JAVA com a agilidade de desenvolvimento em camadas únicas. Em um primeiro momento, é difícil compreender como isso pode ser alcançado, porém o segredo está no fato de o OFBiz ser um *framework* baseado em serviços e na utilização de metalinguagens.

A Figura 5.7 busca mostrar a estruturação do OFBiz. Nela é possível observar que, diferentemente da arquitetura apontada como padrão para JAVA, tanto a camada de lógica de negócios quanto a camada de apresentação podem interagir com a camada de persistência de dados. Segundo a comunidade, isso favorece o desenvolvimento rápido sem comprometer

a estruturação do sistema [8].

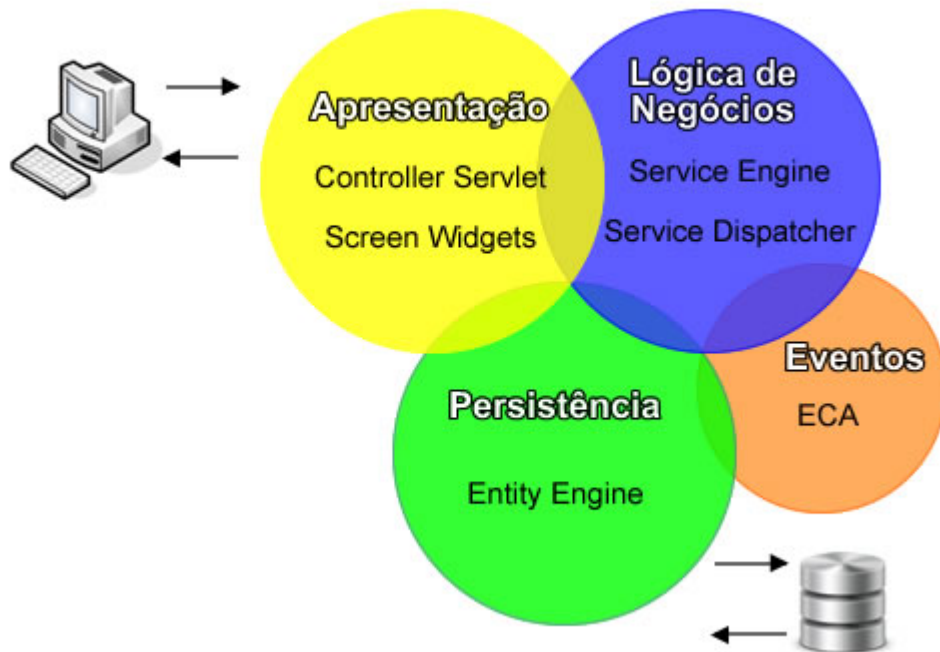


Figura 5.7: Esboço da arquitetura do projeto Open For Business

Outro fator que acelera e facilita o desenvolvimento, como dito anteriormente, é a questão das metadefinições e metalinguagens. Segundo os desenvolvedores [31], essa prática permite tornar tarefas que seriam repetitivas e comuns a várias partes do processo de programação muito mais fáceis de implementar. Garantem, ainda, que isso pode economizar entre 70 a 90% de tempo.

Por fim, além das camadas de apresentação, serviços e dados, é apresentada também a camada referente às ações que o sistema dispara baseado em eventos e condições. Ela interage com a camada de persistência de dados e a de lógica de negócios. Graças a essa característica de programação orientada a eventos, é possível aumentar o grau de reutilização de componentes diminuindo mais ainda o tempo de desenvolvimento e a manutenção de um sistema estruturado.

5.5 Processo recomendado para o desenvolvimento

A Figura 5.8 foi baseada em um diagrama presente em um dos livros de referência do Open For Business [24]. Seu intuito é indicar qual o processo mais adequado para o ciclo de desenvolvimento de componentes dentro do contexto do OFBiz.

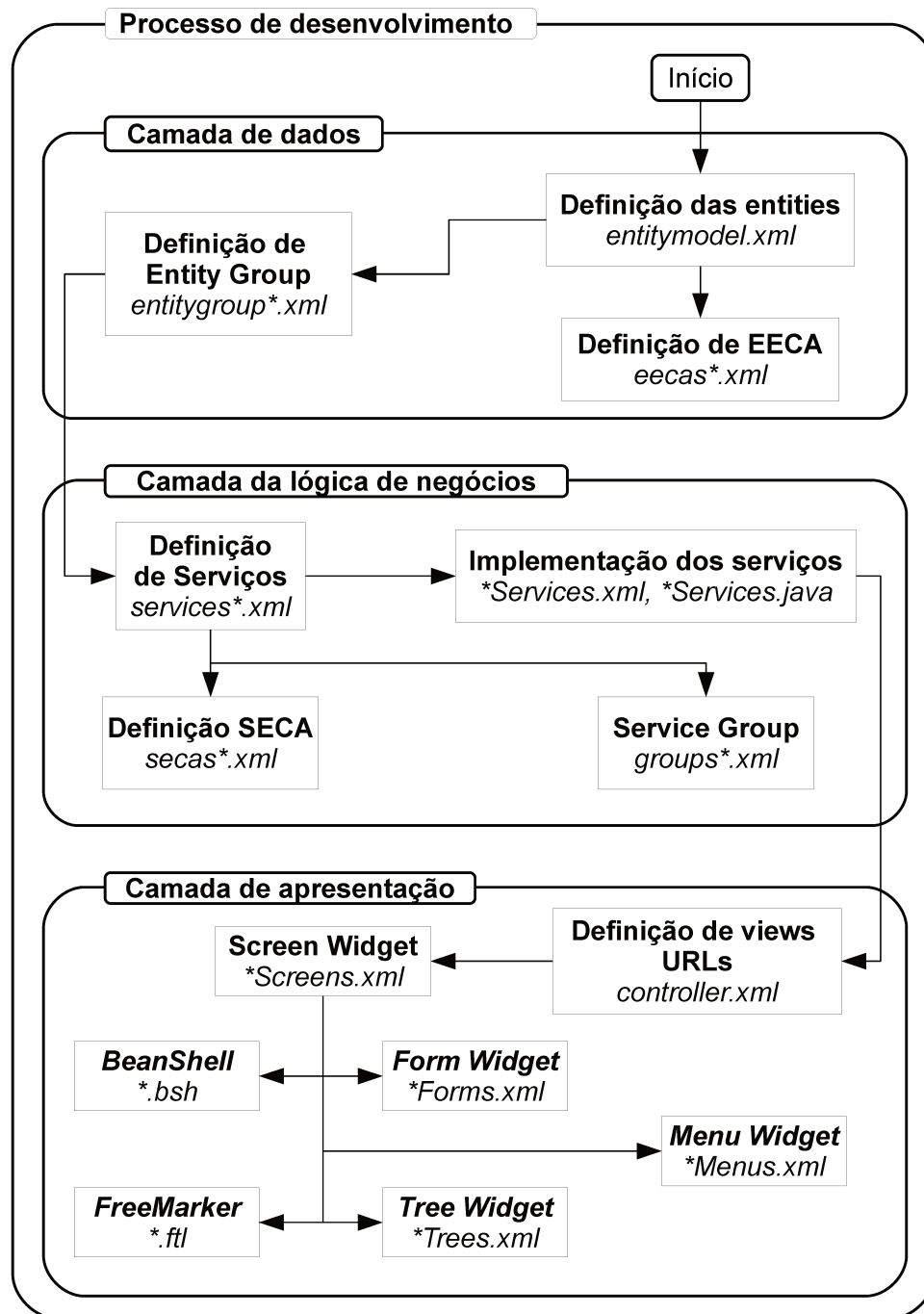


Figura 5.8: Processo recomendado para o desenvolvimento de componentes para o OFBiz

É possível observar que a sequência tida como ideal parte da modelagem da camada de persistência de dados, passa pela camada da lógica de negócios e, então, deve terminar com a criação da interface de usuário, o que é bastante natural.

Além disso, no diagrama da Figura 5.8 são especificados, por alto, os passos para implementação de cada etapa. São apresentadas etapas, dentro de um fluxograma, sendo que cada uma delas contém os arquivos fundamentais para sua implementação. O símbolo asterisco no nome dos arquivos refere-se à convenção de nomes adotada no projeto. Tomando como exemplo o arquivo *entitygroup*.xml*, o símbolo asterisco seria substituído pelo nome do componente ou parte do nome do componente do qual o arquivo faz parte.

Para a camada de persistência de dados, incia-se o trabalho com a definição das tabelas e campos a serem usados, também chamado de *Entity Definition*. De posse das tabelas definidas, é possível agrupá-las para facilitar o acesso (*Entity Group*) caso sejam usados durante o desenvolvimento, vários bancos de dados separados. Ou seja, caso o desenvolvedor opte pelo uso de mais de uma base de dados, o ideal é agrupar as tabelas em grupos definidos no arquivo *entitygroup.xml* para que o acesso por parte do sistema seja mais eficiente.

Ainda para a camada de persistência de dados, recomenda-se que sejam definidas as ações a serem executadas mediante eventos e condições que aconteçam no âmbito das *entities*. Essa definição poderia ser encarada também como algo relevante à camada de lógica de negócios, mas, quando se pensa em manter a coerência entre porções de dados, faz sentido mantê-la nessa camada apenas a título de uma melhor documentação.

Logo em seguida é mostrada a boa prática para o desenvolvimento da camada de lógica de negócios. Segundo a comunidade, o melhor é modelar os serviços que serão implementados, elencando-os todos no arquivo *services*.xml*. Simultaneamente, indica-se que sejam agrupados (*groups*.xml*) de acordo com as necessidades e ainda sejam definidas as ações a serem executadas de acordo com eventos e condições referentes a qualquer serviço, através do arquivo *secas*.xml*.

Com todos os serviços listados e especificados, a próxima etapa é a implementação, seja por meio da *Mini-Language* do OFBiz (**Services.xml*), classes em JAVA (**Services.java*) ou mesmo por qualquer outra das opções disponíveis no *framework*.

Por fim, o diagrama apresenta a sequência indicada para o desenvolvimento da interface gráfica. Em primeiro lugar, indica-se que seja construído o arquivo *controller.xml*, que, como já foi explicado antes, é a fonte das informações necessárias para o sistema reconhecer requisições por URLs. Na realidade, nele não são declaradas apenas as telas, mas também os serviços e suas respectivas URLs. Portanto, trata-se de uma etapa de consolidação, por assim dizer, da criação da camada de lógica de negócios e modelagem da interface.

A partir das telas listadas no arquivo *controller.xml* constrói-se o arquivo **Screens.xml* que possui a implementação das telas na referida metalinguagem. A par da implementação das telas são construídos, também, os arquivos referentes a formulários (**Forms.xml*),

Menus (**Menus.xml*) e as chamadas Trees (**Trees.xml*), que são estruturas utilizadas para hierarquizar formulários ou regiões dentro das telas.

Além de tais definições, o diagrama indica a criação de scripts em *BeanShell* e arquivos *FreeMarker*. O primeiro refere-se a scripts que facilitem a implementação da interface, desde a obtenção de dados, tratamento de informações obtidas a partir de serviços ou, até mesmo, a automatização da criação de elementos como listas presentes na interface.

Já os arquivos *FreeMarker* referem-se a uma técnica usada para a criação de templates para a interface. São, na realidade, uma alternativa à criação das telas usando a metalinguagem baseada em XML. Apesar de constarem no diagrama, a indicação da comunidade é que seja dada preferência aos arquivos XML.

5.6 Funções implementadas pelo OFBiz

Depois de se ter uma ideia da organização do OFBiz como *framework* de desenvolvimento, vale saber também quais são as funções oferecidas para seu uso como ERP propriamente dito.

Em primeiro lugar, há de se explicar que o código do Open For Business, ao menos a parte que mais interessa a um programador, é organizado em quatro diretórios:

- **Framework**, onde estão localizados os códigos responsáveis pelas funcionalidades básicas do *Framework* que consistem na base para o desenvolvimento e funcionamento de outros componentes mais voltado para o usuário final. Três exemplos de componentes desse diretório a citar são: *minilang*, responsável pelo funcionamento da metalinguagem do OFBiz; *widget*, que consiste no mecanismo sobre o qual é desenvolvida a camada de apresentação; e *webtools*, um componente voltado ao usuário para permitir configurações do *Framework* através de uma interface de forma simplificada;
- **applications**, diretório onde os componentes que implementam as principais funcionalidades de um ERP estão localizados e serão citados mais a diante;
- **specialpurpose**, onde são desenvolvidos e distribuídos pela comunidade alguns componentes de finalidade específica como o próprio sistema de comércio eletrônico, e a integração com serviços como Google Checkout e eBay, por exemplo;
- **hot-deploy**, que é o diretório reservado para o desenvolvimento e uso de componentes personalizados. Por convenção, os três diretórios citados acima não devem receber novas funcionalidades, estando reservados apenas para modificações aceitas pela comunidade. Já o *hot-deploy* é indicado como o lugar para se alocar novos componentes personalizados, sendo que o próprio sistema se encarrega do carregamento automático de qualquer componente aí localizado.

As principais funcionalidades implementadas e oferecidas pelo Open For Business estão localizadas nos diretórios *applications* e *specialpurpose*. Seria possível citar cada um dos componentes e explicar quais suas funções, porém muitas das funcionalidades oferecidas pelo sistema são desempenhadas por componentes em conjugado. Dessa forma, são listadas, a seguir, as funções oferecidas com os componentes envolvidos.

- Sistema de Contabilidade e Financeiro - baseado no componente *Accounting*;
- CRM - gestão de clientes, baseado no componente de nome *Party*;
- Recursos Humanos - pelos componentes *Humanres* e *Party*;
- Vendas e Marketing - pelos componentes *Marketing*, *Party*, *Order*, *Workeffort* e *Product*;
- Gestão de Materiais e da Produção - pelo componente *Manufacturing* e *Product*;
- Sistema de Pontos de Venda - baseado no componente *POS*;
- Gestão de Estoques - baseado no componente *Catalog* integrado, principalmente, ao *Ecommerce* e *POS*;
- Comércio Eletrônico - pelo componente *Ecommerce* integrado, principalmente, a *Catalog* e *Product*;
- Pontos de Vendas - pelo módulo *POS* que visa oferecer um sistema para empresas que tenham pontos de vendas espalhados em um grande território;
- Gestão de Conteúdo - pelo componente *Content* que, de modo geral, pode ser considerado um CMS (*Content Management System*) interno do sistema.

Mesmo assim é difícil citar, dentre as funcionalidades de um ERP, quais módulos são responsáveis pela sua implementação no contexto do sistema do OFBiz e, a causa disso, é a natureza de sistema integrado inerente aos ERP.

Um exemplo é o componente *Party* que gira em torno de uma abstração que considera qualquer pessoa ou empresa envolvida em uma negociação ou processo como um participante (*party*). Obviamente, tal componente não desempenha apenas uma função determinada, tal como o cadastro de pessoas para o sistema de recursos humanos, por exemplo. Pelo contrário, implementa a funcionalidade de cadastro e gerenciamento para todos os demais módulos que tenham que fazer uso desse tipo de informação.

Estudando a implementação e a organização do OFBiz fica clara a natureza de integração dos ERP, onde conceitos devem, sempre que possível, ser descritos de forma mais geral possível para permitir a integração entre todas as partes e departamentos de uma empresa.

5.7 Adequação a padrões

Ao se analisar qualquer indústria ou mercado existente, é possível observar que quanto mais importância possuir na vida da sociedade e mais capital movimentar, mais se faz necessária a criação de padrões e normas a serem seguidos.

Do ponto de vista do consumidor, padrões e normas são fundamentais à medida que este deve ter o direito de escolher de quais empresas irá contratar e deve, ainda, ter uma métrica para avaliar a qualidade dos serviços prestados.

Do ponto de vista das empresas, ou daqueles envolvidos no mercado, a criação de padrões facilita o crescimento tecnológico e, idealmente, favorece uma concorrência mais justa.

Apesar de ser um software livre, organizado em torno de uma comunidade, o projeto Apache Open For Business tem a preocupação de se tornar uma solução conhecida e aceita no mercado. O modelo que parece funcionar para o projeto em vários países é o de prestadores de serviços locais que cuidam da adaptação, implantação e treinamento para o uso do sistema.

Para que esse modelo continue funcionando e a meta de se tornar uma solução de importância no mercado seja alcançada, é necessário que o projeto se adeque a padrões usados no mercado de ERPs.

Nesse sentido, a solução adotada pela comunidade do OFBiz foi a implementação dos padrões propostos pelo OAGi (*Open Applications Group*) aplicáveis aos software de gestão empresarial.

Trata-se de uma organização sem fins lucrativos, voltada para o desenvolvimento de padrões, focada principalmente em padrões de negócios baseados em processos para *e-commerce*, *Cloud Computing*, Arquitetura Orientada a Serviços (SOA), *Web Services* e integração entre empresas.

De fato, a própria organização OAGi argumenta que seu padrão OAGIS é o único padrão existente no mundo que realmente oferece padronização para negócios entre indústrias de todas as áreas [10].

Dentre algumas das grandes organizações tecnológicas, que fazem parte do conselho de alta tecnologia do OAGi, estão:

- Cisco Systems;
- Oracle;
- Microsoft;
- SAP;
- Intel.

Em resumo, a adequação do OFBiz aos padrões OAGIS o insere de fato no mercado de ERP, permitindo que haja sua integração com outros sistemas, seja no contexto interno de uma empresa que o adote, ou mesmo na comunicação B2B (*Business to Business*) entre uma empresa que adota o OFBiz e uma outra que utilize o sistema da SAP, por exemplo.

5.8 Considerações finais

Neste capítulo foi apresentado o sistema objeto do estudo de caso proposto neste trabalho, o Apache Open For Business. Foram descritas as principais características dele, como a completa gama de funções, a arquitetura estruturada para um fácil desenvolvimento e a adequação a padrões que a comunidade busca alcançar.

No próximo capítulo, faz-se um estudo a respeito do Sistema Tributário Nacional, para se ter as informações necessárias a fim de analisar a localização do OFBiz segundo aspectos fiscais e contábeis aplicáveis à realidade brasileira.

Capítulo 6

Sistema Tributário Nacional

Qualquer empresa que atue em território brasileiro deve estar consciente e respeitar o Sistema Tributário Nacional. Portanto, a questão tributária é um aspecto fundamental a ser levado em conta para o desenvolvimento e a localização de sistemas de gestão empresarial.

Desde os primórdios do Brasil já existiam impostos. Os primeiros foram as cotas cobradas dos extratores de pau-brasil na época da colônia, os tributos cobrados dos donatários das Capitanias Hereditárias e, mais tarde, as porções exigidas pelo governo português sobre os metais preciosos extraídos do território colonial. A partir da vinda da família real para o Brasil e a posterior independência, começou a se formar o esboço do sistema tributário atual, porém, de maneira geral, os impostos incidiam predominantemente sobre produtos importados.

O sistema evoluiu até a criação do Código Tributário Nacional, em 1966, com a lei 5.172/66 [12], ainda vigente e aceito pela constituição de 1988. Tal código dispõe sobre o Sistema Tributário Nacional e define as normas que regem o direito tributário aplicável à União, aos Estados e aos Municípios. Pode-se dizer que, em comparação aos primórdios, as cobranças foram redirecionadas a impostos internos, taxando atividades industriais, comerciais, imóveis e a renda de profissionais e empresas.

O funcionamento do sistema tributário brasileiro baseia-se nos chamados fatos geradores, que são atividades cotidianas às quais legisladores vinculam o nascimento das obrigações jurídicas de pagar impostos [44], e nas entidades responsáveis pela cobrança dos impostos.

Os principais fatos geradores do sistema brasileiro são: lucro, valor agregado, receita, folha de pagamento, importações, exportações, operações financeiras, remuneração dos dirigentes de empresas e operações e negociações referentes a imóveis. Todos eles dão origem a obrigações de pagamento de impostos que são cobradas pelos órgãos encarregados.

De acordo com a legislação brasileira, há três âmbitos de vigência para o código tributário nacional: a União, os Estados e os Municípios. Cada um deles exerce as com-

petências de cobrança e administração de tributos por meio de entidades eleitas de acordo com as normas do mesmo código. Pode-se considerar quatro órgãos eleitos para desempenhar tal papel, cada um deles responsável pelos respectivos fatos geradores [44]:

- **Secretaria da Receita Federal** - representando a União, é responsável pelos tributos incidentes sobre a renda, produtos industrializados, importação, exportação, operações financeiras, lucro e faturamento das empresas e, por último, sobre a propriedade rural;
- **Instituto Nacional da Seguridade Social** - representando a União, é responsável pelos tributos incidentes sobre a folha de pagamento, também conhecidos por contribuições previdenciárias;
- **Secretarias da Fazenda Estaduais** - presentes em todos os Estados da União, são responsáveis por tributos que incidem sobre a circulação de mercadorias, transportes, comunicações, propriedade de veículos e transmissão de bens por *causa mortis*;
- **Secretarias da Fazenda Municipais** - presentes em cada Município, se encarregam dos tributos incidentes sobre serviços prestados, propriedade urbana e transmissão de bens imóveis.

Os fatos geradores, por sua vez, dão origem às espécies tributárias, que podem ser de cinco tipos [44]:

- **Imposto**, o tributo que não está associado a nenhuma contraprestação direta de serviço ao contribuinte;
- **Taxa**, que é imposta ao contribuinte em contrapartida à prestação de um serviço público;
- **Contribuição de melhoria**, que é o tributo cobrado para compensar o custo de obras públicas para a valorização imobiliária;
- **Contribuição especial**, que são impostos para custear atividades paraestatais, como projetos sociais e intervenções no domínio econômico e de interesse de categorias econômicas ou profissionais;
- **Empréstimo compulsório**, que pode ser instituído exclusivamente por meio de lei complementar em caso de calamidade pública ou guerra externa, ou seja, apenas em casos de urgência.

No Brasil, existem mais de sessenta tributos [22], desde tributos baseados na renda de Pessoas Física e Jurídica até na circulação de mercadorias e produtos pelo território nacional. A seguir são citados e brevemente descritos alguns dos principais tributos brasileiros:

- **IPI** (Imposto sobre Produtos Industrializados) é o tributo federal que incide sobre produtos resultantes da industrialização, mesmo que estes tenham origem em outros países. O valor do IPI é cobrado adicionando seu valor ao valor da nota fiscal do produto e, uma vez que a legislação permite abater o IPI pago sobre matérias-primas, consiste em um imposto não cumulativo. As alíquotas variam de acordo com a classificação de cada produto constante na Tabela de Incidência do Imposto sobre Produtos Industrializados, também conhecida pela sigla TIPI;
- **ICMS** (Imposto sobre a Circulação de Mercadorias e Serviços) é o tributo estadual que incide sobre a movimentação de mercadorias e, em alguns casos, sobre serviços como das prestadoras de telefonia. Quando se trata da entrada de matérias-primas, que constituam o processo produtivo da empresa, não é cumulativo por permitir sua dedução. As alíquotas variam de 7% a 25%, mas são sujeitas a legislação de cada estado;
- **ISS** (Imposto sobre Serviços) é o tributo municipal cobrado sobre serviços prestados. Varia de 2% a 5% e é cobrado pelo município onde o serviço for prestado sem levar em conta a origem do prestador;
- **IRPJ** (Imposto de Renda de Pessoa Jurídica) é o imposto cobrado sobre o lucro das empresas e varia de acordo com o porte de cada uma delas e a opção entre Lucro Real, Presumido ou Arbitrado;
- **IRPF** (Imposto de Renda de Pessoa Física) é o imposto cobrado sobre a renda dos indivíduos assalariados ou autônomos, variando da isenção até alíquotas de 27,5%;
- **COFINS** (Contribuição para o Financiamento da Seguridade Social) é a contribuição federal destinada a financiar a seguridade social, taxando a receita bruta de empresas. Para empresas, que optam pela taxaço pelo lucro real, sua alíquota é de 7,6%, para as demais é de 3%;
- **PIS** (Programa de Integração Social), assim como o COFINS, é taxado da mesma forma sobre a receita, com exceção se for uma alíquota menor. Destina-se a financiar o pagamento do seguro-desemprego e o abono de trabalhadores assalariados com, no máximo, dois salários mínimos;
- **CSSL** (Contribuição Social Sobre o Lucro Líquido) é um tributo com alíquota de 9% que é cobrado de forma semelhante ao IRPJ, onde a empresa paga de acordo com a escolha entre Lucro Real, Presumido ou Arbitrado;
- **INSS** (Instituto Nacional da Seguridade Social) consiste, na realidade, em uma contribuição descontada da folha de pagamento de trabalhadores destinada a esse órgão responsável pela Previdência Social e o pagamento de aposentadorias;

- **FGTS** (Fundo de Garantia por Tempo de Serviço) é o valor de 8% que as empresas pagam sobre o valor da folha de pagamento para que o trabalhador, ao ser dispensado sem justa causa, possa sacá-lo como compensação.
- **IOF** (Imposto sobre Operações Financeiras) é o imposto federal que taxa as operações de financiamento, câmbio, seguro, aplicações e leasing;
- **II** (Imposto sobre Importações) é o tributo que visa proteger a indústria nacional taxando produtos importados em concorrência desleal aos produtos nacionais;
- **IPVA** (Imposto sobre Propriedade de Veículos Automotivos) é o tributo cobrado sobre a propriedade de automóveis, variando de acordo com o ano e tipo do veículo;
- **IPTU** (Imposto sobre Propriedade Territorial e Urbana) é o imposto cobrado sobre o valor de imóveis, variando de acordo com o valor e categoria deles;
- **ITBI** (Imposto sobre Transmissão de Bens Imóveis) é o tributo que incide sobre o ato de lavratura de contrato ou promessa de compra e venda de bens imóveis;
- **TFE** (Taxa de Fiscalização de Estabelecimentos) é uma taxa municipal obrigatória para financiar os serviços de fiscalização;
- **CIDE** (Contribuição de Intervenção no Domínio Econômico) é o tributo que incide sobre a comercialização de petróleo e derivados, de gás natural e derivados, álcool etílico combustível e sobre o pagamento de *royalties* ao exterior;

No trabalho de localização de um software de gestão empresarial, todos esses impostos, contribuições e taxas devem ser levados em conta. Porém, o escopo da localização de aspectos fiscais e contábeis do presente trabalho limita-se ao estudo de localização de aspectos mais ilustrativos do desafio de localização de funcionalidades que possam ser usados para determinar a dificuldade desse tipo de trabalho e a sugestão de uma metodologia a ser adotada.

Com esse intuito, daqui em diante são considerados mais relevantes os tributos de ICMS e IPI para os fins deste estudo.

6.1 Obrigações Acessórias

As espécies tributárias são consideradas como as Obrigações Tributárias que o Estado pode exigir de contribuintes a partir dos fatos geradores correspondentes.

Tais obrigações são constituídas pelas obrigações principais, que são os pagamentos dos tributos propriamente ditos, e as obrigações acessórias, que são os procedimentos para tornar possível a determinação do montante a ser pago.

As obrigações acessórias permitem ao fisco manter o controle sobre as quantias devidas por empresas de acordo com os fatos geradores a que estão sujeitas. Cada tributo possui uma obrigação acessória correspondente. No que diz respeito ao IPI e ao ICMS, as obrigações acessórias são a emissão de documentos fiscais, como a nota fiscal modelo 1 e 1A, e os livros fiscais.

6.1.1 Nota fiscal modelo 1 e 1A

Talvez um dos documentos fiscais mais conhecidos e representativos, a nota fiscal exerce várias funções nos âmbitos comercial, empresarial, fiscal e jurídico.

No campo comercial, a nota fiscal tem por finalidade comprovar transações de compra, venda ou prestações de serviços, registrando as partes envolvidas e o produto ou serviço negociado. No âmbito empresarial permite a apuração de receita e débito e a autorização de crédito de imposto.

Para fins fiscais, é o documento utilizado para respaldar a circulação das mercadorias ou a prestação de serviços. Já no campo jurídico garante o direito do vendedor ou prestador de serviço receber a quantia devida e a satisfação do comprador ou contratante.

A emissão de notas fiscais é obrigatória a todos os contribuintes do ICMS e IPI [5] para acobertar a venda de produtos e prestação de serviços, ficando a salvo das sanções aplicadas pelo fisco ao descumprimento das normas prevista na legislação.

Existem dois modelos de nota fiscal: Modelo 1 e Modelo 1A. Ambos seguem as mesmas normas, possuem os mesmos campos e se destinam ao mesmo fim. A única diferença está no formato. A primeira (Modelo 1) é disposta no chamado formato retrato, ou seja, com a menor dimensão da folha de papel sendo a largura. A segunda (Modelo 1A) é disposta no chamado formato paisagem, em que a menor dimensão da folha de papel é a altura.

Vale a pena apresentar um modelo padrão de nota fiscal e elucidar seus principais campos. A Figura 6.1 ilustra o Modelo 1 de nota fiscal.

No topo do documento são impressos os dados da empresa emitente da nota e se o tipo a nota é de saída ou entrada. A próxima porção do documento é destinada aos dados do destinatário ou remetente, dependendo do tipo da nota.

Nesta porção, além dos mais óbvios, vale ressaltar os campos de *Natureza de Operação* e *CFOP*. Trata-se da descrição da operação à qual a nota está relacionada. O código CFOP, ou Código Fiscal de Operação e de Prestações, estabelecido pelo convênio S/Nº de 15 de Dezembro de 1970, define a natureza exata da operação e é um dos campos mais relevantes da nota fiscal. São números de 4 dígitos, cujo primeiro algarismo define a origem do produto ou serviço de acordo com a Tabela 6.1.

Os três demais algarismos definem a descrição da operação ou prestação. O conjunto de três algarismos 101 junto com os algarismos iniciais 1, 2 ou 3, por exemplo, significa a compra para industrialização ou produção rural. O descritor 1.101, por exemplo, indica

[illegible]

Figura 6.1: Modelo 1 da Nota Fiscal

que se trata de uma compra de um fornecedor no mesmo estado, 2.101 de um fornecedor de outro estado e 3.101 de um fornecedor estrangeiro.

Logo a seguir vem um quadro destinado aos dados dos produtos que são objetos da transação registrada pela nota. Nessa porção podem ser listados quantos produtos forem necessários. O primeiro campo é o do código do produto, que nada mais é que o código adotado para descrevê-lo internamente na empresa, sem nenhuma relação com o fisco. Ao lado há a coluna para a descrição de cada produto, que serve exclusivamente para a descrição e também não segue nenhuma norma específica.

Em seguida há dois campos de grande interesse. O primeiro é o de “Classificação Fiscal”, que é um código de 8 dígitos descritor de cada classe de produtos para facilitar sua tributação pelo IPI. Todos os códigos existentes são elencados na chamada Tabela do IPI ou TIPI [5]. Todas empresas contribuintes do IPI são obrigadas a preencher este campo, sendo que as demais podem preenchê-lo com códigos próprios desde que acompanhados de uma tabela explicativa.

O segundo campo é o da “Situação Tributária” ou CST. Trata-se do Código de Situação

Significado do primeiro algarismo no CFOP	
Algarismo	Significado
1	Entradas ou Aquisições de Serviço do Estado , ou seja, o remetente encontra-se na mesma unidade federativa do destinatário.
2	Entradas ou Aquisições de Serviço de Outros Estados , ou seja, o remetente encontra-se em uma unidade federativa diferente do destinatário.
3	Entradas ou Aquisições de Serviço do Exterior , ou seja, o remetente encontra-se em outro país.
5	Saídas ou Prestações de Serviços para o Estado , ou seja, o destinatário encontra-se na mesma unidade federativa do remetente.
6	Saídas ou Prestações de Serviços para outros Estados , ou seja, o destinatário encontra-se em uma unidade federativa diferente do remetente.
7	Saídas ou Prestações de Serviços para o Exterior , ou seja, o destinatário encontra-se em outro país.

Tabela 6.1: Significado do primeiro algarismo no CFOP

Tributária, que é composto por três dígitos destinados a demonstrar a origem da mercadoria e sua tributação quanto ao ICMS. Esses códigos são obtidos junto a duas tabelas, a Tabela [?], de Origem da Mercadoria que determina o primeiro dígito, e a Tabela [?], de Tributação pelo ICMS que determina os dois outros.

Origem da Mercadoria	Código
Nacional	0
Estrangeira - Importação Direta	1
Estrangeira - Adquirida no mercado interno	2

Tabela 6.2: Códigos de Situação Tributária - CST

Logo depois estão os campos para discriminar as unidades em que são medidas as quantidades de cada produto, a quantidade propriamente dita, o valor por unidade e o valor total. Os últimos campos, no quadro dos dados do produto, são as alíquotas de ICMS e IPI, determinadas a partir do CST e da classificação fiscal, e o valor de IPI para cada produto.

Após o quadro dos dados dos produtos está presente a porção destinada aos dados relevantes para o cálculo dos impostos. Nela são explicitadas, além dos dados referentes ao IPI e ICMS, o valor do frete e do seguro, que, dependendo da venda, pode ser utilizado

Tributação pelo ICMS	Código
Tributada integralmente	00
Tributada com cobrança de ICMS por substituição tributária	10
Com redução da base de cálculo	20
Isenta ou não tributada e com cobrança de ICMS por substituição tributária	30
Isenta	40
Não tributada	41
Com suspensão	50
Diferimento	51
ICMS cobrado anteriormente por substituição tributária	60
Com redução da base de cálculo e cobrança do ICMS por substituição tributária	70
Outras	90

Tabela 6.3: Códigos de Situação Tributária - CST

na base de cálculo dos impostos.

Para cumprir o papel de acobertar a circulação das mercadorias, a nota fiscal deve registrar os dados referentes ao responsável pelo transporte e os volumes transportados. Essa é a função da porção seguinte aos dados para o cálculo dos impostos, registrando os dados de quem transporta a mercadoria e uma descrição da mercadoria sendo transportada, que inclui a quantidade, marca, número, peso bruto e peso líquido.

Por fim, há uma região reservada para dados adicionais, na qual o campo de informações complementares é normalmente destinado ao emissor da nota, para especificar algum dado explicativo do documento, quando necessário, e o campo reservado ao fisco a ser preenchido exclusivamente pelo fisco.

6.1.2 Livros Fiscais

A fim de manter controle sobre todas as transações efetuadas pelas empresas e verificar se nenhuma irregularidade ocorre quanto ao pagamento de impostos, o Estado instituiu a obrigação de empresas manterem os chamados Livros Fiscais.

Há vários tipos de livros fiscais, cada um deles com uma finalidade diferente relacionada a impostos ou obrigações com o fisco. Alguns dos principais livros fiscais são:

- **Livro de Registro de Entradas de Mercadorias**, destinado a registrar todas as aquisições realizadas pela empresa, mesmo aquelas que não acarretem o pagamento de impostos;

- **Livro de Registro de Saídas de Mercadorias**, destinado a registrar todas as operações de vendas realizadas pela empresa. É uma das principais fontes de informação para aferir o montante de impostos a ser recolhido por ela;
- **Livro de Apuração do ICMS**, que cumpre a função de registrar débitos e créditos de ICMS, determinando se a empresa tem ou não encargos a serem pagos ao fisco;
- **Livro de Apuração do IPI**, que funciona de forma análoga ao Livro de Apuração de ICMS, registrando débitos e créditos de IPI para aferir quanto deve ser pago ao fisco;
- **Livro de Apuração do ISS**, que, da mesma forma que outros dois anteriores, registra todos os serviços prestados para poder apurar o ISS devido aos municípios correspondentes;
- **Registro de Inventário**, que destina-se ao controle de mercadorias e materiais não comercializados ou consumidos durante o exercício comercial;

Além desses, há também os Livros de Registro de Controle de Produção e do Estoque, de Registro do Selo Especial de Controle, Registro de Impressão de Documentos Fiscais, dentre outros. Em geral, pode-se dizer que a função deles é manter dados organizados para que o Estado possa fiscalizar as operações das empresas e elas próprias possam manter-se em dia com suas obrigações tributárias.

6.2 Sistema Público de Escrituração Digital - SPED

O projeto do Sistema Público de Escrituração Digital (SPED), criado em 2007, consiste em um esforço de modernizar a forma como as obrigações acessórias são cumpridas pelos contribuintes. O projeto visa definir padrões de arquivos para transmissão, por meio eletrônico, dos documentos que constituem a relação entre o fisco e os contribuintes, usando para tal a certificação digital para garantir a validade jurídica destes documentos digitais [14].

Os principais objetivos do SPED são:

- **Promover a integração ao fisco**, padronizando e compartilhando as informações fiscais e contábeis;
- **Racionalizar e uniformizar as obrigações acessórias dos contribuintes**, estabelecendo um meio para transmissão única de obrigações acessórias distintas para os diferentes órgãos responsáveis;

- **Acelerar a identificação de ilícitos tributários**, facilitando o acesso e cruzamento de informações durante auditorias.

O Sistema Público de Escrituração Digital é composto por vários subprojetos: SPED Contábil (ECD), SPED Fiscal (EFD), Nota Fiscal Eletrônica (NF-e), Nota Fiscal de Serviços eletrônica (NFS-e), CT-e, e-Lalur e Central de Balanços [17].

É importante ressaltar que um dos principais objetivos é a questão de integração e padronização. Isso significa que vários órgãos de fiscalização terão acesso às informações de forma rápida e eficiente [5]. A Figura 6.2 mostra um esquema que demonstra tal integração, onde todos os órgãos de fiscalização poderão consultar e cruzar as informações.

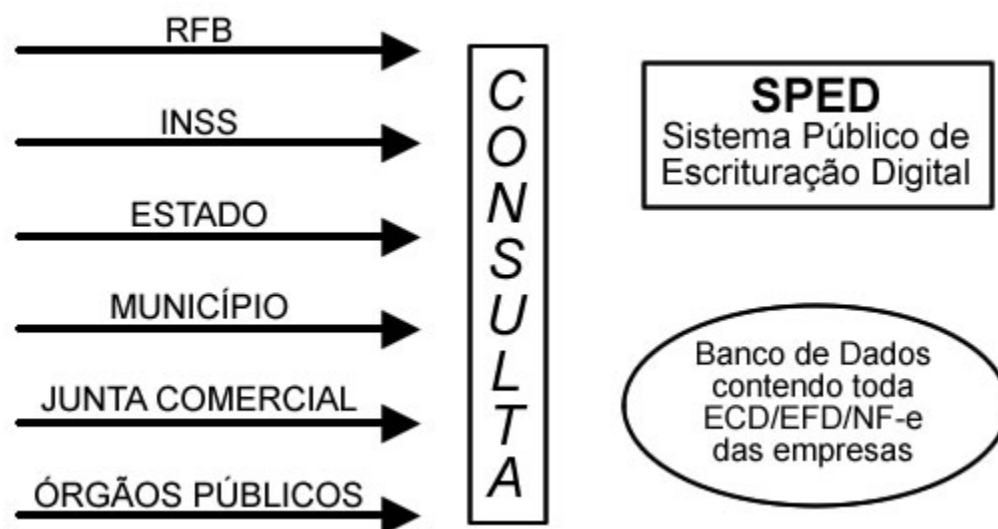


Figura 6.2: Integração que o SPED promete alcançar

Além dessa integração entre vários órgãos, a digitalização de tais dados permitirá à Receita Federal aplicar cada vez mais métodos de fiscalização baseadas em técnicas que automatizem a tarefa, aumentando a eficiência e diminuindo a necessidade da participação de seus funcionários. Será possível, por exemplo, aplicar algoritmos de inteligência artificial para encontrar possíveis práticas ilícitas.

Alguns deles ainda se encontram em fase inicial de desenvolvimento e outros não se adequam a esse estudo de localização. A seguir são citados os três subprojetos com maior potencial de integração com o estudo realizado, com destaque para o equivalente da nota fiscal, que, como já foi ressaltado anteriormente, é o de maior interesse para o presente projeto.

6.2.1 SPED - Contábil - ECD

O ECD, ou Escrituração Contábil Digital, consiste, basicamente, em um esforço para a substituição dos livros de escrituração mercantil por seus equivalentes digitais.

A empresa gera um arquivo digital, também referido como Livro Diário Digital ou Escrituração Contábil Digital, no formato especificado no anexo único à Instrução Normativa RFB nº 787/07, constante na Portaria nº 11.211/2007.

A seguir os responsáveis submetem tal arquivo ao Programa Validador e Assinador (PVA), também fornecido pelo SPED, e, em seguida, o transmitem à Receita com auxílio do o programa Receitanet. Por fim, a empresa tem a opção de impressão de um comprovante da entrega do arquivo.

Já o SPED, por sua vez, extrai um resumo do arquivo composto pelo requerimento, o Termo de Abertura e o Termo de Encerramento. Todas as informações são, então, disponibilizadas à Junta Comercial competente. A partir daí cabe a tal Junta analisar o Livro Diário Digital. O SPED disponibiliza para as empresas a possibilidade de consulta ao andamento e resultado dessa análise [15].

Toda a iniciativa do SPED Contábil é regulamentada pelos seguintes documentos:

- Decreto nº 6.022, de 22 de janeiro de 2007;
- Instrução Normativa nº 787/2007;
- Portaria nº 11.211/2007;
- Instrução Normativa DNRC nº 107/2008;
- Ato Declaratório Cofis nº 36/2007;
- Ato Declaratório Cofis nº 20/2009.

6.2.2 SPED - Fiscal - EFD

O SPED Fiscal resume-se a um arquivo chamado Escrituração Fiscal Digital (EFD), composto por um conjunto de escriturações de documentos fiscais, informações relevantes ao fisco e registros de apuração de impostos referentes às operações e prestações praticadas pelo contribuinte.

Assim como o SPED Contábil, o Fiscal deve ser gerado segundo um padrão, nesse caso definido em Ato COTEPE, e, então, validado e assinado pelo Programa Validador e Assinador, que possibilita, por fim, o envio para a Receita [16].

A Figura 6.3 esquematiza o processo de produção e submissão das informações ao sistema do SPED. O mesmo fluxo nela apresentada pode ser usado para o entendimento do ECD.

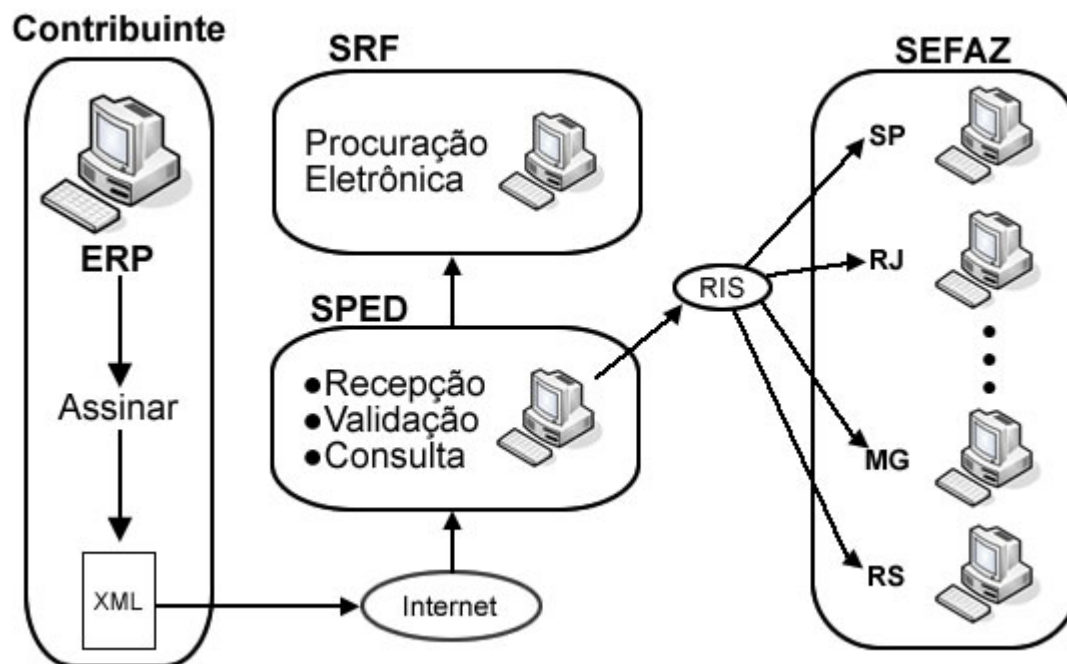


Figura 6.3: Transmissão de dados do EFD para a o SPED. Depois de transmitidos para o SPED são distribuídos para as SEFAZ de cada estado pela chamada Rede de informações (RIS)

Toda a iniciativa do SPED Fiscal é regulamentada pelos seguintes documentos:

- Convênio ICMS nº 143/2006;
- Ajuste Sinief nº 2/2009;
- Ato Cotepe nº 9/2008 e alterações posteriores;
- Ato Cotepe nº 38/2009, que altera o Anexo único do Ato Cotepe nº 9/2008.

6.2.3 NF-e - Ambiente Nacional

O Projeto Nota Fiscal Eletrônica (NF-e), coordenado pelo ENCAT (Encontro Nacional dos Administradores e Coordenadores Tributários Estaduais) e desenvolvido em parceria com a Receita Federal, tem como objetivo a substituição da forma tradicional da emissão da nota fiscal em papel por nota fiscal eletrônica juridicamente válida para todos os fins.

Basicamente, a empresa emissora da nota fiscal eletrônica gera um arquivo eletrônico com as informações fiscais da operação comercial, valida e assina digitalmente tal documento e, então, o transmite para a Secretaria da Fazenda da jurisdição do contribuinte. A

partir disso, é feita a pré-validação do arquivo e a emissão de um protocolo de recebimento [13], desde que validado e aceito o documento.

Além de ser enviada para a Secretaria da Fazenda responsável, a NF-e é enviada também para a Receita Federal, que deve funcionar como repositório nacional de todas as notas fiscais eletrônicas.

A nota fiscal tradicional, não digital, servia dentre outras funções, para acobertar a circulação de mercadorias. Da mesma forma a nota fiscal eletrônica necessita de um artifício para desempenhar a mesma função. Isso é alcançado com o chamado DANFE, ou Documento Auxiliar da Nota Fiscal Eletrônica.

Ao fim do envio da NF-e, o contribuinte tem a opção de imprimir o DANFE para garantir a circulação da mercadoria. Em termos gerais, tal documento auxiliar, além de todas informações do produto ao qual a nota se refere, possui a chave de acesso para que a nota possa ser verificada por fiscais. Essa chave é representada pelo seu código numérico e também por um código de barras unidimensional que facilita sua consulta.

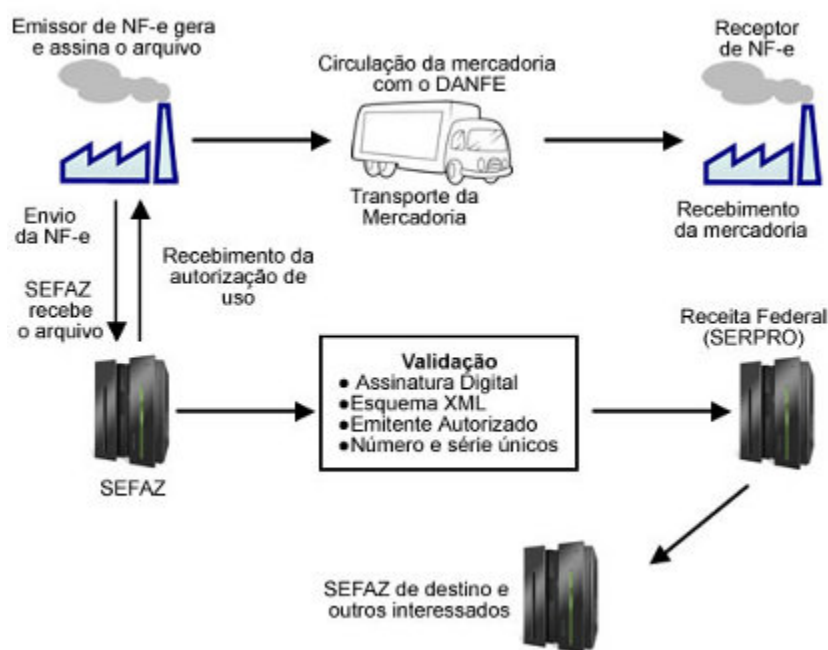


Figura 6.4: Interação entre contribuinte e SEFAZ e o uso do DANFE para a circulação de mercadorias

A Figura 6.4 [5] mostra um esquema do funcionamento da NF-e, como ela é gerada, processada, distribuída entre os órgãos responsáveis e, ainda, como o DANFE acoberta o transporte da mercadoria.

A iniciativa da Nota Fiscal Eletrônica é regulamentada pelos seguintes documentos:

- Ajuste Sinief nº 7/2005 e alterações;

- Convênio ICMS nº 110/2008;
- Protocolo ICMS nº 10/2007 e alterações;
- Protocolo ICMS nº 42/2009;
- Protocolo ICMS nº 55/2007 e alterações;
- Ato Cotepe nº 49/2009.

6.2.4 Certificação Digital utilizada no projeto SPED

Dentro do contexto do sistema de escrituração convencional, a validade e autenticidade dos documentos eram garantidas por assinaturas, papéis especiais, carimbos e outros instrumentos de mesma natureza.

No caso do novo Sistema Público de Escrituração Digital, a validade dos documentos é garantida por mecanismos baseados em certificação digital.

De forma geral, pode-se descrever o sistema todo como sendo formado pelas entidades certificadoras e os emitentes de documentos fiscais. As entidades certificadoras são reguladas e certificadas pelas normas impostas pelo ICP-Brasil (Infraestrutura de Chaves Públicas Brasileira), sendo elas responsáveis pela emissão de certificados digitais utilizados nas assinaturas necessárias para a emissão dos documentos do SPED.

As entidades emissoras, por sua vez, são as empresas, ou prestadores de serviço, que emitem os documentos e os assinam eletronicamente para poder transmiti-los à Receita. Para tal é necessário que possuam credenciais para fazer a assinatura, ou seja, obter os certificados das entidades certificadoras.

Tais credenciais, em outras palavras, são as versões eletrônicas do CPF e CNPJ, chamados de e-CPF e e-CNPJ, respectivamente. Para cada um dos dois há dois tipos: o A1 e o A3.

O primeiro, o tipo A1, gera os pares de chaves pública e privada no disco rígido de um computador e é válido por um ano. Já o segundo, o tipo A3, usa um hardware criptográfico dedicado para a geração das chaves. Tal hardware pode ser um *smart card* ou um *token* criptográfico.

A real diferença entre os dois tipos de certificados é a segurança que proporcionam ao processo. Os do tipo A3, por contarem com hardware criptográficos, são considerados praticamente invioláveis.

Para cada projeto do SPED há especificações que regem como devem ser assinados. Grosso modo, a NF-e e EFD podem ser assinados com qualquer um dos dois tipos. Por outro lado, para o ECD é obrigatório o uso do tipo A3.

6.3 Considerações finais

O estudo do Sistema Tributário Nacional apresentado neste capítulo, teve como principal objetivo apresentar a sua estrutura e os principais aspectos que devem ser levados em conta para a localização de um Sistema de Gestão Empresarial.

Apesar de não ser um estudo minucioso, fica clara a quantidade de detalhes e especificidades existentes no Sistema Tributário. Serve também para dar uma ideia dos esforços da Receita Federal em agilizar o sistema e torná-lo cada vez mais eficiente, lançando mão do uso da tecnologia da informação com o projeto SPED.

Este capítulo encerra a porção do trabalho que foca no levantamento de informações para dar uma base sólida ao estudo de localização proposto. O próximo capítulo consiste em algo de ordem mais prática e relacionado diretamente com o a proposta deste trabalho. Nele é feita uma análise mais detalhada do OFBiz segundo os aspectos levantados nos capítulos anteriores.

Capítulo 7

Análise para Localização

Como já foi explicitado, a intenção deste trabalho é avaliar questões relevantes relacionadas ao Open For Business quanto à sua localização à realidade brasileira.

A literatura existente a respeito das práticas de internacionalização e localização de software, ao menos no contexto acadêmico, costuma focar apenas nos aspectos tradicionais, como o desacoplamento entre o código e os textos que compõem a interface, os formatos numéricos, imagens e símbolos.

Por outro lado, alguns dos documentos existentes nessa área, que advêm de iniciativas do mercado, levam em conta também outros aspectos que devem ser considerados quando se deseja adotar uma postura internacional diante da distribuição de um software. Normalmente, possuem uma visão mais ampla do desafio, levando em consideração o processo de globalização no desenvolvimento, e, ao mesmo tempo, mais prática, abordando até aspectos como a mão-de-obra necessária para se levar adiante um processo dessa natureza.

No entanto, é difícil encontrar, em ambas as fontes, informações relativas à da localização de funcionalidades. Talvez possa até ser argumentado que, ao se localizar funcionalidades de um software, não se está realmente o localizando, mas sim, alterando o software em questão. De fato isso pode ser verdade para algumas categorias de software, porém, ao menos no âmbito dos ERP, fica claro que, se este aspecto não for levado em conta, os sistemas podem nunca vir a ser adotáveis de fato na cultura alvo da localização.

Dessa maneira, é necessário que se leve em consideração, no presente trabalho, outros aspectos de localização, além daqueles tradicionais, incluindo e estudando outros que realmente podem impactar a adoção do software no país.

Nas subseções seguintes, a ideia é abordar tanto os aspectos tradicionais quanto se fazer um apanhado daqueles que a literatura não leva em conta normalmente, até por se tratar de algo específico do domínio dos ERP e da realidade brasileira. Isso servirá de base para, mais adiante, se propor um plano para o estudo do particular caso da localização, escolhendo o que de fato será localizado e onde será necessária uma análise para a sugestão de diretrizes e possibilidades.

Além disso, é feita, também, uma análise das dimensões dos arquivos que contém todos os textos da interface do sistema para que se possa ter uma ideia da ordem do volume requerido da tradução. Completando, é feita também uma análise das traduções já existentes para avaliar o sucesso ou o estágio das localizações, ao menos no que diz respeito à tradução em outras línguas.

7.1 Análise da internacionalização do OFBiz

Antes de se pensar nos aspectos mais específicos e complexos na localização do OFBiz, é necessário considerar aqueles que podem ser chamados de básicos. Para o planejamento da localização é necessário se ter uma ideia da qualidade da internacionalização já existente. Para tal, a seguir são discutidos aqueles aspectos de internacionalização fundamentais para que até a localização mais básica possa ser executada.

7.1.1 Análise do desacoplamento entre texto, código e layout

Por ser um sistema baseado no modelo conceitual MVC (*Model-view-controller*), o OFBiz tem por natureza um bom desacoplamento entre o código e o layout de suas telas. Como visto no Capítulo 5, que aborda a sua Camada de Apresentação, a interface web é gerada pelo chamado *Screen Widget*. Ou seja, a partir de arquivos XML, que descrevem o que se espera da interface, são geradas as páginas HTML.

Na hipótese de ser necessário alterar o formato original da interface do sistema, talvez por aspectos como expansão ou retração muito acentuada do texto que compõem a interface, por exemplo, várias abordagens poderiam ser utilizadas. Pensando-se apenas no HTML gerado pelo *Screen Widget*, é possível formatá-lo com o auxílio de CSS (*Cascading Style Sheets*).

Na realidade, o OFBiz já possui vários *templates* (modelos) para a interface e uma abordagem possível para um projeto de localização que necessitasse adaptar a camada de apresentação talvez fosse a criação de um novo *template*.

De fato, o uso de CSS é algo adotado na implementação do projeto para resolver problemas de internacionalização e localização. O problema para atender a línguas que são escritas da direita para esquerda (RTL - *Right to Left Languages*), por exemplo, é resolvido usando-se justamente esse artifício.

Há uma configuração no *framework* para se optar pela orientação da direita para a esquerda. O que ela faz basicamente é adicionar ao HTML final, algumas classes de CSS para alterar a orientação do texto; uma solução bastante elegante que demonstra o desacoplamento existente entre o código e a interface.

Outro aspecto, que é considerado uma preocupação constante em textos da área, é a questão da codificação de caracteres. Por ser desenvolvido em JAVA, o Open For Busi-

ness possui suporte nativo ao Unicode, o que atualmente resolve quase qualquer problema referente à codificação de caracteres.

Voltando às questões de desacoplamento, falta abordar o que talvez seja o mais importante: o desacoplamento entre os textos que compõem a interface e a interface propriamente dita.

Ainda no Capítulo 5 que abordava a Camada de Apresentação do OFBiz, foi explicado como as traduções para diversas línguas são tratadas no contexto do projeto. Normalmente em sistemas JAVA a solução adotada baseia-se nos arquivos *.properties*, no entanto a comunidade do projeto optou pelo uso de arquivos XML como ilustrado na Figura 5.4.

Basicamente, os arquivos XML com sufixo **.labels*, localizados nas pastas *config* de cada componente possuem uma lista dos itens textuais que compõe a interface. Estes itens são organizados em *tags* de nome *property* e possuem outras *tags* de nome *value* abaixo deles na estrutura XML. O conteúdo das *tags value* são justamente as traduções para as diversas línguas, sendo distinguidas pelo atributo *lang*, que recebe a sigla do *locale* correspondente.

Esses arquivos que possuem os rótulos são utilizados, normalmente, dentro dos arquivos XML referentes ao *Screen Widget*. Quando se deseja carregar um deles para preencher a tela em questão, há métodos que trazem a tradução correta de acordo com a configuração da máquina do usuário. Caso não haja a tradução correspondente ao sistema operacional do usuário, o sistema carrega por padrão o *locale* inglês.

Além de proporcionar um forte desacoplamento entre o texto e o código do programa, essa abordagem facilita a tradução e manutenção das traduções. Isso porque agrupa em um mesmo arquivo todas as línguas. E melhor ainda, agrupa em *tags* todas as traduções para um mesmo componente da interface.

7.1.2 Análise de formatos numéricos e de data

Um dos aspectos esperados a ser alcançado por uma boa internacionalização é que permita adaptar a maneira como datas, formatos numéricos e moedas são tratados na interface de usuário de acordo com o idioma desejado.

Quanto às questões de formatos numéricos e de moeda, o OFBiz é bem internacionalizado e até localizado também. Além de já ter uma lista bastante completa de moedas com seus símbolos e nomes, tem uma integração perfeita com a língua escolhida pelo usuário.

Se o sistema estiver configurado para usar o português brasileiro, por exemplo, seja por detecção automática ou escolha do usuário, haverá uma tentativa de exibir todos os preços na moeda local, o Real no caso.

Na realidade, o sistema se mostra bem adaptado não apenas ao uso local, mas também em âmbito global. Isso porque, no momento do cadastro de produtos no sistema, seja para vendas em loja, pelo sistema de E-commerce ou qualquer outro canal de vendas, é possível cadastrar o preço em várias moedas diferentes.

Cada produto pode ter seu preço configurado em várias moedas, dentre aquelas às quais o OFBiz dá suporte. No caso do Real, o suporte já é nativo. E de fato, será difícil encontrar uma moeda não apoiada, pois para cada *locale* contemplado pelo OFBiz há uma moeda associada.

Em uma outra etapa, quando há a exibição do preço, o sistema exibe o preço do produto preferencialmente na moeda local. Caso não haja um preço cadastrado que atenda a essa condição, a próxima tentativa é exibir o preço em Dólares Americanos.

Essa característica permite, além do uso do sistema em vários países, a interação entre uma empresa com clientes ao redor do mundo. É claro que a manutenção de tantos preços em tantas moedas pode ser complicado, porém talvez seja possível encontrar soluções para conversão automática de preços baseando-se no sistema de serviços e eventos do OFBiz.

O formato numérico, por sua vez, também é levado em conta de forma bastante eficiente na internacionalização do sistema. É também associado ao *locale* escolhido ou detectado a partir das preferências do usuário.

Quando configurado para o português, por exemplo, a separação das casas decimais é a vírgula e o separador do milhar o ponto, assim como era de se esperar. Mesmo moedas estrangeiras, no caso de não haver o preço cadastrado na moeda correspondente à do usuário, são exibidas de acordo com a configuração de formato numérico local.

Em contraste, o formato de datas não é tão bem implementado do ponto de vista da internacionalização. O que acontece é que o formato utilizado por padrão no Open For Business não é nem mesmo o formato utilizado em inglês. Era de se esperar que a data fosse exibida de forma semelhante ao formato americano, por exemplo, na ordem “mês-dia-ano”. Porém o formato utilizado é o mesmo que o JDBC do JAVA utiliza, ou seja, na ordem “ano-mês-dia”.

Analizando-se bem, essa é uma forma bastante prática de implementação, mas não é a melhor quando se pensa no ponto de vista do usuário. O resultado é que o sistema não possui uma interface de configuração, nem mesmo um arquivo de parâmetros, para alterar a exibição e a edição das datas pelo usuário. Claramente isso consiste em uma quebra da heurística 3 de usabilidade, presente no Apêndice A.

Esse problema já foi apontado na comunidade por um francês de nome Nicolas Malin no sistema de gerenciamento de projeto, defeitos e requisições do projeto OFBiz [36], o JIRA Issue. Na época, Nicolas precisava prestar um serviço a um cliente também francês que precisava que a data fosse exibida no formato “dia-mês-ano”.

Ele mesmo submeteu ao projeto um *patch* que corrigia o problema para exibição no formato desejável, que, por coincidência, é o mesmo utilizado em português. O problema é que, do ponto de vista de projeto, a qualidade do *patch* não era condizente com as diretrizes do projeto, além de não ser perfeito.

O *patch* não resolvia totalmente o problema e alguns membros da comunidade ainda argumentaram que traria alguns problemas de compatibilidade com alguns dos componentes

e bibliotecas utilizadas pelo *framework*.

Até hoje essa é uma questão em aberto e precisa ser resolvida. Infelizmente a internacionalização deixa a desejar nesse aspecto. É claro que é possível usar o referido patch para adaptar o sistema ao uso brasileiro, mas isso obriga a se ter uma atenção maior para fazer a manutenção e atualização de uma instalação do sistema e pode não ser tão trivial para a maioria dos usuários.

Quanto à representação das horas, o padrão do sistema é o uso do sistema de 24 horas. Mas há exceções que aparecem em algumas telas do sistema. De qualquer forma, pedidos e outros registros mantidos pelo sistema usam apenas a representação de 24 horas.

Para uma perfeita localização deveria ser possível a escolha do formato a ser utilizado. Na ausência dessa opção, deveria ser mantida ao menos uma consistência entre todas as porções do software.

7.1.3 Análise do uso de imagens, símbolos e cores

A preocupação com o uso de imagens, cores e símbolos, no âmbito da localização, é que a internacionalização deve torná-los desacoplados do código, para que, durante o processo, seja possível alterá-los de forma que não prejudiquem ou ofendam o usuário.

O Open For Business não usa, em geral, muitos ícones e imagens em sua interface, talvez pela sua natureza de software corporativo. Porém, por ser um sistema com interface Web a adição ou a alteração de ícones e imagens já existentes pode ser feita com relativa facilidade.

É possível, por exemplo, alterar os templates *.ftl* incluindo ou alterando diretamente imagens ou ícones onde for necessário o uso desses elementos para melhorar a usabilidade. Outro recurso que pode ser usado para a adição de ícones, principalmente, é a alteração dos arquivos *CSS* (*Cascading Style Sheets*) que compõe os temas visuais do sistema.

Na realidade, a maioria das figuras presentes em uma instalação do sistema consiste em dados do próprio usuário. Talvez um dos componentes que possua mais imagens, ou potencial para tal, é o componente de catálogo. Lá normalmente cada produto possui uma foto associada.

Dessa forma, no geral, o desacoplamento das imagens, ícones e outros símbolos do código do sistema é adequado graças, principalmente, à interface web usada no projeto.

No entanto, um aspecto que de fato não é considerado na interface é o desacoplamento entre imagens e textos em camadas. Se for necessário traduzir uma figura que possua um texto, será necessário refazer a figura por completo. Por outro lado, como há poucas figuras que fazem parte da interface original, apesar de ser uma falha, não representa um problema tão crítico.

Para as cores utilizadas cabe a mesma explicação dada às imagens e símbolos. Elas podem ser facilmente alteradas nos arquivos *CSS*. Como já foi citado anteriormente, de-

pendendo da cultura alvo da localização, ou mesmo o cliente alvo da instalação, talvez se justifique a criação de um tema visual personalizado.

No caso brasileiro, é difícil argumentar ou propor alguma mudança nas imagens, símbolos e cores dos temas visuais já existentes. Pode-se dizer que a cultura corporativa brasileira, apesar de possuir suas peculiaridades, é bastante parecida com a americana ou, em outras palavras, bastante alinhada com a forma ocidental de se fazer negócios. Portanto, esse aspecto da localização, ao menos para contexto brasileiro, perde um pouco de sua importância, deixando de ser uma preocupação muito relevante.

7.1.4 Análise das interfaces estendidas

Em projetos de localização, quando se faz a tradução e adaptação de interfaces estendidas, o foco é na facilidade criada pela internacionalização para se trabalhar com os materiais de apoio ao usuário, como manuais e tutoriais. Em outras palavras, trata-se de quão fácil é a tradução e adaptação de tais materiais por parte dos encarregados de fazê-los.

No caso do OFBiz, infelizmente não parece haver uma preocupação muito grande com os usuários, o que é muito frequente no cenário e código livre. No próprio site do projeto há uma *wiki* com o objetivo de agrupar documentos a respeito do sistema. Porém, no capítulo do usuário final (*End-user*), por exemplo, não há nem mesmo um documento. Simplesmente, não existem documentos cuja finalidade seja apresentar e explicar as funções do sistema para um usuário final. Isso consiste, mais uma vez, em uma quebra das boas práticas da criação de interfaces. Mais precisamente, vai de encontro com a heurística de número 10 presente no Apêndice A.

O que há é uma documentação criada por desenvolvedores, bastante voltada para eles próprios. Mas mesmo assim, trata-se de uma documentação pouco articulada, no sentido de seus componentes muitas vezes não serem coerentes entre si e, frequentemente, difícil de ser consultada.

A impressão que se tem é que a curva de aprendizado para um desenvolvedor do sistema é dificultada pela documentação. De fato, o sistema como *framework* de desenvolvimento tem tudo para ser prático e de fácil uso, porém seu aprendizado não é tão natural quanto seria desejável. Do ponto de vista do usuário final, a ausência de documentação pode, muitas vezes, torná-lo muito difícil de usar, tornando inviável a sua implantação.

Quanto à forma escolhida para a criação das interfaces estendidas pode-se dizer que não cria grandes problemas para sua localização.

Diferentemente de projetos comerciais, que tradicionalmente são distribuídos com algum tipo de documentação impressa, um projeto como OFBiz, baseado em uma comunidade organizada quase que exclusivamente pela própria web, possui o dinamismo e flexibilidade do gerenciamento de conteúdo que esta proporciona.

Levando-se em conta os aspectos levantados no Capítulo 2, que aborda as interfaces estendidas, pode-se dizer que, em primeiro lugar, o layout do texto realmente acomoda a expansão e contração. A separação entre textos e gráficos também é adequada, sendo a escolha de fontes restrita aos padrões utilizados na internet.

A realidade é que o formato escolhido para a documentação é bastante adequado, o que deixa a desejar é o conteúdo da documentação propriamente dita. Outros aspectos a serem comentados são a criação de glossários e listas de acrônimos voltadas para tradutores, que, grosso modo, podem ser considerados uma espécie de documentação da própria documentação.

A criação desses dois artefatos é uma parte fundamental da internacionalização de um projeto, pois não são usadas apenas na tradução das interfaces estendidas; podem servir também como apoio à localização da própria interface do sistema.

Não há uma preocupação da comunidade com esse tipo de documento. Se for considerado apenas o contexto de comunidades de software livre, isso pode até parecer normal, uma vez que a maioria dos projetos existentes apresenta a mesma conduta.

É normal ter o estereótipo em mente de que integrantes dessas comunidades sempre argumentam que a documentação é o próprio código. No entanto, essa é uma postura que deveria ser repensada, pois inibe a adoção do software em larga escala.

Nas próprias listas de discussão do projeto OFBiz, um assunto sempre em pauta é se o desenvolvimento do sistema nesse modelo de comunidade teria sido um erro. Alguns membros argumentam que, se outro modelo, talvez proprietário, tivesse sido adotado, o sistema hoje teria uma participação maior no mercado. Porém, talvez seja necessário considerar que o problema esteja na falta de empenho na criação de documentação que pudesse tornar sua adoção possível por usuários não desenvolvedores.

Concluindo uma análise da internacionalização das interfaces estendidas do OFBiz, é possível dizer que, do ponto de vista do formato escolhido, não há grandes problemas. Já no aspecto de materiais que guiam a internacionalização, o projeto deixa a desejar. É possível argumentar que isso tenha um impacto não só na localização, mas também na própria adoção do sistema em sua língua nativa.

7.2 Análise de Funcionalidades para Localização

Após analisar os aspectos tradicionalmente considerados na localização de um sistema, resta a análise da localização de funcionalidades. Trata-se de necessidades funcionais que surgem pela natureza do software em questão quando aplicado a uma determinada cultura em uma dada região.

No contexto dos sistemas de gestão empresarial, quando usados em português brasileiro, passam a estar sujeitos a requisitos específicos do Brasil. Pode-se identificar facilmente duas

origens de requisitos.

Uma delas é do tipo de indústria que o software atenderá, com suas peculiaridades de funcionamento de acordo com a cultura do país. Outra, mais óbvia e de certa forma mais sistemática, é a questão das exigências do governo ou do fisco.

Pode-se entender a primeira como uma adaptação para um usuário específico do que uma localização. Já a segunda deve, sim, ser entendida como um fator impactante na localização pelo fato de todas as empresas locais estarem sujeitas, ao menos em tese, às mesmas normas e exigências legais.

Na documentação do projeto OFBiz, apesar de escassa, há dois documentos que descrevem boas práticas de localização. O primeiro [48] apresenta recomendações e sugestões sobre como lidar com os arquivos XML, que armazenam os rótulos, e de usar o *Label Manager*, uma ferramenta do próprio projeto para auxiliar na manipulação dos arquivos.

Já o outro documento [45] é um pouco mais profundo, discute questões mais relevantes, e contempla boas práticas e diretrizes para uma boa tradução. Mas o que realmente interessa é a seção *Beyond Translation* (“Além da tradução” em português).

Nessa parte do documento são apontados os aspectos que, na visão da comunidade, fazem parte da localização do sistema para culturas específicas. Lá são citados dois aspectos: a configuração e adaptação do sistema de contabilidade e a do sistema de cálculo de impostos.

No primeiro, a adaptação da contabilidade é apontada como um passo bastante importante e potencialmente trabalhoso. O interessante é que os aspectos não são apenas apontados, mas, sim, associados a alguns arquivos que devem ser alterados para fazer as modificações necessárias. Admite-se, inclusive, a possibilidade de haver a necessidade de alterar os tipos de dados, talvez adicionando informações aos registros que compõem a contabilidade.

Infelizmente o cálculo de impostos, apesar de ser muito relevante, não é abordado em detalhes. É apenas indicado como algo a ser levado em conta. No entanto, há outros documentos que abordam o problema dos impostos.

Há inclusive um documento [38], que está em aberto para que desenvolvedores ou usuários do sistema incluam informações a respeito das exigências tributárias de seus países. Mas o mais importante a se entender a respeito da forma como o OFBiz lida com tributação é que ele se baseia em um modelo de dados chamado de *TaxAuthority* [6][19], que talvez em português pudesse ser chamado de Autoridade Tributária.

Esse modelo de dados permite simplesmente que se defina um órgão responsável pela cobrança e, então, o associe a alíquotas a serem calculadas para produtos ou categorias de produtos, assim como a região geográfica sobre a qual tem influência, e qual conta financeira do livro contábil deve registrar os valores pagos.

Associado a esse modelo de dados há uma série de serviços que rodam para fazer o cálculo dos impostos e aplicá-los às transações efetuadas no sistema. Toda essa estrutura

foi desenvolvida originalmente para funcionar de acordo com as exigências do chamado VAT (*Value Added Tax*), que se adota na Europa. Por isso, para a adaptação ao sistema brasileiro se faz necessária a alteração de tais serviços para que possam ser usados para o cálculo dos tributos de acordo com as normas e exigências brasileiras.

De fato, a localização da contabilidade e do cálculo de impostos são funcionalidades de extrema importância, porém na realidade brasileira não são as únicas a serem consideradas.

Além delas, pode-se identificar outras referentes à legislação e exigências do fisco. Na realidade, as mais fáceis de serem apontadas são as exigências do fisco, que de fato são, pelo menos em seu formato, iguais para todos os contribuintes.

Todos os projetos que compõem o SPED deveriam ser levados em conta para a localização de um ERP para a realidade brasileira. A Escrituração Contábil Digital (ECD), por exemplo, seria algo a ser agregada ao sistema original de contabilidade do OFBiz. Além disto, a Escrituração Fiscal Digital (EFD) também deveria ser incluída em conjunto com o sistema de cálculo de impostos.

A nota fiscal ou seu equivalente digital, por sua vez, é um conceito praticamente inexistente em muitos países. O que há em países como Estados Unidos e Alemanha é conceito de *invoice*, que é justamente o utilizado no projeto OFBiz. É algo bem mais simples do que a nota fiscal brasileira e, por não possuir uma legislação atrelada, requer muito menos detalhes e informações para ser gerada.

Além dos projetos do SPED, há ainda outros aspectos a serem considerados. Um deles é a legislação trabalhista (CLT - Consolidação das Leis Trabalhistas), que regula as relações entre empregado e empregador, a fim de, em tese, trazer tranquilidade e garantias para ambas as partes.

Em conjunto com as leis trabalhistas, pode-se citar ainda as Leis de Segurança do Trabalho que, apesar de fazerem parte da CLT, são bastante complexas e acabam se tornando um problema a ser implementado à parte.

Em suma, pode-se apontar, grosso modo, as seguintes funcionalidades a serem adaptadas ou adicionadas para a localização de um ERP para a realidade brasileira:

- Contabilidade;
- Cálculo de impostos;
- Escrituração Fiscal Digital;
- Escrituração Contábil Digital
- Emissão de Nota Fiscal Eletrônica;
- Cumprimento de leis trabalhistas;

No entanto, para poder se fazer um planejamento melhor, é possível observar como o mercado brasileiro de ERP se organiza. Em primeiro lugar é interessante estudar como empresas internacionais oferecem seus software no mercado nacional.

Grandes empresas, como a SAP, trabalham com um modelo em que empresas nacionais são autorizadas a implementar funcionalidades dessa natureza e prestar serviços para os clientes brasileiros. Ou seja, em outras palavras, é como se a implementação dessas funcionalidades fosse terceirizada.

Quando se analisam as empresas nacionais, pode-se dividi-las entre as grandes e as pequenas. Apesar de haver grandes empresas que detêm grandes porções do mercado, ainda assim as pequenas são significativas, abrangendo micro e pequenas empresas.

Em geral, as grandes empresas tendem a possuir soluções integradas que incluem ao menos aquelas funcionalidades referentes ao SPED. Por outro lado, muitas vezes detalhes referentes às leis trabalhistas e, principalmente, à segurança e medicina do trabalho são feitas por outras empresas e outros sistemas que podem interagir entre si.

Já no cenário das empresas menores que fornecem sistemas de gestão, muitas vezes as soluções não são tão integradas. Há a criação de nichos de prestação de serviço, tanto nas áreas contábeis e fiscais quanto nas que dizem respeito ao cumprimento da legislação trabalhista.

Com isso, é necessário analisar o que faz mais sentido ao se fazer o estudo da localização do OFBiz: considerar a implementação de funcionalidades como componentes extras ou, então, avaliar a facilidade de integrá-lo a outras soluções, seja por meio de interação via software ou exportação de dados.

Tendo em vista a existência de soluções no mercado que trabalham com a terceirização de funcionalidades ou com a prestação de consultoria por outras empresas, talvez faça mais sentido focar na avaliação da possibilidade de integração do sistema.

7.3 Dimensões dos arquivos a serem traduzidos

Para a simples tradução dos componentes do Open For Business pode-se separar o trabalho em 3 etapas, cada uma delas para a tradução dos componentes presentes nos três principais diretórios do sistema. Cada um desses diretórios agrupa componentes com características semelhantes. São eles:

- **Applications** - onde estão localizados os principais módulos do OFBiz, ou seja, aqueles que implementam as principais funções de um ERP;
- **Specialpurpose** - diretório onde são armazenados módulos não menos importantes, porém com funções mais específicas, que complementam aquelas localizadas nos módulos do diretório *applications*;

- **Framework** - onde estão localizados os componentes que talvez implementem as funções mais importantes,, ou seja, aquelas que todos os demais componentes utilizam como base. Merecem destaque, por exemplo, os componentes que tornam possível a metalinguagem (Minilang) e todo o sistema de *widget*.

Porém, nem todos os componentes presentes nesses diretórios realmente possuem rótulos a serem traduzidos. Na realidade, há exceções nos diretórios *framework* e *specialpurpose*, que possuem componentes sem interface com o usuário ou, então, que usam os rótulos presentes em um componente de nome *Common* do diretório *framework*.

Nos apêndices C, D e E são elencados os arquivos de rótulos assim como suas dimensões, dos diretórios *applications*, *specialpurpose* e *framework*, respectivamente.

Diretório	Rótulos pt_BR	Total de rótulos (en)
Applications	1	10215
Specialpurpose	3	1221
Framework	16	2744
Total	20	14180

Tabela 7.1: Dimensões totais dos diretórios a serem traduzidos

Na Tabela 7.1 são mostradas as dimensões totais, em número de rótulos a serem traduzidos, dos três diretórios. Há duas colunas de dimensões, a primeira com o número de rótulos já traduzidos no início do trabalho, ou seja, aquelas com locale **pt_BR**, e a outra com o número total de rótulos, aquelas no locale **en**. Na mesma tabela é apresentado o total de rótulos considerando-se todos os componentes.

Dessa forma, é possível observar que foi necessário traduzir ao todo 14180 rótulos o que representa 99,86% de toda a interface. A tradução foi feita no âmbito do presente estudo.

7.4 Localização em outras línguas

Algo interessante para se analisar, na oportunidade de localizar um sistema, é a sua localização já feita para outras línguas e culturas. No caso do OFBiz, mesmo procurando em todos os documentos da comunidade, não foi possível encontrar nenhum texto que descrevesse uma experiência de localização em outra língua.

Na realidade há menções à adaptação da tributação para alguns países e alguns dicionários de tradução em alemão, holandês e francês. Mas não é possível encontrar uma descrição mais detalhada.

Optou-se, então, por fazer uma análise nos próprios arquivos que compõem a interface para se ter uma ideia da presença da tradução em outras línguas. Para tal implementou-se

um programa em Java para fazer a contagem de todos os textos nos locais presentes no sistema.

A tabela 7.2 mostra a contagem dos rótulos para cada Locale presente, associando-os com os idiomas que representam e mostrando a porcentagem quando comparados ao total de rótulos.

Locale	Idioma	Rótulos	Porcentagem
ar	Árabe	1241	8,75%
cs	Tcheco	437	3,08%
da	Dinamarquês	1053	7,42%
de	Alemão	8964	63,22%
en	Inglês	14180	100%
en_GB	Inglês (Grã-Bretanha)	60	0,42%
es	Espanhol	8140	57,40%
fr	Francês	12183	85,92%
hi_IN	Hindi (Índia)	3858	27,21%
in	Bahasa Indonésia	1	0,007%
it	Italiano	13298	93,78%
ja	Japonês	592	4,17%
nl	Neerlandês	4735	33,39%
pt	Português	413	2,91%
pt_BR	Português brasileiro	20	0,14%
pt_PT	Português de Portugal	635	4,48%
ro	Romeno	8118	57,25%
ru	Russo	7318	51,60%
th	Tailandês	10498	74,03%
zh	Chinês	12283	86,62%
zh_CN	Chinês (China)	795	5,60%
zh_TW	Chinês (Taiwan)	11735	82,75%

Tabela 7.2: Rótulos em cada locale

A partir da Tabela 7.2 é possível observar que, depois do inglês, as línguas mais presentes são o italiano, o chinês e o francês. Seria interessante analisar também a adoção do sistema ao redor do mundo para se ter uma ideia se a localização nas referidas línguas realmente garante a sua adoção. No entanto, não foi possível obter dados que pudessem caracterizar essa situação.

7.5 Considerações finais

Este capítulo de análise para a localização teve como principal função avaliar aspectos como o desacoplamento entre interface e código, formatos numéricos usados pelo sistema e imagens e cores que compõem a interface.

Serviu também para avaliar a dimensão do trabalho à frente através da mensuração dos arquivos envolvidos e das localizações já disponíveis em outros idiomas.

Tal análise servirá como base para o desenvolvimento do próximo capítulo, que trata da localização de componentes e funcionalidades.

Capítulo 8

Localização de Componentes

Tendo sido feita uma análise da internacionalização e de outros aspectos que influenciam a localização, a próxima etapa é de fato começar a localização dos componentes que constituem o OFBiz.

Nas seções seguintes são descritas as etapas para a localização do OFBiz dentro do escopo deste trabalho. Para tal, é usado o processo sugerido no diagrama da Figura 2.4 para organizar a ordem em que as informações são apresentadas.

Dessa forma, pode-se entender que, de posse do código do projeto obtido do sistema de controle de versões da comunidade do OFBiz, a Seção 7.3 foi a etapa de análise que compõe a etapa de planejamento do diagrama da Figura 2.4.

A partir disso, são descritas as etapas de preparação do código, da elaboração de um glossário, da tradução propriamente dita dos rótulos que compõem a interface, do desenvolvimento de funcionalidade e sua integração e assim por diante, tendo-se sempre como diretriz principal o fluxo sugerido no Diagrama da Figura 2.4.

8.1 Preparação do código

Quando se fala em preparação de código para a localização, a ideia é tornar a tradução da interface o mais fácil e eficiente possível. Em alguns sistemas isso pode consistir em um desafio maior que em outros, podendo até ser necessário um trabalho de desacoplamento entre a implementação da interface e os rótulos que a constituem. Tudo depende da qualidade da internacionalização feita pelos desenvolvedores do software.

Como foi visto no Capítulo 7.3, o desacoplamento, na forma arquivos XML com todos os rótulos das telas, entre a interface e o texto que a compõe é bastante adequado.

Dessa maneira, não foi necessário nenhum passo adicional para tornar a tradução eficiente. No entanto, a adição de rótulos em português exigiu a edição de mais de 60 arquivos XML, alguns deles com mais de 15000 linhas. A principal dificuldade foi encontrar uma

forma eficiente de adicionar tais rótulos.

A adição manual deles significa, de certa forma, a meta do trabalho de edição. Por ser um trabalho repetitivo, o ideal seria adicionar as tags XML necessárias para depois preenchê-las com a tradução adequada.

A preparação do código, nesse contexto, foi, então, a adição de tais tags de maneira eficiente. Para tal implementou-se um programa em Java que lia os arquivos XML a serem traduzidos e adicionava as tags para as traduções nos locais **pt** ou **pt_BR**, de acordo com a necessidade.

A escolha entre a adição de uma tag **pt** ou **pt_BR** é feita baseada no comportamento do sistema para a escolha de qual dos dois rótulos utilizar. O que acontece é que, quando o usuário escolhe, por exemplo, “Português Brasileiro” como sua língua favorita, o sistema dará preferência ao conteúdo **pt_BR**. Se não houver a tradução específica ele buscará a tradução **pt** e, em último caso, se não houver nenhuma tradução usará o padrão em inglês **en**.

Para a tradução de línguas como o português, que possui variações de acordo com os países em que é adotada, a boa prática indicada pela comunidade é que se dê preferência pela adição da tag mais geral possível. Assim, o programa responsável pela adição das tags verificava a existência de rótulos **pt** e, caso estivessem presentes, adicionava **pt_BR**. Em caso contrário, a tag **pt** era adicionada.

Na etapa de tradução, a ser descrita mais adiante, foi necessário avaliar, no caso de haver uma tradução anterior sob a tag **pt**, se a tradução em português brasileiro seria igual à anterior. Se esse fosse o caso, seria necessário excluir a tag **pt_BR**.

8.2 Elaboração do glossário

Tanto para a tradução da interface quanto para a tradução de materiais, que compõem a interface estendida de um sistema, é necessário que os termos usados sejam consistentes e sejam condizentes com o jargão utilizado pelos usuários alvo.

Dessa forma, uma das etapas no processo de localização é a elaboração de um glossário que, além de servir para a tradução dos termos mais comuns, cumpre o papel de garantir a consistência da tradução entre diversas porções do software, manuais, tutorias etc.

O glossário deve incluir todas as palavras que pertençam ao escopo e domínio do projeto e possam ser traduzidas de formas diversas. Seu papel é garantir que uma tradução seja escolhida para cada palavra sem que haja ambiguidade e a interface não deixe o usuário confuso.

No projeto OFBiz, documentos como esses já foram publicados em alemão, neerlandês, dinamarquês e francês [48]. Apesar de Francês ser uma das línguas para a qual existem mais rótulos traduzidos, o glossário de traduções para esse idioma é o menos completo.

Por outro, lado o glossário em alemão é bastante extenso e completo. Por isso, serviu de base para a elaboração do glossário para a tradução para o português brasileiro. Basicamente os termos listados em inglês nesse documento foram utilizados para a base do novo glossário e, usando-se um outro programa auxiliar em Java, foram adicionadas outras palavras relevantes.

O resultado é apresentado no Apêndice B, que elenca as principais palavras mais frequentes na interface do sistema e que estão mais relacionadas com o domínio dos ERP.

8.3 Tradução dos rótulos

Depois da elaboração do glossário, o próximo passo foi dar início à tradução da interface propriamente dita. Como mostrado na Seção 7.3, o trabalho para a tradução completa consistiu na tradução de 14180 rótulos distribuídos entre os diretórios *Applications*, *Specialpurpose* e *Framework*.

Com uso da ferramenta citada anteriormente, que adiciona as tags para abrigar as traduções para o português, o trabalho se resumiu à edição do arquivo XML para adicionar as versões em português dos textos, que compõem a interface.

Todo o trabalho foi feito visando a distribuição na comunidade do OFBiz para que, dessa forma, as traduções possam ser adotadas por outros usuários interessados em utilizar o sistema na língua portuguesa.

Para que isso fosse possível, foi necessário seguir as diretrizes indicadas pela comunidade para a edição dos arquivos. Dentre elas há a exigência de que dentro do arquivo XML de rótulos seja mantida a ordem alfabética de todas as tags para cada label e que o *patch* com a adição das traduções siga um formato padrão de nomenclatura.

Basicamente a tradução de cada arquivo consistiu em baixá-lo do sistema de controle de versões do projeto, alterá-lo com as ferramentas auxiliares e, por fim, adicionar as traduções para o português. Depois disso, o *patch* enviado para a comunidade passava pelo crivo de outros desenvolvedores responsáveis pela inclusão de *patches*, os chamados *committers*, e então adicionado ao projeto.

A partir daí, qualquer um que baixasse a versão mais recente de desenvolvimento do sistema (*trunk*) passava a ter acesso aos rótulos traduzidos. Atualmente todas as traduções enviadas já estão disponíveis, ao menos na versão de desenvolvimento do projeto.

Usuários que baixarem uma das versões estáveis do projeto podem aplicar patches para fazer uso do sistema em português. Será assim pelo menos até o próximo lançamento de uma versão estável que inclua todos os *patches* aceitos pelos *committers*.

8.4 Localização de funcionalidades

A localização de funcionalidades pode ser considerada por alguns como sendo mais uma adaptação do que como uma etapa constante em um processo de localização. No entanto, no caso dos ERP com certeza, há funcionalidades que transcendem a simples tradução da interface e realmente podem impactar na adoção do sistema na realidade brasileira.

Anteriormente, na Seção 7.2, foram indicados os principais aspectos funcionais a serem levados em conta para o projeto OFBiz. Deles, pode-se dizer que a contabilidade é uma questão de configuração por alguém da área contábil em conjunto com um especialista no funcionamento do OFBiz. Todos os passos são mostrados na comunidade e, mesmo as alterações que se fazem necessárias no código, são indicadas em detalhes.

O mais complicado, e que foge do escopo deste trabalho, é quanto ao cumprimento de leis trabalhistas. Isso depende de uma série de legislações específicas para cada ramo em que a empresa atue. Há uma dependência de sindicatos e órgãos afins, além de haver também mudanças frequentes nas legislações pertinentes. Em outras palavras, a forma como as leis trabalhistas funcionam no Brasil realmente cria um nicho onde empresas podem atuar em consultoria e terceirização de funcionalidades de um ERP.

Depois disso, pode-se citar as funcionalidades relacionadas ao projeto SPED, que são consideradas de extrema importância para a utilização do sistema na realidade brasileira. E, por fim, o cálculo de impostos, que é algo um pouco mais complicado que a contabilidade por se tratar de configurações que realmente dependem da intervenção no código de métodos e serviços do sistema.

Dessa forma, para se fazer o estudo de localização de funcionalidades, há de se considerar o projeto SPED e o cálculo dos impostos. Para o primeiro existe a possibilidade de implementação de módulos dentro do OFBiz que cuidem de todos os seus aspectos ou, então, foquem na exportação dos dados necessários para sistemas terceirizados cuidarem de todos os aspectos contábeis e fiscais da empresa.

Como já há no país um nicho de mercado que lida com esse modelo de negócios de terceirização, é complicado optar exclusivamente por uma das duas abordagens no presente estudo. A opção escolhida foi a de implementar uma prova de conceito que servisse para avaliar a facilidade de programação dentro do *framework* do OFBiz e gerasse alguns arquivos que pudessem ser utilizados para a terceirização.

Com essa escolha pode-se ter ao menos uma ideia das dificuldades envolvidas para alguém que pense em adotar o OFBiz como sistema de gestão ou uma empresa que pense em usá-lo para prestar serviços. Ao mesmo tempo, parte do trabalho necessário para integrá-lo a um terceirizador de serviços já estará feita.

Já o cálculo dos impostos será abordado do ponto de vista da implementação existente no OFBiz, para, dessa maneira, tornar mais claro quais passos seriam necessários para a adequação às exigências fiscais brasileiras. A escolha por uma abordagem, de certa forma

mais teórica, se deu por haver uma legislação bastante extensa que rege a cobrança de impostos e, mais uma vez, pode ser algo passível de ser feito por empresas terceirizadoras.

É importante ressaltar que a opção pela terceirização é considerada porque uma das principais intenções deste estudo é avaliar a aplicabilidade do OFBiz para micro e pequenas empresas brasileiras. Entende-se que nem sempre essas categorias de empresas dispõem dos meios necessários para manter uma equipe responsável por todos esses aspectos jurídicos e fiscais em sua folha de pagamento.

Nas subseções seguintes são abordados os dois assuntos. A parte do SPED é dividida em uma apresentação e argumentação da prova de conceito, seguida dos principais aspectos referentes à implementação. No final, a forma como o cálculo de impostos é implementada é apresentada.

8.4.1 Prova de conceito para o projeto SPED

Para avaliar a facilidade de localização do OFBiz para trabalhar dentro das exigências do SPED, é necessário escolher um de seus projetos para implementação. Pode-se escolher dentre o ECD, EFD e a NF-e.

Entende-se que ambos ECD e EFD se resumem a uma exportação adequada dos dados referentes à contabilidade e impostos já existentes no OFBiz, para o envio aos órgãos responsáveis. Haveria a necessidade da adição dos campos referentes à classificação fiscal de produtos, por exemplo, assim como algumas outras informações. Mas, no geral, dos três considerados aqui, estes dois parecem ser os mais simples.

O projeto NF-e, por outro lado, parece englobar todos os desafios dos outros dois e é o mais popular no Brasil. Dos três é aquele sobre o qual há mais cobrança por parte das autoridades para que seja cumprido e respeitado. Além disso, algumas empresas são liberadas do ECD ou EFD.

Assim, para que o estudo de localização ganhe um caráter mais abrangente é preferível escolher o projeto NF-e como base para a prova de conceito.

Grosso modo, quando se fala da localização para atender às necessidades da nota fiscal brasileira, está se falando na localização da *invoice* americana. Em outras palavras, é necessário “traduzir” a *invoice* implementada pelo OFBiz para a nota fiscal exigida pelo governo brasileiro.

Pode-se entender a nota fiscal como um documento mais detalhado que a *invoice* americana. Nela há a presença dos itens citados na Seção 6.2, como a classificação fiscal, por exemplo. Então, o primeiro passo é fazer com que o sistema passe a armazenar tais informações em sua camada de persistência de dados, adicionando os campos ao banco de dados. Logo em seguida, o usuário deve ser capaz de preencher tais campos, ou seja, a interface deve ser alterada para que novos campos de entrada e exibição sejam incluídos.

Uma vez que tais dados sejam “conhecidos” pelo sistema, é possível exportá-los para

que sejam tratados por sistemas terceirizados. Há também a possibilidade de tratá-los para que outros módulos implementem a comunicação por *webservices* com a Receita Federal e enviem as notas fiscais eletrônicas já construídas.

Já que uma das principais propostas do projeto Open For Business é sua organização em serviços, pode-se considerar que a implementação da comunicação com os *webservices* da Receita seria a parte mais simples do trabalho. Além disso, se o sistema for preparado para a exportação de dados, automaticamente haverá um caminho indicado tanto para a solução que conta com a terceirização quanto para o caminho da implementação total.

Por isso, a opção para a prova de conceito foi a inclusão de todos os campos necessários e a posterior geração de arquivos XML no formato do projeto NF-e. Com essa abordagem o sistema estaria pronto para a integração com alguns sistemas já existentes no mercado que usam apenas os arquivos XML para se comunicar com a Receita Federal, assinar os documentos, armazenar as respostas e imprimir o chamado DANFE.

Um exemplo de tal sistema é o Uninfe [18], que pode ser instalado no sistema operacional Windows para monitorar um dado diretório em busca de arquivos XML que representem notas fiscais eletrônicas. Ele próprio faz o envio, a assinatura e a impressão do DANFE. Ele é citado aqui pois seria de fato uma opção para micro e pequenas empresas para lidar com as exigências da Nota Fiscal Eletrônica, sendo de fácil instalação e compatível com o sistema operacional mais adotado no Brasil.

Nas próximas duas subseções são apresentados os passos para adição dos campos e a geração dos arquivos XML.

8.4.2 Adição de campos

Como já explicitado, a intenção para a implementação da prova de conceito era demonstrar as principais dificuldades envolvidas para se tornar o OFBiz adotável por empresas brasileiras, do ponto de vista da integração com o SPED.

Pode-se dizer que o principal desse processo é a “tradução” da *invoice*, ou seja, a adição dos detalhes necessários para que ela se torne a nota fiscal brasileira. Encarando tal documento com um nível de abstração maior, podemos dizer que há três categorias de informações relevantes: aquelas referentes à empresa emissora, ao cliente e aos produtos.

Dessa forma, para que uma *invoice* se torne uma nota fiscal é necessário que todas as informações exigidas pelo fisco estejam presentes no sistema e possam ser “adicionadas” ao documento.

Como o objetivo é se fazer uma prova de conceito, optou-se por escolher os principais campos referentes a cada uma das três categorias de forma que uma NF-e básica pudesse ser gerada. Essa escolha partiu da comparação dos dados já modelados no sistema e aqueles que são exigidos pelo fisco.

Para os dados do cliente foi necessário analisar a *entity Person* que já existia no sistema

e era usada para descrevê-los no que diz respeito à *invoice*. Nela o único dado necessário de se adicionar foi o CPF.

A empresa é modelada pela *entity PartyGroup* e, para fins desse estudo, considerou-se necessária a adição dos campos referentes ao CNPJ, à inscrição estadual e ao código CNAE (Classificação Nacional de Atividades Econômicas), usado para classificação das empresas pela Receita Federal.

Por fim, para a adequação dos produtos julgou-se necessária a adição dos dois campos seguintes:

- Descrição - que diferentemente daquela presente no sistema se destina ao uso na nota fiscal;
- NCM, o código de Nomenclatura Comum do Mercosul.

Haveria também a necessidade de adição do CFOP (Código Fiscal de Operações e Prestações), porém, como se trata de uma prova de conceito, foi considerado constante e tratado diretamente na geração do arquivo XML correspondente à NF-e abordado mais adiante.

A adição desse conjunto de campos cumpre o papel de servir como caminho para a avaliação da dificuldade de adaptar o sistema para a integração com o SPED, mas não necessariamente deixá-lo pronto para tal.

Para a adição à camada de persistência de dados dos campos escolhidos seguiu-se os métodos descritos como boas práticas na documentação da comunidade e da literatura [24]. Na realidade, a extensão das tabelas do banco de dados pode ser feita de maneira bem simples no OFBiz, sendo necessário apenas saber qual *entity* será estendida e quais campos deseja-se adicionar.

Não é necessário se preocupar com banco de dados que está realmente sendo utilizado. Basta conhecer os tipos de dados que são reconhecidos pelo *framework* e podem ser estudados nos arquivos constantes no diretório *framework/entity/fieldtype* ou no arquivo *framework/entity/dtd/fieldtype.xsd*.

A extensão em si pode ser feita sem nem mesmo ser necessário editar um dos arquivos originais da distribuição. Dentro do diretório criado para a prova de conceito, foi criado outro subdiretório de nome *entitydef* que armazena as definições referentes à camada de persistência de dados. Lá, o arquivo *entitymodel.xml* armazena o código utilizado para a criação dos campos extras.

Como é possível observar no código da Listagem 8.1, trata-se de um arquivo XML com três elementos que valem a pena ser explicados. São as *<extend-entity>*, que basicamente são a melhor forma de fazer a extensão de *entities* já existentes no sistema. Nessa prova de conceito foram usadas para a extensão das *entities Person, PartyGroup e Product*.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <entitymodel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation=
4   "http://ofbiz.apache.org/dtds/entitymodel.xsd">
5   <!-- ===== -->
6   <!-- ===== Defaults ===== -->
7   <!-- ===== -->
8   <title>Entity of Nfe Component</title>
9   <description>None</description>
10  <copyright></copyright>
11  <version></version>
12
13  <extend-entity entity-name="Person">
14    <field name="cpf" type="id-ne"></field>
15  </extend-entity>
16
17  <extend-entity entity-name="PartyGroup">
18    <field name="cnpj" type="id-ne"></field>
19    <field name="inscricaoEstadual" type="id-ne"></field>
20    <field name="cnae" type="id-ne"></field>
21  </extend-entity>
22
23  <extend-entity entity-name="Product">
24    <field name="codigo" type="id-ne"></field>
25    <field name="descricao" type="description"></field>
26  </extend-entity>
27
28 </entitymodel>

```

Listagem 8.1: entitymodel.xml

O campo de descrição da *entity Product* utilizou o tipo de campo “*description*” que nada mais é que uma cadeia de caracteres de 255 caracteres. Todos os demais utilizaram o tipo “*id-ne*”, que também é uma cadeia de caracteres, mas com apenas 20 caracteres, isso porque todos eles são códigos com menos de 20 dígitos ou letras.

Mas para avaliar a dificuldade de adicionar essas informações ao sistema não bastava apenas adicioná-las à camada de persistência de dados. Foi necessário alterar, também, a interface para que o usuário pudesse gravar, ler e editar tais valores. E mais, uma característica muito prática do OFBiz ficou clara a essa altura.

Tanto as interfaces que utilizam as *entities Person* quanto *PartyGroup* são implementadas de forma que os campos exibidos na tela são os mesmos constantes na camada de persistência de dados. Em outras palavras, não foi necessária nenhuma alteração da interface. A partir do momento em que os campos foram adicionados ao arquivo *entitydef.xml*, mostrado acima, a interface se adequou automaticamente.

Por outro lado, o mesmo não ocorreu para as telas que servem para o manuseio e visualização dos dados referentes ao *Product*. Ao contrário das outras interfaces, ela utiliza recursos mais complexos como campos retráteis baseados em *javascript* e agrupa os campos para melhorar a usabilidade. Faz sentido, pois a *entity Product* possui muitos campos que nem sempre são preenchidos e, portanto, se fossem todos exibidos, poderiam tornar as telas bastante confusas.

Dessa forma, fez-se necessária uma adaptação na tela responsável pela adição, edição e visualização das características do produto. Há outras telas que poderiam ter sido alteradas, porém foi feita a opção de alterar apenas essa pois os novos dados não fariam sentido nas outras, além do que o objetivo principal era apenas demonstrar o conceito.

Surpreendentemente essa alteração custou somente a adição de seis linhas de código no arquivo *applications/product/widget/catalog/ProductForms.xml*. Foi adicionada a declaração dos campos que seriam adicionados à tela e um grupo de campos para acomodá-los. O grupo de campos foi chamado de “Dados para NF-e” e foi posicionado no final da página.

No código da Listagem F.1, apresentada no Apêndice F, é apresentada uma porção do arquivo *applications/product/widget/catalog/ProductForms.xml* que inclui o código resultante da tela alterada. As partes entre os comentários “<!-- adicionado para NFe - ->” (linhas 22 e 37 da listagem) são aquelas que foram adicionadas. A Figura 8.1 mostra a tela resultante.

The screenshot shows a web application interface for editing a product. The top navigation bar includes tabs for WORK EFFORT, PARTICIPANTES, PEDIDOS, and COMUNICAÇÕES. The main header displays the product name 'Product Para: Ethernet Card 10/100 [ID:ETH_BRAND]' and several action buttons: '+ NOVO PRODUTO', '+ NOVO PRODUTO VIRTUAL', 'PÁGINA DO PRODUTO', 'CÓDIGO DE BARRAS', 'EXPANDIR TODOS', and 'RECOLHER TUDO'. The left sidebar contains a search section with a 'Search Products' input, a 'Palavras-chave' field, and a 'Pesquisar' button. Below this is a 'Pesquisa avançada' section with an 'ID da Categoria' field and a 'Pesquisar' button. The sidebar also lists several catalogs: Demo Catalog, Test Catalog, Google Catalog, eBay Catalog, and Rental Catalog. The main form area is titled 'Edit Product' and contains various input fields and sections. The 'Wording And Comment' section includes fields for 'Nome Interno' (Ethernet Card 10/100), 'Marca', 'OEM Party ID', and 'Comentários'. The 'Virtual Product' section includes a 'Categoria Primária' field. The 'Dados' section includes fields for 'Taxa', 'Montante', 'Medidas', 'Frete', 'Diversos', and 'Dados para NF-e'. The 'Dados para NF-e' section is highlighted in yellow and contains a 'Código' field with the value '21434' and a 'Descrição' field with the value 'Placa de Rede'. The 'Atualizar produto' button is visible at the bottom of the form.

Figura 8.1: Tela de Edição de Produtos alterada, já com os campos para NFe

Esse número tão pequeno de alterações necessárias deve-se aos mecanismos de abstração

que o *framework* do OFBiz proporciona. De fato, aquele argumento dos desenvolvedores de que tarefas repetitivas e comuns são eliminadas se mostra verdadeiro. Normalmente em outros sistemas de desenvolvimento seria necessária a associação de cada campo da tela com o correspondente na camada de abstração. No entanto, aqui basta que o nome dos campos na interface e na camada de persistência de dados correspondam.

Claramente isso facilita e acelera o desenvolvimento, mas é necessário bastante cuidado e atenção, pois mecanismos dessa natureza podem causar erros difíceis de depurar.

8.4.3 Geração do XML da NF-e

Uma vez que o modelo adotado para a prova de conceito foi a de uma espécie de tradução da *invoice* nativa do sistema, a ideia para a geração do XML da NF-e foi usar os mecanismo da arquitetura orientada a eventos do OFBiz para, sempre que uma *invoice* fosse criada, uma ação fosse disparada para a criação do XML correspondente.

Tal ação nada mais era do que a chamada de um serviço em Java para a geração de um XML no diretório *runtime/nfe*, criado especialmente para armazenar os arquivos gerados para a prova de conceito.

Foi usado um EECA (*Entity Event Condition Action*), que é disparado sempre que uma *invoice* é criada e armazenada na camada de persistência de dados. O código da Listagem 8.2 mostra o arquivo *eeecas.xml* criado no diretório *entitydef* do componente criado.

A definição do EECA é feita com o elemento *eca*. Para entendê-lo é necessário observar os seguintes elementos:

- “entity=“Invoice””, determina que a *entity Invoice* será aquela observada para gerar os eventos;
- “operation=“create-store””, determina que as operações que gerarão o evento são aquelas de criação e armazenamento;
- “event=“return”” determina que, ao retornar das referidas operações, a ação será disparada;
- “<condition>” é um mecanismo condicional que, no caso, testa se o registro gravado não é vazio;
- “<action>” é de fato a ação disparada a partir do evento. Nesse caso, a invocação do serviço *NfeGeraXml* implementado em Java.

```

1 <?xml version=" 1.0" encoding="UTF-8" ?>
2
3 <entity-eca xmlns:xsi=" http://www.w3.org/2001/XMLSchema-instance"

```

```
4      xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/entity-eca.  
      xsd">  
5  
6      <eca entity="Invoice" operation="create-store" event="return">  
7      <condition field-name="invoiceId" operator="is-not-empty" />  
8      <action service="NfeGeraXml" mode="sync" />  
9      </eca>  
10  
11 </entity-eca>
```

Listagem 8.2: eeca.xml

O serviço *NfeGeraXml* em si foi criado em Java baseando-se nos *schemas* XSD versão 2.0 [11] distribuídos pela Receita Federal. A partir de tais *schemas* foram geradas as classes que representam a estrutura de um arquivo XML da NF-e e adicionado ao projeto como uma biblioteca do componente.

Para validar o que foi feito, foram realizados testes para gerar de fato um arquivo XML. Tais testes basicamente foram criações de pedidos de compra. O principal caso testado foi o pedido de compra de uma placa *Ethernet* no componente *Order*.

Para isso, o sistema pede todos os dados do comprador, incluindo o endereço de entrega e o método de envio. Ao fim da criação do pedido é necessário registrar o recebimento do pagamento. Imediatamente após registrar o pagamento, o sistema gera a *invoice* e, em seguida, a ação descrita acima é disparada gerando o arquivo correspondente no diretório *runtime/nfe*.

Todos os códigos utilizados e os resultados obtidos, como o XML gerado, podem ser encontrados e baixados do seguinte endereço: <http://www.harriss.com.br/localizacaoOFBiz>.

8.4.4 Cálculo de impostos

Originalmente o OFBiz lida com o cálculo e a aplicação de impostos aos produtos vendidos usando um conceito de *TaxAuthority*, ou autoridade tributária. O conceito facilita a integração dessa porção do sistema com a parte contábil, uma vez que ajuda a categorizar cada imposto aplicado e recebido, tornando bastante simples a configuração para atender aos requisitos fiscais.

Em resumo, esse conceito é implementado com base em uma *entity* chamada *TaxAuthorityRateProduct*, que visa descrever como a autoridade tributária se relaciona a determinado produto ou categoria de produtos.

Esta *entity* tem como principais campos os seguintes:

- **ProductStore** determina a qual das lojas gerenciadas pelo sistema o imposto se aplica, podendo abranger todas elas caso tenha o valor *null*;

- **GeoId** (*TaxAuthority*) nada mais é que uma área geográfica de jurisdição da autoridade tributária;
- **GeoId** (*Payer*) modela a localização geográfica do comprador;
- **ProductCategory** representa a categoria de produtos sobre a qual o imposto incide.

Dessa forma, a partir desse modelo é possível lidar com leis tributárias que dependem da localização da empresa vendedora e do consumidor ou mesmo que dependa do tipo de produto.

A fim de lidar com esses dados descritos e armazenados na *entity TaxAuthorityRate-Product*, o sistema usa dois serviços de nome *calcTax* e *calcTaxForDisplay*. O primeiro calcula o valor do imposto e o segundo lida com detalhes como impostos que devem ser incluídos ou não no preço exibido para o cliente.

Ambos os serviços são, na realidade, implementações de duas interfaces de nome *calcTaxInterface* e *calcTaxForDisplayInterface*. Esse conceito de interfaces é exatamente o mesmo utilizado em Java.

Em uma análise de como o cálculo de impostos é implementado no OFBiz, é possível perceber que as preocupações com a jurisdição da autoridade tributária, a categorização de produtos segundo a tributação e a localização do comprador, correspondem a alguns dos requisitos básicos para a adequação ao sistema tributário nacional.

No entanto, a implementação dos serviços responsáveis pelo cálculo de impostos propriamente dito com certeza não se adequam ao funcionamento da tributação brasileira. Na realidade, mesmo nos países onde o OFBiz já é adotado como solução por empresas que necessitam de um ERP, essa implementação parece não ser sempre a ideal.

A *HotWax Media*, uma empresa formada por dois membros respeitados da comunidade do Open For Business, por exemplo, oferece como um de seus principais serviços a adequação do sistema às necessidades fiscais de seus clientes.

Isso pode ser feito de maneira relativamente simples ao fazer implementações alternativas para as já referidas interfaces. No Brasil seria necessário, por exemplo, implementar tais serviços levando em conta a localização da empresa e o deslocamento das mercadorias vendidas.

A implementação poderia ser específica para determinados ramos de atuação ou o mais geral possível tentando-se abranger o maior número de tipos de empresa. Qualquer implementação se torna possível, uma vez que essa abstração para a implementação dos serviços está disponível.

No caso brasileiro seria possível inclusive a implementação desses serviços de forma que dependessem de um *webservice* externo que lidasse com as diversas exigências e legislações. Isso seria condizente com o modelo apontado anteriormente de terceirizadores de funcionalidades.

8.5 Localização de Interfaces Estendidas

No início do trabalho, a intenção era entregar à comunidade ao menos a tradução daquelas interfaces estendidas que fossem direcionadas ao usuário final. Apesar de escassos haviam alguns materiais direcionados a esse público indicando parte da organização dos módulos e o seu funcionamento.

A abordagem escolhida foi de protelar a tradução desses materiais para deixá-la como última etapa do projeto. Além de se adequar ao processo escolhido da Figura 2.4, acreditava-se que mais conteúdo poderia ser gerado e, portanto, a tradução das interfaces estendidas fosse uma tarefa mais rica e que gerasse de fato mais conteúdo útil para brasileiros à procura de documentação.

No entanto, o OFBiz no espaço de tempo de duração deste projeto sofreu várias alterações, a citar a transferência de seus documentos entre diferentes sistemas de gerenciamento de conteúdo. Nessa transferência boa parte das interfaces estendidas direcionadas ao usuário final foram perdidas e, atualmente, o mais comum de ser encontrado são páginas com mensagens de endereço não encontrado.

Não se sabe o motivo, porém isso demonstra mais uma vez a falta de comprometimento da comunidade com o usuário final. A comunidade parece se preocupar mais intensamente apenas com usuários técnicos que possam entender o sistema a partir do próprio código.

Foi feita, então, uma análise de como as interfaces estendidas são publicadas para se ter, de fato, um estudo de localização. Mais uma vez a questão da documentação deixa a desejar. O sistema de documentação oficial simplesmente não possui uma forma de traduzir um documento e associá-lo ao original. Autores que abordam a localização de interfaces estendidas [1] indicam o uso de sistemas de gerenciamento de conteúdo que permitam esse tipo de publicação de conteúdo em várias línguas.

No entanto, o que é possível fazer, no caso do OFBiz, é tentar publicar outro documento na wiki do projeto, mas sem de fato associá-lo ao original. Além disso, um fato que pode ser facilmente observável é que não há documentos dessa natureza em outras línguas senão o inglês. A dúvida é se realmente a comunidade não possui a cultura de prezar pela documentação em outras línguas ou se o simples fato do sistema ser como é desencoraja sua tradução.

Outro ponto a ressaltar é a própria natureza do sistema do ponto de vista do usuário. Quando se dá os primeiros passos para começar a usá-lo, fica claro que ainda não foi feito um trabalho para torná-lo realmente fácil de usar.

Não há agentes de instalação, mecanismos automáticos de atualização e nem mesmo uma ajuda on-line para servir como guia. Não quer dizer que o sistema seja impossível de ser utilizado por usuários iniciantes, contudo a ausência desses materiais reforça a impressão de que a comunidade tem pouca preocupação com o usuário final.

Uma vez que as interfaces estendidas são praticamente inexistentes, não há muito o que

se avaliar nesse contexto.

Porém, com certeza, aí está uma boa indicação do que a comunidade precisa ter como metas para aumentar as chances de adoção do sistema em maior escala.

Uma medida que não seria tão custosa, em termos de esforços, seria a criação de documentação mais detalhada para que usuários de todas os níveis pudessem aprender a usar o sistema. Ao mesmo tempo que essa medida atrairia mais usuários, também incentivaria a entrada de mais desenvolvedores na comunidade.

A criação de ferramentas de apoio à instalação do sistema e atualização automática, também são pontos a serem colocados como metas. Sistemas grandes e largamente adotados, reconhecidos por sua usabilidade, como sistemas da Microsoft e SAP, prezam por conteúdos e ferramentas como estas.

E claro, todos esses passos devem ser dados tendo-se como padrão a adoção de técnicas adequadas de internacionalização para localizações bem sucedidas no futuro.

8.6 Considerações finais

Neste capítulo foram descritos os passos dados no sentido de realmente estudar o caso de localização do Apache Open For Business de um ponto de vista prático. Algumas das etapas representadas no diagrama da Figura 2.4, que sugere um processo para a localização de um software, foram seguidas e descritas.

Em relação à localização de funcionalidades e a adaptação do sistema à realidade fiscal e contábil brasileira, optou-se por fazer o que se chamou de prova de conceito. Dessa forma, foi possível simplificar a tarefa sem prejudicar a investigação dos aspectos relevantes para a tarefa.

Foram analisadas também as interfaces estendidas existentes e avaliadas as possibilidades para sua localização.

A maior dificuldade foi a constatação da escassez desse tipo de material na comunidade e a baixa preocupação com a internacionalização dos materiais existentes.

Ficou claro com esse estudo que, ao menos no aspecto de interfaces estendidas, a comunidade do Apache Open For Business tem bastante trabalho à frente.

Esse capítulo encerra o estudo proposto e o capítulo que segue apresenta os resultados obtidos e tece algumas conclusões.

Capítulo 9

Resultados e Conclusão

O Estudo da Localização do Apache Open For Business foi executado tendo como norteador o diagrama sugerido na Figura 2.4. O primeiro passo foi a obtenção do conteúdo na língua original, que foi simplesmente buscar no site do projeto e da comunidade todos os arquivos e documentos disponíveis.

Pode-se dizer que o planejamento, mostrado como segunda etapa no diagrama, foi, em primeiro lugar, a revisão bibliográfica, o estudo generalizado dos Sistemas de Gestão Empresarial, o estudo do próprio sistema OFBiz e, por fim, o estudo da realidade brasileira nos aspectos fiscais aplicáveis e dentro do escopo do projeto. Em seguida, ainda no contexto do planejamento, foram feitas as análises apresentadas no Capítulo 7, imprescindíveis para se determinar o esforço necessário para o projeto.

A fase de determinação de custos e prazos foi composta pela estimativa de duração do projeto, baseada nos resultados da etapa de planejamento. A preparação do código, nesse caso, foi o pré-processamento dos arquivos com rótulos da interface para a etapa da tradução, conforme explicitado na Seção 8.1.

Ainda seguindo o fluxo do diagrama da Figura 2.4, foi elaborado um glossário como apresentado na Seção 8.2, contendo todos os termos necessário para traduzir os rótulos da interface mantendo a maior consistência possível.

As etapas de Tradução do Software, Desenvolvimento de Funcionalidades e Integração de Funcionalidades foram todas efetuadas segundo a descrição apresentada no Capítulo 8. Nesse mesmo capítulo, mais precisamente na Seção 8.4, foi descrita também a etapa de adaptação da interface, que consistiu na adição de campos necessários à prova de conceito para o projeto SPED.

A localização das interfaces estendidas, um dos últimos passos do diagrama, foi descrita no Capítulo 8.5 no qual foram ressaltadas as dificuldades encontradas e o que foi possível fazer com o material disponível.

Os últimos passos do processo não foram descritos no texto, mas não deixaram de ser considerados. O controle de qualidade teve como base o fato da tradução dos rótulos

ter sido feita sempre considerando e respeitando o jargão utilizado em *software* de gestão empresarial e de seus usuários.

Os testes de funcionalidade, por sua vez, foram considerados não aplicáveis. Uma vez que a tradução da interface, graças ao bom desacoplamento entre a interface e o código, não impacta em nada as funcionalidades originais. Além disso, como a localização de funcionalidades foi analisada através de uma prova de conceito, não há muito a testar além da certeza da geração do arquivo XML da NF-e dentro do esperado para o teste proposto.

Por fim, a aprovação do cliente também não é muito aplicável neste contexto. O mais próximo que pode-se considerar como aprovação do cliente são os aceites da comunidade para os arquivos de rótulos traduzidos. Nesse âmbito, pode-se dizer que a aprovação foi dada se a comunidade do OFBiz for considerada o cliente.

A cobertura da análise da localização do sistema Open For Business foi completa se considerarmos os passos citados desde o Capítulo 7 e o diagrama da Figura 2.4.

O principal material gerado durante este estudo consiste nos arquivos de rótulos traduzidos que agora permitem que todas as porções do software sejam utilizadas em português brasileiro. Para tal foi necessária a tradução de mais de 14000 rótulos.

Além disso, as principais dificuldades na preparação do sistema para a realidade brasileira foram indicadas no Capítulo 8.4, sempre ponderando as possibilidades da implementação das funções por completo no sistema e preparação dele para a integração com outros sistemas que implementassem as funções necessárias.

Mas um dos principais resultados obtidos foi a compreensão maior de alguns aspectos, que fazem parte da localização de um *software* ERP, e da organização de comunidades de *software* livre, que podem afetar a adoção dos projetos em nível internacional.

Com a execução dos passos propostos no diagrama da Figura 2.4, foi possível entender aspectos que impactam na localização de software ERP e delinear quais seriam os passos necessários para a adoção do OFBiz por empresas brasileiras.

Como a literatura já indicava, um bom projeto, que leve em conta a internacionalização desde o início, torna a sua localização muito mais fácil, tornando também sua adoção ao redor do mundo muito mais provável. No entanto, há aspectos em alguns tipos de *software* que são difíceis de se prever e realmente fogem ao alcance dos desenvolvedores.

Trata-se de aspectos muito específicos de cada país ou cultura, como legislação, fiscalização e tributação. No entanto, um software bem projetado do ponto de vista arquitetural pode tornar mais fácil a adição de funcionalidades ou componentes que supram as necessidades criadas por tais aspectos locais.

No caso do Open For Business, que não é apenas um ERP mas também um *framework* de desenvolvimento muito bem estruturado arquiteturalmente, fica clara essa possibilidade de adaptação ou adição de funcionalidade com baixo esforço e consequente baixo custo.

Adaptações, que potencialmente demandariam muito mais linhas de código em outros modelos de desenvolvimento de software, são alcançadas com poucas linhas no OFBiz,

muitas vezes impactando muito pouco em arquivos originais, o que faz com que a possibilidade de atualizações do sistema não seja comprometida. Um exemplo claro disso, foi a facilidade demonstrada para adição de campos às entidades consideradas necessárias para a prova de conceito para a NF-e.

Isso corrobora os argumentos da literatura [33][34] de que sistemas projetados de forma mais modular facilitam sua localização quanto a aspectos funcionais mais dependentes da implementação e ligados à cultura do usuário. A arquitetura do OFBiz se mostrou bastante modular e flexível quando avaliada segundo esse critério.

Mas, apesar dessa facilidade de desenvolvimento, o grande problema está na documentação do projeto OFBiz. A comunidade aparentemente não se preocupa tanto quanto deveria com a qualidade da documentação para induzir uma adoção mais ampla do software. Apesar de pouco explicitado nas seções anteriores, a documentação existente é voltada praticamente só para desenvolvedores. A curva de aprendizagem do *framework* em si pode ser bastante limitada pela dificuldade de entender as metalinguagens utilizadas, por exemplo.

Além disso, como já ressaltado, a documentação para usuários finais é praticamente inexistente, o que torna sua adoção difícil até mesmo para usuários nativos da língua inglesa.

O projeto OFBiz parece se organizar de forma que haja espaço apenas para desenvolvedores experientes e não para usuários finais. De qualquer forma há um lado positivo nisso: esse modelo abre espaço para criação de empresas cujo modelo de negócio seja a prestação de serviços baseados no *framework*.

E essa é uma das principais características que, juntamente com a facilidade de adaptação, faz acreditar que o OFBiz seria sim adotável na realidade brasileira para prover a micro e pequenas empresas uma alternativa barata e descomplicada para sistemas de gestão empresarial. Mas um foco maior na documentação certamente facilitaria o sucesso nesse cenário.

Um paralelo possível de ser feito é a comparação com o projeto Open Office. Apesar de ser um software de natureza totalmente diversa do OFBiz, consiste em um exemplo de sucesso em termos de localização e adoção.

O Open Office, com ramificação recente chamada LibreOffice, possui uma localização completa para o português e, além disso, possui também uma comunidade bastante expressiva no Brasil a ponto de possuir uma versão dele desenvolvida no país e com ampla documentação. Seu nome é BrOffice e é consideravelmente bem adotado por usuários brasileiros.

E aí estão duas características determinantes para o sucesso desse projeto: a força da comunidade local e a documentação disponível. Portanto, pensando no sucesso obtido pelo Open Office, pode-se dizer que a formação de uma comunidade local poderia impulsionar sua adoção no Brasil. Inclusive, o fomento de uma comunidade como essa por parte das

autoridades poderia ser uma forma de investimento com vistas ao desenvolvimento da indústria brasileira.

Este estudo avaliou o caso da localização do Apache OFBiz ao contexto brasileiro e, com as etapas constituintes do processo, é possível apontar algumas possibilidades de trabalhos futuros. Inspirado pela possibilidade apontada de terceirizar serviços, seria possível, por exemplo, criar um sistema que fosse flexível e capaz de adequar sistemas de gestão empresarial às diversidades fiscais e às constantes alterações que sofrem.

Outro tema a ser explorado, mais diretamente relacionado à área de interfaces, seria o desenvolvimento de um sistema de apoio para o controle da produção de interfaces estendidas. Como foi descrito, essa é uma área que deixa a desejar no projeto OFBiz, assim como em muitos outros projetos de código livre. Dessa forma, foco no usuário final pode ser uma oportunidade de contribuir para mudanças positivas na organização de tais projetos e, ainda, para sua adoção em larga escala.

Apêndice A

Heurísticas de Usabilidade

Na área de usabilidade normalmente são aceitas como diretrizes para avaliação e projetos de interfaces as dez heurísticas proposta por Nielsen [41][42].

A seguir, tais heurísticas são reproduzidas para complementar os pontos indicados no presente trabalho.

1. Visibilidade do estado do sistema

O sistema deve sempre manter os usuários informados e conscientes sobre o que está acontecendo, através de *feedback* apropriado e em tempo razoável.

2. Consistência entre o sistema e mundo real

O sistema deve ser expresso em função da linguagem do usuário, com palavras, frases e conceitos familiares ao usuário. Termos específicos do domínio do sistema devem ser evitados. Devem ser seguidas convenções do mundo real, apresentando as informações em uma ordem natural e lógica.

3. Controle do usuário e liberdade

É comum que os usuários cometam erros ao usar a interface. É necessário que funções de "desfazer" e "refazer" sejam apresentadas de forma fácil para que os usuários possam se recuperar de erros e não sejam desestimulados a usar os sistema. Quaisquer mensagens apresentadas devem ser curtas e de fácil compreensão.

4. Consistência e padrões

Os usuários não devem precisar inferir que diferentes palavras, situações ou ações significam a mesma coisa. Convenções da plataforma devem ser utilizadas.

5. Prevenção de erros

Evitar que problemas ocorram é melhor que possuir boas mensagens de erro. Deve-se buscar eliminar condições que possam induzir a erros e, quando não for possível, o sistema deve alertar o usuário e perder alguma espécie de confirmação.

6. Reconhecimento ao invés de *recall*

Minimizar a necessidade do usuário se esforçar para lembrar de objetos, ações ou opções. Todos os elementos que precisem ser levados em conta pelo usuário devem estar devidamente visíveis e destacados. Instruções para o uso devem estar sempre visíveis ou fáceis de serem acessadas.

7. Flexibilidade e eficiência de uso

Atalhos, ou quaisquer meios para acelerar o uso do sistema, devem estar invisíveis do ponto de vista do usuário inexperiente. No entanto, devem estar presentes para que usuários mais experientes possam executar tarefas em menos tempo. Em outras palavras, a interface deve atender a ambos os perfis de usuário.

8. Estética e design minimalista

Diálogos não devem conter informações irrelevantes ou raramente necessárias. Cada informação extra ofuscará as demais, complicando o uso e confundindo o usuário.

9. Ajudar os usuários a reconhecer, diagnosticar e se recuperar de erros

Mensagens de erro devem ser expressas em linguagem clara (sem códigos), indicar com precisão o problema e construtivamente sugerir uma solução.

10. Ajuda e documentação

Idealmente o sistema deveria poder ser usado sem nenhum tipo de documentação. No entanto, esse tipo de material deve ser fornecido. A documentação deve também ser de fácil consulta e ser focada em tarefas do usuário, se possível a nível de cada passo necessário para sua execução.

Apêndice B

Dicionário de Termos Utilizados na localização

Termo Original	Tradução
Accepted	Aceito
Account/Accounts	Conta/contas
Accounting	Contabilidade
Accounting Transaction	Lançamento Contábil
Accounts Payable (AP)	Contas a pagar
Accounts Receivable (AR)	Contas a receber
Activated	Ativado
Adjustment	Ajuste
Admin	Administrador
Agent	Agente
Agreement	Convênio
Amount	Quantia
Apply (payment)	Aplicar (Pagamento)
Approved	Aprovado
Assign	Atribuir
Assigned	Atribuído
Association	Designação
ATP (Available to Promise)	ATP (Disponível para reserva)
Authorize	Autorizar
Back order	Pedido em aberto
Balance Due	Saldo Devedor
Balance Sheet	Balanco patrimonial

Bill Of Materials (BOM)	Estrutura de produtos
Billing	Fatura
Billing Account	Conta para fatura
Cancelled	Cancelado
Capacity	Capacidade
Capture	Debitar
Carrier	Transportadora
Cart	Carrinho
Category	Categoria
Catalog	Catálogo
Channel	Canal
Chart of Accounts	Plano de contabilidade
Classifications	Classificações
Classification Group	Grupo de classificação
CMS	CMS (Sistema de gerenciamento de conteúdo)
Comments	Comentários
Commission	Comissão
Communication	Comunicação
Communication Event	Evento de comunicação
Completion	Conclusão
Completed	Completo
Contact Mech	Mecanismo de contato
Configuration	Configuração
Confirm	Confirmar
Cost Calc	Cálculo de custos
Cost Of Goods Sold (COGS)	Custo de mercadorias vendidas
Custom (Method)	Personalizado (método)
Credit	Crédito
Credit Card	Cartão de crédito
Created	Criado
Date/Dates	Data/Datas
Deactivated	Desativado
Declaration	Resposta
Default	Padrão
Deliverable Product	Produto concluído
Delivery	Entrega
Depreciation	Depreciação

Distributor	Distribuidor
Does not exist	Não existe
Draft	Rascunho
Drop shipment	Drop Shipping
Edit	Editar
EFT Account	Conta para transferência eletrônica
Email	E-Mail
Estimated	Estimado
Event	Evento
Exclude	Excluir
Expense	Custos
Expire	Expirar
Facility	Instalação
Facility Locations	Local da instalação
Fail	Falhas
Feature	Função
Final Draft	Rascunho final
Find ...	Buscar...
Fixed Asset	Ativo fixo
Fixed Cost	Custo fixo
Gift-Card	Vale presente
GL Account	Conta contábil
GL Account Type	Tipo de conta contábil
Government Agency	Orgão governamental
Hide ...	Esconder...
History	Histórico
Hold	Aguardar
Identification	Identificação
Income Statement	Demonstração do Resultado do Exercício
Include	Incluir
In progress	Em andamento
Invitation	Convite
Inventory	Inventário
Inventory Event	Evento de inventário
Inventory Item	Item de inventário
Inventory Transfer	Transferência de Inventário
Invoice	Fatura

Issuance	Emissão
Job Shop	Produtor terceirizado
Liabilities	Passivos
List ...	Listar...
Location	Localização
Login	Login
Lookup ...	Procurar...
Name on account	Nome da conta
Main Page	Página inicial
Maintenance	Manutenção
Mandatory	Obrigatório
Manufacturing Rules	Regras de produção
Manufacturing Instruction	Instruções para produção
Manufacturing Task	Tarefa de produção
Max Retry	Número máximo de tentativas
MRP	MRP
Need	Necessidade
Net book value	Custo original
Net Income	Lucro líquido
Next	Próximo
On hold	Em suspenso
Order	Pedido
Order Item	Item do pedido
Organization	Organização
Override	Sobrescrever
Parent	Elemento Superior
Party	Participante
Payment	Pagamento
Payment Application	Solicitação de pagamento
Payment Group	Grupo de pagamento
Payment Method	Método de pagamento
Payment Transaction	Transação de pagamento
Pending	Pendente
Planned	Planejado
Product	Produto
Product Catalog	Catálogo de produtos
Product Feature	Característica do Produto

Prospect	Cliente em ptencial
Processing	Em processamento
Production Run	Etapa de produção
Promised	Reservado
Promotion	Promoção
Promotion code	Código Promocional
Published	Publicado
Purchase	Compra
Purchase Invoice	Fatura de compra
Purchase Order	Pedido de compra
Qty	Qta
Quantity on hand	Quantidade disponível
Quote	Orçamento
Rating	Avaliação
Receive	Receber
Refund	Reembolso
Release	Liberar
Rejected	Rejeitado
Remaining	Restante
Report	Relatório
Requested	Solicitado
Request	Solicitar
Required	Obrigatório
Requirement	Requisito
Resolved	Solucionado
Return	Devolver
Returned	Devolvido
Retry	Tentar novamente
Revenue	Receita
Review	Analisar/avaliado
Reviewed	Analísado/avaliado
Revised	Revisado
Role	Papel/cargo
Routing	Itinerário
Routing Number	Número de itinerário
Rule	Regulamento
Run Time	Tempo de execução

Sales	Vendas
Sales Invoice	Fatura de vendas
Sales Order	Pedidos de venda
Salvage cost	Custo de recuperação
Schedule	Programação
Scheduled	Programado
Security	Segurança
Segment	Segmento
Sent	Enviado
Sequence	Sequência
Settings	Configurações
Setup	Configuração
Shipment	Remessa
Shipment Plan	Plano de Remessa
Shipment Route	Rota da Remessa
Shipping	Remessa
Ship Group	Grupo de remessa
Snail mail	Correio tradicional
Standard Cost	Custo padrão
Status	Estado
Submitted	Enviado
Subscription	Inscrição
Suppliers	Fornecedores
Survey	Pesquisa
Tax Authority	Órgão Tributário
Tech. Data	Dados técnicos
Terminated	Concluído
Terms	Condições
Time Period	Período
Total	Total
Transaction	Transação
Transaction Entry	Lançamento de transação
Trial Balance	Balancete
Type	Tipo
Unit Price	Preço unitário
UOM	UDM (Unidade de Medida)
User Login	Nome de usuário

Vendor	Fabricante
Variant	Versão
Virtual Variant	Versão virtual
Visit	Visita
Visitor	Visitante

Tabela B.1: Dicionário construído para a localização

Apêndice C

Dimensões para tradução do diretório applications do OFBiz

applications/accounting		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
AccountingErrorUiLabels.xml	0	20
AccountingEntityLabels.xml	0	348
AccountingUiLabels.xml	0	1415
Total		1783

Tabela C.1: Arquivos e números de rótulos do módulo Accounting

applications/commonext		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
CommonExtUiLabels.xml	0	7
CommonExtUiLabels.xml	0	27
Total		34

Tabela C.2: Arquivos e números de rótulos do módulo commonext

applications/content		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
ContentEntityLabels.xml	0	257
ContentErrorUiLabels.xml	0	18
ContentUiLabels.xml	0	470
Total		745

Tabela C.3: Arquivos e números de rótulos do módulo content

applications/humanres		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
HumanResUiLabels.xml	1	315
Total		315

Tabela C.4: Arquivos e números de rótulos do módulo de Recursos Humanos

manufacturing/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
ManufacturingEntityLabels.xml	0	18
ManufacturingReportsUiLabels.xml	0	36
ManufacturingUiLabels.xml	0	398
Total		452

Tabela C.5: Arquivos e números de rótulos do módulo Manufacturing

applications/marketing		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
MarketingEntityLabels.xml	0	9
MarketingUiLabels.xml	0	278
Total		287

Tabela C.6: Arquivos e números de rótulos do módulo Marketing

applications/order		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
OrderEntityLabels.xml	0	81
OrderErrorUiLabels.xml	0	415
OrderUiLabels.xml	0	922
Total		1418

Tabela C.7: Arquivos e números de rótulos do módulo Order

applications/party		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
PartyEntityLabels.xml	0	283
PartyErrorUiLabels.xml	0	81
PartyUiLabels.xml	0	821
Total		1183

Tabela C.8: Arquivos e números de rótulos do módulo Party

applications/product		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
ProductEntityLabels.xml	0	363
ProductUiLabels.xml	0	2123
ProductErrorUiLabels.xml	0	112
Total		2598

Tabela C.9: Arquivos e números de rótulos do módulo Product

applications/workeffort		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
WorkEffortEntityLabels.xml	0	35
WorkEffortUiLabels.xml	0	407
Total		442

Tabela C.10: Arquivos e números de rótulos do módulo WorkEffort

Apêndice D

Dimensões para tradução do diretório SpecialPurpose do OFBiz

specialpurpose/assetmaint/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
AssetMaintUiLabels.xml	0	16
IsMgrUiLabels.xml	0	15
Total		31

Tabela D.1: Arquivos e números de rótulos usados no componente Assetmaint

specialpurpose/ebay/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
EbayUiLabels.xml	0	85
Total		85

Tabela D.2: Arquivos e números de rótulos usados no componente Ebay

specialpurpose/ebaystore/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
EbayStoreUiLabels.xml	0	103
Total		103

Tabela D.3: Arquivos e números de rótulos usados no componente Ebaystore

specialpurpose/ecommerce/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
EcommerceUiLabels.xml	1	306
Total		306

Tabela D.4: Arquivos e números de rótulos usados no componente Ecommerce

specialpurpose/googlebase/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
GoogleBaseUiLabels.xml	0	37
Total		37

Tabela D.5: Arquivos e números de rótulos usados no componente Googlebase

specialpurpose/googlecheckout/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
GoogleCheckoutUiLabels.xml	0	6
Total		6

Tabela D.6: Arquivos e números de rótulos usados no componente Googlecheckout

specialpurpose/myportal/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
MyPortalUiLabels.xml	0	25
Total		25

Tabela D.7: Arquivos e números de rótulos usados no componente Myportal

specialpurpose/oagis/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
OagisUiLabels.xml	1	37
Total		37

Tabela D.8: Arquivos e números de rótulos usados no componente Oagis

specialpurpose/ofbizwebsite/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
OfbizUiLabels.xml	0	43
Total		43

Tabela D.9: Arquivos e números de rótulos usados no componente Ofbizwebsite

specialpurpose/pos/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
PosUiLabels.xml	0	76
Total		76

Tabela D.10: Arquivos e números de rótulos usados no componente POS

specialpurpose/projectmgr/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
ProjectMgrUiLabels.xml	0	235
Total		235

Tabela D.11: Arquivos e números de rótulos usados no componente Projectmgr

specialpurpose/shark/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
SharkUiLabels.xml	1	10
Total		10

Tabela D.12: Arquivos e números de rótulos usados no componente Shark

specialpurpose/webpos/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
SharkUiLabels.xml	0	111
Total		111

Tabela D.13: Arquivos e números de rótulos usados no componente WebPOS

specialpurpose/workflow/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
WorkflowUiLabels.xml	0	19
Total		19

Tabela D.14: Arquivos e números de rótulos usados no componente Workflow

Apêndice E

Dimensões para tradução do diretório Framework do OFBiz

framework/common/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
CommonEntityLabels.xml	0	1411
CommonHelpUiLabels.xml	0	2
CommonUiLabels.xml	14	747
PrefErrorUiLabels.xml	0	7
TemporalExpressionUiLabels.xml	0	24
CommonErrorUiLabels.xml	0	10
CommonPortalEntityLabels.xml	0	4
SecurityextUiLabels.xml	0	55
SecurityUiLabels.xml	0	43
Total		1511

Tabela E.1: Arquivos e números de rótulos comuns a muitos componentes

framework/webtools/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
WebtoolsErrorUiLabels.xml	0	32
WebtoolsUiLabels.xml	0	457
Total		489

Tabela E.2: Arquivos e números de rótulos usados no componente Webtools

framework/bi/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
BiUiLabels.xml	1	47
Total		47

Tabela E.3: Arquivos e números de rótulos usados no componente BI

framework/base/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
DateTimeLabels.xml	0	10
Total		10

Tabela E.4: Arquivos e números de rótulos usados no componente Base

framework/birt/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
BirtUiLabels.xml	0	4
Total		4

Tabela E.5: Arquivos e números de rótulos usados no componente Birt

framework/example/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
ExampleEntityLabels.xml	0	22
ExampleHelpUiLabels.xml	0	1
ExampleUiLabels.xml	1	113
Total		136

Tabela E.6: Arquivos e números de rótulos usados no componente Example

framework/guiapp/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
XuiUiLabels.xml	0	3
Total		3

Tabela E.7: Arquivos e números de rótulos usados no componente Guiapp

framework/minilang/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
DefaultMessages.xml	0	11
MiniLangErrorUiLabels.xml	0	4
Total		15

Tabela E.8: Arquivos e números de rótulos usados no componente Minilang

framework/security/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
SecurityEntityLabels.xml	0	235
Total		235

Tabela E.9: Arquivos e números de rótulos usados no componente Security

framework/service/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
ServiceErrorUiLabels.xml	0	6
Total		6

Tabela E.10: Arquivos e números de rótulos usados no componente Service

framework/webapp/config		
Arquivos Constituintes	Rótulos pt_BR	Total de rótulos
WebappEntityLabels.xml	0	5
WebappUiLabels.xml	0	26
Total		31

Tabela E.11: Arquivos e números de rótulos usados no componente Webapp

Apêndice F

Alteração necessária para adição de campos

A seguir é apresentada a Listagem F.1 com o código do arquivo *applications/product/widget/catalog/ProductForms.xml* referente à adição de campos descrita no Capítulo 8.4.2. Mesmo essa porção de código, foi comprimida, omitindo-se algumas porções para que não ocupasse muito espaço. No lugar dos trechos omitidos foram adicionados alguns comentários indicando a omissão.

```
1 <form name="EditProduct" type="single" target="updateProduct" title=""
  default-map-name="product"
2     header-row-style="header-row" default-table-style="basic-table">
3
4     <alt-target use-when="product==null" target="createProduct"/>
5
6     <field use-when="product==null" name="isCreate"><hidden value="true"
7         /></field>
8
9     <field position="1" use-when="product!=null" name="productId" title=
10        "${uiLabelMap.ProductProductId}" tooltip="${uiLabelMap.
11            ProductNotModificationRecreatingProduct}"><display /></field>
12
13     <field position="1" use-when="product==null&&productId==null
14        " name="productId" title="${uiLabelMap.ProductProductId}"><text
15            size="20" maxlength="20" /></field>
16
17     <!-- - 186 linhas omitidas - -->
18
19     <field use-when="product!=null" position="1" name="lastUpdatedByText
20        " title="${uiLabelMap.ProductLastModifiedBy}:">
21        <display description="[ ${product.lastModifiedByUserLogin} ] ${
22            uiLabelMap.CommonOn} ${product.lastModifiedDate}" also-hidden
23            ="false"/>
24    </form>
```

```

15     </field>
16     <field use-when="product!=null" position="2" name="createdByText"
17         title="{uiLabelMap.CommonCreatedBy}:">
18         <display description="[{$product.createdByUserLogin}] {$
19             uiLabelMap.CommonOn} {$product.createdDate}" also-hidden="
20             false"/>
21     </field>
22     <!-- -adicionado para NFe-->
23     <field position="1" name="codigo" title="Codigo">text size="25"
24         maxlength="255"/></field>
25     <field position="1" name="descricao" title="Descricao">text size="
26         75" maxlength="255"/></field>
27     <!-- -/adicionado para NFe-->
28     <sort-order>
29         <field-group>
30             <sort-field name="productId"/>
31             <sort-field name="productId"/>
32         </field-group>
33     <!-- - 59 linhas omitidas -->
34     <field-group title="{uiLabelMap.CommonMiscellaneous}"
35         collapsible="true" initially-collapsed="true">
36         <sort-field name="returnable"/>
37         <sort-field name="includeInPromotions"/>
38         <sort-field name="taxable"/>
39         <sort-field name="autoCreateKeywords"/>
40     </field-group>
41     <!-- -adicionado para NFe-->
42     <field-group title="Dados para NF-e" collapsible="true"
43         initially-collapsed="true">
44         <sort-field name="codigo"/>
45         <sort-field name="descricao"/>
46     </field-group>
47     <!-- -/adicionado para NFe-->
48 </sort-order>
49 </form>

```

Listagem F.1: Porção do arquivo ProductForms.xml referente à tela de edição de produtos

Apêndice G

Ferramentas Utilizadas

Como apontado no Capítulo 8 foram utilizados alguns pequenos programas para auxiliar a preparação do código a ser traduzido. Esses códigos e outros arquivos utilizados durante todo o processo podem ser encontrados no seguinte endereço:

<http://www.harriss.com.br/arquivosLocalizacao0FBiz>

Referências Bibliográficas

- [1] R. Abou-Lamous, O. Beninatto. *The Guide to Translation and Localization - Communicating with the Global Marketplace*. Lingo Systems, Portland, OR, Estados Unidos, 7^a edition, 2009.
- [2] Acclaro. Software localization process, Abril 2008. <http://www.acclaro.com/software-localization-process> (Acessado em: 14/04/2010).
- [3] S. E. Albertão. *ERP: Sistemas de Gestão empresarial: metodologia para avaliação, seleção e implantação: para pequenas e médias empresas*. Editora Iglu, São Paulo, SP, Brasil, 2005.
- [4] SA Alwabel, AM Ahmed, and M. Zairi. The Evolution of ERP and its Relationship with E-business. *Electronic Business: Concepts, Methodologies, Tools, and Applications*, page 59, 2009.
- [5] O. Reis Azevedo. *SPED: Sistema Público de Escrituração Digital*, volume 1. IOB, São Paulo, SP, Brasil, 2a. edition, 2009.
- [6] J. Cappellato. Taxauthority, 2010. <https://cwiki.apache.org/confluence/display/OFBIZ/VAT> (Acessado em: 20/12/2010).
- [7] S.K. Card, T.P. Moran, and A. Newell. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 23(7):396–410, 1980.
- [8] S. Chen. Developing applications with ofbiz - an overview, 2008. http://www.opensourcestrategies.com/ofbiz/developing_overview.php (Acessado em: 21/05/2010).
- [9] OFBiz Community. Apache ofbiz project, 2009. <http://docs.ofbiz.org/display/OFBADMIN/Adobe+OFBiz+Project+Overview> (Acessado em: 05/06/2009).
- [10] OAGi Architecture Council. Open application group - open standards that open markets, 2007. <http://www.oagi.org/dnn2/AboutUs/ArchitectureCouncil.aspx> (Acessado em: 15/10/2009).

- [11] Ministério da Fazenda. Schemas nf-e - portal nacional da nota fiscal eletrônica, 2010. <http://www.nfe.fazenda.gov.br/portal/schemas.aspx> (Acessado em: 05/11/2010).
- [12] Presidência da República Casa Civil. Lei nº 5.172, 25 de Outubro de 1996. <http://www.planalto.gov.br/ccivil/leis/L5172.htm> (Acessado em: 15/10/2009).
- [13] Receita Federal do Brasil. Portal nacional da nota fiscal eletrônica - descrição da nota fiscal eletrônica, 2009. <http://www.nfe.fazenda.gov.br/portal/descricao.aspx> (Acessado em: 25/07/2009).
- [14] Receita Federal do Brasil. Sistema público de escrituração digital, 2009. <http://www1.receita.fazenda.gov.br/> (Acessado em: 25/07/2009).
- [15] Receita Federal do Brasil. Sistema público de escrituração digital - sped-contábil, 2009. <http://www1.receita.fazenda.gov.br/sped-contabil/como-funciona.htm> (Acessado em: 25/07/2009).
- [16] Receita Federal do Brasil. Sistema público de escrituração digital - sped-fiscal, 2009. <http://www1.receita.fazenda.gov.br/sped-fiscal/como-funciona.htm> (Acessado em: 25/07/2009).
- [17] Receita Federal do Brasil. Sistema público de escrituração digital - universo de atuação, 2009. <http://www1.receita.fazenda.gov.br/sobre-o-projeto/universo-de-atuacao.htm> (Acessado em: 25/07/2009).
- [18] E. M. Ferreira. Uninfe - sumário, 2010. <http://sourceforge.net/projects/uninfe/> (Acessado em: 01/11/2010).
- [19] S. Foga. Tax authorities, 2010. <https://cwiki.apache.org/confluence/display/OFBENDUSER/08+Tax+Authorities> (Acessado em: 20/12/2010).
- [20] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*, volume 206. Addison-wesley Reading, MA, 1995.
- [21] E. Haberkon. *Um bate-papo sobre gestão empresarial com ERP: tudo o que você gostaria de saber sobre ERP e tecnologia da informação, mas ficava encabulado de perguntar*. Editora Saraiva, São Paulo, SP, Brasil, 2007.
- [22] E. Haberkon. *Gestão Empresarial com ERP*, volume 1. Projeto Totvs da Educação, São Paulo, SP, Brasil, 4a. edition, 2008.

- [23] K.B. Hendricks, V.R. Singhal, and J.K. Stratman. The impact of enterprise systems on corporate performance: A study of ERP, SCM, and CRM system implementations. *Journal of Operations Management*, 25(1):65–82, 2007.
- [24] Ruth Hoffman. *Apache OFBiz Cookbook*. Packt Publishing, 2010.
- [25] A. Holzinger. Usability engineering methods for software developers. *Communications of the ACM*, 48(1):71–74, 2005.
- [26] Rupert Howell. *Apache OFBiz Development: The Beginner's Tutorial*. Packt Publishing, 2008.
- [27] E. Huang, R. Haft, and J. Hsu. Developing a roadmap for software internationalization. *Engineering the Global Enterprise, Symbio Group*: <http://www.isp-toin.com/ressources.html> (11.01. 2005), 2001.
- [28] RIC International. Software localization process outline, Junho 2009. <http://www.ricintl.com/software-localization-process.html> (Acessado em: 14/04/2010).
- [29] What is Usability? Resources: About usability - usability professionals' association, 2003. http://www.upassoc.org/usability_resources/about_usability/definitions_of_usability.html (Acessado em: 28/01/2010).
- [30] What is User-Centered Design? Resources: About usability - usability professionals' association, 2003. http://www.upassoc.org/usability_resources/about_usability/what_is_ucd.html (Acessado em: 28/01/2010).
- [31] D. Jones. Mini-language guide, 2008. <https://cwiki.apache.org/confluence/display/OFBIZ/Mini-Language%20Guide> (Acessado em: 23/05/2010).
- [32] D. E Jones. Apache ofbiz project overview, 2009. <http://ofbiz.apache.org/> (Acessado em: 05/06/2009).
- [33] G.E. Kersten, M.A. Kersten, and W.M. Rakowski. Software and culture: Beyond the internationalization of the interface. *Journal of Global Information Management*, 10(4):86, 2002.
- [34] G.E. Kersten, S. Matwin, S.J. Noronha, and M.A. Kersten. The software for cultures and the cultures in software. In *Proceedings of the European Conference on Information Systems, Vienna, Austria*. Citeseer, 2000.
- [35] C. Lewis and J. Rieman. Task-centered user interface design. *Available via ftp. cs.colorado.edu/pub/cs/distrib/clewis/HCI-Design-Books*, 1993.

- [36] N. Malin. Localized date format for end user, 2010. <https://issues.apache.org/jira/browse/OFBIZ-3843> (Acessado em: 07/11/2010).
- [37] C. Mangiron and M. O'Hagan. Game localisation: unleashing imagination with 'restricted' translation. *The Journal of Specialised Translation*, 6:10–21, 2006.
- [38] M. Mosiewicz. Vat, 2010. <https://cwiki.apache.org/confluence/display/OFBIZ/VAT> (Acessado em: 20/12/2010).
- [39] J.L. Nhampossa. The Challenge Of “Translating” Health Information Systems From One Developing Country Context To Another: Case Study From Mozambique. *Innovations through information technology, Turku, Finland*, 2004.
- [40] J. Nielsen. Usability inspection methods. In *Conference companion on Human factors in computing systems*, pages 413–414. ACM, 1994.
- [41] J. Nielsen. Ten usability heuristics. *Useit. com*, 2005.
- [42] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, pages 249–256. ACM, 1990.
- [43] John O'Conner. Java internationalization: Localization with resourcebundles, Outubro 1998. <http://java.sun.com/developer/technicalArticles/Intl/ResourceBundles> (Acessado em: 21/03/2010).
- [44] L. Pavesi Esteves. *Manual Tributário do Empresário*. Qualitymark, 2007.
- [45] J. Le Roux. Tips for ui labels translation, 2010. <https://cwiki.apache.org/confluence/display/OFBIZ/Tips+for+UI+labels+translation> (Acessado em: 20/12/2010).
- [46] P. Russo and S. Boor. How fluent is your interface?: designing for international users. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, New York, NY, USA, 1993. ACM.
- [47] Maxim Sadek, George e Zhukov. *Typographia Polyglotta: A Comparative Study in Multilingual Typesetting*. Association Typographique Internationale, 1997.
- [48] C. Schinzer. Guide to ofbiz-i18n, internationalisation of ofbiz, 2010. <https://cwiki.apache.org/confluence/display/OFBIZ/Guide+to+OFBiz-i18n,++Internationalisation+of+OFBiz> (Acessado em: 20/12/2010).
- [49] M. Sumner. *Enterprise resource planning*. Pearson Education, 2007.

- [50] Usability 101: Introduction to Usability. Jakob nielsen's alertbox, 2003. <http://www.useit.com/alertbox/20030825.html> (Acessado em: 28/01/2010).
- [51] Y.W.E.J. Whitehead Jr. International accessibility of open source software. 2001.
- [52] S. Williams and N. Williams. *The profit impact of business intelligence*. Morgan Kaufmann, 2007.