UNIVERSIDADE FEEVALE

ROBINSON LOVATTO

ESTUDO DE VIABILIDADE SOBRE A MIGRAÇÃO DE UM SISTEMA ORACLE FORMS 6I PARA UM AMBIENTE WEB BASEADO EM TECNOLOGIAS ORACLE

Novo Hamburgo

2013ROBINSON LOVATTO

ESTUDO DE VIABILIDADE SOBRE A MIGRAÇÃO DE UM SISTEMA ORACLE FORMS 6I PARA UM AMBIENTE WEB BASEADO EM TECNOLOGIAS ORACLE

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Feevale.

Orientador: Juliano Varella de Carvalho

Novo Hamburgo

2013

**AGRADECIMENTOS**

Gostaria de agradecer a todos que de alguma maneira contribuíram para a realização desse trabalho de conclusão, em especial à minha mãe, padrasto e namorada pela compreensão e paciência nesse período tão difícil. O apoio de vocês foi fundamental para passar por esta etapa da minha vida.

Aos professores que contribuíram com minha formação e, em especial, ao Prof. Me. Juliano Varella de Carvalho que disponibilizou seu tempo e seus conhecimentos para me orientar.

**Resumo**

Hoje em dia as empresas investem muito dinheiro em softwares, e para ter retorno sobre o investimento, os mesmos devem ser usados por muito tempo. Com o passar dos anos, as tecnologias utilizadas no desenvolvimento se tornam obsoletas, e não mais atendem a determinadas demandas, tampouco novos projetos, podendo até limitar a evolução dos processos de negócio da empresa. Outro agravante é a manutenção do software, que se torna cada vez mais cara, pois o sistema passa por diversas alterações ao longo dos anos. Nesse ponto, surge a questão: o que fazer com o sistema legado? Pode-se descartar o sistema e adquirir um novo, porém essa estratégia está sujeita a muitos riscos; pode-se continuar a manter o sistema, entretanto sob as consequências já descritas; ou ainda pode-se alterar o sistema de alguma maneira para que sua manutenção se torne mais fácil e barata. Neste contexto, o trabalho proposto busca explorar ferramentas de conversão que possam fazer a tradução de código-fonte de um sistema desenvolvido em *Oracle Forms 6i* para uma tecnologia web. Entre as ferramentas de conversão escolhidas estão: *Oracle Forms 11g*, *Oracle Forms Migration Assistant*, Oracle ADF e *Application Express* (APEX). Essas ferramentas convertem o sistema para as tecnologias *Forms 11g*, Java e html. Dessa maneira, a base tecnológica da empresa pode ser atualizada e as demandas dos clientes que excedem a capacidade do software legado, atendidas.

Palavras-chave: Migração de Sistemas. Oracle Forms. Viabilidade.Web. Sistemas Legados.

**Abstract**

Nowadays, companies invest a lot of money on software and to have a return on investment, they should be used for a long time. Over the years, the technologies used in the development become obsolete and no longer meet certain demands, nor new projects and may even limit the evolution of business processes of the company. Another problem is maintenance, which becomes increasingly expensive as the system has undergone several changes over the years. At this point, the question arises: what to do with the legacy system? You can discard the system and acquire a new one, but this strategy is subject to many risks, you can continue to maintain the system, however under the consequences described above or, you can change the system in some way for the maintenance to become easier and cheaper. In this context, the proposed work in aims to explore conversion tools that can translate the source code of a system developed in Oracle Forms 6i to web technology. The chosen conversion tools are: Oracle Forms 11g, Oracle Forms Migration Assistant and Oracle ADF Application Express (APEX). These tools converts the system to Forms 11g, Java and html. Thus, the base technology of the company can be updated and customers demands that exceed the capacity of the legacy software, are met.

Keywords:Migration Systems. Oracle Forms. Feasibility.Web. Legacy Systems.

**Lista de FIGURAS**

[Figura 1 - Engenharia direta e reengenharia 15](#_Toc361257357)

[Figura 2 - Processo de reengenharia 16](#_Toc361257358)

[Figura 3 - Processo de engenharia reversa 18](#_Toc361257359)

[Figura 4 - Componentes do ambiente e sua interação 34](#_Toc361257360)

[Figura 5 - Diagrama Entidade-Relacionamento 35](#_Toc361257361)

[Figura 6 - Cadastro de Tipo\_Pessoa 36](#_Toc361257362)

[Figura 7 - Cadastro de UF 37](#_Toc361257363)

[Figura 8 - Cadastro de Item 38](#_Toc361257364)

[Figura 9 - Cadastro de Venda e Venda\_Item 38](#_Toc361257366)

[Figura 10 - Cadastro de Pessoa 40](#_Toc361257367)

[Figura 11 - Menu 41](#_Toc361257368)

[Figura 12 - Tela Principal 41](#_Toc361257369)

[Figura 13 - Help (biliotecas PL/SQL e de objetos) 42](#_Toc361257370)

[Figura 14 - Compilação em lote no Forms 11g 44](#_Toc361257371)

[Figura 15 - Cadastro de Pessoas convertido com Forms 11g 45](#_Toc361257372)

[Figura 16 - Cadastro de Itens convertido com Forms 11g 45](#_Toc361257373)

[Figura 17 - Arquivos a serem convertidos 46](#_Toc361257374)

[Figura 18 - Log de Conversão 47](#_Toc361257375)

[Figura 19 - Processo de Conversão Forms para APEX 49](#_Toc361257376)

[Figura 20 - Erro no carregamento do arquivo XML do cadastro de pessoas 51](#_Toc361257377)

[Figura 21 - Erro na inserção de registros Tipo Pessoa 52](#_Toc361257378)

[Figura 22 - Cadastro de Itens no Apex 52](#_Toc361257379)

[Figura 23 - Cadastro de Pessoas no Apex 53](#_Toc361257380)

[Figura 24 - Cadastro de Vendas no Apex 54](#_Toc361257381)

[Figura 25 - Cadastro de Itens da Venda no Apex 55](#_Toc361257382)

[Figura 26 - Home (menu Apex) 55](#_Toc361257383)

**LISTA DE QUADROS**

Quadro 1 - Licenciamento Weblogic 63

Quadro 2 - Licenciamento Forms 11g 63

**LISTA DE ABREVIATURAS E SIGLAS**

ADF Application Developer Framework

AJAX Asynchronous Javascript and XML

APEX Application Express

CRUD Create, Read, Update and Delete

DER Diagrama Entidade Relacionamento

GUI Graphical User Interface

HTML HyperText Markup Language

IDE Integrated Development Environment

J2EE Java 2 Platform, Enterprise Edition

Java EE Java Platform, Enterprise Edition

MVC Model-View-Controller

OTN Oracle Technology Network

PDF Portable Document Format

PL/SQL Procedural Language/Structured Query Language

RAD Rapid Application Development

RAM Random Access Memory

SGBD Sistema Gerenciador de Banco de Dados

SOA Service-Oriented Architecture

SP1 Service Pack 1

UF Unidade da Federação

XE Express Edition

XML Extensible Markup Language

**Sumário**

Introdução 10

1 SISTEMAS LEGADOS E REENGENHARIA DE SOFTWARE 13

1.1 TRADUÇÃO DE CÓDIGO-FONTE 17

1.2 ENGENHARIA REVERSA 18

1.3 MELHORIA DE ESTRUTURA DO PROGRAMA 18

1.4 MODULARIZAÇÃO DE PROGRAMA 19

1.5 REENGENHARIA DE DADOS 20

1.6 ANÁLISE DE INVENTÁRIO 20

1.7 ENGENHARIA AVANTE 21

2 CENÁRIO ATUAL E FUTURO DA EMPRESA 22

2.1 CENÁRIO ATUAL 22

2.1.1 O Sistema 22

2.1.2 SGBD 23

2.1.3 Oracle Forms Developer 24

*2.1.3.1 Características* 24

*2.1.3.2 Limitações* 25

2.1.4 Arquitetura 26

2.1.5 Recursos Humanos 26

2.2 TECNOLOGIAS E FERRAMENTAS SUBSTITUTAS 27

2.2.1 Oracle ADF 27

2.2.2 Oracle Forms Services 11g 28

2.2.3 Oracle Forms Migration Assistant 29

2.2.4 APEX 29

3 TESTES 31

3.1 PLANEJAMENTO DOS TESTES 32

3.2 AMBIENTE DE TESTES 33

3.2.1 Sistema de Testes 34

4 PROCESSO DE CONVERSÃO 43

4.1 ORACLE FORMS SERVICES 11G 43

4.2 ORACLE FORMS MIGRATION ASSISTANT 46

4.3 ORACLE ADF 48

4.4 APEX 49

5 COMPARATIVO DAS FERRAMENTAS DE CONVERSÃO 57

5.1 ROTEIRO DAS ENTREVISTAS 57

5.2 ENTREVISTAS 58

5.3 CUSTOS DAS FERRAMENTAS 62

5.4 ANÁLISE DOS RESULTADOS DAS CONVERSÕES 64

CONCLUSÃO 67

Referências Bibliográficas 69

Introdução

Hoje em dia, as empresas investem muito dinheiro em sistemas de software. Para ter retorno sobre o investimento, esses precisam ser usados por muito tempo. Eles costumam ter a vida útil prolongada e ainda são essenciais para o funcionamento das empresas (SOMMERVILLE, 2003). Normalmente, softwares legados têm tecnologia obsoleta, e em determinado ponto de sua existência limitarão a evolução dos processos de negócio da empresa.

Segundo Sommerville (2003), nesses casos se julga necessária uma avaliação do sistema legado para que uma das seguintes estratégias seja escolhida:

• Descartar completamente o sistema;

• Continuar a manter o sistema;

• Transformar o sistema de alguma maneira para facilitar sua manutenção;

• Substituir o sistema por um novo.

Pensando sobre as estratégias propostas por Sommerville (2003), percebe-se que “Descartar completamente o sistema” e “Substituir o sistema por um novo” devem andar juntas, pois, nos tempos atuais, é improvável que uma empresa informatizada descarte o sistema para voltar aos formulários de papel. Da mesma forma, não faz sentido que seja implantado um sistema novo e o sistema antigo seja mantido paralelamente (quando ambos se destinam para a mesma atividade), considerando-se que o novo sistema está estável e atendendo aos processos de negócio da empresa.

A segunda opção proposta por Sommerville (2003) seria continuar a manter o sistema. Essa é a estratégia mais segura, porém a manutenção se torna cada vez mais difícil com o passar do tempo, devido a inúmeras modificações e, ainda tem-se o obstáculo de contornar as demandas que não podem ser atendidas devido a alguma limitação da tecnologia. Cabe salientar ainda que, para Peters e Pedrycz (2001), essa estratégia tem custo de manutenção muito elevado. Eles estimam que, para cada dólar gasto com manutenção e aprimoramento de um sistema, esses mesmos custos podem aumentar em até 80 centavos de dólar para cada ano subsequente às modificações.

Já a opção de “Transformar o sistema de alguma maneira para facilitar a manutenção” pode ser uma boa estratégia. Quando a tecnologia é substituída por uma mais moderna, elimina, na maioria das vezes, a limitação tecnológica e os riscos pertinentes a uma nova implantação.

Peters e Pedrycz (2001) consideram que, quando a manutenção do sistema torna-se inviável, a atualização (*upgrade*) passa a ser útil. Isso é válido quando as necessidades dos clientes excedem a capacidade do software legado, conforme citado anteriormente. Nesses casos, os projetos de reengenharia são considerados.

Sommerville (2003) aponta a tradução de programa como uma das alternativas de reengenharia de software, ou seja, quando um software traduz os códigos-fonte da linguagem atual para uma linguagem alvo. Essa alternativa transformaria o sistema e facilitaria sua manutenção, mas só é viável economicamente se a empresa adquirir ou desenvolver uma ferramenta que traduza a maior parte do código-fonte.

Em contrapartida, Paula Filho (2003) sugere que uma ferramenta para automatizar a geração de código-fonte deve ser desenvolvida internamente somente quando é absolutamente indispensável, ou seja, quando a ferramenta não existe no mercado ou o custo é abusivo. Descarta ainda as ferramentas de domínio público ou da área acadêmica, por não garantirem suporte e evolução.

Existem algumas ferramentas no mercado que se propõem a fazer essa tarefa, tais como: *CodePorting C#2Java* - esta ferramenta converte projetos C# para Java e promete usabilidade, produtividade, simplicidade, confiabilidade, administração facilitada e poderosas customizações (CODEPORTING, 2012); *Composer CipherSoft* – converte aplicações Oracle *Forms* e PL/SQL para Java (CIPHERSOFT, 2012); *Telescope for Jaguar* – converte Oracle *Forms* para Java, trata-se de uma solução semi-automatizada, pois precisa de análise da empresa desenvolvedora antes da migração (IBROWSE, 2012) . Existem ainda ferramentas que se destinam a facilitar a atualização de uma tecnologia para outra: o *Oracle Forms Migration Assistant* é uma ferramenta que atualiza as aplicações Oracle *Forms* 6i para Oracle *Forms* 11g. Embora esse tipo de ferramenta tenha evoluído, não são raros os casos em que os desenvolvedores devem intervir e corrigir possíveis erros manualmente.

O trabalho em questão foi realizado em uma empresa com tecnologia baseada em Oracle, banco de dados e plataforma de desenvolvimento – Oracle *Forms* 6i e Oracle *Reports* 6i. A última atualização da Oracle para essas ferramentas foi em setembro de 2007, e elas não receberam mais suporte ou novos recursos nessa versão, desde então (ORACLE, 2012). No documento *Oracle Application Development Tools Statement of Direction: Oracle Forms, Oracle Reports and Oracle Designer* (ORACLE PRODUCT MANAGEMENT, 2012), a Oracle afirma que não descontinuará a tecnologia, mas recomenda fortemente que a mesma seja atualizada para a última versão (11g).

Com essa situação, surgiu a necessidade de identificar uma tecnologia web para migração dos sistemas, ou realizar o *upgrade* para *Oracle Forms* e *Reports* 11g, tanto para atualizar a base tecnológica da empresa, quanto para atender as demandas de clientes que a tecnologia atual não permite mais. Foi necessário, portanto, determinar a viabilidade da migração para esta tecnologia através de variáveis como: recursos da nova tecnologia, custos com recursos humanos e investimento em ferramentas.

Neste contexto, este trabalho buscou explorar tecnologias web disponíveis que possam substituir a tecnologia da empresa por outra mais recente, ou que possam facilitar a manutenção do sistema e atender as demandas dos clientes. Buscou também esboçar o custo de migração, considerando-se: custo de ferramentas auxiliares (para tradução de código-fonte) e custo com recursos humanos (programadores/analistas).

O trabalho ainda teve como propósito testar algumas ferramentas disponíveis no mercado para tradução de código-fonte, e avaliar a quantidade de ajustes/erros que devem ser corrigidos no código gerado pela ferramenta. Dessa forma, a empresa teria um ponto de partida no planejamento do projeto de migração.

Esse trabalho está dividido em cinco capítulos. O primeiro descreve a dificuldade de manutenção dos sistemas legados, sobretudo quando esses têm como base uma tecnologia obsoleta, tornando a manutenção muito onerosa. Discute também atividades que podem contornar essa situação. No segundo capítulo é abordado o cenário atual da empresa, foco do trabalho, discute as tecnologias utilizadas, bem como arquitetura de software, recursos humanos e faz uma breve apresentação das tecnologias que poderiam ser utilizadas para migração. O terceiro capítulo apresenta a metodologia de testes, detalha o ambiente de testes e o sistema desenvolvido para este fim, bem como os critérios que foram considerados na avaliação das ferramentas de conversão. No quarto capítulo foi abordado o processo de conversão do sistema de testes utilizando as ferramentas propostas, as particularidades de cada uma, bem como os resultados do sistema convertido e a facilidade com que cada ferramenta é utilizada. No quinto e último capítulo foram feitos os testes com os usuários e posteriormente, entrevistas para avaliar as diferenças entre o sistema de origem e os sistemas convertidos.

1. SISTEMAS LEGADOS E REENGENHARIA DE SOFTWARE

Segundo Sommerville (2008), devido ao tempo e esforço necessários para desenvolver um sistema complexo, os sistemas baseados em computadores têm vida longa. Algumas vezes é muito oneroso e arriscado descartar sistemas críticos com poucos anos de uso. Para Sommerville (2008, p. 25):

Os sistemas legados são sistemas sociotécnicos baseados em computadores e que foram desenvolvidos no passado, frequentemente utilizando tecnologias mais antigas ou obsoletas. Esses sistemas não apenas incluem hardware e software, mas também processos e procedimentos legados - velhas formas de fazer coisas que dificilmente são mudadas porque estão baseadas em software legado.

Sistemas técnicos baseados em computadores são sistemas que incluem hardware e software, mas não incluem procedimentos e processos. Exemplos desse tipo de sistemas são televisores e processadores de texto. Eles têm um propósito, mas o conhecimento do propósito não faz parte do sistema. Já os sistemas sociotécnicos, além de componentes de hardware e software, incluem também procedimentos e processos, ou seja, incluem conhecimento de como o sistema deve ser usado para alcançar um objetivo maior. Sistemas desse tipo gerem regras de negócio, políticas organizacionais e incluem pessoas como parte integrante do sistema (SOMMERVILLE, 2003).

Após um sistema ser considerado operacional – quando o sistema está sendo usado pelos usuários em ambiente de produção -, qualquer trabalho de modificação é considerado como manutenção. Não podemos comparar a manutenção de hardware com a manutenção de software. A manutenção de hardware consiste em troca de peças desgastadas para prolongar a vida útil do produto. Já a manutenção de software não sofre com desgaste, “os pontos-e-vírgulas não caem do final das linhas”, portanto, consiste em modificar o sistema para atender alguma nova demanda do cliente, corrigir defeitos ou tratar situações não previstas na especificação de requisitos do software (PFLEEGER, 2004).

Pfleeger (2004) afirma também que os softwares desenvolvidos são evolutivos, exceto nos casos mais simples. O software não é modificado simplesmente porque o cliente deseja fazer algo diferente, mas, em alguns casos, a própria natureza do sistema se modifica. Por exemplo, softwares para emissão de notas fiscais sofrem mudanças constantes devido a mudanças na legislação.

É interessante observar que a manutenção não é uma atividade que ocorre somente uma vez. Um produto bem feito pode passar por inúmeras versões durante seu ciclo de vida. Portanto, é imprescindível que a manutenção seja planejada desde o início do projeto e que as manutenções posteriores mantenham o padrão estabelecido (SCHACH, 2009).

Para Schach (2009), é mais fácil criar uma nova versão de software do que fazer manutenção, porém a diferença dos custos não torna a primeira opção tão interessante. Dessa forma, se vê a importância de desenvolver o sistema pensando em facilitar sua manutenção pós-entrega.

Para Sommerville (2003), manter o sistema é uma estratégia segura, porém a manutenção se torna cada vez mais difícil com o passar do tempo, devido às inúmeras modificações que o sistema pode ter sofrido. Ainda corre-se o risco de a tecnologia em que o software legado foi desenvolvido, perder o suporte com o fornecedor, ou passar a limitar a evolução dos processos de negócio da empresa devido a alguma limitação tecnológica. Cabe salientar ainda que, para Peters e Pedrycz (2001), manter o sistema tem custo de manutenção muito elevado. Eles estimam que, para cada dólar gasto com manutenção e aprimoramento de um sistema, esses mesmos custos podem aumentar em até 80 centavos de dólar para cada ano subsequente às modificações.

Dessa forma, buscam-se estratégias para contornar as situações apresentadas. Pode-se adotar a estratégia de atualização (*upgrade*) de sistema, quando as necessidades dos clientes excedem a capacidade do sistema legado. Nesse caso, os projetos de reengenharia são considerados (PETERS; PEDRYCZ, 2001).

Geralmente, sistemas legados têm a documentação precária ou inexistente, tornando-se difícil de o software ser entendido por pessoas que não participaram do desenvolvimento. Peters e Pedrycz (2001), dizem que há três escolhas básicas a serem feitas na abordagem a um sistema legado: atualização, redesenvolvimento e reengenharia.

Apenas para definir engenharia de software, Pfleeger (2004) diz que ela é a utilização de conhecimento sobre computadores e computação para ajudar a resolver problemas, mas antes disso é essencial entender a natureza do problema. O problema, se complexo, deve ser dividido em vários problemas menores que tenham relação entre si. Dessa forma, é possível elaborar a solução a partir de componentes que resolvam vários de seus aspectos.

Para explicar reengenharia de software, Pressman (2006) faz a seguinte analogia: considere qualquer produto de tecnologia que tenha servido bem a você. Você o utiliza regularmente, mas ele está ficando velho. Quebra com frequência, leva mais tempo do que gostaria para reparar e deixou de representar a tecnologia mais recente. O que fazer? Se o produto é hardware, você provavelmente vai jogá-lo fora e comprará um modelo mais novo. Mas se for software, feito sob encomenda, essa opção pode não ser possível. Você precisará reconstruí-lo. Terá de criar um produto com funcionalidade adicional, melhor desempenho e confiabilidade, e manutenibilidade aperfeiçoada. Isso é o que chamamos de reengenharia. “A reengenharia de software se ocupa de reimplementar sistemas legados para que sua manutenção seja mais fácil. A reengenharia pode envolver redocumentar, organizar e reestruturar o sistema, traduzir o sistema para uma linguagem de programação mais moderna, modificar e atualizar a estrutura e os valores dos dados do sistema. A funcionalidade do software não é modificada e, normalmente, a arquitetura do sistema também permanece a mesma.” (SOMMERVILLE, 2003)

Conforme mostra a Figura 1, a diferença entre a engenharia convencional e a reengenharia de software é o ponto por onde se começa em cada uma. Na engenharia, o processo se inicia com uma especificação escrita, enquanto que na reengenharia o próprio sistema que sofrerá a reengenharia servirá como especificação para o novo sistema. A engenharia cria um novo sistema a partir de um projeto e implementação, utilizando uma especificação para tal. A reengenharia criará o sistema reengenheirado[[1]](#footnote-1) compreendendo e transformando o sistema já existente.

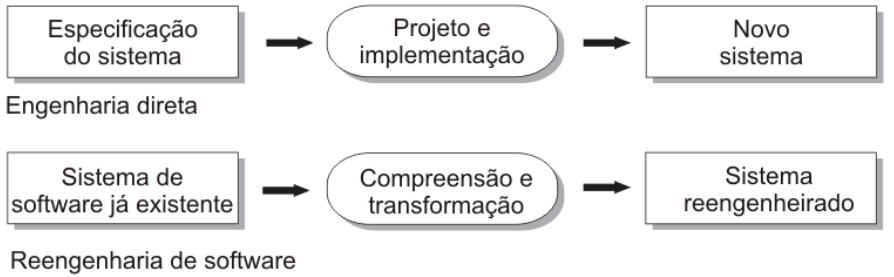


Figura 1 - Engenharia direta e reengenharia

**Fonte: Sommerville (2003)**

Sommerville (2003) divide o processo de reengenharia nas seguintes atividades:

1. **Tradução de código-fonte**: O programa é convertido de uma linguagem de programação antiga para uma versão mais moderna da mesma linguagem ou para uma linguagem diferente. No processo de reengenharia, esse processo ajuda a reduzir custos, mas não é obrigatório.

2. **Engenharia reversa**: o programa é analisado e as informações são extraídas dele, a fim de ajudar a documentar sua organização e funcionalidade.

3. **Melhoria de estrutura do programa**: A estrutura de controle do programa é analisada e modificada, a fim de torná-la mais fácil de ser lida e compreendida.

4. **Modularização de programa**: As partes relacionadas do programa são agrupadas e, quando for apropriado, a redundância é removida. Em alguns casos, esse estágio pode envolver a transformação de arquitetura.

5. **Reengenharia de dados**: Os dados processados pelo programa são modificados, a fim de refletir as mudanças feitas nele.

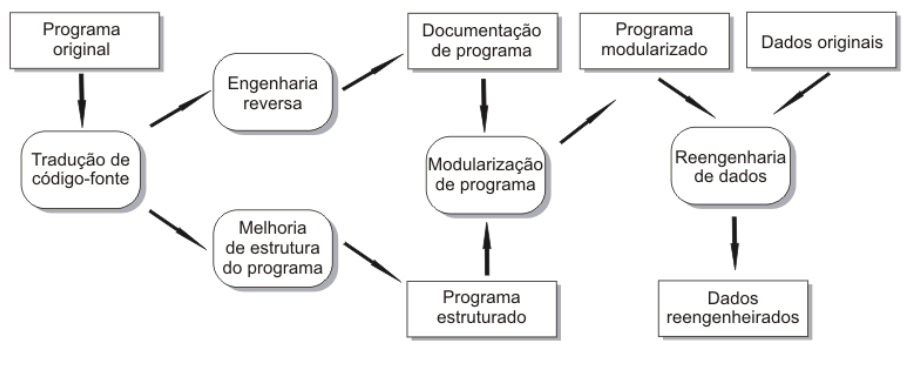


Figura 2 - Processo de reengenharia

**Fonte: Sommerville (2003)**

É interessante salientar que o processo de reengenharia não necessita da execução de todos os processos acima. Por exemplo, caso a tecnologia ainda seja considerada recente, não é necessário realizar a tradução de código-fonte. Não é necessário realizar engenharia reversa para recuperação de documentos, caso a reengenharia dependa inteiramente de ferramentas automatizadas.

Diferentemente de Sommerville (2003), Pressman (2006) divide a reengenharia de software nas seguintes atividades:

1. **Análise de inventário:** Inventário de todas as aplicações da organização.
2. **Reestruturação de documentos:** Processo de documentação das aplicações.
3. **Engenharia reversa:** Sistema pronto passa pela reengenharia reversa para obter-se um maior entendimento sobre o sistema.
4. **Reestruturação de código:** O código é reestruturado para ser mais fácil entendê-lo e mantê-lo.
5. **Reestruturação de dados:** Fortalecer a estrutura de dados para facilitar a adaptação e aperfeiçoamento do programa.
6. **Engenharia avante:** Recupera informações do software existente para aperfeiçoar sua qualidade global.

Nos subcapítulos seguintes, será feito um pequeno mapeamento sobre os processos de reengenharia descritos por Sommerville (2003) e Pressman (2006), explicando sucintamente as atividades de cada um e comparando ambos, caso existam semelhanças.

* 1. Tradução de Código-Fonte

O programa é convertido de uma linguagem de programação antiga para uma versão mais moderna da mesma linguagem, ou para uma linguagem diferente. Sommerville (2003) apresenta alguns motivos para que a tradução aconteça, levando em consideração de que é necessária uma ferramenta automatizada para viabilizar a tradução:

* **Atualização da plataforma de hardware**: A organização pode mudar seu hardware e é possível que o compilador não suporte o novo hardware.
* **Escassez de pessoal habilitado**: em se tratando de uma linguagem antiga, é cada vez mais difícil conseguir pessoal habilitado e esses se tornam muito disputados pelas empresas.
* **Mudanças na política organizacional**: a empresa pode padronizar o uso de uma linguagem específica para reduzir custos com apoio ao software.
* **Falta de suporte ao software:** os fornecedores podem ter encerrado suas atividades ou encerrado o suporte ao produto.
  1. Engenharia Reversa

Sommerville (2003) e Pressman (2006) são unânimes na definição de engenharia reversa. Trata-se do processo de analisar um software e recuperar seu projeto e especificação. Sommerville (2003) ressalta que engenharia reversa não é o mesmo que reengenharia, pois na engenharia reversa tem-se por objetivo a extração de informações a partir do código-fonte; a reengenharia tem por objetivo produzir um novo sistema para facilitar a manutenção. A engenharia reversa é um dos processos da reengenharia. Contudo, ela pode ser usada sem ser seguida da reengenharia, apenas para recuperação de documentos do projeto.

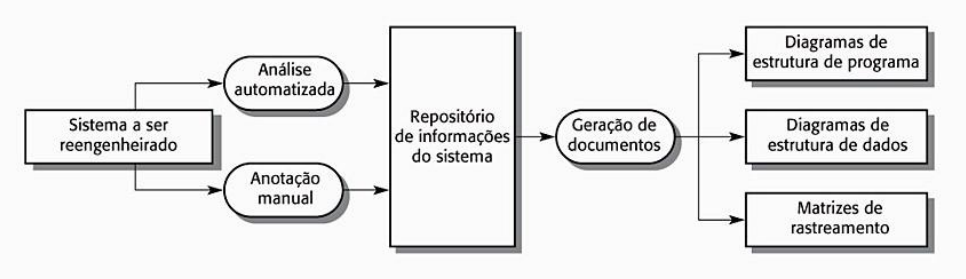


Figura 3 - Processo de engenharia reversa

**Fonte: Sommerville (2003)**

Embora a engenharia reversa possa ser feita manualmente, é sempre indicado que seja usada alguma ferramenta que auxilie a análise para reduzir custos. Pressman (2006) diz que o objetivo é conseguir um grau de abstração mais alto que o código-fonte.

Pressman (2006) descreve um processo de Reestruturação de documentos na reengenharia de software, que nada mais é do que criar documentos, se não existem, ou atualizar os documentos existentes. Sommerville (2003) trata da documentação no processo de Engenharia Reversa.

* 1. Melhoria de Estrutura do Programa

Para Sommerville (2003), muitos sistemas legados não são bem estruturados, pela falta de conhecimento de engenharia de software por parte dos programadores. Dessa forma, as estruturas de controle acabam se tornando complexas e a lógica de controle não intuitiva. Essa situação pode ter sido causada também pelas manutenções regulares.

Como possíveis problemas, Sommerville (2003) ressalta a utilização de dialetos que não sejam padrão de linguagem, pois estes podem não ser reestruturados corretamente por ferramentas automatizadas que utilizam código padrão. Outros problemas seriam: a possível perda de comentários no código, perda ou desatualização da documentação e o tempo necessário, pois os algoritmos de reestruturação são complexos e exigem grande poder computacional.

Esta etapa pode ser comparada com a Reestruturação de código proposta por Pressman (2006) que tem por objetivo modificar o código-fonte em um esforço de torná-lo mais ameno em modificações futuras, ou seja, mantém o programa original, mas melhora a qualidade do código. A reestruturação de código de Pressman (2006) tende a se concentrar nos detalhes de projeto de módulos. Se o esforço de reestruturação se estender e atingir a arquitetura de software, a reestruturação se transforma em engenharia avante (será tratada logo em seguida).

* 1. Modularização de Programa

Sommerville (2003) diz que a modularização de programa é o processo de reorganizar um programa, a fim de analisar as partes relacionadas e agrupá-las em um único módulo. Segundo ele, esse processo facilita a remoção de redundâncias nos componentes agrupados, otimiza suas interações e simplifica suas interfaces com o restante do programa.

Pressman (2006) considera essa atividade como parte do processo de reestruturação de código, pois ele prevê um mapeamento dos módulos para descobrir qual seu nível de acoplamento ou intercâmbio entre si para fazer a reestruturação.

* 1. Reengenharia de Dados

Assim como é necessário fazer a reengenharia de software, em alguns casos, pode ser necessária também a reengenharia de dados. Sommerville (2003) aponta os seguintes motivos que levam à reengenharia de dados:

* **Degradação dos dados:** Diminuição da qualidade dos dados com o passar do tempo. Mudanças de ambiente podem não estar sendo refletidas nos dados.
* **Limites inerentes inseridos nos dados:** Restrições referentes à quantidade dos dados processados. Hoje em dia, trabalhamos com uma quantidade gigantesca de dados, dessa forma, essa limitação não deve existir.
* **Evolução da arquitetura:** exemplo clássico de evolução de arquitetura é quando uma empresa decide migrar os dados de vários arquivos separados para um sistema gerenciador de banco de dados (SGBD).

Pressman (2006) trata do mesmo assunto no processo Reestruturação de dados. Ele afirma que antes de qualquer reestruturação de dados deve ser feita uma análise do código-fonte. Dessa forma, todas as declarações em linguagem de programação que contém definições de dados, descrição de arquivos, entradas e saídas e descrições de interfaces são avaliadas. Após a análise de dados, inicia-se o reprojeto de dados.

* 1. Análise de Inventário

Pressman (2006) sugere a documentação de todas as aplicações da empresa, mesmo que de uma maneira simplificada em planilha eletrônica. Essa documentação deve ter informações relevantes como: criticidade para o negócio, longevidade, manutenibilidade. A planilha deve ser revisada periodicamente, a fim de atualizar informações e encontrar sistemas candidatos à reengenharia.

* 1. Engenharia Avante

A engenharia avante realiza técnicas de reengenharia de software e depois de engenharia para desenvolver novas funcionalidades para o software. Pressman (2006) diz que a engenharia avante aplica conceitos e técnicas de engenharia de software para recriar uma aplicação existente. Segundo ele ainda, na maioria das vezes, a engenharia avante não cria simplesmente um equivalente moderno de um programa antigo. Em vez disso, são integrados na reengenharia novos requisitos do usuário e, possivelmente, nova tecnologia. O programa redesenvolvido expande as funcionalidades do programa antigo.

Nesse capítulo foram tratadas algumas possibilidades para modernização de sistemas legados. Apesar de suas limitações e custo elevado de manutenção, muitas empresas ainda são resistentes quando o assunto é modernização ou evolução de software. A reengenharia de software tem várias etapas, porém deve-se verificar quais se aplicam melhor ao sistema que deve ser modificado. No próximo capítulo será apresentado o cenário atual e futuro da empresa, tecnologias utilizadas, arquitetura, recursos humanos e escolha das tecnologias substitutas.

1. CENÁRIO ATUAL E FUTURO DA EMPRESA

Neste capítulo será apresentado o cenário atual e futuro da empresa. No cenário atual são apresentadas as tecnologias utilizadas: software de desenvolvimento, banco de dados; assim como o sistema, arquitetura e recursos humanos. É feita também uma breve descrição das ferramentas que poderiam ser substitutas para o trabalho de desenvolvimento e/ou migração.

* 1. CENÁRIO ATUAL

A empresa foco do trabalho possui, hoje, cerca de 500 usuários que rodam aplicativos cliente-servidor. As aplicações foram desenvolvidas em *Oracle Forms 6i* e são mantidas por equipes internas. A empresa conta com um sistema que possui em torno de 1750 fontes entre *Forms* e *Reports*. Segundo Housinger (2011), para fazer a migração para o *Oracle Forms 11g,* a linguagem de programação permanece a mesma e a diferença no desenvolvimento está, basicamente, na remoção e adição de alguns recursos. Por isso, sem uma análise mais detalhada, esta parece ser a tecnologia mais viável para uma migração.

Independente da ferramenta utilizada para conversão, tanto se a tecnologia escolhida for Java ou *Oracle Forms 11g*, algumas modificações manuais, por vezes, são necessárias. Por exemplo, mudando a arquitetura de cliente-servidor para arquitetura de três camadas, teremos um servidor de aplicação “no meio do caminho”. Desse modo, comandos que interagem com o sistema operacional da máquina cliente, devem ser alterados para que continuem funcionando dessa forma, caso contrário, a interação será feita com o sistema operacional do servidor de aplicação.

* + 1. O Sistema

O sistema em questão atende as necessidades de indústria e varejo. Cada módulo está em um *owner/schema* diferente no banco de dados Oracle e ambos interagem entre si através do uso de *Database Links* e sinônimos públicos. Os módulos são divididos da seguinte maneira:

* Contabilidade e Patrimônio;
* Estoque;
* Exportação / Importação;
* Financeiro;
* PCP;
* Representantes (Força de Vendas);
* Vendas (Interno);
* Fiscal (SPED);
* Retaguarda Varejo.
  + 1. SGBD

O conceito de banco de dados relacional foi descrito pela primeira vez pelo Dr. Edgar F. Codd, por volta de 1970. No início, não estava claro que o modelo relacional alcançaria sucesso comercial. Entretanto, em 1977, foi fundada a precursora da *Oracle Corporation*, a empresa *Software Development Laboratories* (SDL). Em 1979, foi lançado o Oracle V2 como o primeiro banco de dados relacional do mundo. Então a SDL mudou o nome para *Relational Software Incorporated* (RSI) (GREENWALD; STACKOWIAK; STERN, 2007). Atualmente, o modelo de banco de dados mais aceito é o modelo relacional (ASHDOWN; KYTE, 2011).

No cenário atual da empresa é utilizado o Sistema Gerenciador de Banco de Dados Oracle na versão 10g. Segundo Smith (2005), o Oracle 10g é o banco de dados que teve o maior número de funcionalidades adicionadas pela empresa, até então. Essa versão teve mudanças fundamentais na forma de operação em *data center*, além de redução de custos, gerenciamento aprimorado, adição de recurso contra erro humano a fim de aumentar a disponibilidade e melhorias na segurança e desenvolvimento de aplicações.

* + 1. Oracle Forms Developer

O *Forms Developer* é um ambiente de produtividade da Oracle - RAD (*Rapid Application Development*). Com ele pode-se construir aplicações que, rapidamente, podem dar manutenção às bases de dados Oracle. Essas aplicações podem ser tanto em ambiente cliente-servidor quanto em arquitetura de três camadas, possibilitando o acesso através da internet (FERNANDES, 2002).

O *Forms Builder* é a principal ferramenta desse ambiente, pois é a ferramenta de desenvolvimento de aplicações e manutenção de banco de dados. Além do *Oracle Forms Builder*, esse ambiente de desenvolvimento ainda conta com as seguintes ferramentas, segundo Fernandes (2002):

• *Graphics Builder*: essa ferramenta permite que sejam criadas aplicações para o usuário interagir graficamente com os dados do banco de dados;

• *Project Builder*: gerenciar visualmente os vários componentes que compõem uma aplicação;

• *Procedure Builder*: serve para editar, compilar, testar e debugar código PL/SQL;

• *Schema Builder*: criar, copiar, modificar, apagar objetos de banco de dados e seus relacionamentos;

• *Query Builder*: construir *queries* de banco de dados em suas aplicações;

• *Translation Builder*: pode ser usado para traduzir sua aplicação em até 40 idiomas.

Como o *Forms Builder* é a principal ferramenta do pacote, nesse trabalho somente ela será abordada.

* + - 1. Características

O principal recurso do *Oracle Forms* é a produtividade. Para construir uma aplicação de cadastro simples é necessário somente seguir um guia passo a passo, sem a necessidade de programação. A própria ferramenta faz a conexão com o banco de dados, mediante *login* do usuário, e também as tarefas básicas de: inserção, exclusão, atualização, navegação entre registros e pesquisa.

Outros recursos interessantes apontados pela Oracle (2012):

• Implantação de aplicativos na internet;

• Aplicações escaláveis;

• Construir grandes aplicações consistentes e sustentáveis rapidamente, trabalhando com métodos visuais de fácil compreensão.

* + - 1. Limitações

Em se tratando de aplicações corporativas em que o objetivo principal seja a manutenção de dados – CRUD (Inserção, Recuperação, Atualização e Exclusão de dados) – o *Oracle Forms* é uma excelente ferramenta (FERNANDES, 2002). Conforme descrito acima nos recursos apontados pela Oracle, esta tecnologia é de alta produtividade, deixando a complexidade do desenvolvimento para o algoritmo em si e eximindo a interface gráfica dessa complexidade.

Como limitação do *Oracle Forms 6i*, nesse contexto, temos a criação de interfaces gráficas antigas e sem atratividade. Esta tecnologia está presente no mercado há muitos anos e não sofreu nenhuma grande atualização, exceto quando passou de modo caractere para GUI (interface gráfica com o usuário). Entretanto, as telas criadas no *Oracle Forms 6i* continuam com a aparência de sistemas antigos.

Outra limitação é a falta de suporte com o fornecedor. A Oracle não fornece suporte ao *Forms 6i* desde janeiro de 2005, sendo que o suporte poderia ser estendido até janeiro de 2008 (ORACLE, 2012). É importante que o fornecedor da tecnologia utilizada seja uma empresa estável e forneça suporte aos eventuais problemas. Dessa forma, o cliente não ficará desprovido quando necessitar de apoio.

Fazendo uma breve análise das atividades de reengenharia de software descritas por Sommerville (2003) e Pressman (2006) no capítulo 1, a modernização do sistema poderia ser resolvida de duas maneiras: com a tradução do código-fonte para uma linguagem diferente ou para uma versão mais nova da mesma; ou ainda, utilizando o conceito de engenharia avante. Na tradução de código-fonte, poderia ser resolvida a limitação de falta de suporte do fornecedor, uma vez que a tecnologia seria atualizada para sua última versão ou substituída por uma mais recente. Com a engenharia avante seria possível utilizar uma nova tecnologia, dessa forma a aplicação poderia ser modificada de tal maneira que se tornasse mais atrativa para o usuário, graficamente e operacionalmente.

* + 1. Arquitetura

A arquitetura utilizada na empresa atualmente é a cliente-servidor. Nesta arquitetura, o *Forms Server Runtime Engine* – componente responsável pela execução do programa desenvolvido em *Oracle Forms* –, e toda a lógica da aplicação, são geralmente instaladas e executadas na máquina do usuário, que acessa diretamente o servidor de banco de dados (HOUSINGER, 2011).

No ambiente da empresa, os arquivos de código fonte (fmb) e os executáveis (fmx) ficam armazenados em um servidor de arquivos, que tem seu acesso controlado por regras de rede definidas, além de restrições de bancos de dados. O *Forms Server Runtime Engine* é instalado na máquina cliente. Dessa forma, a interface gráfica e a lógica da aplicação ficam no servidor de arquivos (arquivo fmx) e são executadas a partir da máquina cliente. Após executado, o arquivo fmx acessa diretamente o banco de dados através da máquina cliente.

* + 1. Recursos Humanos

Para determinar o cronograma de um projeto e estimar o esforço e os custos associados, deve-se saber quantas pessoas estarão trabalhando nele, quais tarefas cada pessoa realizará, bem como suas habilidades e experiência para realizarem seu trabalho de maneira eficiente (PFLEEGER, 2004). Schach (2009) ainda diz que sem engenheiros de software competentes e bem treinados, qualquer projeto está fadado ao fracasso. Entretanto, somente a presença das pessoas certas não garante o sucesso. Além disso, é necessário que a equipe esteja organizada de maneira que o trabalho seja produtivo e haja cooperação mútua.

A empresa foco do trabalho conta com 63 colaboradores, entre desenvolvedores, analistas e suporte, sendo que cada um deles têm atividades bem definidas. Desses colaboradores, 57 trabalham com *Forms/Reports* e somente 6 com Java*.* Eles são divididos em equipes por módulos, tais como: suporte, comercial, PCP, financeiro, contabilidade, entre outros. As áreas mantém contato entre si para que os objetivos da empresa sejam atendidos e os clientes satisfeitos com o resultado.

* 1. TECNOLOGIAS E FERRAMENTAS SUBSTITUTAS

Considerando-se que o cenário atual da empresa é baseado na tecnologia *Oracle Forms Developer 6i*, obviamente, a tecnologia sucessora desta – *Oracle Forms Services 11g* -, será uma das candidatas para a migração. Outra tecnologia a ser considerada é o Java, que é uma plataforma completa para criação de aplicativos corporativos e web, além de ser gratuita, portável e possuir uma ampla comunidade global.

Segundo Schach (2009), a escolha da tecnologia mais adequada em um projeto é uma atividade complicada. Deve-se escolher a tecnologia “mais adequada”, ou seja, a que tenha o melhor custo-benefício. Em exemplo dado por Schach (2009), uma empresa que trabalha há 30 anos com Cobol e tenha em seu quadro de funcionários 200 pessoas que trabalham com essa tecnologia, é improvável que a tecnologia mais adequada de desenvolvimento seja Java. Para isso, novos programadores teriam que ser contratados, ou o pessoal existente teria que ser treinado na nova tecnologia.

Pfleeger (2004) destaca outro problema na escolha da tecnologia: deve ser escolhida a tecnologia certa para resolver o problema e que, ao mesmo tempo, atraia o cliente. Isto porque pode ser feito um trabalho maravilhoso para resolver um problema e, mesmo assim, a aplicação não alcançar sucesso comercial. Assim como pode ser apressado o lançamento de um produto de modo que ele se torne um sucesso comercial, mas ao longo do tempo fracasse devido à falta de suporte ou aperfeiçoamento.

* + 1. Oracle ADF

O *Oracle Application Developer Framework* (Oracle ADF) é um *framework* de desenvolvimento Java EE, disponibilizado pela Oracle e suportado pelo ambiente de desenvolvimento *Oracle JDeveloper 11g*. O Oracle ADF simplifica o desenvolvimento Java EE e minimiza a necessidade de escrita de código que implementa a estrutura da aplicação. Dessa forma, o esforço pode ser direcionado para os recursos da aplicação e não para a codificação. O Oracle ADF é focado na experiência de desenvolvimento para fornecer uma abordagem visual e declarativa para o desenvolvimento Java EE, através da ferramenta *Oracle JDeveloper 11g*. (O'BRIEN e SHMELTZER, 2011).

O Oracle ADF é baseado no modelo de desenvolvimento MVC (*Model-View-Controller*). Uma aplicação MVC é separada em camadas da seguinte forma: camada de modelo (*model*) que é responsável pela interação com as fontes de dados e em executar as regras de negócio; camada de visualização (*view*) que é responsável pela interface de usuário da aplicação e; a camada controladora (*controller*) que gerencia o fluxo da aplicação e atua como interface entre as camadas de modelo e visualização. Separar as aplicações nessas três camadas facilita a manutenção e o reaproveitamento de componentes da aplicação. A independência de cada camada para com as outras resulta em um baixo acoplamento, assim como a arquitetura orientada a serviços (SOA) (O'BRIEN e SHMELTZER, 2011).

* + 1. Oracle Forms Services 11g

O *Oracle Forms Services 11g*, componente do *Oracle Fusion Middleware*, é uma tecnologia há muito estabelecida para projetar e construir aplicações empresariais de forma rápida e eficiente. A continuidade dessa tecnologia permite que os clientes preservem seus investimentos e, ao mesmo tempo, permite a atualização e integração das aplicações existentes para tirar proveito das tecnologias web e arquiteturas orientadas a serviço (SOA) (THACKER, 2012).

O *Oracle Forms Services 11g* é otimizado para implantar aplicações *Forms* em ambientes multi-camadas. As aplicações são desenvolvidas com o *Oracle Forms Developer*, integrante do *Forms Services,* o código fonte é criado (\*.fmb), depois é compilado para um arquivo executável (\*.fmx). O arquivo fmx é executado (interpretado) pelo *Forms Runtime*, componente do *Forms Developer*. Os aplicativos desenvolvidos no *Forms Developer* são implantados no *Oracle Fusion Middleware* (*Forms Services*). Esses aplicativos são executados na camada intermediária (servidor de aplicação) e a interface de usuário é apresentada na camada cliente como um *applet* Java no navegador do cliente (THACKER, 2012).

* + 1. Oracle Forms Migration Assistant

O *Oracle Forms Migration Assistant*, componente do *Oracle Fusion Middleware*, atualiza os recursos obsoletos no código PL/SQL da aplicação desenvolvida em *Forms 6i* para a versão equivalente no *Forms 11g*. Dessa forma, as aplicações desenvolvidas em *Forms 6i* podem ser migradas para a versão 11g. A ferramenta possui uma versão assistente (GUI) e uma versão em linha de comando (HOUSINGER, 2011).

Essa ferramenta permite que seja convertido um arquivo por vez ou que seja feita a conversão de vários arquivos ao mesmo tempo, ou seja, conversão em lote. Editando as propriedades da ferramenta, é possível definir *strings* para busca e substituição e ainda editar as mensagens de erro, caso haja algum problema na conversão. Os *logs* indicam onde se encontram os problemas da aplicação para que seja possível a alteração de forma manual (HOUSINGER, 2011).

* + 1. APEX

O *Oracle Application Express* (APEX) é uma ferramenta de desenvolvimento web rápida e declarativa para bancos de dados Oracle. Está disponível sem custo algum e para todas as edições de banco de dados Oracle. Usando somente um *web browser* é possível desenvolver e distribuir aplicações profissionais que sejam rápidas e seguras (ORACLE, 2012).

O APEX era conhecido anteriormente como HTML-DB, que surgiu a partir do WebDB. Esses produtos apresentam a mesma metodologia de desenvolvimento, assistentes de ajuda (*wizards*) executados em um *web browser* que ajudam a criar componentes de aplicação que também rodam em um *browser*. O APEX cria componentes em pacotes PL/SQL que produzem a interface do usuário baseada em navegador. Componentes APEX podem incluir formulários, relatórios e gráficos. Alguns recursos interessantes do APEX: possibilidade de criar aplicativos ricos, estender as aplicações APEX com lógica JavaScript, exportar qualquer relatório para PDF, mudar a aparência das aplicações, entre outros(GREENWALD; STACKOWIAK; STERN, 2007).

Neste capítulo foi apresentado o cenário atual da empresa e as tecnologias que podem ser utilizadas para migração. Embora o *Oracle Forms 6i* seja uma ferramenta muito produtiva, não é mais suportado pela Oracle, além de ter uma interface antiga e que não agrada aos usuários. Por esses motivos, seria importante que o *Oracle Forms* fosse atualizado para a versão mais recente ou substituído por outra tecnologia.

No próximo capítulo será apresentado o ambiente de testes e o sistema desenvolvido para esse fim. Os testes servirão para comparar o sistema de origem com os sistemas convertidos. Além disso, será apontada a metodologia utilizada para os testes, assim como seu planejamento.

1. TESTES

Este capítulo tem por objetivo apresentar a metodologia de testes utilizada, bem como os critérios que serão considerados na avaliação das ferramentas apresentadas na seção 2.2 e ainda, detalhar o ambiente de testes e o sistema desenvolvido para este fim.

Para Myers (1976 apud PETERS e PEDRYCZ, 2001), “teste é o processo de execução de um programa com a intenção de encontrar erros”. O teste de software determina quando um sistema de software pode ser liberado comercialmente. Com a crescente complexidade dos sistemas, não é surpreendente que boa parte do orçamento do desenvolvimento seja gasto diretamente nos seus testes.

Schach (2009) considera a atividade de teste um componente essencial do processo de software e que deve ser praticada ao longo de todo o seu ciclo de vida. Não adianta testar um software somente quando ele for finalizado, porque podem ser encontrados problemas que afetam toda a sua estrutura e ele tenha que ser reprojetado, sendo que isso poderia ter sido resolvido fazendo testes em cada fluxo de trabalho.

Schach (2009) divide os testes em duas categorias: testes que não se baseiam em execução e testes baseados em execução. Os testes que não se baseiam em execução têm como métodos: a revisão de código, percorrendo-o e lendo-o cuidadosamente; e a análise matemática, técnica para demonstrar que um produto atende suas especificações. Os testes baseados em execução constituem-se de um processo de análise do comportamento do software, tomando-se como base, os resultados obtidos na sua execução em um ambiente conhecido e com entradas selecionadas.

Peters e Pedrycz (2001) definem outras estratégias de testes: teste estático versus teste dinâmico e teste de caixa branca versus teste de caixa preta. O teste dinâmico requer que o software seja executado com dados de teste. O seu objetivo principal é testar o comportamento do software, detectando seus erros. O teste estático não necessita que o software seja executado. Essa abordagem se baseia na demonstração do programa, análise de irregularidades, inspeções e revisões de código. Os compiladores modernos podem ajudar nos testes estáticos, pois eles analisam o código sintaticamente e semanticamente. Os testes de caixa preta e caixa branca são apenas exemplos das classes fundamentais de abordagens de teste de software, são comparáveis aos testes dinâmicos e estáticos, respectivamente.

O teste de caixa preta enfatiza entradas, saídas e princípios funcionais do módulo ou software, buscando encontrar “comportamentos não esperados” na execução. Nesse caso, o conteúdo da caixa é oculto. O teste de caixa branca enfatiza o projeto detalhado do software, ao invés de funções (caixa preta). Nessa abordagem, as instruções, caminhos e ramificações do software são analisados para fins de execução correta.

Embora Peters e Pedrycz (2001), além de Schach (2009), usem métodos com nomes diferentes, as atividades realizadas são as mesmas. Dessa forma, os testes neste trabalho serão feitos com a abordagem de teste baseado em execução de Schach (2009) ou teste dinâmico de Peters e Pedrycz (2001). As opções de teste estático ou teste que não se baseia em execução não serão utilizadas, pois o compilador do *Oracle Forms* faz boa parte do trabalho: analisa sintaxe e semântica.

* 1. PLANEJAMENTO DOS TESTES

Para testar as ferramentas apresentadas no capítulo 2, foi desenvolvido um sistema de cadastro em *Oracle Forms 6i,* que será abordado na seção 3.2.1. O código fonte do sistema (arquivos com extensão .fmb) será usado como entrada nessas ferramentas para obter como resultado, o código fonte na tecnologia alvo.

Os critérios usados nos testes serão: o método de conversão (linha de comando ou GUI, processamento em lote ou individual), tempo de conversão, curva de aprendizado e a qualidade do resultado (analisar se serão necessários ajustes manuais no código gerado, baseando-se no comportamento do software).

Os critérios utilizados estão todos relacionados e foram escolhidos para que seja possível determinar qual é o processo menos oneroso. O método de conversão está ligado diretamente com o tempo total de conversão e a curva de aprendizado. Sendo escolhido o método em linha de comando, o usuário levará um tempo maior para se adaptar à ferramenta, a curva de aprendizado é maior, logo o tempo de conversão total será maior.

Da mesma forma ocorre com o processamento em lote ou unitário, ele não afeta diretamente a curva de aprendizado, mas o tempo de conversão será maior se forem convertidos arquivos individualmente. A qualidade dos resultados não depende dos demais critérios, mas sim das ferramentas de conversão e afeta diretamente os custos do projeto. Se houver muito código a ser redesenvolvido, o projeto acaba se tornando inviável.

* 1. AMBIENTE DE TESTES

Para realização dos testes, foram utilizados dois computadores. O principal motivo para isso é que o ambiente seja o mais próximo possível de um ambiente de três camadas. Um dos computadores servirá como servidor de banco de dados e de aplicação e o outro como máquina cliente e ambiente de desenvolvimento *Forms 6i*.

O primeiro computador possui a seguinte configuração: Windows 7 Ultimate x64 SP1*,* 320 GB de disco rígido, 4GB RAM e processador Intel Core 2 Duo T6670. Este computador servirá como máquina cliente para acesso ao sistema, além de ambiente de desenvolvimento *Forms 6i*. Para isso, se fez necessária a instalação do *Oracle Forms 6i*, bem como o *Java Runtime Edition*, necessário para execução do *applet* do sistema em *Forms 11g.*

O segundo computador tem por configuração: Windows 7 Ultimate x86 SP1, 500 GB de disco rígido, 4GB RAM e processador Intel Core i3-2310M. Neste computador foram instalados o ambiente de desenvolvimento *Oracle Forms 11g*, *JDeveloper* com *ADF 11g,* o banco de dados *Oracle Express Edition (XE) 11g* e o *Oracle Forms Migration Assistant*, que faz parte do pacote *Oracle Forms Services 11g.* O *Application Express* (APEX) será utilizado online pelo site apex.oracle.com.

Na Figura 4, podem-se observar os componentes presentes no ambiente de testes, conforme parágrafos anteriores, e sua interação. O funcionamento do APEX não se encontra na imagem, pois é utilizado o ambiente de testes disponibilizado pela Oracle (apex.oracle.com), ambiente este atualizado e homologado.

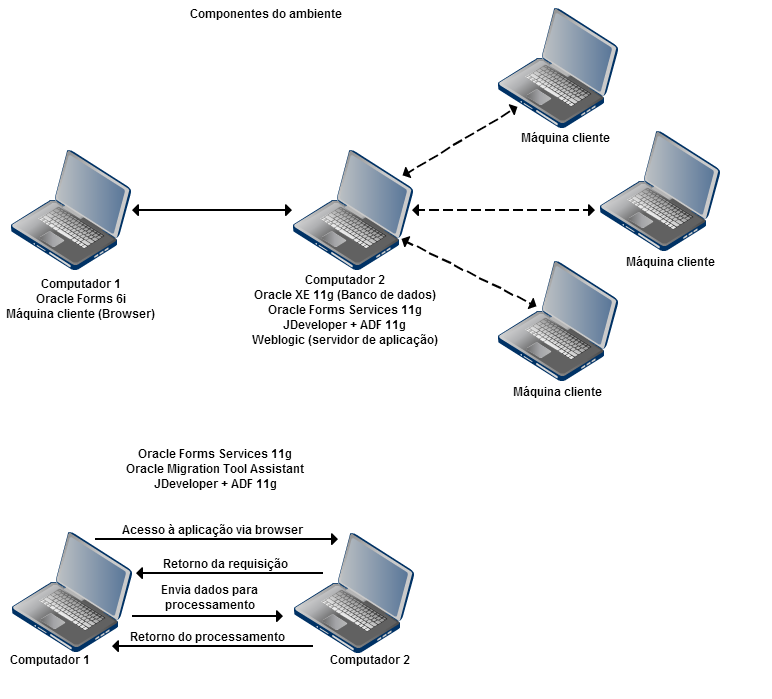


Figura 4 - Componentes do ambiente e sua interação

**Fonte: o autor**

* + 1. Sistema de Testes

Um pequeno sistema foi criado e servirá de entrada para os testes. Como o objetivo principal é a análise dos resultados obtidos nas conversões, o sistema é simples. As entidades selecionadas no Diagrama Entidade Relacionamento (DER), ilustradas na Figura 5, foram utilizadas porque são comuns em qualquer sistema comercial.

O DER da Figura 5, foi modelado com a ferramenta BrModelo na versão 2 e será utilizado como base na elaboração dos formulários (*forms*) de teste. O diagrama conta com cinco entidades: tipo\_pessoa, pessoa, uf, venda e item. Posteriormente, o relacionamento venda\_item será transformado em tabela do banco de dados, pois possui cardinalidade n..n entre as entidades venda e item.

Explicando o diagrama sucintamente:

* Tipo\_pessoa: informa se a pessoa cadastrada é um cliente, fornecedor, transportadora, revendedor, ou outro;
* Pessoa: armazena o cadastro de pessoas;
* UF: registra o cadastro de unidades federativas;
* Venda: cadastra as vendas com informações principais e valores totalizados. A tela que interage com essa entidade é parecida com um cadastro de pedidos ou notas fiscais;
* Item: guarda o cadastro de itens de estoque e vendas;
* Venda\_item: cadastra os itens de uma venda. A tela é parecida com o cadastro de itens de um pedido ou nota fiscal.

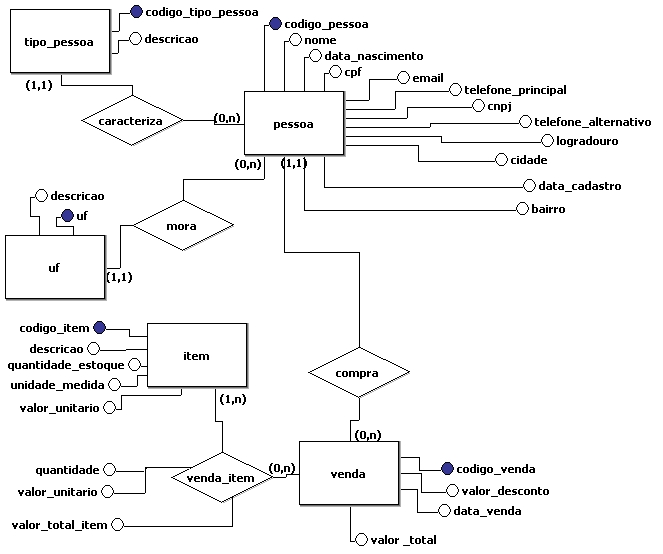


Figura 5 - Diagrama Entidade-Relacionamento

**Fonte: o autor**

Após a definição do DER, iniciou-se o desenvolvimento do sistema em *Oracle Forms 6i.* As telas foram desenvolvidas seguindo-se o DER e os recursos adicionais de cada tela são descritos nos parágrafos seguintes.

As telas de cadastro de tipo\_pessoa e uf, respectivamente Figura 6 e Figura 7, são as mais simples. Elas têm somente itens de texto e só é possível a inserção, atualização e exclusão de dados, além de pesquisa.

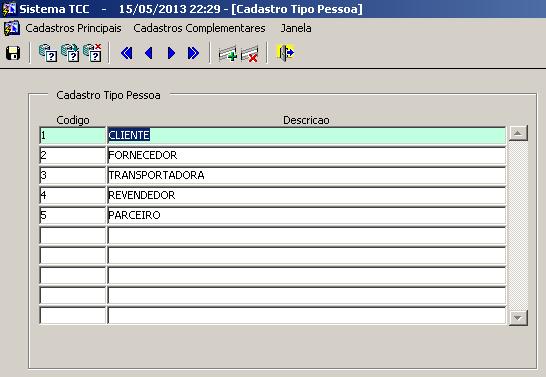


Figura 6 - Cadastro de Tipo\_Pessoa

**Fonte: o autor**



Figura 7 - Cadastro de UF

**Fonte: o autor**

A tela de cadastro de item, Figura 8, tem uma *trigger* de aplicação (*trigger* no lado cliente) que calcula o valor total do produto em estoque (que não é salvo no banco de dados) e um item de lista para o campo unidade\_medida, onde são mostradas todas as opções disponíveis.

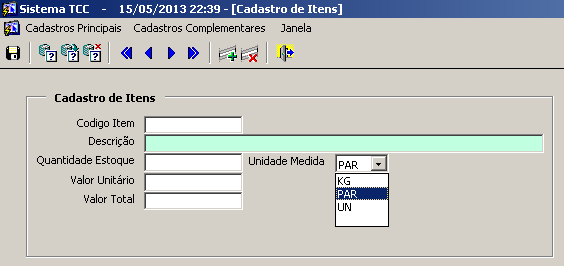


Figura 8 - Cadastro de Item

**Fonte: o autor**

As informações de venda e venda\_item são encontradas na mesma tela, Figura 9. Existem *triggers* de aplicação para os cálculos total dos itens, total das mercadorias, total da venda (total das mercadorias subtraindo o valor de desconto), validação do campo quantidade, preenchimento automático dos campos descrição e valor unitário quando é informado o código do item e preenchimento automático da data de venda. Quando qualquer alteração é feita nos itens, todos os cálculos são atualizados.

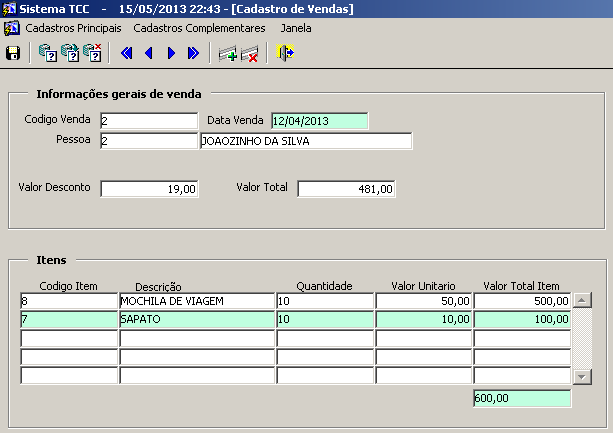


Figura 9 - Cadastro de Venda e Venda\_Item

**Fonte: o autor**

A tela de cadastro de pessoa (Figura 10) é a mais importante nos testes, pois foram utilizados componentes que foram substituídos ou considerados obsoletos pela Oracle na versão 11g. São eles:

* OLE2: *package* que permite a integração de uma aplicação *Oracle Forms* com o sistema Operacional *Windows,* além de algumas ferramentas da *Microsoft* como o *Microsoft Office* (ORACLE, 2013). No sistema de testes, a tela de cadastro de pessoas utiliza esse recurso para gerar um arquivo em *Excel* com os dados cadastrados,
* TEXT\_IO: *package* que provê uma maneira de ler e escrever dados em um sistema de arquivos (BETHKE, 2000). Ela é utilizada no cadastro de pessoas para gerar um arquivo com o formato texto (.txt) a partir dos dados cadastrados;
* GET\_FILE\_NAME: recurso utilizado para selecionar o caminho onde um arquivo será lido ou salvo. É utilizado para selecionar o caminho onde o arquivo no formato txt será gerado;
* WHEN-MOUSE-LEAVE: se trata de uma *trigger* de aplicação (evento) que dispara quando o ponteiro do mouse deixa o item que contém a *trigger*. No sistema de testes, ela é usada somente para disparar a mensagem “Perdeu a chance de gerar o arquivo”, quando o ponteiro do mouse deixa o botão “Gerar txt” (item que contém a *trigger*)*.*

Segundo o documento de migração da Oracle (HOUSINGER, 2011), os componentes OLE2, TEXT\_IO e GET\_FILE\_NAME continuam funcionando no *Forms 11g*, porém eles interagem com o sistema operacional do servidor de aplicação. Para interagirem com o sistema operacional da máquina cliente, deve ser anexada a biblioteca *WebUtil* no *form* e os componentes devem ser substituídos por CLIENT\_OLE2, CLIENT\_TEXT\_IO e CLIENT\_GET\_FILE\_NAME, respectivamente. Na conversão, houve essa substituição para que o comportamento desses componentes permanecesse o mesmo: interação com o sistema operacional da máquina cliente. A *trigger* WHEN-MOUSE-LEAVE foi descontinuada no *Forms 11g*, pois gera um tráfego de rede muito grande (HOUSINGER, 2011).

Dessa forma, poderá ser visto como as ferramentas de conversão se comportarão frente aos componentes que se comportam de maneira diferente na versão 11g e também um componente que foi descontinuado. Os componentes OLE2, TEXT\_IO e CLIENT\_GET\_FILE\_NAME foram selecionados porque são frequentemente utilizados, principalmente, em sistemas financeiros, onde são gerados arquivos bancários. O WHEN-MOUSE-LEAVE está presente no sistema de testes porque foi descontinuado por uma causa nobre - o desempenho -, seu funcionamento gera um tráfego de rede muito grande, inadmissível para um ambiente de três camadas.

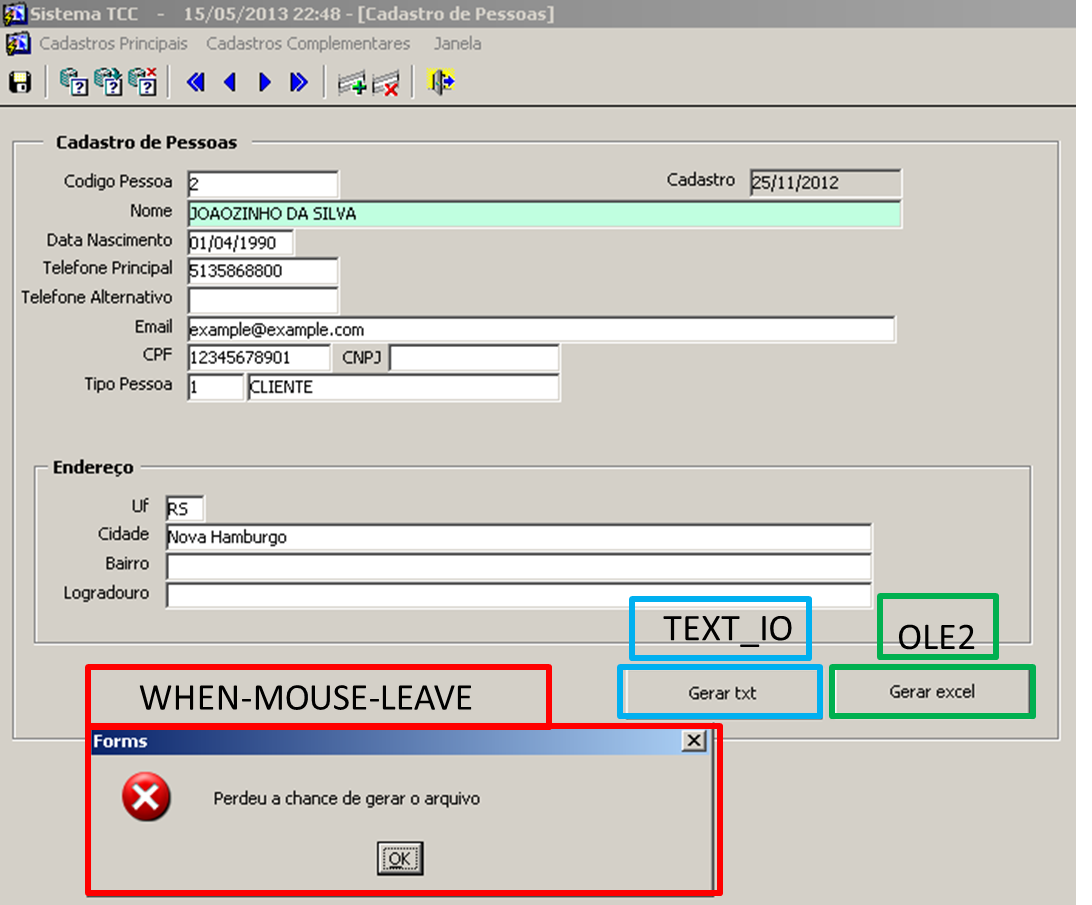


Figura 10 - Cadastro de Pessoa

**Fonte: o autor**

Com o objetivo de deixar o sistema mais organizado, foi criado um menu e tela inicial. O menu conta com a chamada de cada tela, exceto a tela principal, e ainda uma barra *toolbar* com funcionalidades comuns como: salvar, inserir, apagar, pesquisar e outras (Figura 11). A tela principal somente mostra uma imagem (Figura 12).

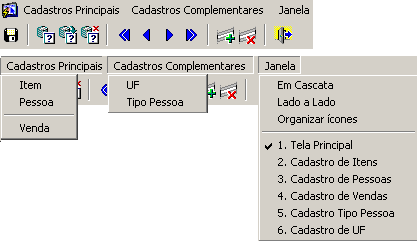


Figura 11 - Menu

**Fonte: o autor**

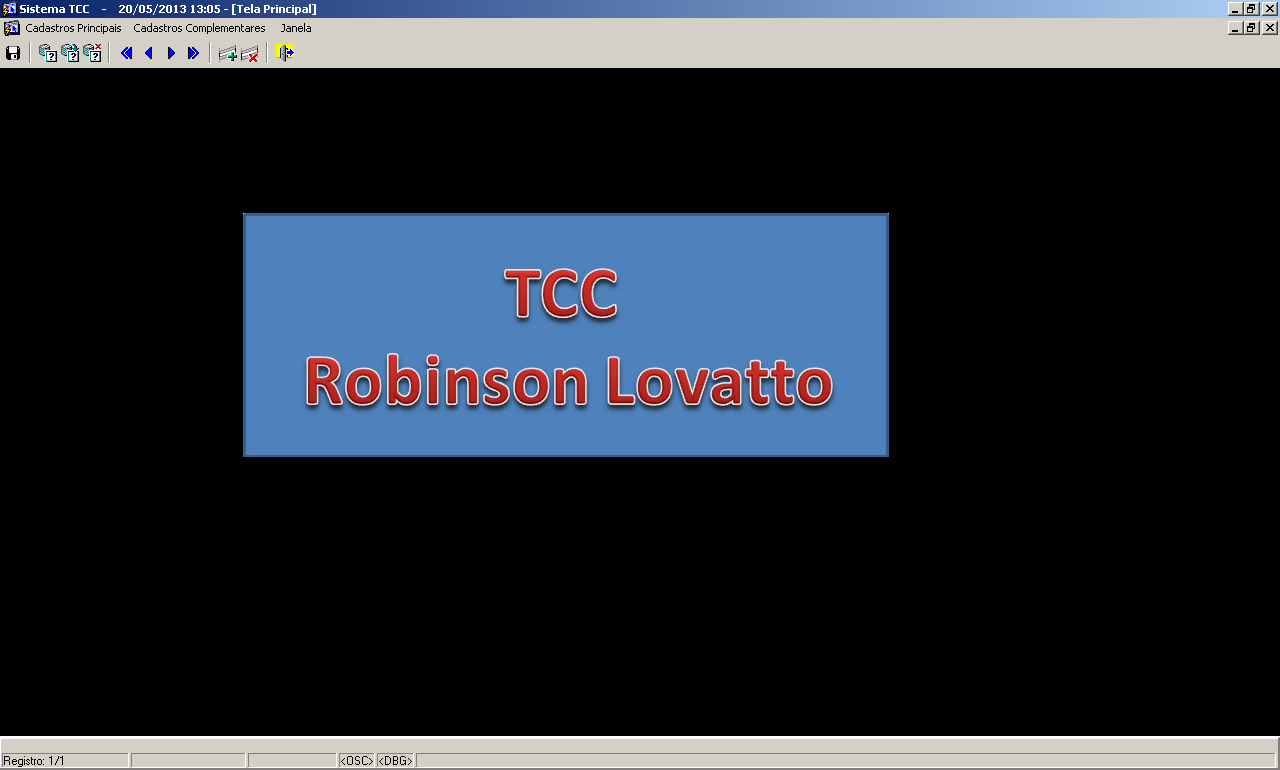
****

Figura 12 - Tela Principal

**Fonte: o autor**

Para explorar um pouco mais as funcionalidades do *Oracle Forms 6i*, foram desenvolvidas uma biblioteca PL/SQL e uma biblioteca de objetos – que nada mais é do que um repositório de objetos do *Oracle Forms* como: *triggers*, *canvas* e outros. A biblioteca PL/SQL foi anexada em todas as telas e os objetos da biblioteca de objetos, copiados. A biblioteca PL/SQL serve para controlar os objetos criados na biblioteca de objetos. Em conjunto, esses componentes formam um *Help* com algumas informações sobre a tela e o item selecionado. Na Figura 13 essa funcionalidade pode ser conferida, como pode ser visto no campo valor\_desconto que foi selecionado. Além do *Help,* ainda foi adicionada a data e hora de acesso à barra de título da tela.

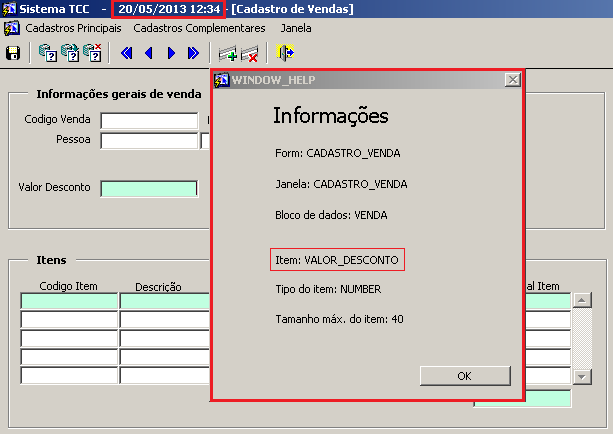


Figura 13 - Help (biliotecas PL/SQL e de objetos)

**Fonte: o autor**

Neste capítulo foram apresentadas as abordagens de alguns autores sobre testes, além do planejamento dos testes e o ambiente de testes, desenvolvido exclusivamente para avaliar os programas de conversão. Os testes já são parte fundamental do processo de desenvolvimento de software e recebem investimentos cada vez maiores. Embora os autores utilizem nomes distintos para as técnicas de teste, analisando as atividades, pode-se concluir que se tratam de técnicas iguais ou, ao menos, parecidas. O sistema desenvolvido para os testes, apesar de simples, possibilitará o cumprimento dos objetivos estabelecidos. No próximo capítulo será abordado o processo de conversão, utilizando as ferramentas propostas e analisando os resultados das conversões.

1. PROCESSO DE CONVERSÃO

Neste capítulo será abordado o processo de conversão do sistema de testes utilizando as ferramentas propostas (*Oracle Forms Services 11g*, *Oracle Forms Migration Assistant*, *Oracle ADF* e APEX), as particularidades de cada uma, bem como os resultados do sistema convertido e a facilidade com que cada ferramenta é utilizada.

* 1. Oracle Forms Services 11g

A principal diferença entre o *Forms 6i* e o *Forms 11g* é a arquitetura. Enquanto o primeiro funciona, principalmente, em arquitetura cliente-servidor, *o Forms 11g* funciona somente em arquitetura de três camadas.

O processo de conversão é muito simples. Basta somente abrir o código fonte do *Forms 6i*, arquivo com extensão fmb, no *Forms 11g,* salvá-lo e depois compilá-lo. Para isso, podem ser abertos vários arquivos no programa, porém isso é feito individualmente. Na compilação, todos os arquivos abertos podem ser selecionados com auxílio da tecla CTRL do teclado, porém eles são compilados individualmente em sequência, conforme Figura 14. Depois de compilar a primeira tela, pressiona-se o botão OK e a próxima é compilada, e assim, sucessivamente, até que o processo seja finalizado. Para converter os menus e bibliotecas o processo é o mesmo.

De todas as telas e bibliotecas desenvolvidas e posteriormente convertidas, somente o cadastro de pessoas e a tela principal tiveram algumas particularidades a considerar. A compilação não indicou nenhum erro ou componente obsoleto. Porém, no cadastro de pessoas, a *trigger* WHEN-MOUSE-LEAVE não funcionou, simplesmente foi ignorada. Os recursos OLE2 e TEXT\_IO continuaram funcionando, entretanto, os arquivos gerados eram salvos somente no servidor de aplicação. Para contornar essa situação, foi anexada a biblioteca WebUtil no programa e os itens foram substituídos por CLIENT\_OLE2 e CLIENT\_TEXT\_IO, respectivamente.

O recurso GET\_FILE\_NAME, usado para selecionar um arquivo, tanto no salvamento quanto na abertura, não funcionou corretamente. Sempre que acionado, ocorria um travamento na aplicação, embora a documentação da Oracle indique que ele funciona no servidor de aplicação tal como OLE2 e TEXT\_IO. Este recurso foi substituído por CLIENT\_GET\_FILE\_NAME. Dessa forma, foi possível salvar e abrir arquivos localizados na máquina cliente.

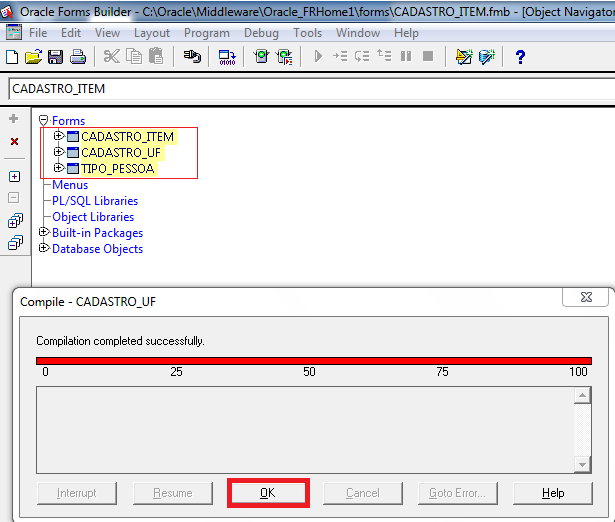


Figura 14 - Compilação em lote no Forms 11g

**Fonte: o autor**

Na tela principal, o carregamento da imagem não estava funcionando. O código que faz esse carregamento estava sendo usado em uma *trigger* WHEN-NEW-FORM-INSTANCE contida em um item do tipo imagem. Porém, segundo Housinger (2011), essa *trigger* passou a funcionar somente em nível de bloco, dessa forma, ela não estava sendo disparada. A solução foi passar o código para a trigger WHEN-NEW-FORM-INSTANCE do bloco.

Abaixo podem ser conferidas as telas de cadastro de pessoas (Figura 15) e cadastro de itens (Figura 16) que foram convertidas utilizando o *Oracle Forms Services 11g*.

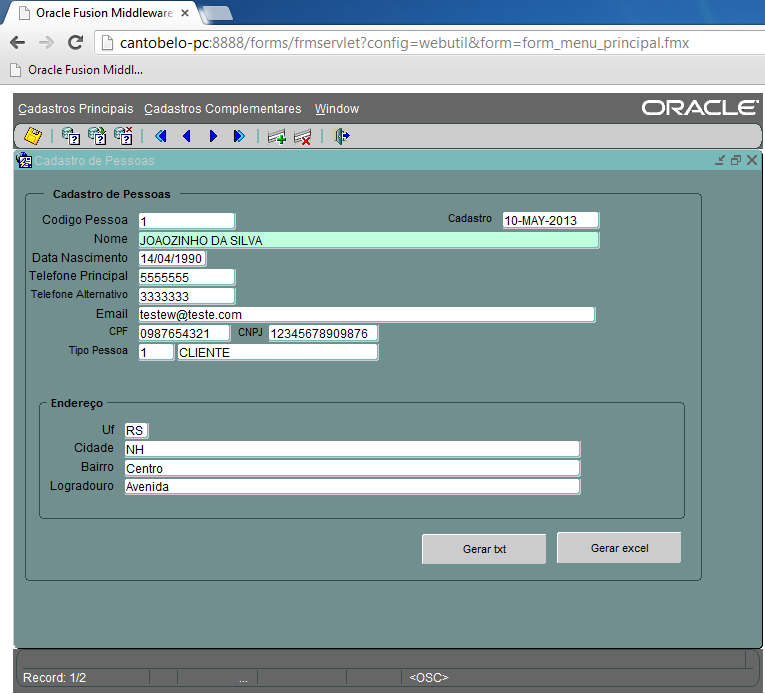


Figura 15 - Cadastro de Pessoas convertido com Forms 11g

**Fonte: o autor**

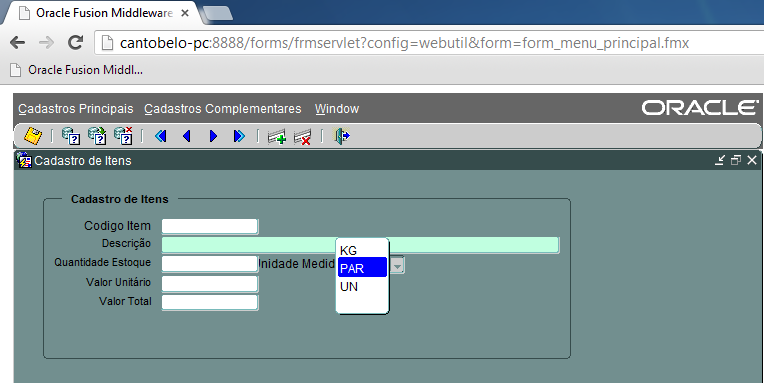
****

Figura 16 - Cadastro de Itens convertido com Forms 11g

**Fonte: o autor**

* 1. ORACLE FORMS MIGRATION ASSISTANT

O *Oracle Forms Migration Assistant* é uma ferramenta que se encontra no mesmo pacote de instalação do *Forms 11g*. A conversão é simples e rápida. Um assistente é aberto e explica a cada passo o que deve ser feito. Vários arquivos podem ser selecionados e convertidos ao mesmo tempo, porém sem interação do usuário como no *Forms 11g*. A conversão pode ser realizada também com linhas de comando, porém como é necessário criar um *batch* (programa em lote) para automatizar a conversão e a versão GUI é extremamente fácil de ser usada, os testes não foram prosseguidos no modo linha de comando.

A Figura 17 mostra o *Oracle Forms Migration Assistant* em versão GUI. Como pode ser visto na imagem, todos os arquivos a serem convertidos foram selecionados.

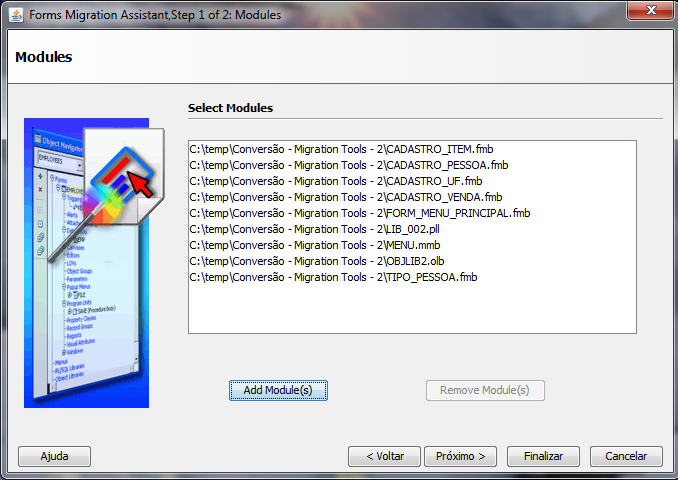


Figura 17 - Arquivos a serem convertidos

**Fonte: o autor**

Ao final do processo de conversão, um *log* é gerado, Figura 18. Esse *log* contém informações como o caminho e nome do arquivo, data e hora de início e fim da conversão, alterações realizadas pelo conversor e possíveis erros. A conversão de todos os arquivos juntos demorou somente um segundo.

Para facilitar ainda mais o processo, existe um arquivo chamado *search\_replace.properties* que contém algumas configurações, entre elas, a possibilidade de usar a ferramenta “Localizar/Substituir”. Nesse arquivo foram substituídos os componentes OLE2, TEXT\_IO e GET\_FILE\_NAME por CLIENT\_OLE2, CLIENT\_TEXT\_IO e CLIENT\_GET\_FILE\_NAME, respectivamente. Na conversão, como pode ser visto na Figura 18, o componente GET\_FILE\_NAME foi substituído automaticamente pela ferramenta.

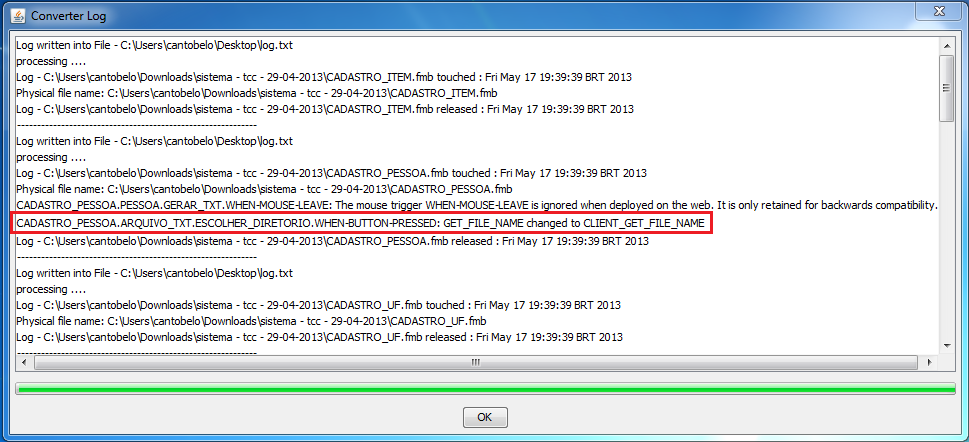


Figura 18 - Log de Conversão

**Fonte: o autor**

Na conversão do sistema de testes, somente no cadastro de pessoas e na tela principal o conversor apresentou algumas considerações. No cadastro de pessoas, sobre o componente WHEN-MOUSE-LEAVE, o *log* diz que a *trigger* é ignorada em ambiente web e só é mantida por questões de compatibilidade com versões anteriores, dessa forma, esse recurso não funcionou. Sobre o recurso GET\_FILE\_NAME, o *log* diz que é um recurso obsoleto e o compilador não mostrará erros ou avisos, nem mesmo em tempo de execução. Porém como foi substituído por CLIENT\_GET\_FILE\_NAME, esse componente funcionou corretamente, mediante a inclusão da biblioteca WebUtil.

Na tela principal, o carregamento da imagem não funcionou. O *log* indicou que existe a *trigger* WHEN-NEW-FORM-INSTANCE em um item do tipo imagem e essa *trigger* passou a funcionar somente em nível de bloco, conforme dito por Housinger (2011). Dessa forma, a *trigger* foi movida para o bloco e a imagem foi carregada corretamente.

A compilação dos arquivos não é automática, por esse motivo foi criado um *batch* para automatizar a tarefa. Após a conversão, os arquivos foram compilados. Um *log* é gerado para cada arquivo, diferente do *log* de conversão, e quando há algum erro de compilação, o arquivo fmx (executável do *Forms*) não é criado. Somente no cadastro de pessoas houve erro de compilação. Como o *Oracle Forms Migration Assistant* substituiu o componente GET\_FILE\_NAME por CLIENT\_GET\_FILE\_NAME, a compilação indicou que o último deve ser declarado. Após anexar a biblioteca WebUtil o problema foi resolvido.

As telas referentes a essa conversão não são mostradas aqui, pois ficaram graficamente idênticas às da conversão anterior.

* 1. ORACLE ADF

O Oracle ADF é um *framework* J2EE completamente integrado a IDE *JDeveloper*. Este *framework* simplifica o desenvolvimento de aplicações usando um método visual e declarativo (DAVELLAR, 2012).

Complementando o *JDeveloper* e o Oracle ADF, existe uma ferramenta chamada *JHeadstart*. Esta ferramenta facilita o desenvolvimento adicionando uma nova abordagem, especificando metadados e gerando uma aplicação completa de uma só vez (ORACLE, 2013).

Um dos principais componentes do *JHeadstart* é o *Forms2ADF*. Este componente permite a geração de aplicações ADF baseadas em uma aplicação *Oracle Forms* (ORACLE, 2013). Esta opção é interessante, principalmente, para empresas onde a maioria dos desenvolvedores implementam em *Oracle Forms,* pois o *back office* pode continuar usando *Oracle Forms*.

Apesar de ser uma tecnologia interessante, não foi possível avaliar a ferramenta. Ao contrário do que acontece com a maioria de seus produtos, a Oracle não disponibiliza o *Forms2ADF* para *download* na OTN (*Oracle Technology Network*). O *JHeadstart* possui uma versão de avaliação, mas somente é permitido o download do *Forms2ADF* para os clientes que possuem a licença do *JHeadstart,* e esta custa em torno de U$ 1750,00 por usuário (ORACLE, 2013). Por esse motivo, não foi possível avaliá-la.

* 1. APEX

Embora o *Oracle Forms 6i* ainda atenda as necessidades da maioria das empresas, muitas delas querem uma experiência diferente para seus usuários. Com o *Oracle Application Express* (APEX), os desenvolvedores podem atender a demandas dos usuários oferecendo maior interatividade e capacidades Web 2.0 (JENNINGS, 2012).

O processo de conversão pode ser visto na imagem abaixo (Figura 19):

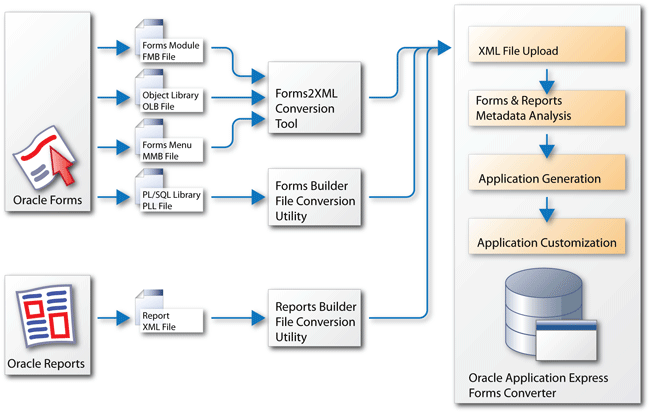


Figura 19 - Processo de Conversão Forms para APEX

**Fonte: ORACLE (2013)**

Segundo Jennings (2012), quando uma aplicação necessitar ser convertida para APEX, essa atividade não se resume a simplesmente carregar o código-fonte, clicar em alguns botões e a aplicação é gerada automaticamente. O projeto de conversão serve, na verdade, para fornecer um *design* inicial da aplicação. Após o carregamento dos arquivos no APEX, a aplicação deve ser analisada e incrementada até que fique próxima da original.

O APEX possui algumas limitações em relação ao *Oracle Forms*. Conceitos como *canvas* e *windows* não são suportados pelo APEX, suas páginas são criadas em HTML, tendo a possibilidade de integração com *JavaScript* e AJAX. Não é possível fazer inserção ou atualização de múltiplos registros ao mesmo tempo – a menos que seja uma página tabular, onde são exibidos vários registros ao mesmo tempo. As diferenças de implementação são tantas que a maior parte da lógica da aplicação antiga não será mais relevante (JENNINGS, 2012).

O primeiro passo do processo de conversão é converter todos os arquivos para XML, dessa forma, eles podem ser carregados em um projeto de migração no APEX. Para isso, é utilizada uma ferramenta chamada *Forms2XML* (Figura 19) para converter os arquivos com extensão fmb (*forms*), olb (bibliotecas de objetos) e mmb (menus); os arquivos com extensão pll (bibliotecas PL/SQL) devem ser convertidos com o *Forms 11g*. Contudo, o *Forms2XML* somente suporta blocos de dados que sejam baseados em uma tabela ou *view* de banco de dados, ou seja, não suporta blocos baseados em sinônimos ou que não tenham uma tabela associada. O *Forms2XML* foi introduzido no *Oracle Developer Suite 9i*, apesar disso, Jennings (2012) afirma que a ferramenta funciona com versões anteriores, apesar de indicar que o arquivo seja convertido para o *Forms 9i* ou *10g* e, posteriormente, XML (JENNINGS, 2012).

Após a conversão dos arquivos, é necessário criar um *workspace*. Um *workspace* é um banco de dados virtual privado que permite que múltiplos usuários trabalhem dentro de uma mesma instância do APEX, mantendo seus objetos, dados e aplicações privados (JENNINGS, 2012).

Um projeto de migração deve ser criado para que os arquivos, já no formato XML, possam ser carregados para o APEX. Após o carregamento dos arquivos, o projeto é finalizado. Apesar de a maior parte da lógica do *Forms* não funcionar no APEX, o código ainda assim é carregado e pode ser utilizado como referência (JENNINGS, 2012).

Com o projeto de migração finalizado, a aplicação poderá ser gerada. Ao gerar a aplicação uma página *Home* padrão é definida. As páginas são criadas conforme os arquivos XML carregados no projeto. Existe a opção de criar páginas adicionais para outras tabelas como parte da aplicação. O APEX oferece também vários temas para serem escolhidos como interface de usuário.

O sistema desenvolvido para testes foi convertido para XML a fim de ser carregado em um projeto de migração no APEX. Já no carregamento dos arquivos, pôde-se perceber que os blocos de dados não baseados em tabelas não são suportados. No carregamento do programa de cadastro de pessoas, o erro da Figura 20 foi exibido. Para contornar essa situação, o XML foi alterado manualmente, removendo-se todas as referências ao bloco não baseado em tabela. O bloco removido era responsável por gerar os arquivos nos formatos txt e xls da aplicação. O restante das telas não apresentaram erros no carregamento dos arquivos XML.

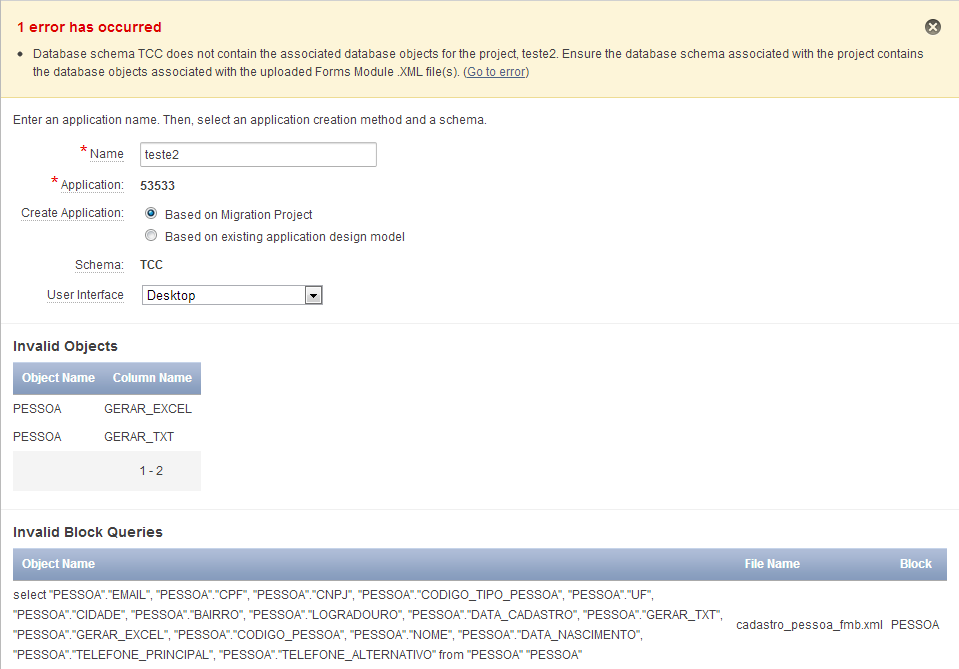


Figura 20 - Erro no carregamento do arquivo XML do cadastro de pessoas

**Fonte: o autor**

Após gerar a aplicação, foi percebido que todos os programas perderam a capacidade de auto incremento da chave primária, impossibilitando a inserção de registros. Como a chave primária era populada via aplicação (*Forms*) e não com uma *trigger* de banco de dados e a conversão não considera o código fonte da aplicação *Forms*, deve-se refazer o código no APEX para popular as chaves primárias. Se a chave fosse populada por uma *trigger* de banco de dados, não seria necessário reescrever o código.

Analisando tela por tela, nota-se o seguinte:

- UF e Tipo Pessoa: como essas telas são cadastros simples, a única questão revista foi a chave primária, conforme mencionado acima. Na Figura 21 pode ser visto que o código da tabela aparece na tela (1), mas não é possível informar um valor. Após tentar inserir um registro, ocorre erro no banco de dados (2) informando que não pode ser inserido *null* no campo.

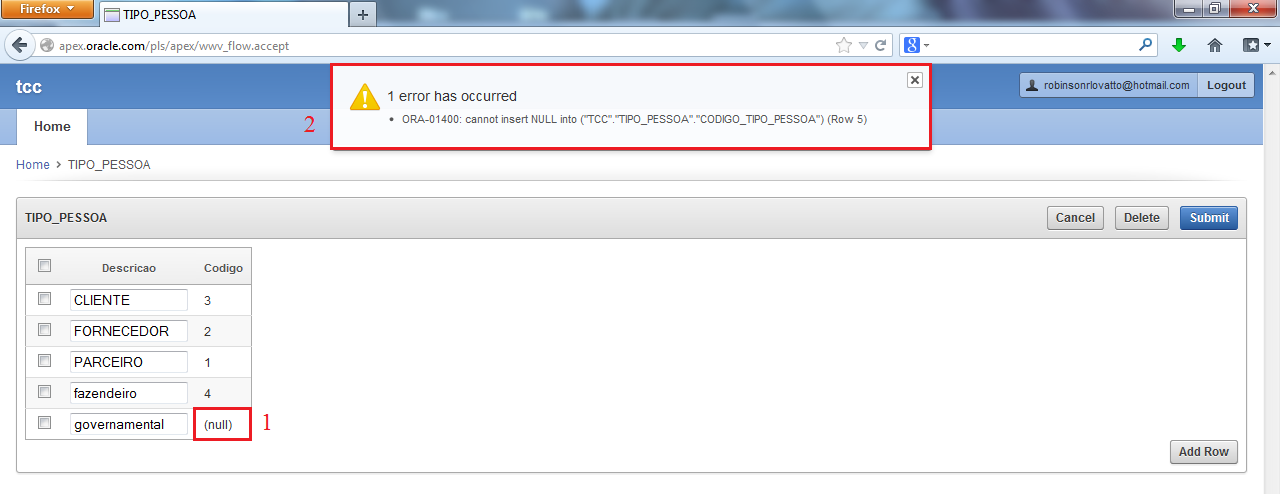


Figura 21 - Erro na inserção de registros Tipo Pessoa

**Fonte: o autor**

-Item: nessa tela (Figura 22), além da chave primária, foi perdido o componente *list item* do campo unidade\_medida (1) e o campo valor\_total “perdeu” a fórmula (2) - onde multiplicava o campo quantidade\_estoque pelo campo valor\_unitario. O *list item* passou a permitir qualquer valor, desde que respeite as restrições de banco de dados.

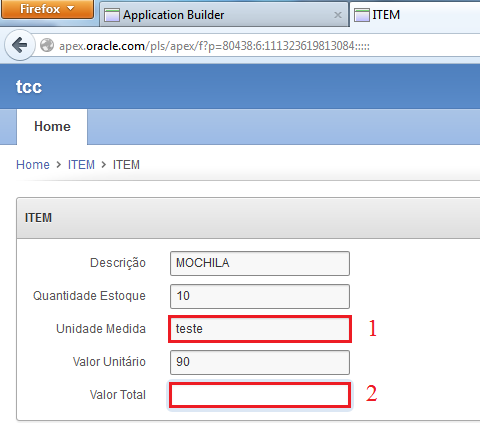


Figura 22 - Cadastro de Itens no Apex

**Fonte: o autor**

-Pessoa (Figura 23): assim como nos outros programas, o código da chave primária foi “perdido”, além do preenchimento automático dos campos data\_cadastro (4) e descrição\_tipo\_pessoa (3). No campo tipo\_pessoa do sistema desenvolvido em *Forms 6i*, havia a possibilidade de inserir o código de tipo pessoa, onde era validado, ou selecionar o registro através de uma lista de valores contendo o código e a descrição; no APEX, o campo contendo o código de tipo de pessoa não foi carregado e a descrição é exibida através de um componente *list item* (1). O campo UF era populado com a sigla da unidade federativa e continha uma lista de valores com a sigla e descrição, além de uma validação quando o usuário inseria o valor manualmente; no APEX o campo UF tornou-se um componente *list item* com a descrição (2).

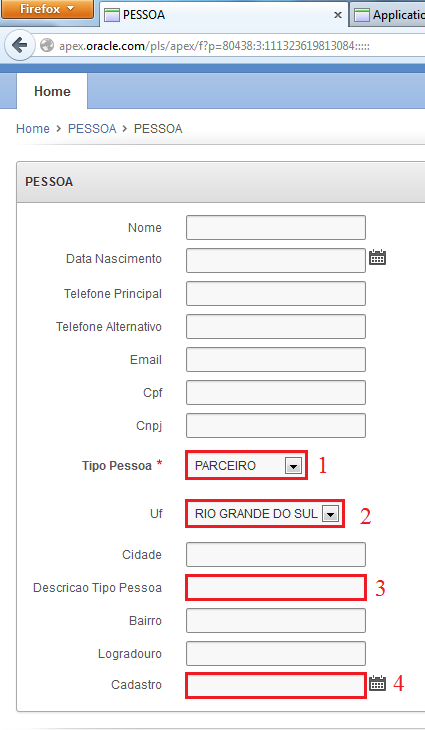


Figura 23 - Cadastro de Pessoas no Apex

**Fonte: o autor**

-Venda (Figura 24 e Figura 25): além da chave primária, há outras questões importantes a serem revistas. Essa tela se trata de um mestre detalhe entre as tabelas venda e venda\_item, ou seja, o código fonte do *Forms* contém dois blocos de dados na mesma tela (cada um baseado em uma das tabelas). No APEX, ao gerar a aplicação, esses dois blocos foram criados em páginas diferentes, ou seja, perderam o vínculo existente entre eles no sistema de testes. Na página venda (Figura 24), os campos data\_venda (2) e nome\_pessoa (4) não são preenchidos automaticamente e a lista de valores do campo Pessoa foi substituída por um *list item*(1). Na página venda\_item (Figura 25), a lista de valores para o campo codigo\_item foi substituída por um *list item* contendo a descrição do item (9). Como no *Forms* a mesma lista de valores do codigo\_item também popula o campo valor\_unitario, no APEX o valor\_unitario se tornou um *list item* com a descrição do item (8). Um campo chamado Rowid[[2]](#footnote-2) apareceu automaticamente (5). Os cálculos contidos em ambas as páginas não funcionam:

- Valor Total item (6) = quantidade (7) \* valor\_unitario (8): mesmo que o APEX contivesse a fórmula que faz esse cálculo, isso não seria possível, pois o valor unitário estaria populado com o valor “teste”;

- Total de mercadorias = somatório do valor total de todos os itens, esse campo sequer foi carregado no APEX;

- Valor Total (3) = total\_mercadorias – valor\_desconto. O campo total\_mercadorias não existe no APEX e estaria em uma página diferente do valor\_total, dessa forma, o cálculo não seria possível.

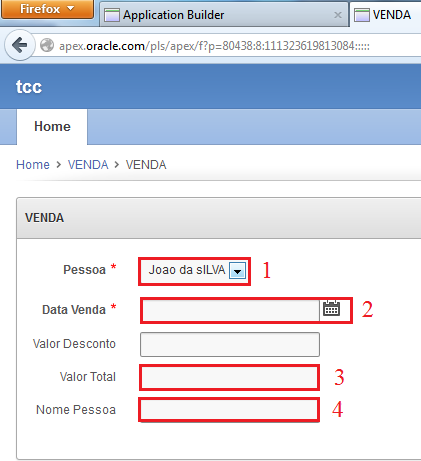


Figura 24 - Cadastro de Vendas no Apex

**Fonte: o autor**

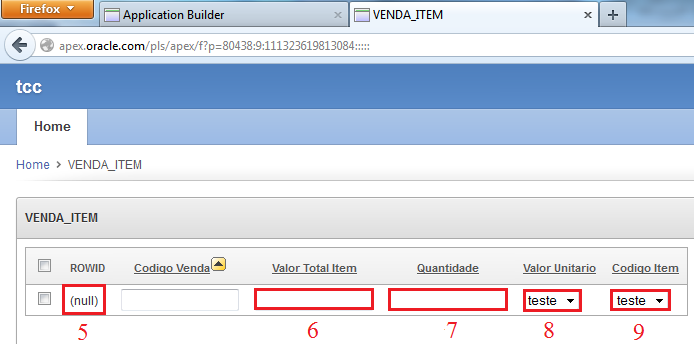


Figura 25 - Cadastro de Itens da Venda no Apex

**Fonte: o autor**

Nenhum dos recursos das bibliotecas PL/SQL e de objetos funcionaram. A tela principal não foi considerada pelo APEX, pois contém somente um bloco não baseado em tabela e, como dito anteriormente, esse tipo de bloco não é suportado pela ferramenta de conversão. O menu também não foi considerado pelo APEX, mas, em vez disso, uma página *Home* foi criada automaticamente, conforme Figura 26.

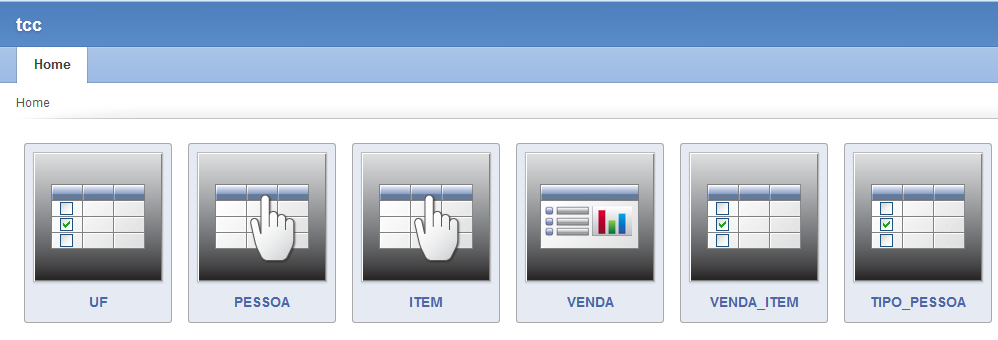


Figura 26 - Home (menu Apex)

**Fonte: o autor**

Com a análise feita até aqui, pode-se perceber que os resultados mais satisfatórios na conversão foram os produzidos para a tecnologia *Forms 11g*, independentemente da ferramenta utilizada. Tanto o *Forms 11g* quanto o *Oracle Forms Migration Assistant*, demandam uma curva de aprendizado muito baixa, além de oferecerem resultados muito próximos da aplicação original. O *Oracle Forms Migration Assistant* se sobressai perante as outras ferramentas por possuir a funcionalidade de alterar o código-fonte automaticamente, conforme definido pelo usuário no arquivo *search\_replace.properties.* O APEX, conforme citado anteriormente, deve ser usado apenas para um *design* inicial. A sua utilização em um projeto de conversão, demandaria muito redesenvolvimento, tornando o projeto muito oneroso.

No próximo capítulo serão apresentados os resultados dos testes com os usuários, a entrevista feita com eles avaliando as diferenças entre os sistemas de teste e convertidos, bem como a análise final dos resultados das entrevistas e conversões.

1. COMPARATIVO DAS FERRAMENTAS DE CONVERSÃO

Neste capítulo serão apresentados os resultados das entrevistas com os usuários que realizaram os testes nos sistemas, visando identificar as diferenças entre o sistema de teste e os sistemas convertidos, e apresentar uma análise sobre os resultados obtidos tanto das entrevistas quanto das ferramentas de conversão.

* 1. ROTEIRO DAS ENTREVISTAS

A entrevista é uma das fontes de informações mais importantes para um estudo de caso. Além de fornecer dados importantes, pode-se, mesmo seguindo uma linha consistente de investigação, tornar as questões da entrevista mais flexíveis ao invés de rígidas (RUBIN & RUBIN, 1995 apud YIN, 2005).

Yin (2005) diz que uma das formas de conduzir uma entrevista é a forma espontânea. Essa forma de entrevista permite uma conversa amigável e flexível com o entrevistado. Além de conversar sobre um assunto ou fato específico, o entrevistado fica livre para dar a sua opinião ou interpretação sobre eventos relacionados ao assunto principal.

Outro tipo é a entrevista focada (MERTON, FISKE e KENDALL, 1990 apud YIN, 2005). Nesse tipo de entrevista, o respondente é entrevistado por um curto período de tempo. Embora a conversa continue sendo espontânea, é comum que se siga um conjunto de questões pré-definidas. Um dos objetivos desse tipo de entrevista é simplesmente confirmar fatos que se acredita estarem estabelecidos. Dessa forma, as perguntas devem ser elaboradas com cautela para que o entrevistador pareça ingênuo acerca do assunto e o entrevistado possa contribuir com novos comentários (YIN, 2005).

Na entrevista realizada para este trabalho, foi utilizada a abordagem de entrevista focada. Através das conversões do sistema e testes posteriores, foi possível chegar a uma conclusão a respeito do sistema convertido que causa menor impacto aos usuários que irão utilizá-lo e a ferramenta mais adequada para conversão. As questões da entrevista foram elaboradas de tal maneira que o entrevistador pareça ingênuo acerca do resultado, conforme sugerido por Yin (2005), mas que também possam confirmar suas percepções e os entrevistados possam fornecer novos comentários.

Como a configuração do ambiente de testes é complexa, o número de entrevistados foi pequeno, pois os testes passaram a depender, principalmente, da mobilidade e tempo do entrevistador em alcançar os entrevistados. Foram selecionadas três pessoas, cada uma com um perfil diferente: um usuário, um desenvolvedor/analista e um testador/suporte. Dessa maneira, foi possível realizar os testes e entrevistas a partir de três visões distintas.

O objetivo principal dos testes não foi avaliar as funcionalidades do sistema em si, mas sim fazer uma análise comparativa do comportamento dos diferentes sistemas. Para isso, utilizou-se a abordagem de teste que se baseia em execução (SCHACH, 2009) ou, ainda, de caixa preta (PETERS e PEDRYCZ, 2001). Desse modo, foi possível observar o impacto causado aos usuários, já que estavam utilizando um sistema convertido para outra tecnologia. Se o impacto fosse grande, geraria insatisfação nos usuários e, possivelmente, seria necessário investimento em treinamento.

As entrevistas ocorreram da seguinte forma: após os testes de cada sistema, o entrevistado era questionado sobre alguns pontos desse, expressando sua opinião a respeito do sistema testado e comparando com o sistema original (*Forms 6i*). A única exigência nos testes foi a ordem dos cadastros realizados: cadastro\_uf e cadastro\_tipo\_pessoa, cadastro\_pessoa e cadastro\_item e por último, cadastro\_venda; exigência necessária por causa das restrições de banco de dados.

* 1. ENTREVISTAS

Abaixo seguem os questionamentos da entrevista e as respostas recebidas, sendo que para cada pergunta, consta a resposta dos três usuários.

*O que você observou em relação ao sistema desenvolvido em Forms 6i:*

*-Quanto à interface?*

Usuário: Simples.

Testador: Faltam alguns efeitos gráficos.

Desenvolvedor: Interface amigável. Simples, mas direta.

-*Quanto à usabilidade (facilidade de uso)?*

Usuário: Difícil para quem não conhece *Forms 6i*.

Testador: Fácil utilização quando se sabe os comandos de atalho.

Desenvolvedor: Fácil de usar. A navegação entre os campos está OK. Muito fácil de criar um item , um cliente e lançar uma venda.

-*Quanto ao desempenho?*

Usuário: Rápido.

Testador: Rápido.

Desenvolvedor: O desempenho é bom. O tempo de resposta para salvar ou consultar um item é bem rápido.

-*Algo mais?*

Usuário:

Testador: O sistema é funcional, o recurso do *Help* auxilia bastante na utilização.

Desenvolvedor: O sistema é bem simples de utilizar, porém necessita que seja instalado um *client* para rodar.

Após a leitura das respostas, percebe-se que a interface do sistema original é considerada simples por todos os entrevistados, o testador inclusive indicou que “faltam efeitos gráficos”. Sobre a usabilidade pode-se considerar o sistema fácil de utilizar para quem conhece os comandos de atalho. O desempenho é considerado bom ou rápido por todos os entrevistados. Abaixo, respostas sobre o sistema convertido utilizando o *Forms 11g*.

*O que você observou em relação ao sistema convertido utilizando o Forms 11g:*

-*Quanto à interface?*

Usuário: É mais moderna que a anterior.

Testador: Mudou para melhor, se tratando de uma interface web, os cantos arredondados tornaram a aparência mais leve.

Desenvolvedor: Interface idêntica a do sistema em 6i.

-*Quanto à usabilidade (facilidade de uso)?*

Usuário: Difícil para quem não conhece os atalhos.

Testador: Houve mudança das teclas de atalho, isso dificultou um pouco.

Desenvolvedor: A única dificuldade são as teclas de atalho que não são as mesmas do sistema em 6i. O restante é igual ao sistema anterior na usabilidade.

-*Quanto ao desempenho?*

Usuário: Não é tão rápido quanto o sistema anterior.

Testador: Acredito que por se tratar de um sistema web, o desempenho perante o 6i aparenta ser mais lento, isso também pode ser devido aos efeitos gráficos.

Desenvolvedor: Bom desempenho.

-*Quais as semelhanças e/ou diferenças em relação ao sistema desenvolvido em Forms 6i?*

Usuário: As teclas de atalho são diferentes.

Testador: Há semelhanças entre as barras de ferramentas, entre o design das telas, mas ficou claro que os efeitos gráficos tornam o sistema diferente, com aparência mais bonita.

Desenvolvedor: A interface é a mesma. Então não foram encontradas dificuldades na operação do sistema. A grande vantagem do sistema desenvolvido em *Forms 11g* é que ele roda em um *browser*.

-*Algo mais?*

Usuário:

Testador: No *Forms 6i* tinha uma mensagem informando “Perdeu a chance de gerar o arquivo” no cadastro de pessoas, mas no *Forms 11g* ela não apareceu.

Desenvolvedor:

Após os testes com o sistema convertido em *Forms 11g*, os entrevistados consideraram a interface igual, ou melhor, ao sistema desenvolvido em *Forms 6i*. Isso faz com que se possa acreditar que a conversão utilizando essa ferramenta deixa o sistema com a aparência muito semelhante ao sistema de origem, ou seja, há um resultado positivo em relação à interface gráfica.

Sobre a usabilidade os entrevistados ressaltaram que as teclas de atalho mudaram e isso dificultou o uso do sistema. Entretanto, segundo Thacker (2012), existe um arquivo chamado *fmrweb.res*, que permite refazer o mapeamento das teclas de atalho do teclado. Em algumas plataformas, nem todas as combinações de teclas são permitidas. Thacker (2012) exemplifica que na plataforma Windows, combinações como Alt+F4 (Fechar janela), F1 (Ajuda), teclas de edição ou do teclado *extended* (*Pause*, *ScrollLock*, *PrintScreen*, teclado numérico) não podem ser remapeadas. Ainda assim, o sistema convertido pode ficar parecido com o sistema de origem, em se tratando de atalhos de teclado.

Ao falar sobre o desempenho, dois entrevistados perceberam que o sistema estava mais lento. Algo que é normal em um sistema web, desde que essa diferença de desempenho não prejudique o trabalho dos usuários. O testador informou que a mensagem “Perdeu a chance de gerar o arquivo” não apareceu nessa versão de sistema, funcionalidade que deve ser reprojetada, já que não existe equivalente no *Forms 11g*. Abaixo são exibidas as respostas referentes ao sistema convertido com o *Oracle Forms Migration Assistant*.

*O que você observou em relação ao sistema convertido utilizando o Oracle Forms Migration Assistant:*

-*Quanto à interface?*

Usuário: Não notei nenhuma diferença quanto ao sistema anterior.

Testador: Muito semelhante ou igual ao sistema anterior.

Desenvolvedor: Idêntica à interface do sistema em *Forms 11g.*

-*Quanto à usabilidade (facilidade de uso)?*

Usuário: É a mesma do sistema anterior, mas depois de utilizar os outros sistemas ficou mais fácil.

Testador: Assim como no *Forms 11g*, a questão dos atalhos serem desconhecidos de certo modo atrapalha.

Desenvolvedor: Idêntica à usabilidade do sistema em *Forms 11g*.

-*Quanto ao desempenho?*

Usuário: Acredito ser igual ao sistema anterior.

Testador: Se tratando de uma aplicação web, o desempenho da aplicação aparentou ser mais lenta que a do 6i, mas não apresentou diferença perante o 11g.

Desenvolvedor: Idêntico ao desempenho do sistema em *Forms 11g*.

-*Quais as semelhanças e/ou diferenças em relação ao sistema desenvolvido em Forms 6i?*

Usuário: Não encontrei nenhuma diferença quanto ao sistema anterior.

Testador: Assim como no 11g as diferenças perante o 6i são claras e perceptíveis, desde a aplicação rodar na web, aos efeitos gráficos e tonalidades de cores, que tornam o sistema com uma aparência mais leve e agradável ao usuário. Comparando ao 11g, as aplicações são idênticas.

Desenvolvedor: Operacionalmente os sistemas são iguais. A vantagem do sistema convertido com o *Oracle Forms Migration Assistant* sobre o 6i é que roda via *browser*. Com isso elimina a instalação do *client*.

-*Algo mais?*

Usuário:

Testador:

Desenvolvedor: Visualmente e operacionalmente igual ao sistema convertido com o *Forms 11g*.

Analisando as respostas dos entrevistados sobre o sistema convertido com o *Oracle Forms Migration Assistant*, pode-se notar que ele foi considerado igual ao sistema convertido com o *Forms 11g*, tanto na interface, quanto na usabilidade e desempenho. As diferenças das teclas de atalho do teclado podem ser corrigidas tal como no *Forms 11g*.

Inicialmente fora definido no anteprojeto que as entrevistas seriam feitas com os gestores da empresa a qual se refere esse trabalho. No entanto, no decorrer das atividades, julgou-se mais válido fazer as entrevistas com os usuários que testariam os sistemas. Podendo eles, dessa forma, comparar o comportamento dos sistemas, de acordo com o que puderam observar.

* 1. CUSTOS DAS FERRAMENTAS

Os custos estimados das ferramentas de migração só foram calculados para o *Forms 11g* e o *Oracle Forms Migration Assistant*, pois essas ferramentas se mostraram mais propícias aos objetivos do trabalho. Esses custos devem ser calculados de acordo com as regras de licenciamento de produtos Oracle, onde existem duas métricas: *Named User Plus* ou *Processor*. *Named User Plus* é utilizado em ambientes onde os usuários podem ser identificados e contados facilmente ou quando seu custo-benefício é superior ao *Processor*. Essa métrica inclui tanto pessoas, quanto dispositivos operados por máquinas. Todos que estiverem utilizando o programa deverão estar licenciados. O parâmetro *Processor* é utilizado quando não há como contar o número de usuários utilizando os sistemas, um exemplo são os sistemas de *e-commerce*. Nesse caso, cada núcleo do processador do servidor é contado e multiplicado por um fator estabelecido pela Oracle no *Oracle Processor Core Factor Table*. Esse fator é determinado pelo modelo do processador (ORACLE, 2009).

A empresa na qual se baseia esse trabalho conta com um servidor que possui 2 processadores Intel Xeon E5649 de 6 núcleos cada, ou seja, 12 no total. Como esse modelo de processador tem fator 0.5 no *Oracle Processor Core Factor Table* (ORACLE, 2013), calcula-se:

Número de licenças = 12 (núcleos) \* 0.5 (fator) = 6 licenças do tipo *Processor*

Na simulação de custos abaixo, é utilizada a licença perpétua do *Weblogic Standard*, servidor de aplicação utilizado pelo *Forms 11g*, mas existe também a possibilidade de adquirir uma licença de 1 a 5 anos, porém com um custo-benefício menor (ORACLE, 2013). No Quadro 1, podem-se comparar os valores referentes ao licenciamento do *Weblogic* utilizando cada métrica:

Quadro 1 - Licenciamento Weblogic

|  |  |  |  |
| --- | --- | --- | --- |
| Métrica | Número de licenças | Valor por licença | Valor total |
| Named User Plus | 57 (colaboradores que utilizam Forms) | 435,00 | 24.795,00 |
| Processor | 6 | 21.728,00 | 130.368,00 |

Fonte: adaptado de Oracle (2013)

O *Forms 11*g segue as mesmas regras de licenciamento do *Weblogic* (ORACLE, 2009). No Quadro 2 pode-se avaliar o investimento necessário para licenças perpétuas dessa ferramenta:

Quadro 2 - Licenciamento Forms 11g

|  |  |  |  |
| --- | --- | --- | --- |
| Métrica | Número de licenças | Valor por licença | Valor total |
| Named User Plus | 57 (colaboradores que utilizam Forms) | 999,00 | 56.943,00 |
| Processor | 6 | 49.974,00 | 299.844,00 |

Fonte: adaptado de Oracle (2013)

Os custos do *Oracle Forms Migration Assistant* são os mesmos do *Forms 11g*, pois ambas ferramentas fazem parte do mesmo pacote.

* 1. ANÁLISE DOS RESULTADOS DAS CONVERSÕES

Os resultados da entrevista atenderam ao seu propósito, que era confirmar as percepções obtidas pelo entrevistador, durante o processo de conversão. Sendo o *Forms 11g* sucessor do *Forms 6i*, naturalmente, espera-se que a conversão para essa tecnologia seja a que apresenta a menor quantidade de erros e/ou ajustes manuais por parte dos desenvolvedores. Cabe avaliar agora qual a ferramenta que melhor se adapta ao processo de conversão, valendo-se de que tanto o *Forms 11g* quanto o *Oracle Forms Migration Assistant* convertem o sistema para a tecnologia *Oracle Forms 11g*.

Desse modo, como as ferramentas citadas geraram sistemas muito semelhantes, com o mesmo impacto sobre os usuários, a ferramenta a ser escolhida deve apresentar vantagens no processo de conversão e quanto aos recursos financeiros disponibilizados.

O ambiente de desenvolvimento *Forms 11g* é a ferramenta mais fácil de ser utilizada no processo de conversão, devido a sua compatibilidade e semelhança com o *Forms 6i*. Os desenvolvedores que realizariam a conversão não necessitariam de nenhum treinamento adicional, mas se encontrassem algum componente obsoleto, seria útil estarem munidos do manual de migração da Oracle para saberem qual componente é equivalente no *Forms 11g*. Qualquer alteração que fosse necessária poderia ser feita na própria ferramenta, não sendo necessárias ferramentas adicionais.

Como desvantagem na utilização dessa ferramenta, os componentes que foram mantidos no *Forms 11g* por motivos de compatibilidade, não mostram nenhum erro quando as telas são compiladas e, simplesmente não funcionam na execução do programa. Outra desvantagem seria o processo totalmente manual que é realizado para selecionar os arquivos com extensão fmb, um a um para serem abertos, e depois um a um para serem compilados; essa ferramenta não permite a criação de um arquivo *batch* para automatizar essas tarefas.

O *Oracle Forms Migration Assistant* é uma ferramenta de conversão muito rápida e simples, que possui os modos linha de comando e GUI. Nesse trabalho foi utilizada a versão GUI, pois é uma interface amigável e não necessita de nenhum conhecimento específico para utilizá-la; o *wizard* explica cada passo da conversão. Os desenvolvedores poderiam fazer uso dessa ferramenta sem qualquer dificuldade, mas é necessário um mínimo de conhecimento para editar o arquivo *search\_replace.properties*, conhecimento esse facilmente adquirido na documentação da Oracle.

O tempo de conversão utilizando o *Oracle Forms Migration Assistant* é muito baixo, pois a ferramenta é rápida e podem ser selecionados vários arquivos de uma vez para serem convertidos. Outros pontos que contam a favor são a funcionalidade “Localizar/Substituir” e o *log* que é gerado, pois através do *log* identificam-se os componentes obsoletos e também as correções que devem ser realizadas. Como pontos negativos: a ferramenta não compila as telas convertidas, é necessário criar um arquivo *batch* para automatizar essa tarefa; é necessária a utilização do *Forms 11g* para fazer as correções que são descritas no *log* de conversão. Outro ponto negativo a destacar é que não é possível anexar bibliotecas PL/SQL automaticamente. No final da seção 4.2 temos um exemplo disso: o recurso GET\_FILE\_NAME é substituído por CLIENT\_GET\_FILE\_NAME, mas ocorre erro na compilação, pois a biblioteca que contém o recurso CLIENT\_GET\_FILE\_NAME ainda não havia sido anexada.

Na conversão realizada com o APEX não foram obtidos resultados tão positivos quanto nas outras. Os arquivos fmb são convertidos para xml via linhas de comando, mas pode-se criar um arquivo *batch* para automatizar a tarefa. Após isso, deve ser criado um projeto de migração no APEX e os arquivos devem ser carregados individualmente, esse processo demanda um tempo razoável. Conforme mencionado por Jennings (2012), a migração para APEX serve apenas como *design* inicial, a maior parte das funcionalidades do sistema desenvolvido em *Forms 6i* não foi considerada. O layout ficou completamente diferente e os campos de somatório não funcionaram. Os blocos de dados que não são baseados em tabelas de banco de dados não foram carregados, o uso desse tipo de bloco é muito comum no *Forms 6i*. A quantidade de funções a serem redesenvolvidas foi tanta que o sistema nem mesmo ficou funcional, dessa forma, ele não foi disponibilizado para os testes dos usuários.

Infelizmente, não foi possível testar a tecnologia Oracle ADF, devido às restrições já apresentadas. O código-fonte gerado com essa tecnologia é em linguagem de programação Java. A possibilidade de fazer essa conversão resultaria em uma discussão mais profunda, pois seria outra opção a ser avaliada.

Apesar de serem necessários alguns ajustes, os sistemas resultantes das conversões utilizando o *Forms 11g* e o *Oracle Forms Migration Assistant* ficaram funcionais para as suas tarefas básicas: inserção, exclusão, atualização e recuperação de dados. A conversão para APEX não ficou funcional e seria necessário redesenvolver boa parte das funcionalidades do sistema, o que foge do escopo desse trabalho. A conversão com Oracle ADF não foi possível devido a restrições de custo, conforme já mencionado.

Analisando as respostas das entrevistas e de acordo com o que já havia sido apresentado anteriormente, percebe-se que a tecnologia “alvo” *Forms 11g* gera um impacto pequeno sobre os usuários. No processo de conversão, levando-se em consideração os pontos observados, chegou-se a conclusão de que a melhor forma de fazer a conversão é utilizando o *Oracle Forms Migration Assistant* para automatizar as tarefas tanto quanto possível. Entretanto, como as correções devem ser feitas utilizando o *Forms 11g,* as duas ferramentas devem ser usadas em conjunto. Como ambas fazem parte do mesmo pacote, somente uma licença de software é necessária. Indica-se que o processo de conversão seja realizado no *Oracle Forms Migration Assistant*, principalmente, pela velocidade de conversão e informações apresentadas no *log*. Caso o arquivo com extensão fmb não tenha nenhum componente obsoleto, a conversão é automática. Caso contrário, entra em cena o *Forms 11g*, onde, após analisar o *log*, pode-se partir direto para a fonte do problema. Quando uma nova biblioteca PL/SQL deve ser anexada (caso do WebUtil) ou ainda, a biblioteca tem algum componente obsoleto, o *Forms 11g* deve ser usado para corrigir o problema e/ou anexar a biblioteca. Dessa forma, pode ser realizada a conversão de uma grande quantidade de arquivos, sem maiores problemas.

CONCLUSÃO

Apesar das limitações e custo elevado de manutenção dos sistemas legados, muitas empresas ainda são resistentes quando o assunto é modernização ou evolução de software. A reengenharia de software é uma atividade que pode ajudar no processo de evolução de sistema, pois o software pode ser modernizado com custo e risco reduzidos. A reengenharia de software tem várias etapas, porém devem ser verificadas quais se aplicam melhor ao sistema que deve ser modificado. A tradução de código-fonte, citada por Sommerville (2003), é o processo de reengenharia mais bem visto pelos autores que foram utilizados na revisão bibliográfica, pois a tecnologia do sistema pode ser alterada sem que a empresa tenha os elevados riscos e custos de redesenvolvimento de software.

A empresa citada neste trabalho utiliza a tecnologia *Oracle Forms 6i* como ferramenta de desenvolvimento. Embora essa seja uma ferramenta muito produtiva, não é mais suportada pela Oracle, além de ter uma interface antiga e pouco atrativa aos usuários. Por esses motivos, seria interessante que o *Oracle Forms* fosse atualizado para a sua versão mais recente ou substituído por outra tecnologia. A Oracle recomenda fortemente que esta tecnologia seja atualizada para sua última versão, a 11g. Embora alguns recursos tenham se tornado obsoletos e outros tenham sido adicionados, o *Oracle Forms 11g* ainda possui a mesma metodologia de desenvolvimento visual e declarativa, ou seja, o desenvolvimento é feito, basicamente, através da adição de componentes na tela e alteração de propriedades desses componentes.

A fim de orientar a empresa no projeto de migração, foi criado um sistema utilizando a tecnologia *Oracle Forms 6i* com o objetivo de testar diferentes ferramentas de conversão e, posteriormente, avaliá-las. Por isso, nesse trabalho, os testes estão presentes no processo de conversão para avaliar os recursos de cada ferramenta utilizada, e também depois das conversões, identificar as alterações necessárias em cada sistema convertido.

Após as alterações necessárias para os sistemas ficarem inteiramente funcionais, foi elaborada uma entrevista. Essa entrevista foi respondida por três pessoas de perfis diferentes: um usuário, um analista/desenvolvedor e um testador/suporte; logo após eles testarem o sistema original e os sistemas convertidos. Baseando-se nas respostas obtidas e também na observação do processo de conversão, foi possível verificar o impacto causado nos entrevistados pelo uso dos diferentes sistemas, assim como fazer uma análise comparativa entre eles. A entrevista serviu ao seu objetivo principal que era confirmar as percepções obtidas pelo entrevistador durante todo o processo de conversão. Dessa forma, foi possível chegar a uma conclusão sobre a ferramenta que melhor se adapta ao processo de conversão, nesse caso.

Através das observações realizadas, chegou-se a conclusão de que a ferramenta que pode melhor auxiliar a empresa no processo de conversão é o *Oracle Forms Migration Assistant,* tanto pelos recursos disponíveis quanto pela baixa curva de aprendizado. Entretanto, muitas das alterações apontadas por essa ferramenta em seu *log* de conversão, devem ser corrigidas utilizando o *Forms 11g.* Dessa forma, julga-se que ambas as ferramentas devem ser utilizadas em conjunto, porém o *Oracle Forms Migration Assistant* deve ser utilizado tanto quanto possível, para automatizar as tarefas de conversão. Trabalhando dessa forma, poderá ser feita a conversão de uma grande quantidade de arquivos sem problemas.

**Referências Bibliográficas**

ASHDOWN, Lance; KYTE, Tom.**Oracle Database Concepts 11g Release 2 (11.2).**2011.Disponível em: <http://docs.oracle.com/cd/E11882\_01/server.112/e25789/intro.htm#i68236>**.** Acessoem: 03 nov. 2012.

BETHKE, Fred, et al. **Oracle® Forms Developer and Oracle Reports Developer:** Common Built-in Packages Release 6i**.**2000.Disponível em: <http://download.oracle.com/otn\_hosted\_doc/forms/forms/A73152\_01.pdf>**.** Acessoem: 20 abr. 2013.

CODEPORTING. **CodePorting C#2Java**. Disponível em: <http://codeporting.com/apps/csharp-2-java>. Acesso em: 25 ago. 2012.

COMPOSERCIPHERSOFT. **Products Overview**. Disponível em: <http://www.ciphersoftinc.com/products/>. Acesso em: 25 ago. 2012.

CYRAN, Michele. **Oracle Database Concepts 10g Release 1 (10.1).** 2003.Disponível em: <download.oracle.com/docs/cd/B12037\_01/server.101/b10743.pdf >**.** Acessoem: 03 nov. 2012.

DAVELLAR, Steven, et al.**Oracle JHeadstart 11g for ADF Release 11.1.2:** Developer’s Guide. 2012.Disponível em: <http://download.oracle.com/consulting/jhsdevguide1112.pdf>**.** Acesso em: 10 abr. 2013.

FERNANDES, Lúcia. **Oracle 9i para Desenvolvedores Curso Completo**. Rio de Janeiro: Axcel Books, 2002.

GREENWALD, Rick; STACKOWIAK, Robert; STERN, Jonathan.**Oracle Essentials:** Oracle Database 11g. 4 ed. Sebastopol, CA: O’Reilly Media, 2007.

HOUSINGER, Art, et al. **Oracle® Forms Upgrading Oracle Forms 6i to Oracle Forms 11g.**2011. Disponível em: <http://docs.oracle.com/cd/E24269\_01/doc.11120/e24478/whymigr.htm>. Acesso em: 25 ago. 2012.

IBROWSE CONSULTORIA. **Telescope for Jaguar**. Disponível em: <http://www.ibrowse.com.br/ibsite/site/produtoseservicos.xhtml>. Acessoem: 25 ago. 2012.

JENNINGS, Terri. **Oracle Application Express Application Migration Guide, Release 4.1.** 2012.Disponível em: <http://docs.oracle.com/cd/E23903\_01/doc/doc.41/e21675.pdf>.Acessoem: 30 abr. 2013.

O'BRIEN, Shaun; SHMELTZER, Shay.**Oracle Application Development Framework Overview.**2011. Disponível em: <http://www.oracle.com/technetwork/developer-tools/adf/adf-11-overview-1-129504.pdf>**.** Acesso em: 03 nov. 2012.

ORACLE. **1970sDefyingConventionalWisdom.** Disponível em: <http://www.oracle.com/us/corporate/profit/p27anniv-timeline-151918.pdf>**.** Acesso em: 03 nov. 2012.

ORACLE. **Forms 6i Documentation**. 2007. Disponível em: <http://www.oracle.com/technetwork/documentation/6i-forms-084462.html>. Acesso em: 25 ago. 2012.

ORACLE.**HOW TO - Performing OLE on the client using WebUtil.**Disponível em: <http://www.oracle.com/technetwork/developer-tools/forms/howto-ole-090332.html>.Acesso em: 20 abr. 2013.

ORACLE. **Oracle Application Express 4.2: Overview.** Disponível em: <http://www.oracle.com/technetwork/developer-tools/apex/overview/index.html>**.** Acesso em: 03 nov. 2012.

ORACLE. **Oracle Forms 6i andReports 6i Licensing.** Disponível em: <http://www.oracle.com/us/corporate/pricing/forms-reports-070602.pdf >**.** Acessoem: 29 set. 2012.

ORACLE.**Oracle Forms and Reports**.Disponível em: <https://shop.oracle.com/pls/ostore/f?p=dstore:product:0::NO:RP,6:P6\_LPI:4509906074781805720005>**.** Acessoem: 04 jun. 2013.

ORACLE.**Oracle JHeadstart 11g:** Developing JEE Applications has never been so easy**.** Disponível em: <http://www.oracle.com/technetwork/developer-tools/jheadstart/overview/jheadstart11gleaflet-131834.pdf>**.** Acesso em: 10 abr. 2013.

ORACLE.**Oracle JHeadstart - Frequently Asked Questions.** Disponível em: <http://www.oracle.com/technetwork/developer-tools/jheadstart/jheadstart-faq-085254.html#bm0>**.**Acesso em: 10 abr. 2013.

ORACLE. **Oracle Processor Core FactorTable**. 2013. Disponível em: <http://www.oracle.com/us/corporate/contracts/processor-core-factor-table-070634.pdf>. Acessoem: 04 jun. 2013.

ORACLE.**Oracle WebLogic Server Standard Edition**.Disponível em: <https://shop.oracle.com/pls/ostore/product?p1=OracleWebLogicServerStandardEdition&p2=&p3=&p4=&sc=ocom\_oracleweblogicserverstandardedition>. Acesso em: 04 jun. 2013.

ORACLE. **Software InvestmentGuide**. 2009. Disponível em: <http://www.oracle.com/us/corporate/pricing/sig-070616.pdf>. Acesso em: 04 jun. 2013.

ORACLE.**Supporting Forms Developer Tools: Graphics Builder.Disponível em:<http://www.oracle.com/technetwork/developer-tools/forms/fe1-2c-089646.html>. Acessoem: 06 out. 2012.**

ORACLE.**Supporting Forms Developer Tools: Procedure Builder.Disponível em:<http://www.oracle.com/technetwork/developer-tools/forms/fe1-3b-089686.html>. Acessoem: 06 out. 2012.**

ORACLE.**Supporting Forms Developer Tools: Project Builder.Disponível em:<http://www.oracle.com/technetwork/developer-tools/forms/fe1-3a-099363.html>. Acessoem: 06 out. 2012.**

ORACLE.**Supporting Forms Developer Tools: Query Builder.Disponível em:<http://www.oracle.com/technetwork/developer-tools/forms/fe1-3d-089656.html>. Acessoem: 06 out. 2012.**

ORACLE.**Supporting Forms Developer Tools: Schema Builder.Disponível em:<http://www.oracle.com/technetwork/developer-tools/forms/fe1-3c-097815.html>. Acessoem: 06 out. 2012.**

ORACLE.**Supporting Forms Developer Tools: Translation Builder.Disponível em:<http://www.oracle.com/technetwork/developer-tools/forms/fe1-3e-097809.html>. Acesso em: 06 out. 2012.**

ORACLE.**What is Forms Developer?.Disponível em:<http://www.oracle.com/technetwork/developer-tools/forms/fe1-088499.html>. Acessoem: 06 out. 2012.**

ORACLE PRODUCT MANAGEMENT.**Oracle Application Development Tools Statement of Direction: Oracle Forms, Oracle Reports and Oracle Designer.** 2012. Disponível em: <http://www.oracle.com/technetwork/issue-archive/2010/toolssod-3-129969.pdf>**.** Acesso em: 22 out. 2012.

PAULA FILHO, Wilson de Pádua. **Engenharia de software**: Fundamentos, métodos e padrões. 2.ed. Rio de Janeiro, RJ: LTC – Livros Técnicos e Científicos Editora S.A., 2003. 602 p.

PEAKE, David. **Converting Forms:** Modernize Oracle Forms Applications with Oracle Application Express**.**2009.Disponível em: <http://www.oracle.com/technetwork/issue-archive/2009/09-may/o39browser-093206.html>**.** Acesso em: 30 abr. 2013.

PETERS, James F.; PEDRYCZ, Witold. **Engenharia de software.** Rio de Janeiro: Campus, 2001. 602 p.

PFLEEGER, Shari Lawrence. **Engenharia de software**: Teoria e Prática. 2 ed. São Paulo: Prentice Hall, 2004. 537 p.

PRESSMAN, Roger S. **Engenharia de software**. 6 ed. São Paulo, SP: McGraw-Hill, 2006.

SCHACH, Stephen R.**Engenharia de software**: Os paradigmas clássico & orientado a objetos. 7. ed. São Paulo, SP: McGraw-Hill, 2009. 618 p.

SMITH, Gordon C. **OracleDatabase 10g Release 2: A Revolution in Database Technology.** 2005.Disponível em: <http://www.oracle.com/technetwork/issue-archive/2005/05-sep/bwp-overview-10gr2-060205-131882.pdf>**.** Acessoem: 03 nov. 2012.

SOMMERVILLE, Ian. **Engenharia de software**. 6. ed. São Paulo, SP: Pearson Addison Wesley, 2003. 580 p.

SOMMERVILLE, Ian. **Engenharia de software**. 8. ed. São Paulo, SP: Pearson Addison Wesley, 2007. 568 p.

THACKER, Swati. **Oracle Fusion Middleware Forms Services Deployment Guide 11*g*Release 2 (11.1.2).**2012.Disponível em: <http://docs.oracle.com/cd/E24269\_01/doc.11120/e24477/intro.htm#i1009368>**.** Acesso em: 25 out. 2012.

YIN, Robert K. **Estudo de caso:** Planejamento e Métodos. 3. ed. Porto Alegre: Bookman, 2005. 212 p.

1. Termo utilizado por Sommerville (2003) que se refere a um sistema que passou por reengenharia. [↑](#footnote-ref-1)
2. Endereço físico de um registro no banco de dados. [↑](#footnote-ref-2)