Revista Engenho, vol.9 – Junho de 2014

A IMPORTÂNCIA DA QUALIDADE NO DESENVOLVIMENTO DE *SOFTWARE*

CRISTIANE CANDIDO PEREIRA
Centro Universitário Padre Anchieta
Ane.cpereira@gmail.com

CARLOS EDUARDO CÂMARA
Centro Universitário Padre Anchieta
ccamara@anchieta.br

PETER JUNIOR JANDL
Centro Universitário Padre Anchieta
pjandl@anchieta.br

RESUMO

Qualidade no desenvolvimento de *software* está relacionada ao fato de antecipar, satisfazer, estar em conformidade com os requisitos necessários para o cliente. Porém, não basta a qualidade existir, ela deve também ser reconhecida pelo cliente. Para que isso ocorra, o desenvolvedor precisa entender que o problema não está *no software* em si, mas sim na forma como ele é desenvolvido e na necessidade de aplicar os conceitos de qualidade. O objetivo da presente pesquisa é demonstrar por meio de um estudo de caso e aplicação de testes que a qualidade do produto é algo extremamente importante, e, consequentemente, mas a qualidade do processo pode assegurar a qualidade no produto.

Palavras-Chave: engenharia de *software*; qualidade de *software*; testes de *softwa*re.

ABSTRACT

Quality in *software* development is related to anticipate, satisfy, comply with the necessary requirements for the client. However, it is not enough for the quality to exist, it must also be recognized by the client. For this to happen, the developer needs to understand that the problem is not in the software itself, but how it is developed and on the need of applying the concepts of quality. The objective of the present research is to show, through a study case and tests application that the product quality is extremely important, and, consequently, the quality of the process can ensure the quality of the product.

Keywords: *software* engineering; *software* quality; *software* testing.

Revista Engenho, vol.9 – Junho de 2014

INTRODUÇÃO

Atualmente, as empresas desenvolvedoras de *software* buscam um objetivo comum que é produzir *software* com alto nível de qualidade. A preocupação com a qualidade deixou de ser um diferencial competitivo e passou a ser um pré-requisito básico para participação ativa no mercado, afinal um único sistema é capaz de integrar todos os departamentos de uma empresa e o cliente deseja um sistema que traga solução rápida e eficiente. "*A importância de um projeto de software pode ser definida com uma única palavra – Qualidade"* (Pressman, 2006).

É exatamente nesse contexto que a engenharia de *software* tem ganho espaço dentro das empresas, contribuindo com métodos, ferramentas e metodologias avançadas para obter maior nível de qualidade. Ela muda continuamente à medida que novos métodos, melhores análises e o mais amplo entendimento evolui, aprimorando o projeto.

Segundo Paula (2009) "*O que decide a qualidade é a comparação com os respectivos requisitos: O confronto entre a promessa e a realização de cada produto*", porém qualidade é algo relativo. O que é qualidade para uma pessoa pode ser uma falha para outra. Por esse relativismo que atender aos requisitos do cliente é importante. Para que a qualidade seja alcançada, são utilizadas técnicas como validação e verificação de software e testes de sistema que são imprescindíveis para atingir a qualidade no *software*.

A qualidade no *software* utilizando dessas técnicas traz benefícios a empresa como aumento da produtividade, reduz os possíveis defeitos no sistema, o retrabalho é menor e o índice de satisfação do cliente maior. Não basta que a qualidade exista, ela deve ser reconhecida pelo cliente.

A qualidade de *software* se inicia no levantamento de requisitos que é uma parte de grande importância no desenvolvimento, segundo Mello *"é entender aquilo que o cliente deseja ou o que o cliente acredita que precisa e as regras do negocio ou processos do negocio"* , pois é desse ponto que se inicia a qualidade de um software.

Partindo dessa necessidade de conhecer essas técnicas, nesse trabalho foram realizadas análises com bases em pesquisas procurando focar nos processos de verificação, validação e teste de *software*. Nosso objetivo é mostrar quais procedimentos devem ser tomados, quais opções existem e como podem ser realizados. Para isso, apresentaremos um estudo de caso utilizando como base essa análise, testando algumas opções e procedimentos que foram pesquisados, apresentados e aplicados.

VALIDAÇÃO E VERIFICAÇÃO DE SOFTWARE

"A validação tem como finalidade verificar se o sistema está de acordo com sua especificação. Se o sistema atende ás expectativas do cliente, ou seja, a validação avalia se a construção segue os requisitos pré-estabelecidos" (Kosciansk, 2006).

A verificação se refere aos conjuntos de atividades para garantir que o *software* seja implementado corretamente, ela identifica defeitos e possíveis problemas.

"Esses processos envolvem verificar por meio de inspeção e revisão, em cada estágio, desde a definição dos requisitos dos usuários até o desenvolvimento do programa " (Sommerville, 2003).

Essa atividade pode utilizar técnicas propostas pela norma IEEE (*Institute of Eletrical and Eletronics Engineers*) Standard for Software Test Documentation STD 829 – 1998, que é uma terminologia padrão de Engenharia de *Software*. A partir dessa norma é possível identificar as tarefas mínimas e documentos recomendados para cada nível de integridade de *software*. O nível de integridade pode ser entendido como criticidade e complexidade do *software*.

A utilização dos documentos de teste facilita na padronização e organização da execução do processo, além de facilitar o trabalho do testador, existem várias formas de criar esses documentos, podem ser encontrados modelos em site como IEEE.

Essa norma descreve oito documentos para a atividade de teste de um programa de *software*, esses documentos são usados em três áreas: no planejamento (Plano de teste), na execução (Especificação de teste) e nos relatos (Relatórios de teste).

Plano de Teste: Identifica as funcionalidades a serem testadas com ênfase nas datas, pessoas envolvidas e riscos. A figura 1 mostra algumas informações básicas para um plano de teste, a tabela foi criada através dos modelos IEEE.

Nome do Projeto:	Nome da Funcionalidade:	Versão:
Data Início:	Data Fim:	

Funcionalidade:	Tempo Despendido (h):
Contador:	Criticidade:

Figura 1. Plano de Teste (tabela desenvolvida através dos modelos IEEE)

Especificação de Teste: "*As especificações de teste são artefatos que contêm os detalhes dos testes a serem realizados. Uma especificação é reaproveitada quando testes similares e são*

realizados de diferentes marcos de um projeto. Tipicamente, uma especialização é desenhada apenas uma vez, mas o teste que ela descreve pode ser executado muitas vezes" (Paula, 2009). A figura 2 mostra uma estrutura para o relatório de especificação, com as informações a serem preenchidas de acordo com as especificações, conforme o modelo IEEE.

Objeto de Teste:		
Descrição do Caso de Teste:		
Pré-Condição:		
Dados de Entrada:		
ID	**Passo**	**Procedimento**
1	P1	
	P2	
	P3	
	V1	
2	P4	
	P5	
	V2	
Resultado Esperado: As operações deverão funcionar corretamente cumprindo todas as regras acima citadas.		

Figura 2. Especificação de Teste (tabela desenvolvida através dos modelos do site IEEE)

Conforme Blanco (2012) "*Especificação Projeto de Teste: Identifica os casos, os procedimentos e critérios de aprovação.*

Especificação Casos de Teste: Dados de entrada, resultados esperados, ações e condições gerais para executar o teste. O o que interessa nesse caso é o que é obtida no final, a confiabilidade depende da qualidade do teste.

Especificação de Procedimento de Teste: Identifica quais passos são seguidos para executar o teste.

Relatório de Incidente: Documenta qualquer evento que ocorra durante a atividade de teste e que necessite de análise posterior (erros). A figura 3 mostra um modelo de estrutura e informações necessárias para preenchimento, conforme o modelo IEEE.

Nome do Projeto:

ID	Funcionalidade	Descrição	Roteiro	Resultado Esperado	Descrição do Erro

Figura 3. Relatório de Incidente (tabela desenvolvida através dos modelos do site IEEE)

Relatório de Resumo: Mostra um resumo dos resultados das atividades associados à especificação projeto de teste. A figura 4 mostra as informações que podem ser usadas como preenchimento do relatório de teste conforme o modelo IEEE.

Relatório de Teste

Nome do Projeto:	
Data Inicio Teste:	Data fim Teste:

Números dos testes	
Casos de teste criados antes do teste:	
Casos de teste criados durante o teste:	
Casos de teste executados:	
Casos de teste com sucesso:	
Casos de teste com erro:	

Percentual	
Casos de testes executados:	
Casos de teste executados com sucesso:	
Casos de testes com incidência de erro:	

Figura 4. Relatório de Teste (tabela desenvolvida através dos modelos do site IEEE)

Diário de teste: Registro cronológico dos dados relevantes.

Relatório de Encaminhamento de Item: Identifica os itens encaminhados para testes no caso de equipes distintas serem responsáveis pelas tarefas.

Visando facilitar a documentação, pode ser feita uma união dos documentos de Plano de Teste, Especificação de Teste, Especificação de Projeto de Teste e Especificação dos Casos de Teste.

De acordo com Myers, citado por Blanco (2012) *"Infelizmente não é possível testar todas as entradas de dados e suas centenas ou milhares de combinações possíveis. Criar casos de testes para todas essas possibilidades é impraticável, pois levaria muito tempo e seria economicamente inviável".*

A verificação e validação revisam e analisam o *software* nas diversas fases dos processos de desenvolvimento, abrangem nos documentos de requisitos, os diagramas de análise de projetos e o próprio código fonte, as revisões são consideradas técnicas estáticas por não envolverem a execução do produto.

TESTES DE *SOFTWARE*

Teste é uma atividade fundamental para a qualidade ser assumida no *software*, o principal objetivo é revelar falhas, um teste bem sucedido identifica defeitos que ainda não foram descobertos e que podem ser corrigidos pelo programador. Um teste bem eficiente é aquele que é projetado para descobrir o maior número de erros possíveis.

Segundo Sommerville (2003), "*teste de software consiste em um processo que é utilizado com o intuito de descobrir evidencias de problemas de software".*

Para definir essas evidencias, é preciso saber as diferenças dos conceitos relacionados aos testes, conforme Dias (2007):

"Defeito: É um ato inconsistente cometido por um individuo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta, exemplo uma instrução ou comando incorreto.

Erro: É uma manifestação concreta de um defeito num artefato de software, diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado

inesperado na execução constitui um erro. Isso pode ser causado por desvio da especificação, modelagem mal feita, erro de programação, pressão de tempo, especificação de requisitos.

Falha: É o comportamento operacional do software diferente do esperado, um processamento incorreto, uma falha pode ser causada por diversos erros e alguns erros podem nunca causar uma falha."

A figura 5 mostra a diferença entre esses conceitos.

Figura 5. Defeito x Erro x Falha (revista Qualidade de *Software*)

"Teste de sistema é na verdade uma série de diferentes testes cuja finalidade principal é exercitar por completo o sistema baseado em computador. Apesar de cada teste ter uma finalidade distinta, todos trabalham para verificar se os elementos do sistema formam adequadamente integrados e executam as funções a eles alocados" (Pressman, 2006). Algumas finalidades do teste podem ser de :

"*Teste de recuperação: No qual o software é forçado de diversos modos a falhar para verificar se a recuperação é adequadamente realizada.*

Teste de segurança: São feitas várias invasões impróprias no sistema ou alterações dos arquivos gerados pelo sistema e verificam-se os mecanismos de proteção são capazes de protegê-lo" (Pressman, 2006).

Testes de *stress*: Executa no sistema uma quantidade de recursos anormais em grande volume para verificar o quanto o sistema é sensível.

Teste de desempenho: É analisado o desempenho do sistema durante a execução, verificando qual é o comportamento integrado entre o *software* e o *hardware*.

Teste Funcional: Verifica se todos os requisitos foram cumpridos de acordo com as regras de negocio, garantindo de que não existam diferenças entre os requisitos funcionais e o comportamento do *software* construído, o teste não se preocupa com o código em si, eles são realizados a partir da seleção dos Casos de Uso baseados na especificação.

TÉCNICAS EM TESTE DE *SOFTWARE*

O teste de *software* é uma das técnicas mais onerosas do processo de desenvolvimento. Porém o rigor e o custo depende da criticidade da aplicação, essas técnicas são classificadas de acordo com a origem das informações utilizadas para estabelecer os requisitos de testes, essas técnicas podem ser classificadas como funcional e estrutural.

A técnica estrutural conhecida também como teste de caixa branca trabalha diretamente com o código fonte, avalia aspectos como: teste de condição fluxo de dados, caminhos lógicos, laços; "*Os aspectos nessa técnica dependem da complexidade do componente de software" (*Dias,2007), ela permite uma verificação mais precisa de comportamento ela é realizada durante todo o processo de desenvolvimento.

A técnica funcional ou teste de caixa preta *"identifica erros de interface, acesso ao banco de dados, desempenho e velocidade"* (Koscians*k,2006)*; Os dados de entrada são fornecidos e o resultado é comparado a resultados previamente conhecidos, haverá sucesso se o resultado recebido foi igual ao esperado, ela é aplicável em todos os níveis de teste.

Existe também a definição de teste de caixa cinza no qual são focados os mecanismos dos componentes e não do sistema em si, ele ignora os componentes internos e focaliza apenas as saídas geradas em resposta a entrada e condições de execução selecionadas.

Conforme Sommerville (2003) "*o processo de teste deve evoluir em estágios, os testes devem ser realizados incrementalmente, em conjunto com a implantação do sistema*".

Esses processos podem ser divididos em estágios:

Teste de Unidade: Explora a menor unidade do projeto, procura falhas ocasionadas por defeitos de lógica e de implementação de cada modulo*," são testados individualmente ou em grupos de unidades para garantir que eles operam corretamente" (Sommerville, 2003)*, podem ser baseados nas especificações de classes e na codificação do código.

Teste de Módulos (Subsistema): São executados procedimentos em componentes dependentes, como uma classe de objetos que foram integrados.

Teste de Integração: Provoca falhas associando entre os módulos quando eles são integrados, testa quais componentes são combinados e avalia a interação entre eles, esse teste geralmente é executado durante o desenvolvimento.

Teste de Sistema: Busca falhas por meio de utilização como se fosse o usuário final, verifica se o produto satisfaz os requisitos funcionais e não funcionais e testa as propriedades emergências do sistema, identifica possíveis erros entre o *hardware*, o *software* e o banco de dados, é baseado no projeto e na arquitetura do *software*.

Teste de Aceitação: Costuma ser o estágio final do processo de teste do sistema e simulando rotinas de operações de modo a verificar se seu comportamento está de acordo com o solicitado, "*ele pode revelar erros e omissões nas definições de requisitos, quando os recursos na verdade não atendem ás necessidades dos usuários ou quando o desempenho do sistema é inaceitável" (Sommerville, 2003).*

Teste de Regressão*: "Consiste em aplicar todos os testes já aplicados nas versões anteriores em todos os ciclos e atualizações*", (Dias, 2007), verifica se as alterações não causam nenhum efeito indesejado e se o sistema mantém a conformidade com os requisitos especificados, são mais utilizados na manutenção do sistema.

Técnicas de testes devem ser vistas como complementares e a questão está em como as utilizar de forma que as vantagens de cada uma seja melhor explorada em uma estratégia que leve a uma atividade de teste de boa qualidade, que seja eficaz e de baixo custo.

O objetivo central de toda metodologia de teste é maximizar a sua cobertura e a quantidade potencial de defeitos que podem ser por ela detectados.

TESTES AUTOMATIZADOS

Conforme Souza (2013), "*automação de teste é o uso de software para controlar a execução do teste de software, a comparação dos resultados esperados com os resultados reais, a configuração das pré-condições de teste e outras funções de controle e relatório de testes"*, é testar um *software* com outro *software*.

São usados *scripts* padronizados que podem ser mais rápidos na execução dos testes e na detecção dos erros do que a forma de teste manual.

Utilizando testes automatizados é possível aumentar a consistência e abrangência, reduzir o tempo ou esforço, diminuir o custo, aumentar a qualidade do produto final.

O ideal é automatizar tarefas repetitivas, cálculos matemáticos, testes de regressão e funcionalidades criticas no sistema.

Funcionalidades pouco usadas, protótipos e funcionalidades que exigem inspeção visual são pouco recomendadas.

Existem ferramentas comerciais e *OpenSource* disponíveis para auxiliar o desenvolvimento de automação de software, ferramentas específicas para cada tipo de teste e *software*:

Selenium: É usado para automatizar navegadores em várias plataformas, realizando testes funcionais.

JUnit: É um *framework* simples para escrever testes repetíveis, orientado a Java, realiza testes de unidade.

JMeter: Projetado para testar aplicações *web*, mas expandiu para outras funções de teste, realizado para teste de desempenho (performance).

MODELOS DE TESTE

Modelo TMAP: O modelo teste TMAP (*Test Management Approach*) é uma abordagem que pode ser aplicada em todas as situações de teste e em combinação com qualquer outra metodologia de desenvolvimento de sistema, ele oferece ao testador uma série de elementos para seu teste como técnicas de especificação de teste, infraestrutura de teste, estratégia de teste, organização de teste, ferramentas de teste, entre outros.

TMap é composto das seguintes fases e etapas, conforme a figura 6 mostra:

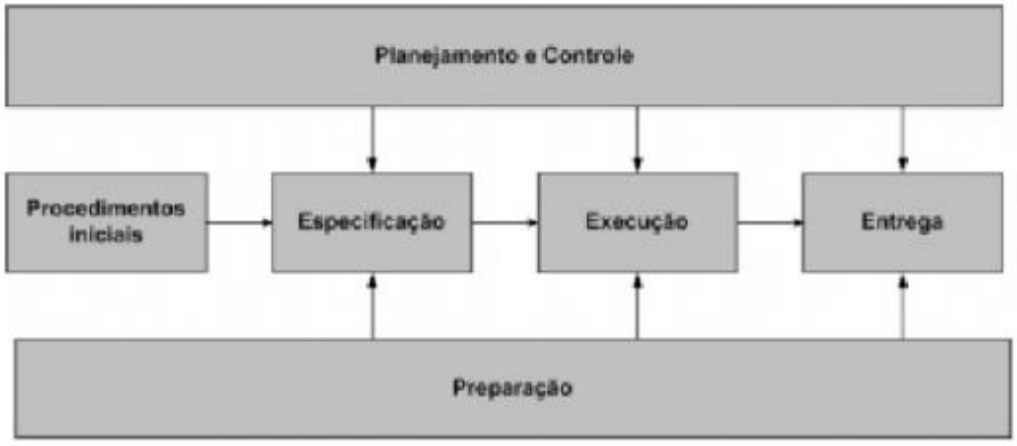

Figura 6. Fases e Etapas TMap (Gerencia de projeto de teste Segundo o Modelo do PMI por Emerson Rios,2003)

Segundo Rios (2003), *"as etapas de Planejamento e Controle e Preparação seguem em paralelo as demais etapas, pois continuam em andamento durante todo o projeto. O produto da fase de planejamento, uma vez concluído, deve ser acompanhado e controlado. Na etapa de preparação os ambientes de gerencia de mudanças e de gerencia de configuração são adequados ao projeto de testes, além de ser preparada a infraestrutura a ser utilizada no projeto (Hardware, Software e Pessoal)".*

Modelo V: A estrutura do modelo em V é uma aproximação estruturada de teste que pode ser usada em todas as metodologias de desenvolvimento. Segundo Silva "*o modelo que foi definido por Paul Rook em 1980, foi apresentado como um modelo alternativo ao modelo Waterfall e enfatizava a importância nos testes em todo o processo de desenvolvimento e não somente ao término do processo".*

Esse modelo introduz a criação de testes de dados e cenários de teste durante o ciclo de desenvolvimento do *software*, em geral, reforça a ideia de que o teste não é uma fase, mas uma parte integrante do ciclo de desenvolvimento do *software*, o qual trabalha com diferentes níveis de teste como: teste de unidade, teste integração, teste de sistema e teste de aceitação.

A principal ideia é sempre testar a mesma coisa, *porém* com focos diferentes, o modelo em V possui dois "lados", um lado é especifico para verificação que é o ciclo de vida de desenvolvimento e o outro é especifico para a validação que é o ciclo de testes. Basicamente, o modelo segue as fases citadas na figura 7:

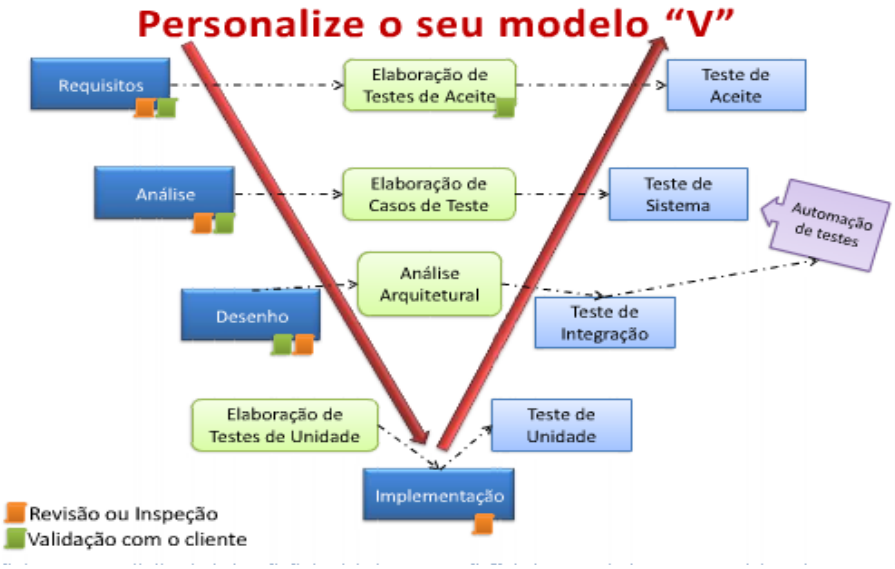

Figura 7. Fases e Etapas "V" (Técnicas de Teste no Ciclo de Desenvolvimento de *Software* por Camilo Ribeiro ,2010)

A vantagem nesse modelo é que segundo Camilo Ribeiro "*a fase de teste começa no inicio do ciclo, os planos de teste são detalhados em cada fase do ciclo o que ajuda a compreender melhor qual a origem do problema*".

A desvantagem, segundo o mesmo autor "*é que apesar da sua implementação em todas as fases continua a não ser flexível suficiente e é necessário maior número de feedback entre todas as fases do ciclo*".

ESTUDO DE CASO – LIBRE OFFICE

Libre Office é um pacote de componentes para escritório gratuito, é desenvolvido por voluntários do mundo todo através de fundações sem fins lucrativos. É uma extensão aberta do StarOffice e uma continuação de desenvolvimento da suíte OpenOffice. Seus componentes são: processador de textos (Write), planilha de cálculo (Calc), apresentações (Impress), gráficos vetoriais (Draw), banco de dados (Base), editor de fórmulas matemáticas (Math), o pacote possui inúmeras vantagens entre elas: sem taxas de licenciamento, código aberto, multiplataforma, extenso suporte a idiomas, é compatível com vários Sistemas Operacionais (como Windows, Linux e MAC OS).

*Possui grande compatibilidade com extensões de arquivos, suporte online gratuito, entre outras inúmeras vantagens (*Tutorial *LibreOffice).*

Esse estudo de caso visa testar algumas funções do componente processamento de texto Write, que além dos recursos usuais de um processador de texto, fornece também características como, exportação para PDF, integração de banco de dados, ferramentas de desenho incluídas, entre outras funções muito similar ao MS Word.

Para verificar a finalidade do *software* foi utilizado o teste funcional das funções de cartão de visitas, envelopes, abrir documentos salvos no Word. Na função de recuperação de documentos foi aplicado o teste de recuperação, com objetivo de verificar se essas funções funcionam corretamente. E finalmente, validar o uso das mesmas.

Os modelos dos relatórios foram desenvolvidos através dos modelos sugeridos site IEEE e o site do Grupo de Testadores de *Software* (Blog editado por Anne Caroline, mestre em Ciência da Computação pela UFCG).

Hipótese Abrir Documentos Salvos no *Word*: Como muitos documentos são salvos em *Word* é importante que o *Libre* Office abra os documentos, sem alterar formatações que foram estabelecidas no *Word* (veja resultado na figura 9).

Hipótese Recuperação de Dados: Como existem riscos é importante que quando o sistema fechar de forma inesperada, ele recupere todos os dados e as formatações do documento (veja resultado nas figuras 10, 11,12 e 13).

Hipótese Cartão de Visita: Independente de que forma que ele seja feito, num cartão de visita é importante que a impressão seja de acordo com o que foi estabelecido na configuração, respeitando o tamanho, *layout* e os dados informados (veja resultado nas figuras 14 e 15).

Hipótese Envelopes: A impressão do envelope deve ser correta, com o endereço legível, posição aceitável e que tamanho das letras seja legível (veja resultado nas figuras 16 e 17).

Hipótese Personalizar Barra de Ferramentas: A barra deve ser criada conforme foi feita a personalização, os atalhos criados devem funcionar corretamente e deve ser possível alterar a barra personalizada (veja resultado nas figuras 18, 19).

Hipótese Conversor de Documentos: Arquivos com extensão doc devem ser convertidos sem alterações ou erros para o formato oxps (veja resultado nas figuras 20, 21).

Foi escrito um relatório de plano de teste (veja resultado na figura 8) para cada função do sistema testado, esse plano de teste foi respeitado e executado com êxito, os resultados negativos foram registrados no relatório de incidentes, informando quais foram os erros.

LibreOffice The Document Foundation	Qualidade Plano de Teste	Libre Office

Funcionalidade: Envelope	**Tempo Despendido (h):** 1h
Contador: 02	**Criticidade:** Baixa

Objeto de Teste:

- Validar uso do assistente para criar documento no modelo Envelope.
- Verificar impressão usando um envelope.

Descrição do Caso de Teste:

- O aplicativo deverá atender corretamente a configuração fornecida pelo usuário (nas 3 abas: Envelope, Formato e Impressora).
- A impressão no envelope deverá ser realizada com sucesso.

Pré-Condição:

- *O usuário deverá informar dados do Destinatário e Remetente.*
- *Existir uma impressora compatível para impressão no envelope.*

Dados de Entrada:

ID	Passo	Procedimento
1	P1	Executar o aplicativo LibreOffice 4.2 (Opção texto)
	P2	Selecionar a opção Texto
	P3	Acessar o menu a opção Envelope (Inserir>>Envelope)
	V1	O aplicativo deverá exibir a tela do assistente para Envelope
	P3	Preencher os dados do campo Destinatário e Remetente
	P4	Clicar no botão Inserir
	P5	Se necessário ajustar a largura da caixa de texto do Destinatário e Remetente
2	P7	Clicar no ícone da Impressora
	V2	Verificar se o envelope foi impresso corretamente.

Resultado Esperado: As operações deverão funcionar corretamente cumprindo todas as regras acima citadas.

Figura 8. Plano de teste executado (tabela desenvolvida para teste do *Libre Office*)

Foram encontradas divergências e dificuldades nas funções de cartões de visita, abrir documentos salvos pelo *Word* e na Recuperação de documentos, as divergências foram:

- A função de abrir documentos no *Word* não respeita algumas formatações como tabelas, que abrem fora de formatação, cortando algumas colunas da tabela (veja resultado na figura 9);

Relatório de Incidente

Nome do Projeto: Libre Office - Write

Cont.	ID	Funcionalidade	Descrição	Roteiro	Resultado Esperado	Descrição do Err
01	1	Cartão de Visita	Inserir dados na aba Pessoal	1- Acessar conteúdo cartão de visitas – aba Pessoal 2- Preencher os campos	1- Seja intuitivo sem duplicidade de informação 2- Fácil preenchimento	1- Existe dois lugares pa inserir o nome 2- Não existe formataçã validação para CEP, telefone e e-mail 3- Campo pais e estado que ser escrito
01	1	Cartão de Visita	Inserir dados na aba Comercial	1- Acessar conteúdo cartão de visitas – aba comercial 2- Preencher os campos	1- Seja intuitivo sem duplicidade de informação 2- Fácil preenchimento	1- Não existe formataçã validação para CEP, telefone e e-mail 2- Campo pais e estado que ser escrito
01	1	Cartão de Visita	Inserir dados na aba Opções	1- Acessar conteúdo cartão de visitas	1- Fácil entendimento	*1-* Check "Sincronizar conteúdo" esta confus quando ele é usado p gerar o documento, a informações preenchi não aparece
01	2	Cartão de Visita	Conferencia dos dados	1- Após preencher os dados, clicar em novo documento	1- Fácil visualização 2- Fácil retorno as informações anteriores para alteração	1- Não aparece nenhum informação na confer 2- Para fazer alterações necessário refazer toc os passos de acesso

Figura 9. Tabela de relatório de Incidente (tabela desenvolvida para teste do *Libre Office*)

- A função de recuperação de dados não funciona, pois não recupera dados que não foram salvos *(veja resultado na figura 10, 11, 12 e 13).*

Figura 8. Plano de teste executado

O resultado foi aceitável, porém foram encontradas divergências e dificuldades nas funções de cartões de visita, abrir documentos salvos pelo Word e na Recuperação de documentos, as divergências foram:

teste de recuperação...|

A função de abrir documentos no Word não respeita algumas formatações como tabelas:

Figura 10. Imagem do texto a ser recuperado (Imagem desenvolvida para teste do *Libre Office*)

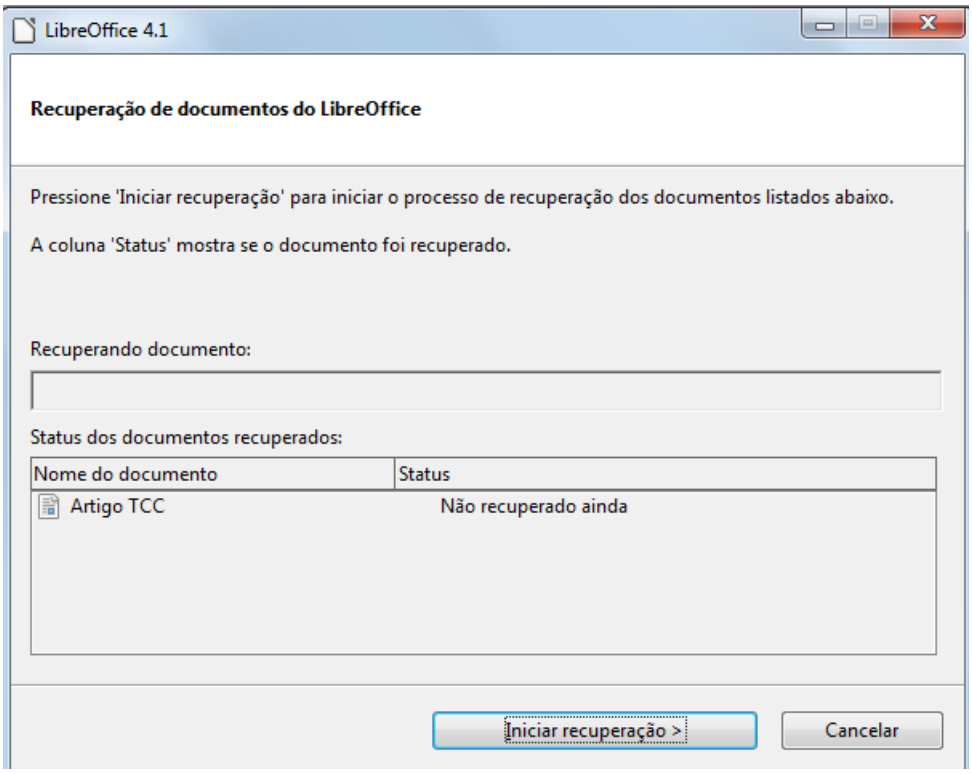

Figura 11. Tela para iniciar a recuperação de dados (*Libre Office*)

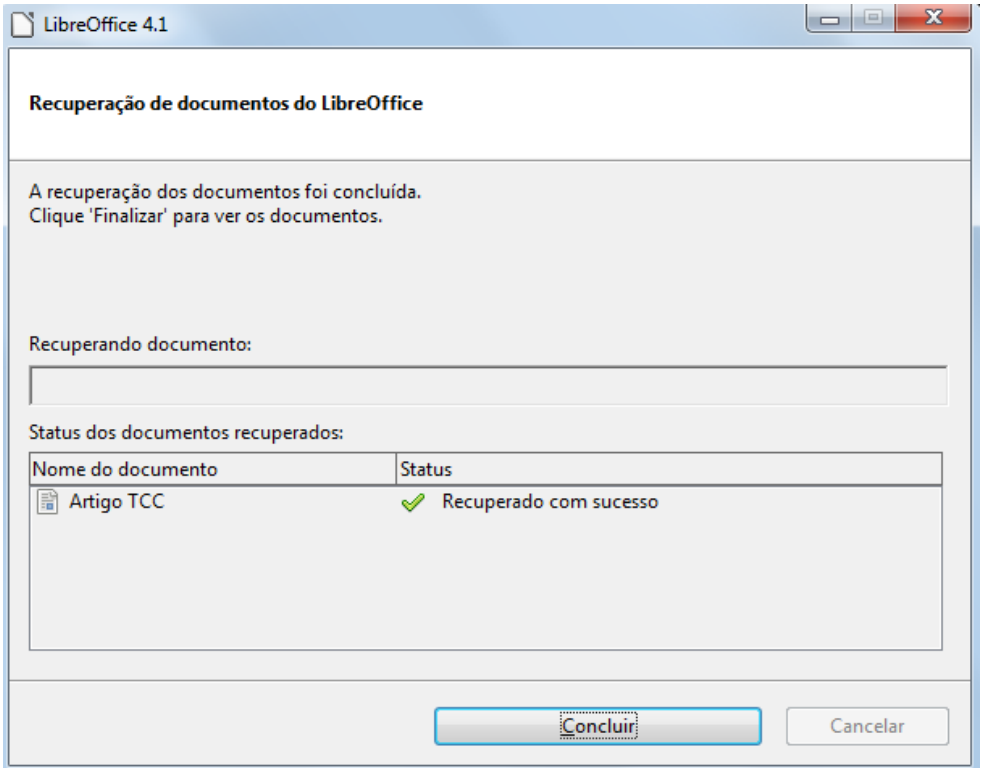

Figura 12. Tela informando os dados recuperados (*Libre Office*)

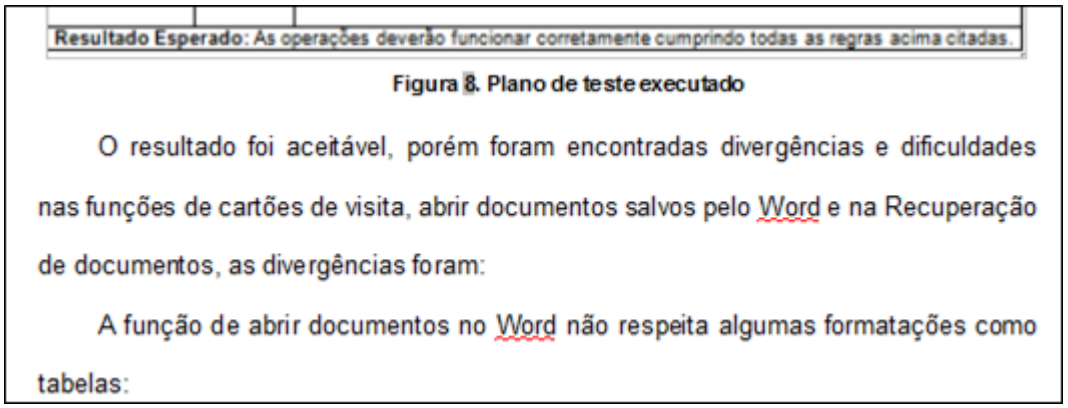

Figura 13. Imagem com texto recuperado (Imagem desenvolvida para teste do *Libre Office*)

- Na função cartões de visitas, alguns campos são confusos e não existe formatação e nem validação para as informações como CEP, telefone e e-mail, a visualização da impressão é difícil e a impressão não é econômica, pois fica muito espaço sem aproveitamento na folha e ao visualizar impressão não aparece as informações (veja resultado na figura 14 e 15).

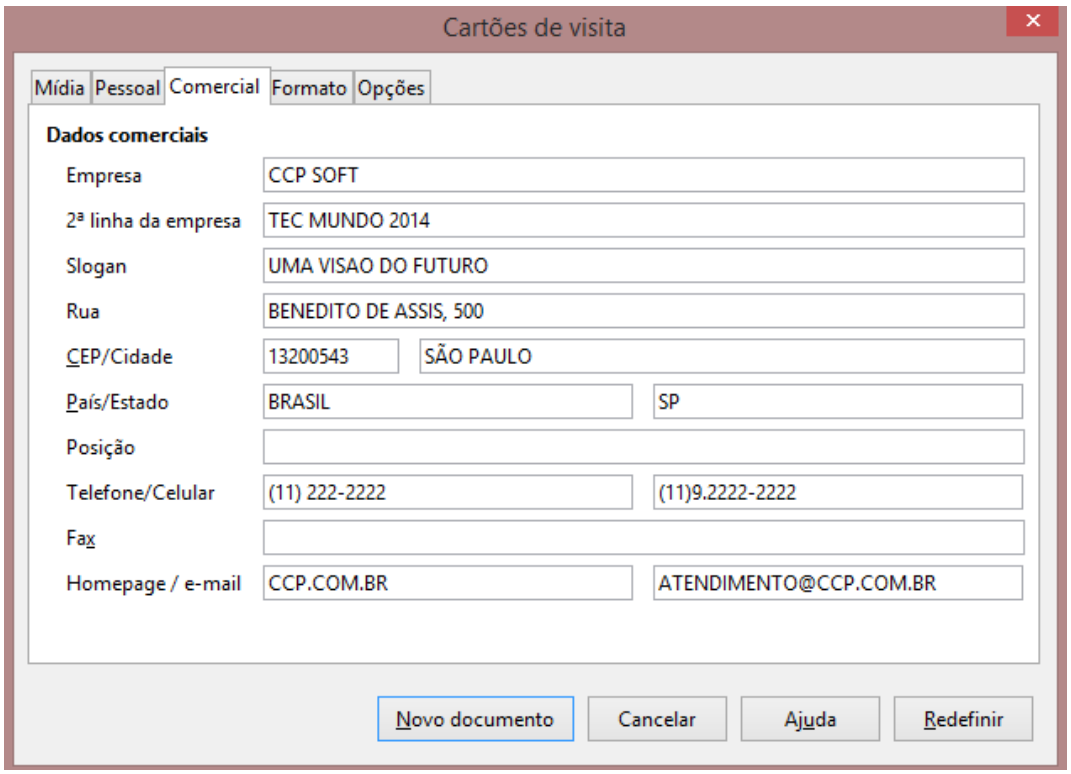

Figura 14. Tela para preenchimento cartão de visita (*Libre Office*)

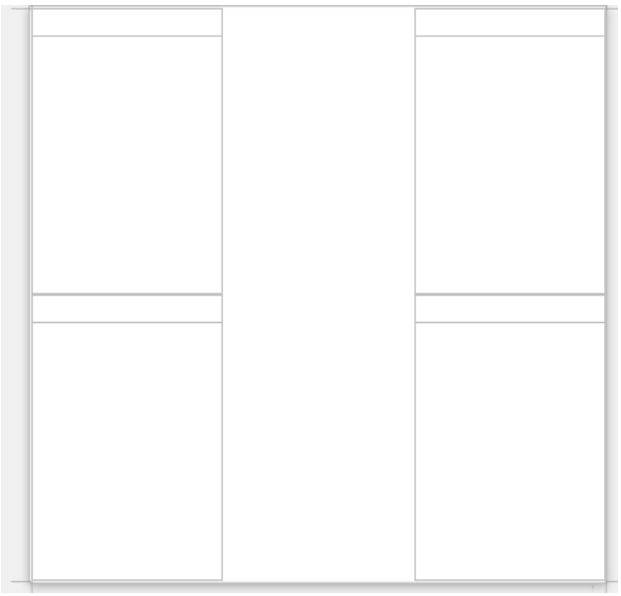

Figura 15. Tela de impressão dos cartões de visita (*Libre Office*)

- A função de envelope não apresentou divergências (veja resultado na figura 16 e 17)

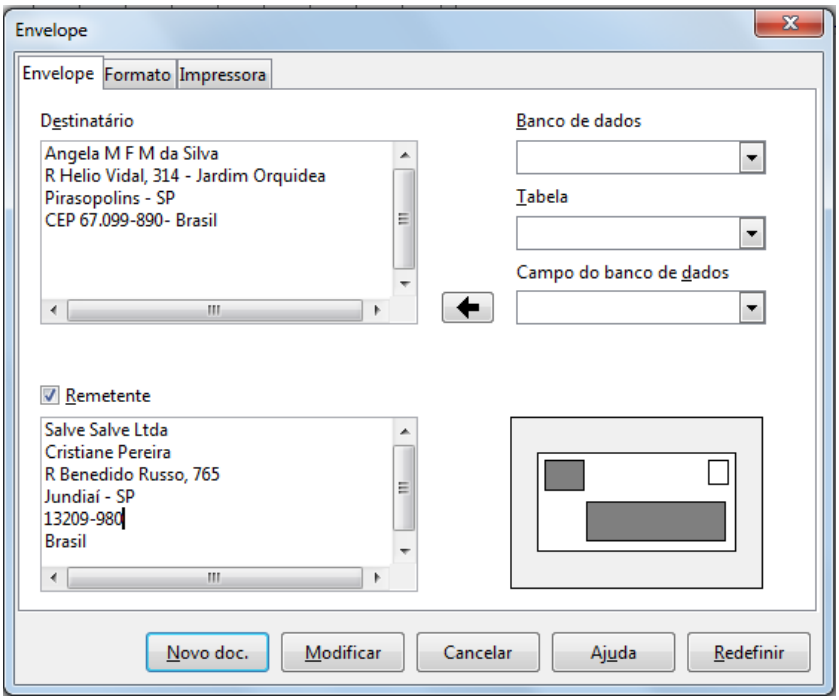

Figura 16. Tela de preenchimento para envelope (*Libre Office*)

Salve Salve Ltda
Cristiane Pereira
R Benedido Russo, 765
Jundiaí - SP
13209-980
Brasil

Angela M F M da Silva
R Helio Vidal, 314 - Jardim Orquidea
Pirasopolins – SP
CEP 67.099-890- Brasil

Figura 17. Envelope impresso (*Libre Office*)

- A função de personalização de barra de ferramentas não apresentou divergências (veja resultado na figura 18 e 19)

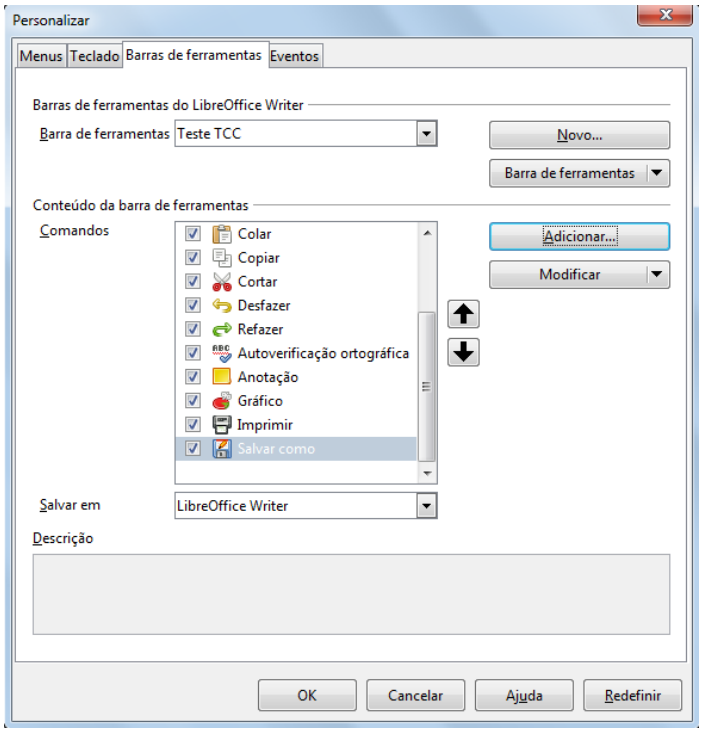

Figura 18. Criando barra de ferramentas (*Libre Office*)

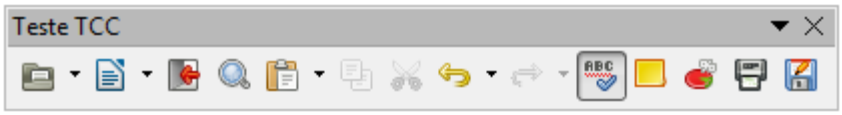

Figura 19. Barra de ferramentas criada (*Libre Office*)

- A função conversor de documentos não apresentou divergências (veja resultado na figura 20 e 21).

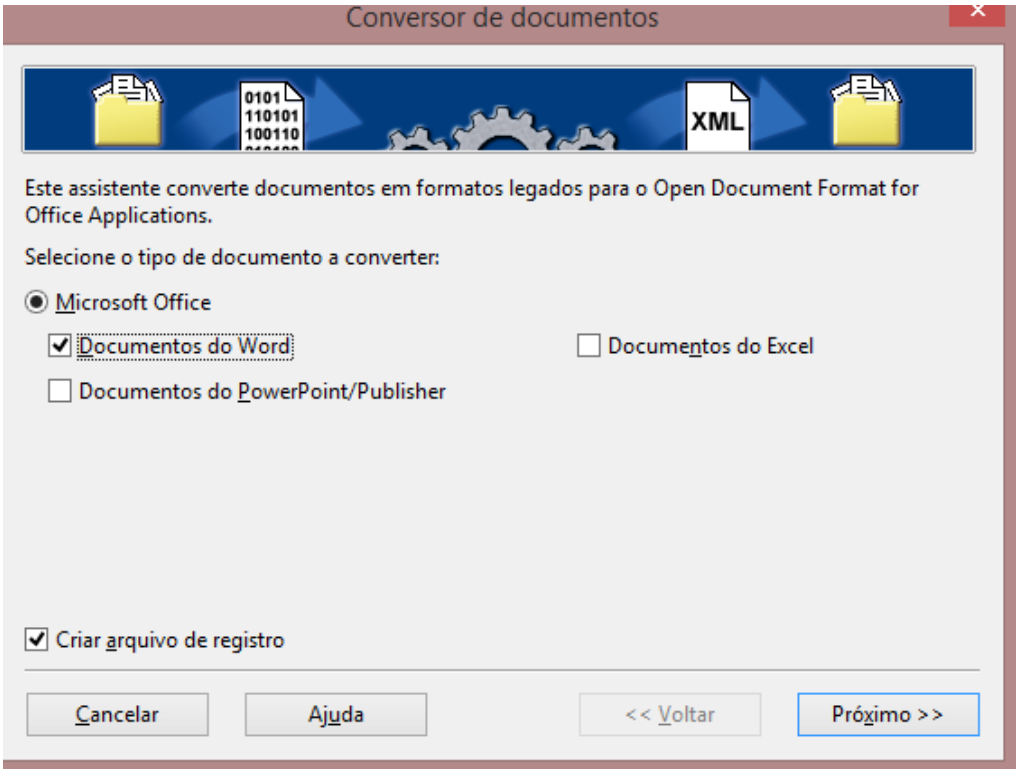

Figura 20. Tela para selecionar tio de documento (*Libre Office*)

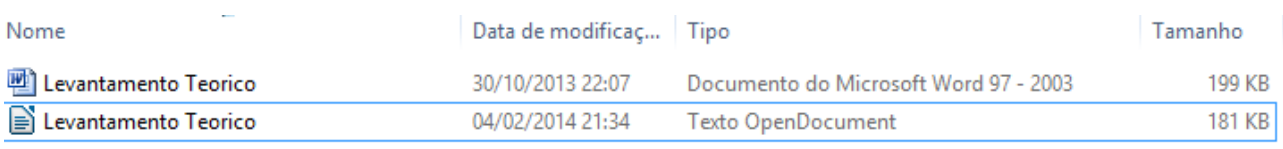

Figura 21. Tela com o tipo de documento alterado (*Libre Office*)

Todas as divergências encontradas foram documentadas no relatório de incidentes, conforme a figura 22 mostra.

LibreOffice The Document Foundation	***Qualidade*** **Relatório de Incidente**	**Libre Office**

Relatório de Incidente

Nome do Projeto: Libre Office - Write

Cont.	ID	Funcionalidade	Descrição	Roteiro	Resultado Esperado	Descrição do Erro
01	1	Cartão de Visita	Inserir dados na aba Pessoal	1- Acessar conteúdo cartão de visitas – aba Pessoal 2- Preencher os campos	1- Seja intuitivo sem duplicidade de informação 2- Fácil preenchimento	1- Existe dois lugares para inserir o nome 2- Não existe formatação e validação para CEP, telefone e e-mail 3- Campo pais e estado tem que ser escrito
01	1	Cartão de Visita	Inserir dados na aba Comercial	1- Acessar conteúdo cartão de visitas – aba comercial 2- Preencher os campos	1- Seja intuitivo sem duplicidade de informação 2- Fácil preenchimento	1- Não existe formatação e validação para CEP, telefone e e-mail 2- Campo pais e estado tem que ser escrito
01	1	Cartão de Visita	Inserir dados na aba Opções	1- Acessar conteúdo cartão de visitas	1- Fácil entendimento	1- Check "Sincronizar conteúdo" esta confuso, quando ele é usado para gerar o documento, as informações preenchidas não aparece
01	2	Cartão de Visita	Conferencia dos dados	1- Após preencher os dados, clicar em novo documento	1- Fácil visualização 2- Fácil retorno as informações anteriores para alteração	1- A visualização é muito pequena e escura 2- Para fazer alterações é necessário refazer todos os passos de acesso
01	2	Cartão de Visita	Impressão	1- Após visualizar clicar para imprimir	1- Facilidade de recortar os cartões 2- Economia de papel	1- Não existem linhas para facilitar o corte dos cartões 2- Existe muito desperdício de espaço na folha impressa
03	1	Abrir documento salvo no word	Tabelas	1- Abrir documentos que possuem tabelas	1- Abrir sem alterações	1- Tabela abre fora de formatação cortando parte da tabela

Figura 22. Relatório de teste preenchido (tabela desenvolvida para teste do *Libre Office*)

Utilizadas nos testes as técnicas de unidade e sistema, verifica-se cada função atende às hipóteses esperadas. O relatório de teste é preenchido informando a quantidade de validações realizadas, quantas foram bem sucedidas ou mal sucedidas, os casos de teste são preenchidos de acordo com as validações que são informadas no plano teste, o resultado é apresentado na figura 23.

LibreOffice The Document Foundation	Qualidade Relatório de Teste	Libre Office

Relatório de Teste

Nome do Projeto: Libre Office	
Data Inicio Teste: 16/10/2013	**Data fim Teste**: 04/02/2014

Números dos testes	
Casos de teste criados antes do teste:	10
Casos de teste criados durante o teste:	4
Casos de teste executados:	10
Casos de teste com sucesso:	6
Casos de teste com erro:	4

Percentual	
Casos de testes executados:	100%
Casos de teste executados com sucesso:	60%
Casos de testes com incidência de erro:	40%

Figura 23. Relatório de Teste (tabela desenvolvida para teste do *Libre Office*)

CONCLUSÕES

Uma das formas importantes para ter qualidade no *software* é realizar o processo de teste corretamente: criar um plano de teste a ser seguido, preencher os relatórios de incidente e relatórios de teste.

Uma documentação de plano de teste bem elaborada e especificada é indispensável, para que o testador possa executar todos os passos e que o mesmo consiga validar e reportar os erros de todas as definições estabelecidas junto ao cliente.

É importante também que o grupo de testes siga a metodologia e a cumpra para que a produtividade e qualidade aumente, além disso as equipes de teste e desenvolvimento devem ser distintas, "*pois conforme algumas experiências realizadas é muito difícil um desenvolvedor realizar*

as duas funções (desenvolver e testar), podendo estes encobrirem seus próprios enganos de modo que a eficácia dos testes seja praticamente nula" (Rocha,2011).

Utilizando corretamente as regras pré-definidas e respeitando a técnica de teste é possível atender a real necessidade do cliente e atingir a qualidade no *software*. No entanto, para que isso seja bem feito é necessário tempo, detalhe de extrema importância que a maioria dos projetos não possui.

Teste de software não é um processo barato, pelo contrário é um dos processos mais caros, porém em num *software* bem testado o custo de manutenção é menor, quanto antes o erro for descoberto no processo, menor o custo, o que leva a uma grande economia, redução no custo e a confiança e satisfação do cliente.

No estudo de caso dessa pesquisa, foi utilizado como técnica principal, o teste de sistema, no qual foram encontrados diversas falhas, defeitos e alguns pontos que podem ser melhorados.

A área de teste será uma área de grande evolução e disseminação no mercado, dificilmente em uma fábrica de *software* de grandes empresas não haverá a necessidade de testadores no meio de milhares de desenvolvedores.

A qualidade de *software* não deve ser vista como algo simples e de nenhum interesse, mas sim como sendo uma área de grande valor para o mercado e principalmente para que clientes possam estar criando sistemas ágeis, rápidos e com qualidade.

REFERÊNCIAS BIBLIOGRÁFICAS

AMARAL, Davi; GOMES, Renato; JESUS, Rodrigo Passos; ARAUJO, Tiago Marques; GOULART, Elias E. . Metodologias de Teste de Software. Disponível em <http://seer.uscs.edu.br/index.php/revista_informatica_aplicada/article/view/993>. Acessado em 03/11/2013.

BLANCO, Mariana Zanuzzio. Documentação de teste baseado na Norma IEE 829 – estudo de caso: "Sistema de apoio a tomada de decisão, 2012. Disponível em < http://revistatis.dc.ufscar.br/index.php/revista/article/view/18 >. Acessado em 03/11/2013.

CAMPOS, Fabio Martinho Campos. TMap Next Test Management Approach – As 4 Essências do TMap Next. Disponível em: < http://www.linhadecodigo.com.br/artigo/3221/tmap-next-test-

management-approach-as-4-essencias-do-tmap-next-parte-3.aspx >. Acessado em 29/09/2013.

CAROLINE, Anne. Grupo Testadores de Software - Modelo de Roteiro de Testes. Disponível em < http://gtsw.blogspot.com.br/2009/08/modelo-de-roteiro-de-testes.html>. Acessado em 03/11/2013.

DIAS, Arilo Claudio Dias, Neto. *Introdução a Teste de Software.* Engenharia de Software Magazine. Edição 01. Editora SQL Magazine, 2007.

DIAS, Arilo Claudio Dias, Neto. *Planejamento de teste a partir de casos de uso.* Engenharia de Software Magazine. Edição 06. Editora SQL Magazine, 2008.

IEEE Standard for Software Test Documentation. Disponível em <http://faculty.ksu.edu.sa/mohamedbatouche/SWE%20434/IEEE%20Std%20829%20-%201998.pdf>. Acessado em: 12/11/2013.

HOHN, Erila Nina; MALDONADO, Jose Carlos; FABBRI, Sandra C.P.F. Um estudo de caso do arcabouço de conhecimento e melhoria de processo de teste – KITest.
Disponível em < http://www.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_RT_366.pdf> .
Acessado em 03/11/2013.

JMETER, The Apache Software Foundation. Disponível em < http://jmeter.apache.org/>.
Acessado em 10/11/2013.

JUNIT. Disponível em < http://junit.org/>. Acessado em 10/11/2013.

KOSCIANSKI, André; SOARES, Michel dos Santos. *Qualidade de Software*. Ed. 2. Editora Novatec, 2006.

LIBRE OFFICE. The Document Foundation. Disponível em: <http://www.libreoffice.org/>. Acessado em 15/10/2013.

MELLO, Leandro Cicero da Silva. Levantamentos de requisitos. Disponível em: < http://www.ice.edu.br/TNX/storage/webdisco/2011/03/11/outros/7799b4179c3e64b7f8ab3ec4cfdbea22.pdf>. Acessado em 05/12/2013.

OLIVEIRA. Sandro R. Bezerra. Conceitos Fundamentais de Qualidade de Software. Disponível em <http://www.ufpa.br/srbo/Disciplinas/CBCC_CBSI_Mestrado_Qualidade/Aulas/Aula02.pdf>. Acessado em: 14/11/2013.

PAULA, Wilson de Pádua, Filho. *Engenharia de Software: Fundamentos, Métodos e Padrões*. Disponível em: http://aulasprof.6te.net/Arquivos_Aulas/07-Proces_Desen_Soft/Livro_Eng_Soft_Fund_Met_Padroes.pdf, recuperado em 21/08/2013.

PAULA, Wilson de Padua, Filho. *Engenharia de Software: Fundamentos, Métodos e Padrões*. Ed. 3. Editora LTC, 2009.

PRESSMAN, Roger S. *Engenharia de Software*. Ed.6. Editora McGraw – Hill Interamericana, 2006.

PRIMÃO, Aline Pacheco; RIBEIRO, Patric da Silva; KREUTZ, Diego Luis. Estudo de Caso: Técnicas de Teste como parte do Ciclo de Desenvolvimento de Software. Disponível em < http://www.sirc.unifra.br/artigos2010/4.pdf>. Acessado em 03/11/2013.

RIBEIRO, Camilo. Técnicas de Teste no Ciclo de Desenvolvimento de Software,2010. Disponível em <http://www.slideshare.net/camiloribeiro/tcnicas-de-teste>. Acessado em: 12/11/2013.

RIOS, Emerson. Gerencia de Projeto de Testes Segundo o Modelo do PMI, 2003. Disponível em: <http://www.bfpug.com.br/islig-rio/Downloads/Gerencia_Projeto_Testes_PMI.pdf >. Acessado em 14/04/2014.

ROCHA, Camila. Estudo de caso da qualidade de software na Metodologia V-Model e sua interação com metodologias ágeis, 2011. Disponível em: <http://www.fatecsp.br/dti/tcc/tcc0028.pdf>. Acessado em 12/11/2013.

SELENIUM HQ, Browser Automation. Disponível em < http://www.seleniumhq.org/>. Acessado em 10/11/2013.

SILVA, Fernando Rodrigues. Testes de software – Níveis de testes. Disponível em <http://www.devmedia.com.br/testes-de-software-niveis-de-testes/22282>. Acessado em 12/11/2013.

SOMMERVILLE, Ian. *Engenharia de Software*. Ed.6. Editora Pearson Education Limited, 2003.

SOMMERVILLE, Ian. *Engenharia de Software*. Ed.8. Editora Pearson Education Limited, 2007.

SOUZA, Eduardo Freitas. Introdução a Automação de Teste de Software. Centro de estudos avançados Intellecta. Disponível em: < http://www.slideshare.net/eduardofsouza9/automacao-de-testes-de-softwares>. Acessado em 22/09/2013.