UNIVERSIDADE FEDERAL DE OURO PRETO
ESCOLA DE MINAS
ENGENHARIA DE PRODUÇÃO

MARCO TÚLIO DA SILVA GOMES

TESTES E QUALIDADE EM *SOFTWARES*: UM ESTUDO DE CASO EM UM SISTEMA DE PLANEJAMENTO E CONTROLE DE PRODUÇÃO

OURO PRETO - MG
2012

MARCO TÚLIO DA SILVA GOMES
marcotuliosg@gmail.com

TESTES E QUALIDADE EM *SOFTWARES*: UM ESTUDO DE CASO EM UM SISTEMA DE PLANEJAMENTO E CONTROLE DE PRODUÇÃO

Monografia submetida à apreciação da banca examinadora do curso de Engenharia de Produção da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Engenheiro de Produção.

Orientador: Prof. André Luís Silva

OURO PRETO - MG
2012

Marco Túlio da Silva Gomes

Monografia julgada e aprovada, em 1º de outubro de 2012 como parte dos requisitos necessários para a obtenção de grau de Engenheiro de Produção pela Universidade Federal de Ouro Preto.

BANCA EXAMINADORA

_____

Prof. André Luís Silva
Universidade Federal de Ouro Preto - DEPRO
Professor orientador

_____

Prof.ª Silvana Prata Camargos
Universidade Federal de Ouro Preto - DEPRO
Examinadora

_____

Prof.ª Zirlene Alves da Silva Santos
Universidade Federal de Ouro Preto - DEPRO
Examinadora

Aos meus pais e ao meu irmão.

AGRADECIMENTOS

A Deus, que é Vida;

Aos meus pais Edison e Sebastiana, meu irmão, Sérgio, e demais familiares, que estão e estarão sempre presentes em minha vida;

Aos amigos, de perto e de longe, que tornam os dias mais divertidos e incentivaram a conclusão desta etapa. Em especial, Tamara Oliveira e Cassius Ribeiro, exemplos profissionais, e Antero Sobreira pela ajuda na revisão do texto;

Aos colegas da Universidade Federal de Ouro Preto, de sala ou de trabalho, que muito contribuíram para meu crescimento;

Aos professores da Universidade Federal de Ouro Preto, que se dedicaram na missão de ensinar, em especial, professor André Silva, pela orientação e disponibilidade;

À empresa EFX Tecnologia, por disponibilizar e autorizar o uso do *software* para este estudo.

Muito obrigado a todos!

[...] a tecnologia é, sim, uma condição necessária, mas não é suficiente. Para termos os benefícios quando instalamos a nova tecnologia também precisamos mudar as regras que reconhecem a existência da limitação. Puro bom senso.

Eliyahu M. Goldratt

RESUMO

As novas tecnologias mudaram a forma com que as organizações lidam com seus sistemas de informação, levando ao aumento da complexidade e novos desafios no desenvolvimento dos *softwares*. Este trabalho teve por objetivo analisar a importância do uso de testes em *softwares* no contexto dos sistemas de informações dedicados à manufatura, com vistas à melhoria da qualidade, utilizando para tanto um estudo caso em um sistema de planejamento e controle de produção. O estudo seguiu técnicas de teste encontradas na literatura, em especial o particionamento de equivalências, em busca de encontrar falhas no *software*. Casos de testes foram planejados e conduzidos para sete interfaces do *software*, nos quais foram encontradas e descritas as situações de conformidade, conformidade parcial e não-conformidade com os aspectos relacionados à funcionalidade, confiabilidade e usabilidade do sistema. Discutiu-se, pela análise do caso, a relevância dos testes de *software* relacionada a qualidade no contexto das tecnologias de informação.

Palavras-chave: Engenharia de *software*. Testes em *software*. Qualidade. Sistemas de Planejamento e Controle de Produção.

ABSTRACT

New technologies have changed the way organizations handle their information systems, leading to increased complexity and new challenges in the development of software. This study aimed to analyze the importance of using software testing in the context of information systems dedicated to manufacturing, with a view to improving quality, using both a case study on a system of planning and control of production. The study followed testing techniques found in the literature, particularly the partitioning of equivalence, seeking to find flaws in the software. Test cases were designed and conducted for seven interfaces of software, in which were found and described the situations of compliance, partial compliance and non-compliance with aspects related to the functionality, reliability and usability of the system. It has been argued, for examining the case, the relevance of software testing related to quality in the context of information technology.

Key-words: *Software* Engineering. *Software* Testing.Quality.Production Planning & Control Systems.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CBIS	*Computer-BasedInformation System*
CMM	*Capability Maturity Model*
DSS	*Decision Support Systems*
ERP	*Enterprise Resource Planning*
GIS	*Geographical Information Systems*
GQS	Garantia da Qualidade em *Software*
IA	Inteligência Artificial
ISO	*International Organization for Standardization*
MIS	*Management Information Systems*
MPS	*Master Production Schedule*
MRP	*Manufacturing Resource Planning*
PCP	Planejamento e Controle da Produção
PMP	Plano Mestre de Produção
SAD	Sistemas de Apoio a Decisão
SE	Sistemas Especialistas
SI	Sistema de Informação
SIG	Sistemas de Informações Gerenciais
SPT	Sistemas de Processamento de Transações
SQA	*Software Quality Assurance*
TI	Tecnologia de Informação
TPS	*Transaction Processing Systems*

SUMÁRIO

1 INTRODUÇÃO

Um novo ambiente se estabeleceu na transição entre os séculos XX e XXI em vários contextos (econômico, produtivo, organizacional, social) e apresenta novos desafios aos indivíduos, empresas, governos e à sociedade como um todo. Segundo Santos (2004) "o momento presente é um ponto de inflexão entre a era da certeza e do raciocínio lógico (era industrial), e uma nova era caracterizada pela imprecisão, pelo futuro desconhecido e pelo número infinito de possibilidades que se apresentam (era do conhecimento)".

De acordo com Lastres (2000), nesta nova conjuntura existe a absoluta relevância dos conhecimentos científicos e tecnológicos desenvolvidos e utilizados. Mesmo havendo outras considerações envolvendo ciência e tecnologia, de modo geral, pode-se entender a tecnologia como sendo a aplicação prática de ciências fundamentais, a fim de alcançar objetivos determinados.

Dentro da conceituação de tecnologia, Stonebraker e Leong (1994 *apud* PEDROSO, 1999) fazem uma subdivisão em cinco categorias distintas: tecnologia de processos, tecnologia de materiais, tecnologia de produtos e serviços, tecnologia de informação e tecnologia de gestão. Considerando a atuação nestas diversas frentes, cada categoria traz contribuições particulares para o desenvolvimento econômico e social.

Para Santos (2004), o conhecimento é mais importante do que qualquer outro fator de produção da era industrial sejam recursos naturais, trabalho humano, mão-de-obra ou mesmo capital. Beal (2008) afirma que a informação já é considerada o principal insumo e, em muitos casos, o principal produto das organizações. Lastres (2000) também destaca que os elementos fundamentais identificados da dinâmica da chamada nova ordem mundial são a informação, o conhecimento e as tecnologias de informação.

De fato, observa-se nas organizações uma mudança de prioridades, com a maior valorização das atividades intelectuais que envolvem maior conhecimento e tecnologias associadas. Segundo Sommerville (2007) praticamente todos os países dependem de sistemas complexos baseados em computadores, sendo que a manufatura e distribuição industriais assim como os sistemas financeiros estão completamente automatizados.

Enfocando a Tecnologia da Informação (TI), Laurindo e Rotondaro (2006) a veem como a fonte de criação de novas estratégias de negócio, de novas estruturas organizacionais e de novas formas de relacionamento entre empresas e entre empresas e seus consumidores.

Reforçando este pensamento, Kalakota e Robinson (2002) afirmam que os administradores precisam se concentrar nas novas tendências (tecnologia, hábitos de compra do consumidor, serviços e processos, organizações e tecnologia empresarial) que estabelecem as estratégias lucrativas de negócios no futuro.

As TI possibilitam o processamento do conhecimento, no qual a informação deixa de ser tratada como uma pequena burocracia preestabelecida no interior de uma máquina para ser trabalhada de forma livre e aberta dentro de uma empresa (GOMES e RIBEIRO, 2004, p. 154).

No mesmo raciocínio, Foina (2006, p.19) afirma que as tecnologias usadas para o tratamento da informação na empresa, formam um arcabouço tecnológico consistente e dimensionado para suportar os sistemas, normas e procedimentos que efetivamente tratam as informações que fluem por essa tecnologia.

Laurindo e Rotondaro (2006, p. 68) afirmam que há uma grande expectativa acerca das aplicações da Tecnologia da Informação, que possibilitam novas alternativas de estratégias de negócios e novas possibilidades para as organizações, como é o caso do "e-business".

Kalakota e Robinson (2002, p. 24) conceituam o *e-business* como "uma estratégia global de redefinição dos antigos modelos de negócios, com o auxílio de tecnologia, para maximizar o valor do cliente e os lucros".

Internamente às organizações, a TI permite que as diversas áreas e processos das empresas sejam interligados e coordenados. Permite também que estes processos sejam viabilizados ou mesmo reinventados (LAURINDO e ROTONDARO, 2006, p. 69).

No âmbito da gestão da cadeia de suprimentos, Gomes e Ribeiro (2004) afirmam que também existem tecnologias e sistemas de informações que permitem um intercâmbio entre fornecedores e clientes, facilitando a transferência dos dados de reposição dos estoques e da demanda do ponto-de-vendas até o fornecedor, não só de produtos, como também dos componentes e materiais.

A influência nos processos descritos anteriormente do advento de um novo padrão sócio-técnico baseado nas tecnologias da informação, e sua difusão (cada vez mais intensa através da economia e sociedade mundiais, embora de forma irregular e desigual), exigindo

restruturações econômicas, sociais e políticas (LASTRES, 2000). Deste modo, o envolvimento dos profissionais de todas as áreas, incluindo a engenharia, com as novas tecnologias, torna-se, ao mesmo tempo, uma condição para manterem-se no mercado e para garantir a competitividade da organização neste mercado.

Contudo, há também um grande questionamento sobre os reais ganhos advindos dos investimentos em TI. Goldratt *et al* (2000) afirmam que "a tecnologia é, sim, uma condição necessária, mas não é suficiente". Isso acontece porque muitas vezes o que se esperava da tecnologia não se reflete em números positivos na empresa.

Laurindo e Rotondaro (2006) exemplificam situações problemáticas com que as empresas frequentemente se deparam, quando não identificam, analisam e melhoram seus processos antes da aplicação dos sistemas de informação: a ilusão de que a simples implementação do sistema automatizará o processo de integração, e a criação de conflitos pessoais e setoriais na organização e inflexibilidade de cada área organizacional em detrimento de uma maior integração.

Sommerville (2007, p.24) também afirma que os fatores humanos, sociais e organizacionais são frequentemente críticos para determinar se o sistema atende seus objetivos com sucesso ou não.

Apesar da relevância desta questão para o desempenho esperado dos *softwares* em ambientes empresariais, discussões sobre onde se deve aplicar maiores investimentos para melhoria, seja durante o desenvolvimento ou na realização paralela de testes, ainda estão presentes.

Este estudo enfatizou a importância dos testes em *softwares* nos sistemas de apoio ao planejamento e controle da produção, a fim de obter alguns benefícios e limitações presentes na aplicação de técnicas deste tipo.

1.1 FORMULAÇÃO DO PROBLEMA

É notável como os *softwares* estão inseridos em muitos ramos de atividade humana, proporcionando significativos ganhos de produtividade. Stair (1998, p. 17) afirma que os sistemas de informação são usados em todas as áreas funcionais (divisões de operação) dos negócios e também em quase todas as indústrias ou ramos. No entanto, na medida em que isto acontece, eleva-se também o grau de dependência por parte das empresas e das pessoas pelo bom desempenho destes sistemas.

Pressman (1995, p. 22) relatou o surgimento de uma crise no horizonte do desenvolvimento de *software*, com problemas que poderiam ser caracterizados a partir de uma série de perspectivas diferentes: estimativas de prazo e de custo frequentemente são imprecisas, a produtividade das pessoas da área de *software* não ter acompanhado a demanda por seus serviços, e a qualidade de *software* às vezes ficando abaixo do desejado.

Ainda segundo o autor, a insatisfação do cliente com o sistema "concluído" ocorre muito frequentemente. Os projetos de desenvolvimento de *software* normalmente são levados a efeito apenas com um vago indício das exigências do cliente, e a comunicação entre o cliente e o desenvolvedor de *software* é muito fraca.

Falhas em *softwares* podem provocar prejuízos em termos de tempo, financeiros ou materiais. Charette (2005, p. 43) afirma que as falhas deste tipo acontecem em todos os países, grandes e pequenas empresas, em organizações comerciais, sem fins lucrativos e governamentais. O autor ainda revela que os custos de negócios e sociais desses fracassos estão em torno dos bilhões de dólares, por ano. Quanto mais tardiamente forem detectados os erros no *software*, maiores serão os custos envolvidos para correção da falha.

Alguns problemas provocados por falhas em *softwares* ganham destaque na mídia e causaram impactos muito negativos para a imagem das empresas envolvidas. Inúmeros outros casos de erros em *softwares* ocorrem cotidianamente em pequenas, médias e grandes organizações, sem que seja dada a devida atenção, e tomadas as medidas preventivas para que isso não ocorra.

O desenvolvimento de um *software* deve, neste contexto, portanto, garantir a máxima confiabilidade possível. No mesmo sentido, uma organização ao escolher implantar um *software* para auxiliá-la no gerenciamento de suas informações, deve se precaver quanto à adequação do mesmo aos requisitos funcionais por ela exigidos.

Os métodos, procedimentos e ferramentas da engenharia de *software* têm sido adotados sucessivamente para a melhoria dos *softwares*, evitando variados transtornos e prejuízos oriundos das falhas apresentadas pelo sistema após sua implantação.

Considerando o contexto apresentado, propôs-se para nortear este estudo a seguinte questão problema: **Qual a contribuição da aplicação de métodos de teste, com foco no particionamento de equivalências, em um sistema informatizado de planejamento e controle de produção?**

1.2 JUSTIFICATIVA

A Engenharia de Produção envolve um conjunto de subáreas do conhecimento dentre as quais a Engenharia Organizacional, definida pela Associação Brasileira de Engenharia de Produção (ABEPRO) como sendo o "conjunto de conhecimentos relacionados à gestão das organizações, englobando em seus tópicos o planejamento estratégico e operacional, as estratégias de produção, a gestão empreendedora, a propriedade intelectual, a avaliação de desempenho organizacional, os sistemas de informação e sua gestão e os arranjos produtivos" (ABEPRO, 2008). Dentro deste contexto, os sistemas de informação devem constituir importante ponto de atenção para as organizações no sentido de alcançar com sucesso suas metas, de acordo com a estratégia adotada.

Segundo Barros Filho & Tubino (1998), o foco do assunto normalmente encontrado em trabalhos na área de Planejamento e Controle da Produção (PCP), recai quase sempre em sistemas computacionais (*softwares*), sua utilização e aplicações. Os autores ainda afirmam que é evidente a pouca ênfase dada aos pré-requisitos para a operacionalização adequada destes sistemas, e sistematização do ambiente fabril para receber tais ferramentas computacionais, ou quem sabe, ainda que inicialmente, dispensá-las.

Os sistemas PCP, sejam referentes a bens ou a serviços, devem funcionar sem a incidência de falhas durante sua execução, a fim de evitar que todo o planejamento e controle feito até então seja comprometido, parcial ou totalmente, o que pode gerar transtornos e prejuízos à(s) empresa(s) que o utilizam o *software*.

Assim como para qualquer outro tipo de produto, bem ou serviço, os *softwares* devem ter garantida a qualidade do produto final. Segundo Pressman (2006) revisões e outras atividades de SQA (*Software Quality Assurance*) podem e efetivamente descobrem erros, mas elas não são suficientes. O autor ainda afirma que para encontrar o maior número possível de erros, testes devem ser conduzidos sistematicamente e casos de testes devem ser projetados usando técnicas disciplinadas.

Para Magela (2006), os testes são uma necessidade inerente a qualquer engenharia. Ainda assim, segundo Pressman (2006) só recentemente começou-se a entender a importância dos testes de *software* sistemáticos e tecnicamente completos.

A relevância dos testes em *software* reside, de modo geral, na verificação e validação da adequação de um determinado sistema em uma dada organização. Do ponto de vista mais

pragmático, permitem a antecipação de falhas geradoras de custos (muitas vezes elevados) relacionados, por exemplo, ao abandono do projeto de implantação, retrabalhos e aos erros. Segundo Pressman (2006), em resposta a esses problemas, estão sendo adotadas práticas de engenharia em toda a indústria de *softwares*.

Charette (2005, p. 43) declara que a falha de *software* é na maior parte das vezes previsível e evitável, e que infelizmente, a maioria das organizações não vê a prevenção de falhas como uma questão urgente, apesar dos riscos. O autor coloca ainda que entender por que essa atitude persiste não é apenas um exercício acadêmico, mas tem enormes implicações para os negócios e a sociedade.

O benefício óbvio das revisões técnicas formais é a descoberta precoce dos defeitos de *software*, de forma que cada defeito possa ser corrigido antes do passo seguinte do processo de engenharia de *software*. Ao detectar e suprimir uma grande quantidade desses erros, o processo de revisão reduz substancialmente o custo dos passos subsequentes nas fases de desenvolvimento e manutenção (PRESSMAN, 2006).

A escolha pelo teste de um sistema de controle de produção desenvolvido para indústrias de pequeno porte, a fim de verificar de maneira mais efetiva, a contribuição de um método de teste em *software*, foi feita considerando a disponibilidade restrita de tempo e recursos, para um estudo mais amplo. Por sua vez, a escolha do método de teste para este estudo levou em consideração a restrição de acesso ao código-fonte de *softwares* de planejamento e controle da produção já existente no mercado. A alternativa encontrada foi a de teste funcional com a função de validar os requisitos funcionais do *software*, sem se prender ao funcionamento interno de um programa.

1.3 OBJETIVOS

1.3.1 Objetivo geral

Analisar a importância do uso de testes em *softwares* no contexto dos sistemas de informações dedicados à manufatura.

1.3.2 Objetivos específicos

Com o intuito de alcançar o objetivo geral delimitaram-se metas mais específicas dentro do trabalho. São elas:

- Realizar um levantamento na literatura especializada dos conceitos relacionados a teste de *software* bem como dos tipos de testes mais comuns, sua aplicabilidade, e benefícios que podem proporcionar às organizações;
- Realizar um levantamento na literatura referente à técnica/método de teste empregado no projeto.
- Definir, especificar e realizar um conjunto de testes, segundo a técnica/método de particionamento de equivalências em um sistema de planejamento e controle de produção.
    - Especificar casos de teste
    - Executar casos de teste
    - Analisar casos de teste.

1.4 ESTRUTURA DO TRABALHO

Este estudo esta estruturado em cinco capítulos, onde:

O capítulo 1 apresenta a introdução, a formulação do problema, a justificativa do trabalho, os objetivos e a estrutura. O capítulo 2 contém uma revisão bibliográfica da literatura sobre tecnologia e sistemas de informação, engenharia de *software* e tipos de testes de *software*. O capítulo 3 consiste em caracterizar a metodologia da pesquisa. O capítulo 4 mostra as análises e discussões relativas aos assuntos investigados. O capítulo 5 apresenta a conclusão do estudo, suas limitações e recomendações para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O objetivo deste capítulo é apresentar um referencial teórico com conceitos, classificações e outras considerações pertinentes sobre os principais temas que fundamentam o estudo: tecnologia e sistemas de informação, engenharia e qualidade de *software* e os testes de *software*, testes caixa-preta e particionamento de equivalências.

2.1 TECNOLOGIA DA INFORMAÇÃO E SISTEMAS DE INFORMAÇÃO

Os conceitos de Tecnologia da Informação (TI) e Sistemas de Informação (SI) têm ganhado significativa notoriedade, e sua importância praticamente unânime, à medida que as organizações passaram a perceber o valor da informação. Para Stair (1998, p. 5), "o valor da informação está diretamente ligado à maneira como ela ajuda os tomadores de decisões a atingirem as metas da organização".

Vários autores apresentam diferentes abordagens sobre as terminologias de TI e SI, podendo ser mais ou menos abrangentes.

Rezende (2005, p. 32) conceitua a TI como "recursos tecnológicos e computacionais para guarda de dados e geração de informação". O autor ainda afirma que a TI está fundamentada nos seguintes componentes: *hardware* e seus dispositivos periféricos; *software* e seus recursos; sistemas de telecomunicações; gestão de dados e informações.

Já para Foina (2006, p.17) a TI é um "conjunto de métodos e ferramentas, mecanizados ou não, que se propõe a garantir a qualidade e pontualidade das informações dentro da malha empresarial".

Por outro lado, as definições de SI, em geral, refletem a ideia de sistema voltado ao tratamento de informações composto por um conjunto de elementos interdependentes formando um todo unitário para se atingir objetivos. Para Batista (2006, p.19) um SI pode ser definido como "todo e qualquer sistema que possui dados ou informações de entrada que tenham por fim gerar informações de saída para suprir determinadas necessidades."

De modo semelhante, Stair (1998, p. 11) define SI como uma série de elementos ou componentes inter-relacionados que coletam (entrada), manipulam e armazenam (processo), disseminam (saída) os dados e informações e fornecem um mecanismo de *feedback*. Esta definição é ilustrada pela Figura 1, a seguir:

Figura 1 – Componentes de um sistema de informação

Fonte: STAIR, 1998, p. 11

Apesar da associação, muitas vezes imediata, feita entre SI e computadores, Stair (1998) afirma que os sistemas de informação também podem ser manuais, como sistemas de

informação manuais de análise de ações, e define então um conceito próprio para os sistemas de informação computadorizados da seguinte maneira:

> O sistema de informação baseado em computador (CBIS - *Computer-basedinformation system*) é composto pelo hardware, *software*, banco de dados, telecomunicações, pessoas e procedimentos, que estão configurados para coletar, manipular, armazenar e processar dados em informação. (STAIR, 1998, p. 13)

Stair (1998) assim como Rosini e Palmisano (2003) propõem classificações semelhantes para os sistemas de informação, que possuem relação com os diferentes níveis/dimensões de decisão existentes no ambiente empresarial:

i. Sistemas de processamento de transações (SPT) ou *Transaction Processing Systems (TPS*): conjunto organizado de pessoas, procedimentos, bancos de dados e dispositivos usados para processar transações de negócio. O processamento inclui a coleta de dados de entrada, a realização de cálculos, o armazenamento da informação no banco de dados, e/ou produção de vários documentos e relatórios. Entre as aplicações comuns dos SPT estão o processamento de pedidos, fatura, controle de estoque, contas a pagar e a receber, compras, recebimento e expedição, folha de pagamento e contabilidade em geral (Stair, 1998). Os SPT atendem as necessidades do nível operacional da organização (Rosini e Palmisano, 2003).

ii. Sistemas de informações gerenciais (SIG) ou *Management Information Systems* (MIS): é um agrupamento organizado de pessoas, procedimentos, bancos de dados e dispositivos usados para oferecer informações de rotina aos administradores e tomadores de decisões (Stair, 1998). Os SIG tipicamente fornecem relatórios pré-programados gerados com dados e informações do sistema de processamento de transações. São orientados quase exclusivamente para eventos internos e servem como base para as funções de planejamento e controle e tomada de decisão em nível gerencial (Rosini e Palmisano, 2003).

iii. Sistemas de Apoio a Decisão (SAD) ou *Decision Support Systems* (DSS): é um grupo organizado de pessoas, procedimentos, bancos de dados e dispositivos usados para dar apoio à tomada de decisões referentes a problemas específicos (Stair, 1998). Desenvolvidos para atender as necessidades do nível estratégico da organização, utilizando as informações internas geradas pelos outros sistemas e

oferecem ainda a informação das fontes externas, tais como nível de preços dos concorrentes (Rosini e Palmisano, 2003).

iv. Sistemas Especialistas (SE) ou *Expert Systems*: é um agrupamento organizado de pessoas procedimentos, banco de dados e dispositivos usados para gerar um parecer especializado ou sugerir uma decisão em uma área ou disciplina.

Uma classificação diferente para os SI é proposta por Batista (2006), de acordo com a sua forma de utilização e o tipo de retorno dado ao processo de decisões:

i. Sistemas empresariais básicos: são aqueles utilizados para realizar tarefas rotineiras da empresa, essenciais para conduzir a organização.

ii. Sistemas de automação de escritório: toda e qualquer tecnologia de informação que possui como objetivo principal aumentar a produtividade dos trabalhadores que manipulam informações de escritório.

iii. Sistemas de informação gerencial (SIG): oferecem um conjunto de relatórios resumidos sobre o desempenho da empresa, os quais são utilizados para a realimentação do planejamento operacional. Permitem a utilização de relatórios, consultas e visualização de dados, que são ferramentas incorporadas a algum tipo de gerenciador de banco de dados.

iv. Sistemas de suporte à decisão (SSD): sistemas que possuem interatividade com as ações do usuário, oferecendo dados e modelos para a solução de problemas semiestruturados e focando a tomada de decisões.

v. Sistemas de suporte executivo (SSE): sistemas que dão suporte ao desenvolvimento do planejamento estratégico da empresa e ajudam a definir os objetivos a serem estabelecidos. Focalizam a alta administração da empresa.

vi. Sistemas especialistas: sistemas ligados ao campo da inteligência artificial (IA), que utiliza o computador para assistir, ou mesmo substituir, os tomadores de decisão.

vii. Sistemas de informação geográfica (GIS): é o conjunto de programas, equipamentos, metodologias, dados e pessoas (usuários) perfeitamente integrados, de maneira a tornar possível a coleta, o armazenamento, o processamento e a

análise de dados georreferenciados, bem como produzir informações a partir dos processamentos dos dados obtidos

A Tabela 1 sintetiza as classificações propostas pelos autores pesquisados para os sistemas de informação empresariais.

Tabela 1 – Classificações para os sistemas de informação empresariais

Stair (1998)	**Rosini e Palmisano (2003)**	**Batista (2006)**
Sistemas de processamento de transações (SPT)	Sistemas de informação transacionais/operacionais (SIT)	Sistemas empresariais básicos
		Sistemas de automação de escritório
Sistemas de informações gerenciais (SIG)	Sistemas de informação gerenciais (SIG)	Sistemas de informação gerencial (SIG)
Sistemas de apoio a decisão (SAD)	Sistemas de apoio à decisão (SAD)	Sistemas de suporte a decisão (SDD)
		Sistemas de suporte executivo (SSE)
Sistemas especialistas (SE)	Sistemas de informação especialistas, sistemas de automação (SE, SA)	Sistemas especialistas
-	-	Sistemas de informação geográfica (GIS)

Fonte: Elaborada pelo autor, a partir do elenco de autores citados

Stair (1998, p. 17) cita uma série de benefícios esperados dos SI: valor agregado aos produtos (bens e serviços), maior segurança, melhor serviço, vantagens competitivas, menos erros, maior precisão, produtos de melhor qualidade, aperfeiçoamento no sistema de saúde, aperfeiçoamento das comunicações, maior eficiência, maior produtividade, administração mais eficiente, mais oportunidades, carga de trabalho reduzida, custos reduzidos, tomadas de decisões financeiras e gerenciais superiores, maior e melhor controle sobre as operações.

Prates e Ospina (2004) afirmam que os investimentos em TI visam alcançar um ou mais dos três objetivos operacionais independentes: aumentar a continuidade (integração funcional, automação intensificada, resposta rápida), melhorar o controle (precisão, acuidade, previsibilidade, consistência, certeza), e proporcionar maior compreensão das funções produtivas (visibilidade, análise, síntese).

Em contrapartida, a adoção de uma TI implica em custos para a organização. Segundo Giurliani (1999 *apud* Prates e Ospina, 2004) os custos mais facilmente mensuráveis seriam aqueles relacionados a *hardware* e *software* (despesas com compra e/ou *leasing* de equipamentos, upgrades, atualizações); gerenciamento (redes, sistemas, banco de dados); suporte (*helpdesk*, treinamento, viagens, manutenção); desenvolvimento (aplicações, testes e

documentação, customização emanutenção); e taxas de comunicação (aluguel de linhas de comunicação de dados, taxas de acesso a servidores).

Um ponto de equilíbrio entre os custos de implantação/manutenção de uma TI e os benefícios advindos desta TI deve ser analisado e buscado pelas empresas, a fim de que o resultado final do processo seja o mais positivo possível em termos organizacional e financeiro.

2.2 ENGENHARIA DE *SOFTWARE*

Segundo Rezende (2005, p. 1) a Engenharia de *Software* caminha em paralelo com os Sistemas de Informação, para auxiliar as organizações a tomarem decisões sobre o foco de seu negócio empresarial ou de sua atividade pública.

Para Bauer (1969 *apud* Pressman 2006) engenharia de *software* é a "criação e a utilização de sólidos princípios de engenharia a fim de obter *softwares* econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais". Já na definição do *Institute of Electrical and Electronics Engineers Standards* - IEEE (1993 *apud* Pressman 2006), engenharia de *software* é "aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção do *software*; isto é, a aplicação da engenharia ao *software"*. E de acordo com Sommerville (2007, p.5) a engenharia de *software* é "uma disciplina de engenharia relacionada com todos os aspectos da produção de *software*, desde os estágios iniciais de especificação do sistema até sua manutenção, depois que este entrar em operação".

Vários autores relacionam o surgimento e crescimento da engenharia de *software* com a preocupação crescente pela qualidade de *software*, num cenário cada vez mais complexo.

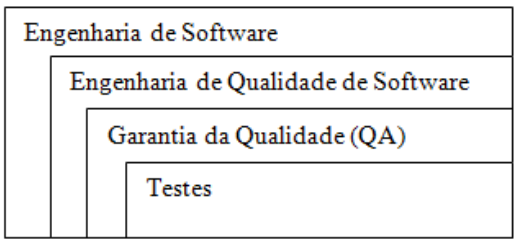
> Qualquer abordagem de engenharia deve se apoiar num compromisso organizacional com a qualidade. Gestão da Qualidade Total, Seis Sigmas (*Six Sigma*) e filosofias análogas levam à cultura de um processo contínuo de aperfeiçoamento, e é essa cultura que, em última análise, leva ao desenvolvimento de abordagens cada vez mais efetivas para a engenharia de *software*. A base em que se apoia é o foco na qualidade. (PRESSMAN, 2006, p.17).

De acordo com Tonsig (2008, p.65) os fundamentos científicos para a engenharia de *software* envolvem o uso de modelos abstratos e precisos que permitem especificar, projetar, implementar e manter sistemas de *software*, avaliando e garantindo sua qualidade.

2.3 QUALIDADE DE *SOFTWARE*

Segundo Molinari (2008b), a Engenharia de *Software* é mostrada como uma grande ciência, uma área de construção e gerência de projetos de *software*, dentro da qual se encontra uma subárea de Qualidade, que por sua vez, tem internamente o que se chama GQS (Garantia da Qualidade em *Software*) ou SQA (*Software Quality Assurance*). O autor ainda afirma que teste é uma área que tem se destacado devido à importância dada cada vez mais pelas empresas. Estas relações estão representadas na Figura 2.

Figura 2 – Contexto de qualidade e testes em Engenharia de *Software*



Fonte: Molinari (2008b)

Magela (2006, p. 346) afirma que existem quatro dimensões a serem consideradas para avaliar, qualificar e melhorar um *software*: pessoas, tecnologia, processo e produto. Mas o autor afirma que os processos de desenvolvimento de *software* junto com os produtos são o verdadeiro alvo de qualidade dos padrões internacionais.

Considerando a dimensão da qualidade que se refere ao "produto", de acordo com Sommerville (2007, p. 9) um bom *software* deve possuir os seguintes atributos: facilidade de manutenção, confiança, eficiência e usabilidade. Já de acordo com o modelo da norma internacional ISO/IEC 9126, publicada em 1991 são critérios de qualidade de *software*: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.

A qualidade do produto depende da qualidade do processo pelo qual o *software* foi desenvolvido. Considerando a dimensão da qualidade referente ao "processo", vários modelos foram propostos ou adaptados, para melhorar a qualidade do processo de desenvolvimento de *softwares*. Moraes e Laurindo (1999) publicaram um estudo sobre a importância de processos sistemáticos sobre teste e qualidade de *softwares*, baseados nomodelo da ISO 9000-3 (Diretrizes para a aplicação da ISO 9001 ao desenvolvimento, fornecimento e manutenção de *software*), e no Modelo de Maturidade de Capacitação (CMM-*CapabilityMaturityModel*).

Para melhorar os atributos de qualidade, o ciclo de desenvolvimentode *software* deve contemplar, independente da adoção de um modelo específico para isso, etapas diversas que busquem a redução da probabilidade de defeitos que culminam em perda da qualidade do *software*, podendo ocasionar prejuízos ao desenvolvedor e ao cliente.

2.4 TESTES DE *SOFTWARE*

Teste de *software* é definido por Humphrey (1989) como a execução de um programa para encontrar seus defeitos. Segundo Pressman (2006), uma vez gerado o código-fonte, o *software* deve ser testado para descobrir (e corrigir) tantos erros quanto possível antes da entrega ao cliente. Para Tonsig (2008, p. 242) todo *software* codificado deve sofrer rigorosos e exaustivos testes na busca incessante de erros e consequente eliminação dos mesmos. Sommerville (2007) define que teste é um processo voltado a atingir a confiabilidade do *software*.

Magela (2006) afirma que o teste do *software* é a verificação e validação do *software* através de sua execução em um processador. Para o autor, verificar consiste em atestar que o *software* está sendo construído corretamente (seguindo as especificações), e validar consiste em atestar que o *software* desejado esteja sendo construído (atendendo os requisitos do usuário).

Sommerville (2007), afirma que o processo de teste de *software* tem duas metas distintas: (1) demonstrar ao desenvolvedor e ao cliente que o *software* atende aos requisitos, e (2) descobrir falhas ou defeitos no *software* que apresenta comportamento incorreto, não desejável ou em não conformidade com as especificações.

Dijkstra (1972 *apud* Sommerville, 2007) afirma que os testes podem somente mostrar a presença de erros, não nunca sua ausência. Os erros advindos da não eliminação antecipada durante as fases de testes podem provocar sérios problemas e prejuízos às organizações, como exemplificado na Tabela 2.

Tabela 2 – Exemplos de problemas reais relacionados a *softwares*

Ano(s)	Empresa	Resultado
2004	Hewlett-Packard Co.	Problemas com o sistema ERP contribuíram para perda de $160 milhões
2004-05	UK Inland Revenue	Erros de *software* contribuíram para uma perda de 3,45 bilhões de dólares em crédito de imposto pago em excesso
1999	Hershey Foods Corp.	Problemas com o sistema ERP contribuíram para perda de $151 milhões
1999	Nasa	Um bit inesperado no conjunto de instruções do programa, devido a falta de testes de forma integrada do sistema, levou a perda da sonda Nasa Mars Polar Lander durante o pouso em Marte.
1996	Fox Meyer Drug Co.	Sistema ERP de 40 milhões de dólares sistema abandonado após a implantação, forçando empresa à falência.
1994-95	Disney	CD do jogo baseadono filme do Rei Leão, falhava ou não funcionava corretamente, pois não foi testado em modelos de computadores...
1994	Chemical Bank	Erro de *software* causou um total de 15 milhões de dólares deduzidos de 100.000 contas de clientes.

1993	Greyhound Lines Inc.	Sistema de reservas de ônibus falha repetidamente após iniciar, contribuindo para a perda de receita de US $ 61 milhões de dólares.

Fonte: Adaptado de Charette (2005) e Molinari (2008a, p. 52)

Portanto, os testes, enquanto etapa do processo de desenvolvimento de um *software*, devem ter como meta projetar uma série de casos de testes que têm uma grande probabilidade de encontrar erros naquele *software*, melhorando assim sua qualidade.

Os testes em *software*, como área da Engenharia de *Softwares*, agregam um conjunto de métodos e técnicas bem definidos a serem aplicados a problemas práticos. Segundo Pressman (2006), as técnicas de teste fornecem diretrizes sistemáticas para projetar testes que (1) exercitam a lógica interna e as interfaces de cada componente do *software*, e (2) exercitam os domínios de entrada e saída do programa para descobrir erros na função, no comportamento e no desempenho do programa.

2.4.1 Tipos de testes de *software*

O conjunto de métodos e técnicas que podem compor os testes de *software* é amplo e dinâmico, sendo muitas vezes influenciado pelo surgimento de novas tecnologias. Vários autores apresentam diferentes propostas para identificar ou classificar os tipos e estratégias de testes de *software*.

Pfleeger (2004) identifica os testes acompanhando os estágios do processo de desenvolvimento do *software*, sendo formada por duas grandes etapas: teste de programa (teste de unidade e teste de integração) e teste de sistema (teste funcional, teste de desempenho, teste de aceitação, teste de instalação). Esta sequência das etapas de teste está representada na Figura 3.

Figura 3 – Etapas de teste, segundo Pfleeger (2004)

Fonte: Pfleeger (2004, p. 276)

Magela (2006) apresenta uma visão macro dos testes de *software* cuja sequência é composta por: teste de unidade (especificação, estado e implementação), teste de integração, teste de sistema (funcionais, não-funcionais e de ambiente) e teste de distribuição (aceitação e documentação).

Já Molinari (2008b) utiliza dimensões para agrupar os tipos de teste, sendo que cada dimensão apresenta um escopo diferente e específico: a primeira dimensão (estado do teste, “o momento”), a segunda dimensão (técnica do teste, “como vou testar”), a terceira dimensão (metas do teste, “o que tenho de testar”), e a quarta dimensão (onde será o teste, “o ambiente do teste”).

A Tabela 3 apresenta os tipos e classificações segundo alguns autores para estratégias e técnicas de teste em *software*.

Tabela 3 – Classificações para testes de *software* por diversos autores

Pfleeger (2004)	**Magela (2006)**	**Pressman (2006)**	**Molinari (2008a)**
Teste de programa	Teste de unidade	Teste caixa-branca (estrutural/caixa de vidro)	1ª dimensão: estado do teste
Teste de unidade	Teste de especificação ou caixa-preta	Teste de caminho básico	Teste de unidade
Teste de integração	Teste de estado ou caixa-transparente	Teste de estrutura de controle	Teste de integração
Teste de sistema	Teste de implementação ou caixa branca	Teste de condição	Teste de sistema
Teste funcional	Teste de algoritmo	Teste de fluxo de dados	Teste de aceitação
Teste de desempenho	Teste de fluxo de controle	Teste de ciclo	2ª dimensão: técnica do teste
Teste de aceitação	Teste de comando	Teste caixa-preta (funcional/comportamental)	Teste operacional
Teste de instalação	Teste de ramos	Métodos de teste baseados em grafo	Teste negativo-positivo
	Teste de condição	Particionamento de equivalência	Teste de regressão
	Teste de caminho	Análise de valor-limite	Teste de caixa-preta
	Teste polifórmico	Teste de matriz ortogonal	Teste de caixa-branca
	Teste de integração	Métodos de teste orientados a objetos	Teste beta
	Teste do sistema	Teste baseado em erro	Teste de verificação de versão
	Testes funcionais	Métodos de teste aplicáveis ao nível de classe	3ª dimensão: metas do teste
	Testes não-funcionais	Teste aleatório para classes OO	Teste funcional
	Teste de carga	Projeto de casos de teste interclasse	Teste de interface
	Teste de volume,	Teste de várias classes	Teste de performance e teste de carga
	Teste de configuração,	Testes derivados dos modelos de comportamento	Teste de aceitação do usuário
	Teste de compatibilidade,	Teste de ambientes, arquiteturas e aplicações especializadas	Teste de estresse
	Teste de segurança,	Teste de IGU	Teste de volume
	Teste de temporal,	Teste de arquitetura cliente/servidor	Teste de configuração
	Teste de recuperação,	Teste da documentação e dispositivos de ajuda	Teste de instalação
	Teste de facilidade de uso	Teste de sistemas de tempo real	Teste de documentação
	Teste de disponibilidade		Teste de integridade
	Teste de documentação		Teste de segurança
	Teste de ambiente		4ª dimensão: onde será o teste
	Teste de distribuição		Teste de aplicações mainframe
	Teste de aceitação		Teste de aplicações client
	Teste de documentação		Teste de aplicações server
			Teste de aplicações network
			Teste de aplicações web

Sommerville (2007)
Teste de sistemas
Teste de integração
Teste de releases
Teste de desempenho
Teste de componentes
Teste de interfaces
Projeto de casos de teste
Teste baseado em requisitos
Teste de partições
Teste estrutural
Teste de caminho

Fonte: Elaborada pelo autor, a partir do elenco de autores pesquisados

2.4.2 Teste estrutural (caixa-branca) e teste comportamental (caixa-preta)

Uma separação bastante referenciada pelos autores é entre os testes caixa-branca e testes caixa-preta. Pressman (2006, p. 318) ressalta que estas duas categorias de teste podem ser aplicadas para qualquer produto que passe por engenharia: caixa-branca, sabendo como é o trabalho interno de um produto; e caixa-preta, conhecendo-se a função específica que o produto foi projetado para realizar.

A abordagem de teste caixa-branca (também chamado teste estrutural) é usada para projetar casos de teste nos quais estes são derivados do conhecimento da estrutura e da implementação do *software* (Sommerville, 2007). O teste caixa-branca é uma filosofia de projeto de casos de teste que usa a estrutura de controle descrita como parte do projeto ao nível de componente para derivar casos de teste (Pressman, 2006). Testes caixa-branca são testes para verificar se as operações internas são realizadas de acordo com as especificações e que todos os componentes internos sejam adequadamente exercitados. Uma abordagem complementar aos testes caixa-branca são os chamados testes caixa-preta.

O teste caixa-preta, também chamado de teste comportamental, visa validar os requisitos funcionais do *software*, sem se prender ao funcionamento interno de um programa. (PRESSMAN, 2006, p. 327). Estes testes demonstram que cada função está plenamente operacional, e ao mesmo tempo, procuram erros em cada função. Consistem em utilizar a especificação e chamar cada método passando o devido parâmetro e validando o resultado esperado. Ou seja, não se olha como foi codificado o método, e sim seu resultado (Pfleegler, 2004).

Testes de caixa-preta podem ser aplicados em nível de teste de unidade, teste de integração, teste de sistema e teste de aceitação do usuário. Existem técnicas específicas de testes funcionais ou métodos de projeto de casos de testes, tais como: teste de equivalência de classe, teste de valor limite, teste por tabela de decisão, *parwise testing*, teste de transição de estado e teste de análise de domínio (MOLINARI, 2008b, p. 149). Na sequência será pormenorizado o método de particionamento de equivalência.

2.4.3 Particionamento de equivalências

Particionamento de equivalências, também referenciado na literatura como teste de equivalência de classes (*equivalence class testing)* (Molinari, 2008b), teste de particionamento em categorias (Pezzè & Young, 2008), ou ainda teste de partições (Sommerville, 2007), é segundo Pressman (2006, p. 329), um “método de teste que divide o domínio de um programa em classes de dados, das quais os testes podem ser derivados”.

Na definição de Molinari (2008b, p. 150) a equivalência de classes é uma técnica de redução da quantidade de casos de testes (ou dados de testes) que se baseia em descobrir, em termos lógicos, conjuntos de situações (ou classes) que produzem ou gerem o mesmo comportamento na aplicação.

Segundo Sommerville (2007, p. 365) este método de teste se baseia na ideia de que os dados de entrada e os resultados de saída de um programa geralmente caem em classes (chamadas partições de equivalência ou domínios) diferentes com características comuns, como números positivos, números negativos e seleções de menu. Assim, uma abordagem sistemática pode ser usada para projetar casos de teste baseando-se na identificação de todas as partições de um sistema ou componente. O autor lembra ainda que entradas válidas e inválidas também formam partições equivalentes, sendo que as entradas que estão fora de outras partições escolhidas testam se o *software* trata as entradas de maneira correta.

Pressman (2006, p. 329) afirma que as classes de equivalência podem ser definidas de acordo com as seguintes diretrizes:

1. Se uma condição de entrada especifica um intervalo, uma classe de equivalência válida e duas inválidas são definidas.

2. Se uma condição de entrada exige um valor específico, uma classe de equivalência válida e duas inválidas são definidas.

3. Se uma condição de entrada especifica um membro de um conjunto, uma classe de equivalência válida e uma inválida são definidas.

4. Se uma condição de entrada é booleana, uma classe de equivalência válida e uma inválida são definidas.

Ainda de acordo com Sommerville (2007), classes de equivalência de entrada são conjuntos de dados em que todos os membros do conjunto devem ser processados de modo equivalente. Classes de equivalência de saídas são saídas de programa com características comuns, de maneira que elas possam ser consideradas como uma classe distinta. Na Figura 4, é feita a representação do particionamento de equivalência, na qual cada partição é mostrada como uma elipse:

Figura 4 – Particionamento de equivalência

Fonte: Sommerville (2007, p. 366)

Para Pezzè e Young (2008, p. 186) o teste de particionamento, é um método de teste que divide o conjunto infinito de casos de teste possíveis em um conjunto finito de classes,

com o propósito de derivar um ou mais casos de teste de cada classe. Ainda segundo os autores, o método é constituído através de três passos fundamentais: (i) decompor a especificação em funcionalidades testáveis independentes, (2) identificar valores representativos e (3) gerar especificações de casos de teste.

Este método busca definir um caso de teste que descobre classes de erros, tendo assim como vantagem a redução do o número total de casos de testes que precisam ser desenvolvidos (PRESSMAN, 2006 e MOLINARI, 2008b).

Esta técnica é simples e pode (e deve) ser usada de forma intuitiva pela maioria dos testadores. É uma técnica que trabalha com dedução lógica. (MOLINARI 2008b, p. 150)

O uso desta técnica associado aos outros princípios que compõem os testes de *softwares* permite a elaboração de casos de testes mais consistentes, com maiores chances de obter êxito na busca por problemas de qualidade nos sistemas em teste.

3 METODOLOGIA

Este capítulo visa descrever a abordagem metodológica utilizada para o desenvolvimento do trabalho final de graduação, com a definição das técnicas e métodos empregados em sua condução.

3.1 CARACTERIZAÇÃO DA PESQUISA

Segundo Miguel *et al* (2012) a caracterização do tipo de trabalho possibilita o desenvolvimento dos métodos e técnicas mais adequados. Desta forma, se buscou classificar este estudo dentro de critérios conhecidos na literatura para a definição do escopo metodológico a ser seguido. A pesquisa foi caracterizada quanto a critérios referentes ao seu objetivo, à sua abordagem, à sua forma ou tipo e ao método de pesquisa.

3.1.1 Objetivo

De acordo com Wazlawick (2008, p. 37), uma pesquisa pode ter finalidade técnica ou científica. Os trabalhos técnicos apenas usam o conhecimento já disponível, enquanto os trabalhos científicos devem, além de usar o conhecimento já disponível, criar novos conhecimentos, associando-os dentro de uma estrutura coerente àqueles que já são conhecidos.

Pela definição exposta, o presente trabalho pode ser caracterizado como técnico, pois foram utilizados conhecimentos já existentes com o objetivo de realizar uma aplicação prática em um estudo de caso.

3.1.2 Abordagem

Miguel *et al* (2012, p. 47) consideram que a escolha da abordagem da pesquisa precede a escolha do método de pesquisa propriamente dito. A classificação usualmente encontrada na literatura para abordagens em trabalhos acadêmicos é em abordagem qualitativa e abordagem quantitativa, embora alguns autores adotem a combinação destas duas como uma terceira abordagem.

A característica mais marcante da abordagem quantitativa é o ato de mensurar variáveis de pesquisa, para a partir delas prover dados para realização do teste das hipóteses. (MIGUEL *et al*, 2012, p. 47). Já a abordagem qualitativa, em contraste com a pesquisa quantitativa, possui como característica distintiva a ênfase na perspectiva do indivíduo que está sendo estudado (BRYMAN, 1989 *apud* MIGUEL *et al*, 2012).

Wazlawick (2008) relata que "os projetos de pesquisa que envolvem métodos qualitativos - por exemplo estudo de caso, intervenção e outros - têm a preocupação de empregar técnicas com foco principalmente na argumentação, e não apenas em aspectos numéricos."

Considerando os critérios apresentados, este estudo pode ser classificado como de abordagem predominantemente qualitativa, por apresentar uma ênfase na interpretação de evidências não quantificáveis do objeto.

3.1.3 Forma ou tipo

Ribeiro e Zabadal (2010, p. 30) afirmam que os estudos científicos, de forma geral, podem ter caráter exploratório, descritivo, explanatório ou preditivo, sendo que ainda alguns autores incluem o tipo explicativo.

Exploratório é aquele em que há novos aspectos, novos enfoques ou até novas áreas a serem descobertas, examinando um tema ou área pouco estudados, podendo abrir campo para muitas e novas pesquisas (Ribeiro e Zabadal, 2010).

Descritivo é aquele que busca descrever fenômenos ou sujeitos de pesquisa relacionados a dúvidas remanescentes de estudos exploratórios já feitos. Tem como requisito o conhecimento prévio considerável da área de estudo (Ribeiro e Zabadal, 2010).

Explanatório é aquele que descreve relações, inclusive relação do tipo causa e efeito. Não são medidas as características dos objetos, mas as relações entre eles. (Ribeiro e Zabadal, 2010).

Preditivo é aquele que pretende apresentar possíveis cenários ou consequências. Procura predizer os resultados de um fenômeno. (Ribeiro e Zabadal, 2010).

Considerando esta forma de classificação, pode-se dizer que a presente pesquisa apresenta maior proximidade de um estudo caráter exploratório, uma vez que estuda uma área em crescente mudança, e visou oferecer informações sobre o objeto da pesquisa.

3.1.4 Método de pesquisa

Miguel *et al* (2012) citam como principais métodos de pesquisa adotados na Engenharia de Produção e Gestão de Operações: o levantamento tipo *survey*, o estudo de caso, e a modelagem e simulação. Já Ribeiro e Zabadal (2010) classificam, de forma geral, os métodos de pesquisas em pesquisa bibliográfica, simulação, experimentação, pesquisa *survey*, estudo de caso e intervenções (pesquisa-ação).

O estudo de caso é um trabalho de caráter empírico que investiga um dado fenômeno dentro de um contexto real contemporâneo por meio de análise aprofundada de um ou mais objetos de análise (casos) (MIGUEL *et al*, 2012).

Segundo Yin (2010), o estudo de caso, como qualquer outro método de pesquisa, complementa os pontos fortes e as limitações dos outros tipos de pesquisa. Ainda segundo o autor, este método permite reter as características holísticas e significativas dos eventos da vida real.

A metodologia utilizada nesta monografia para análise dos resultados é o estudo de caso. Assim sendo, os resultados estão restritos ao objeto de estudo analisado, não podendo ser generalizados para todos os *softwares* ou modelos de desenvolvimento de *software*.

3.2 TÉCNICAS DE COLETA DE DADOS

O presente trabalho utilizou de alguns princípios já existentes na literatura sobre procedimentos para condução de testes em *software* com o intuito de estudar o caso específico. O modelo de processo de testes de *software* na Figura 5 ilustra os passos fundamentais deste processo, que foram utilizados para gerar e registrar os dados e informações do estudo.

Figura 5 – Modelo de processo de testes de *software*

Fonte: Sommerville (2007)

Os dados referentes aos testes executados foram coletados por meio de formulários próprios, identificado como casos de teste, nos quais foram registrados todas as observações, os dados de entrada, as saídas esperadas, os resultados obtidos e seu grau de conformidade, as figuras das telas da interface do *software*, as figuras com as mensagens de erro ou advertência exibidas pelo *software* durante a condução dos testes.

Observações sobre o comportamento do *software* durante a execução dos testes também foram registradas nos formulários de cada um dos casos de teste contidos no desenvolvimento do estudo.

3.3 ÁREA DA PESQUISA

A temática desta pesquisa contempla as áreas de Engenharia de *Software*, Engenharia da Qualidade e Tecnologia da Informação, e os sistemas dedicados ao planejamento e controle da produção de empresas. É importante ressaltar que as várias dimensões precisam ser consideradas, uma vez que são interdependentes.

O objeto do caso de estudo é um *software* (Mini PCP) da área de planejamento e controle da produção, no qual foi verificada a aplicação de testes de *software*, previamente planejados, que levaram em consideração conceitos já existentes na literatura.

4 ESTUDO DE CASO: TESTANDO O *SOFTWARE* MINIPCPC

Este capítulo visa apresentar o estudo de caso da aplicação de um método de teste de *software* em um sistema de planejamento e controle da produção, denominado MiniPCP, contemplando a descrição das funcionalidades do *software*, o plano de teste a ser executado e sua especificação, os resultados dos testes e uma discussão acerca dos resultados encontrados.

4.1 *SOFTWARE*: MINIPCP PLANEJAMENTO E CONTROLE DA PRODUÇÃO

O s *softwares* para Planejamento e Controle da Produção (PCP) são sistemas de informação que auxiliam a gerência das empresas a lidar com seus recursos de maneira eficiente, sendo, portanto, um sistema de informação aplicado. Existem diversas opções destes sistemas no mercado, com diferentes características. Neste estudo foi adotado o *software* Mini PCP, um sistema de controle de produção desenvolvido para indústrias de pequeno porte pela empresa EFX Tecnologia, a qual autorizou seu uso neste trabalho (Anexo A). Segundo o desenvolvedor, o Mini PCP é u m *software* pequeno que oferece recursos avançados, normalmente encontrados somente nos grandes sistemas de gestão. A versão de demonstração d o *software* (MiniPCP 1.8.1.220) foi obtida diretamente por download no site www.minipcp.com.br.

Por não se tratar da versão comercial para uso em ambiente de produção, mas sim uma versão de demonstração, o *software* apresenta algumas limitações, conforme informado pelo desenvolvedor: algumas tabelas com informações básicas estão com dados predefinidos e não podem ser alterados; limitação no número de registros que podem ser adicionados ao cadastro; não permite redefinir nome e caminho do banco de dados para uso em rede; a tela de aviso de demonstração é exibida aleatoriamente durante o uso do sistema.

O sistema Mini PCP é composto por nove recursos principais: MRP - Gestão de Materiais, Plano Mestre de Produção, Capacidade, Previsão de Vendas, Programação de Produção, Gestão de Compras, Controle de Estoque, Cadastros e Segurança. A Tabela 4 apresenta os principais recursos e funcionalidades do *software* Mini PCP.

Tabela 4 – Recursos e funcionalidades principais do *software* Mini PCP

Recurso/módulo	**Descrição das funcionalidades do recurso/módulo**
MRP - Gestão de Materiais	MRP, ou planejamento de necessidades de materiais, ajuda a manter os estoques de insumos em níveis ideais de forma que a produção não pare por falta de componentes e que o espaço de armazenamento não seja ocupado desnecessariamente. O cálculo de MRP inclui: Demanda pendente; Demanda de produção programada; Estoque projetado; Sugestão de compra; Mini PCP gera também relatórios para impressão, que podem ser exportados para diversos formatos.
Plano Mestre de Produção	Através do Plano Mestre de Produção (MPS) é possível obter uma visão clara e detalhada de como conduzir o processo. Para isso, o usuário seleciona o período de cálculo e define um ciclo de produção, que pode ser diário, semanal ou mensal. Também há a opção de informar os dias para abertura de ordens. Utiliza os dados de cadastro para calcular o tempo de produção e lançar as melhores datas, assegurando que o produto esteja pronto para atender a demanda no momento certo.
Capacidade	Permite que o usuário cadastre todas as etapas de produção, incluindo detalhes de maquinários e mão de obra. Junto com essas informações o programa registra o tempo

Recurso/módulo	Descrição das funcionalidades do recurso/módulo
	gasto em cada processo, bem como o número de pessoas que operam o recurso e a quantidade produzida por vez.
Previsão de Vendas	A previsão de vendas é uma das origens de informação que Mini PCP utiliza no cálculo de MPS. O sistema exibe somente produtos cujos tipos são sinalizados para venda. Basta escolher um item e preencher a quantidade prevista de vendas para cada mês.
Programação de Produção	As ordens são programadas para datas definidas pelo usuário, gerando dados para os cálculos de MRP. Permite ao usuário abrir novas ordens ou verificar o fluxo de produção, permite o acompanhamento das ordens até sua finalização, integrando empenho, saída de estoque e a entrada de itens produzidos. Além disso, gera um quadro comparativo para cada ordem aberta, indicando os insumos necessários para a fabricação, a quantidade de matéria-prima em estoque e a quantidade restante do processo.
Gestão de Compras	Permite cadastrar ordens de compra e visualizar o cadastro completo dos pedidos realizados/em andamento. Para registrar um novo pedido, o usuário seleciona um dos fornecedores cadastrados e inclui dados como condição de pagamento, previsão de entrega e observações. Os itens do pedido também são selecionados pela base do sistema, e o usuário deve apenas inserir a quantidade e o preço unitário. O programa lista os pedidos em aberto e seus respectivos itens. O usuário pode cadastrar o recebimento do pedido na mesma medida em que a mercadoria é entregue, pois o sistema se incumbe de deduzir as quantidades do total solicitado. Essa função permite o cadastro em lotes separados, facilitando a operação para indústrias que precisam rastrear os lotes do fabricante. Além disso, é possível obter maior controle sobre o pedido: mesmo que o recebimento venha em parcelas, o programa desconta apenas a quantidade recebida, alterando a situação do pedido para 'Parcialmente recebido' e apontando a quantidade pendente.
Controle de Estoque	Registrar com facilidade a entrada/saída de materiais e as suas movimentações (ajuste, inventário, venda e devolução). A busca pode ser feita de acordo com o tipo do produto (matéria prima, produto acabado, revenda e outros) ou através de seu código. Inventário de estoque: oferece recursos para que o usuário visualize os produtos por tipos, alterando não apenas a quantidade física como também o valor unitário de cada lote. Nesta seção também é possível cadastrar novos lotes e excluir registros anteriores. Consulta por lote: uma tela que permite consultar a movimentação de todos os lotes de produtos. É possível realizar a busca por tipo, além de aprovar ou rejeitar lotes com apenas alguns cliques.
Cadastros	Com suporte para informações diversas, fica a critério do usuário decidir quais campos são relevantes para cada registro. Por exemplo: o usuário possui a opção de preencher os códigos dos produtos manualmente (estipulando números únicos), mas não há problemas em deixar o registro em branco, pois o sistema se ocupa de fazer a numeração automaticamente. Fornece acesso ao cadastro de: Custo unitário padrão; Preço unitário de venda; Lote mínimo para compra/produção; Validade; Estoque mínimo; Lead Time de compra/produção; Lead Time para liberação; Essas informações permitem editar e cadastrar um número ilimitado de produtos, insumos e materiais. Desta forma, você registra com a mesma facilidade a matéria prima, o produto acabado e até a mão de obra. **Padrões para produtos e materiais:** Em Mini PCP o destaque fica por conta do usuário, que pode conferir e alterar os padrões utilizados pelo sistema. O ideal é que o cadastro de novos tipos, grupos e unidades proporcione uma perfeita integração às necessidades de seu empreendimento. **Fornecedores**: Com o cadastro de fornecedores do Mini PCP você terá acesso a uma ficha completa, que inclui dados como razão social, CNPJ, Inscrição Estadual, endereço, contato, site e email. Além disso, as telas recebem uma função de pesquisa para localizar todos os registros com rapidez.

Recurso/módulo	Descrição das funcionalidades do recurso/módulo
	Pedidos de compra: Permite cadastrar os pedidos de compra, fornecendo a condição de pagamento e previsão de entrega. Você seleciona os fornecedores do cadastro e inclui todos os itens que fazem parte do pedido, enquanto o sistema se encarrega de calcular a quantidade total e preencher os dados dos produtos já cadastrados.
Segurança	Para acessar Mini PCP, é necessário fornecer suas credenciais de acesso. O programa identifica o usuário e registra suas ações. Assim, é possível a realização de uma auditoria detalhada que ajuda a identificar e corrigir erros de operação do sistema.

Fonte: EFX Tecnologia (2012)

Ainda que se trate de um sistema de controle de produção para pequena indústria, o conjunto de módulos e funcionalidades se mostra amplo para execução de uma suíte de teste completa para fins deste estudo, e, portanto, optou-se por realizar testes em apenas dois dos módulos fundamentais do sistema: Cadastros e Gestão de Compras.

4.2 ETAPAS DO PROCESSO DE TESTE

Para se testar um *software*, é preciso considerar as etapas de teste envolvidas. Esta seção tem como objetivo apresentar todo o processo do caso de teste, desde a etapa de planejamento, passando pela especificação, execução, e culminando na análise dos testes.

4.2.1 Planejamento do teste

O objetivo principal da etapa de planejamento é elaborar a estratégia de testes para o projeto e definir como esta será implementada. Devendo tudo isso ser documentado no Plano de Testes. Segundo Sommerville (2007, p. 353), os planos de teste devem incluir uma descrição de itens a serem testados: o cronograma de teste, os procedimentos para gerenciamento do processo de teste, o hardware e os requisitos de *software*, além de quaisquer outros problemas de teste que possam surgir.

As diretrizes do Plano de Teste foram derivadas e desenvolvidas tendo como base o padrão IEEE 829 de documentação teste de *software* (*IEEE Standard for Software and System Test Documentation*, 1998) e na NBR ISO/IEC 12119 - Tecnologia de informação - Pacotes de *software* - Teste e requisitos de qualidade. Foram feitas algumas simplificações nas diretrizes, como em relação a cronograma, custos e pessoas envolvidos, uma vez que o plano de teste não faz parte de um projeto real, e sim de estudo acadêmico.

4.2.2 Plano de Testes

4.2.2.1 Identificador do plano de testes

Mini PCP 1.8.1.220 - PT 1.0 - Este é um plano de teste para o *software* Mini PCP - Planejamento e Controle da Produção para pequena indústria, desenvolvido pela EFX Tecnologia, e faz parte de um estudo de caso sobre testes de *software*.

4.2.2.2 Objetivos dos testes

Executar testes tendo em vista encontrar possíveis erros no *software* Mini PCPC, utilizando a abordagem de testes caixa-preta e o método de particionamento de equivalências, testando a conformidade quanto aos aspectos de funcionalidade, confiabilidade e usabilidade.

4.2.2.3 Escopo dos testes

O escopo deste plano de testes restringe-se à versão de demonstração de Mini PCP 1.8.1.220; e ao nível de teste funcional dos requisitos do sistema.

4.2.2.4 Referências a documentos relevantes

Número de ordem	Tipo do material	Referência bibliográfica
1	Site	Mini PCP - Recursos (www.minipcp.com.br)
2	Padrão	IEEE 829 Standard for *Software* and System Test Documentation, 1998
3	Norma	NBR ISO/IEC 12119 -Tecnologia de informação - Pacotes de*software* - Teste e requisitos dequalidade
4	Livro	MOLINARI, L. Testes funcionais de *software*. Florianópolis: Visual Books, 2008b.

		224 p. ISBN 978-85-7502-234-4.
5	Livro	PEZZÈ, M.; YOUNG, M. Teste e análise de *software*: processos, princípios e técnicas. Tradução de Bernardo Copstein e Flavio Moreira Oliveira. Porto Alegre: Bookman, 2008. 512 p. ISBN 978-85-7780-262-3.

4.2.2.5 Itens a testar

Número de ordem	Item	Referência às Especificações de Testes
1	Cadastros / Produtos e Insumos	[FormProdutos]
2	Cadastros / Linhas de produção e recursos	[FormCapacidade]
3	Cadastros / Fornecedores	[FormFornecedores]
4	Cadastros / Clientes	[FormClientes]
5	Cadastros / Vendedores	[FormVendedores]
6	Vendas / Pedido de venda	[FormVendasPedidos]
7	Compras/ Pedido de compra	[FormComprasPedidos]

4.2.2.6 Itens que não serão testados

Número de ordem	Item	Comentários
1	MRP - Gestão de Materiais	Estes recursos não serão testados por estarem fora do escopo deste plano de teste (outros tipos de testes podem ser mais adequados) e/ou pela limitação da versão de demonstração
2	Plano Mestre de Produção	
3	Capacidade	
4	Previsão de Vendas	
5	Programação de Produção	
6	Controle de Estoque	
7	Segurança	
8	Vendas / Movimentação de pedido	
9	Compras/ Recebimento de pedido de compra	

4.2.2.7 Aspectos a testar

Número de ordem	Item	Comentários
1	Funcionalidade	Presença das funções especificadas
2	Confiabilidade	Reconhecer as violações da sintaxeestabelecida para entrada de dados.
3	Usabilidade	Inteligibilidade, apresentação e organização

4.2.2.8 Aspectos que não serão testados

Número de ordem	Aspecto	Motivo
1	Eficiência	Não haver exigência normativa quanto a estes aspectos, e não estão no escopo do Plano de Teste.
2	Manutenibilidade	
3	Portabilidade	

4.2.2.9 Abordagem

Estas instruções descrevem o teste funcional (teste caixa-preta). O teste estrutural não está incluído porque requer a disponibilidade do código-fonte.

O principal método adotado é o do particionamento de equivalências. A intenção é que particionando o espaço de entrada em classes, gerando dados de teste de forma a escolher dados para cada partição, tenha-se uma amostra de casos com maior chance de incluir regiões “especiais” ou sujeitas a erro do espaço de entrada.

4.2.2.10 Ambiente - Hardware

Os testes foram realizados em um microcomputador com as seguintes configurações:

Fabricante: Acer, Modelo Aspire 1810TZ

Processador: Genuine Intel® CPU U 4100 1.30 GHz

Memória RAM: 2,00 GB

4.2.2.11 Ambiente - *Software*

Sistema Operacional: Windows 7 Home Premium – 64 bits Versão do *Software* testada: Mini PCP 1.8.1.220 - Versão de demonstração

4.2.2.12 Ferramentas de testes

Os testes serão executados manualmente, sem o uso de ferramentas especiais ou automatizadas.

4.2.3 Especificação e execução dos casos de teste

Nesta etapa, considerou-se que a principal abordagem a ser adotada seria a do particionamento de equivalências, ou seja, a utilização de classes de equivalência para especificar os casos de testes, de modo a reduzi-los, sem, no entanto, trazer prejuízos à eficácia em encontrar erros. Sommerville (2007, p. 365) afirma que uma boa regra prática para seleção de casos de teste é escolher os casos de teste no limite das partições mais os casos próximos ao ponto médio da partição.

A intenção é que particionando o espaço de entrada em classes, gerando dados de teste de forma a escolher dados para cada partição, tenha-se uma amostra de casos com maior chance de incluir regiões "especiais" ou sujeitas a erro do espaço de entrada. (PEZZÈ e YOUNG, 2008)

A geração dos cenários e consequentemente dos casos de teste faz uso do documento de requisitos do sistema (casos de uso), de modo a percorrer as funcionalidades do sistema em busca de possíveis erros, que possam ter passado despercebidos por etapas anteriores de teste.

Na especificação de testes, foram relacionados: os aspectos a serem testados, os detalhes da abordagem, identificação dos testes, procedimentos de teste, e os casos de teste.

Com base nas especificações de testes previamente definidas, os casos de testes foram sendo executados e devidamente documentados.

4.2.3.1 Identificador da especificação de teste

Mini PCP 1.8.1.220 - ET 1.0- Especificação de teste funcional dos módulos Cadastros e Gestão de Compras.

4.2.3.2 Aspectos a serem testados

Número	Requisito	Comentários
1	Funcionalidade	Presença das funções especificadas
2	Confiabilidade	Reconhecer as violações da sintaxe estabelecida para entrada de dados.
3	Usabilidade	Inteligibilidade, apresentação e organização

4.2.3.3 Detalhes da abordagem

Os aspectos a serem testados no *software*, em conformidade com a NBR ISO/IEC 12119 foram:

Funcionalidade: As propriedades adicionais do produto devem ser descritas para assegurar a capacidade funcional do produto: verificação se a entrada é aceitável e proteção contra consequências danosas decorrentes de erro de usuário. Valores-limite específicos devem ser fornecidos no teste: valores máximos ou mínimos; comprimento de chaves; número máximo de registros em um arquivo; número máximo de critérios de busca.

Confiabilidade: O programa deve reconhecer as violações da sintaxe estabelecida para entrada de dados. No caso de o programa reconhecer uma entrada como errônea ou indefinida ele não deve processá-la como uma entrada permitida.

Usabilidade: a usabilidade abrange os seguintes aspectos:

Inteligibilidade: As mensagens de erro devem fornecer informações detalhadas, explicando a sua causa ou forma de correção.

Apresentação e organização: As mensagens do programa devem ser projetadas de forma que o usuário possa diferenciá-las facilmente pelo tipo, por exemplo: confirmação; solicitações; advertências; mensagens de erro.

Os formatos de tela de entrada, de relatórios e de outras entradas e saídas sejam projetados para serem claros e com boa apresentação e organização. Algumas alternativas poderiam ser:

- campos alfanuméricos sejam alinhados pela esquerda;

- campos numéricos sejam alinhados pela direita;

- em tabelas, pontos decimais e vírgulas sejam colocados na mesma linha vertical;

- limites dos campos sejam reconhecíveis;

- campos obrigatórios sejam reconhecíveis como tal;

- na detecção de falhas de entrada, as mesmas sejam imediatamente realçadas na tela;

- quando ocorrer uma mudança no conteúdo da tela, o usuário seja alertado por um sinal auditivo ou visual.

Para verificação da conformidade (não existência de erros), foi adotada a seguinte escala de cores, para representar os resultados encontrados nos testes.

	Conforme	Saída de acordo com o resultado esperado
	Parcialmente conforme	Saída de acordo com o resultado esperado, com alguma ressalva
	Não-conforme	Saída com resultado diferente do resultado esperado

Algumas categorias de foram estabelecidas para facilitar a identicação dos procedimentos de teste, sendo elas:

Teste de inclusão de registro:

Para verificar a robustez do *software* em impedir registros indevidos que pudessem comprometer a funcionalidade, confiabilidade ou usabilidade do *software*, foram feitos testes de inclusão usando partições de equivalências para os campos, sendo de modo geral, testada a reação do *software* a três classes: entradas nulas, entradas válidas e entradas inválidas.

Teste de entrada duplicada:

Para verificação se o *software* permitia a inclusão de registro duplo (repetidos) nas funções de cadastro, foram feitos testes com a tentativa de inclusão de dois registros com valores idênticos para os diversos campos.

Testes de exclusão de registro:

Entrada do teste: tentativa de exclusão de um registro com dependência de outro
Saída esperada: Alerta: advertência ao usuário.

A saída do *software* correspondendo à saída esperada é considerada conformidade.

4.2.3.4 Identificação dos casos de teste

Número	Caso de teste	Identificação do caso de teste
1	Cadastros / Produtos e Insumos	[FormProdutos]
2	Cadastros / Linhas de produção e recursos / Recursos	[FormCapacidade]
3	Cadastros / Fornecedores	[FormFornecedores]
4	Cadastros / Clientes	[FormClientes]
5	Cadastros / Vendedores	[FormVendedores]
6	Vendas / Pedido de venda	[FormVendasPedidos]
7	Compras/ Pedido de compra	[FormComprasPedidos]

Identificação	[FormProdutos]					
Itens a testar	Cadastros / Produtos e Insumos					
Entrada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**		
	Tipo	Nulo	Produto acabado	Matéria prima		
	Grupo	Nulo	Nacionais	Importados		
	Código	Nulo	001	MP-0002		
	Descrição	Nulo	Descrição 1	Descrição 2		
	Descrição adicional	Nulo	Descrição Adic.1	Descrição Adic.2		
	Unidade	Nulo	L	g		
	Custo unitário padrão	Nulo	-100,00	+	-	,,
	Preço unitário de venda	Nulo	-200,00	+	-	,,
	Estoque mínimo	Nulo	-100	+	-	,,
	Lote mínimo de compra/produção	Nulo	-10	+	-	,,
	Validade (meses)	Nulo	-24	+	-	,,
	Leadtime compra/produção	Nulo	-5	+	-	,,
	Leadtime liberação	Nulo	-10	+	-	,,
Saída esperada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**		
	Tipo	Alerta: obrigatório	Produto acabado	Matéria prima		
	Grupo	Nulo	Nacionais	Importados		
	Código	Alerta: obrigatório	001	MP-0002		
	Descrição	Nulo	Descrição 1	Descrição 2		
	Descrição adicional	Nulo	Descrição Adic.1	Descrição Adic.2		
	Unidade	Nulo	L	g		
	Custo unitário padrão	Nulo	Alerta: advertência	Alerta: inválido		
	Preço unitário de venda	Nulo	Alerta: advertência	Alerta: inválido		
	Estoque	Nulo	Alerta:	Alerta: inválido		

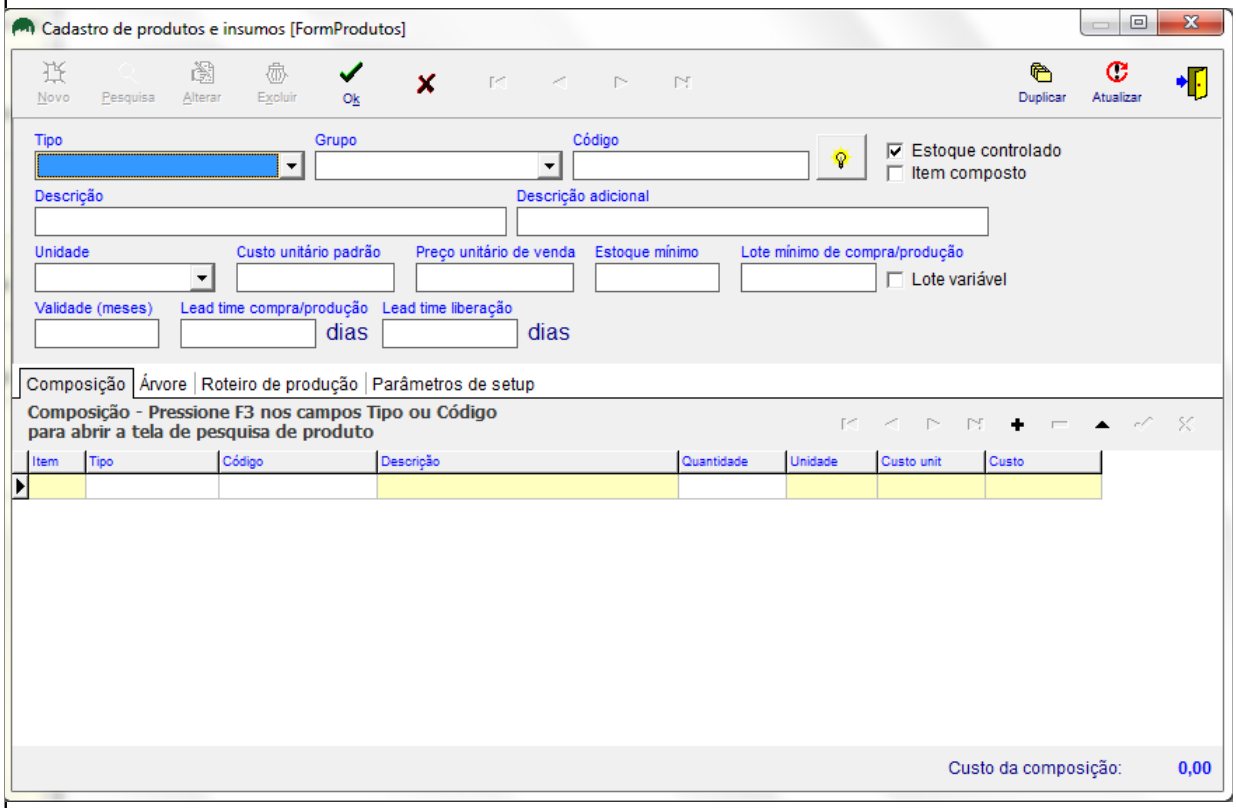
	mínimo		inválido	
	Lote mínimo de compra/produção	Nulo	Alerta: inválido	Alerta: inválido
	Validade (meses)	Nulo	Alerta: inválido	Alerta: inválido
	Leadtime compra/produção	Nulo	Alerta: inválido	Alerta: inválido
	Leadtime liberação	Nulo	Alerta: inválido	Alerta: inválido

4.2.3.5 Caso de teste 1: Cadastros / Produtos e Insumos

4.2.3.5.1 Ambiente

A interface do *software* testada no caso de teste 1 está representada na Figura 6.

Figura 6 – Caso de teste 1 - Ambiente de teste

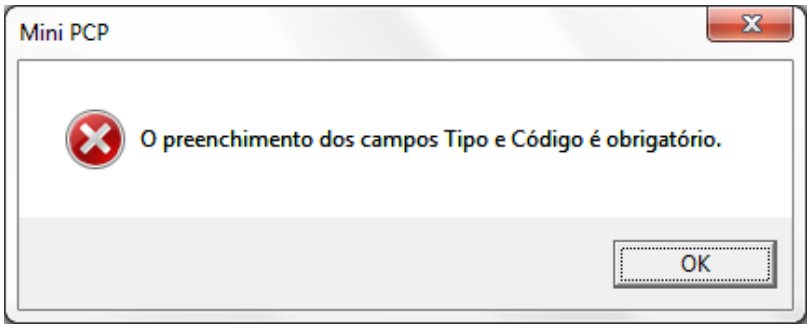


Fonte: EFX Tecnologia, MiniPCP (2012)

4.2.3.5.2 Relatório

Testes de inclusão de registro
Para atributos deixados nulos, embora o sistema exiba mensagem de alerta para os campos obrigatórios (Figura 7), estes não estão devidamente sinalizados como tal na interface do *software*.

Figura 7 – Caso de teste 1 - Mensagem de erro 1

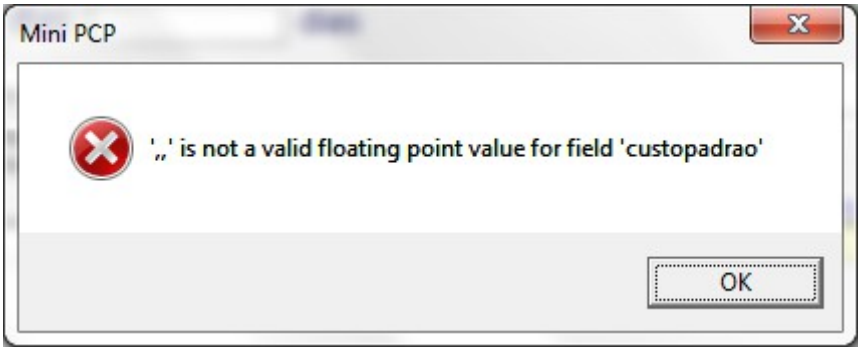


Fonte: EFX Tecnologia, MiniPCP (2012)

Valores negativos foram aceitos, sem restrição, nos campos Custo unitário padrão, Preço unitário de venda, Estoque mínimo, Lote mínimo de compra/produção, Validade (meses), Leadtime compra/produção, Leadtime liberação, por serem campos do tipo *floating point* ou *interger*. Esperavam-se mensagens de erro de advertência ou de valor inválido para estes campos.

Os valores (caracteres) considerados inválidos (+ - ,,) para os campos do tipo *floating point* ou *interger* receberam alerta (Figura 8), não sendo aceitos para inclusão.

Figura 8 – Caso de teste 1 - Mensagem de erro 2

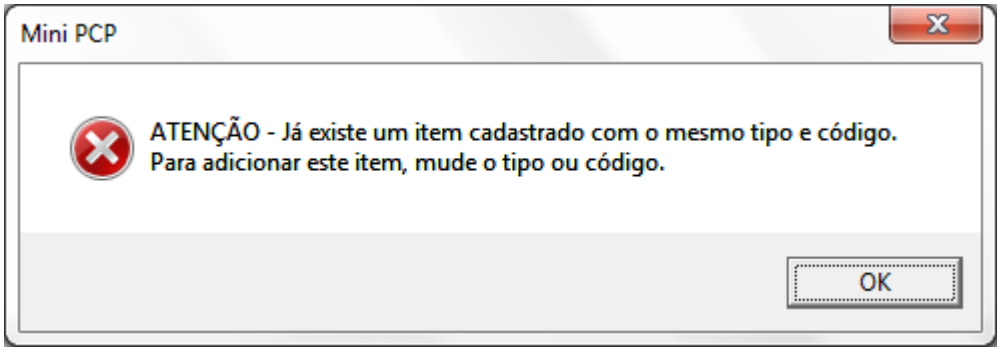


Fonte: EFX Tecnologia, MiniPCP (2012)

Demais caracteres alfabéticos (letras e símbolos) para campos numéricos (*floating point* ou *interger*) não foram aceitos (sequer para digitação) pelo *software*.

Teste de inclusão de registro duplicado
Resultado: | | Conforme. Mensagem exibida na Figura 9.

Figura 9 – Caso de teste 1 - Mensagem de erro 3

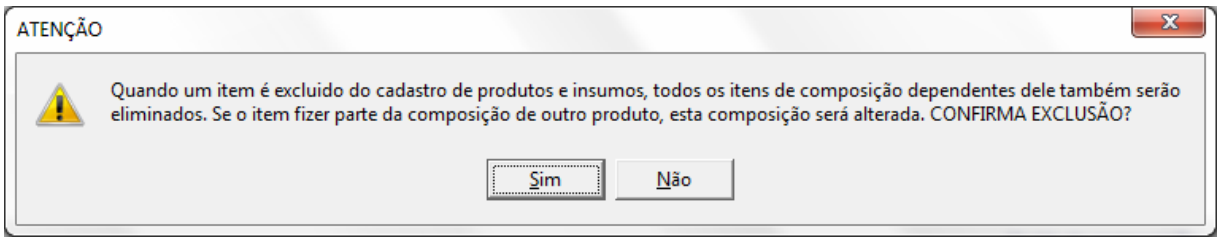


Fonte: EFX Tecnologia, MiniPCP (2012)

Teste de exclusão de registro

Resultado: | | Conforme. Mensagem exibida na Figura 10.

Figura 10 – Caso de teste 1 - Mensagem de advertência 1



Fonte: EFX Tecnologia, MiniPCP (2012)

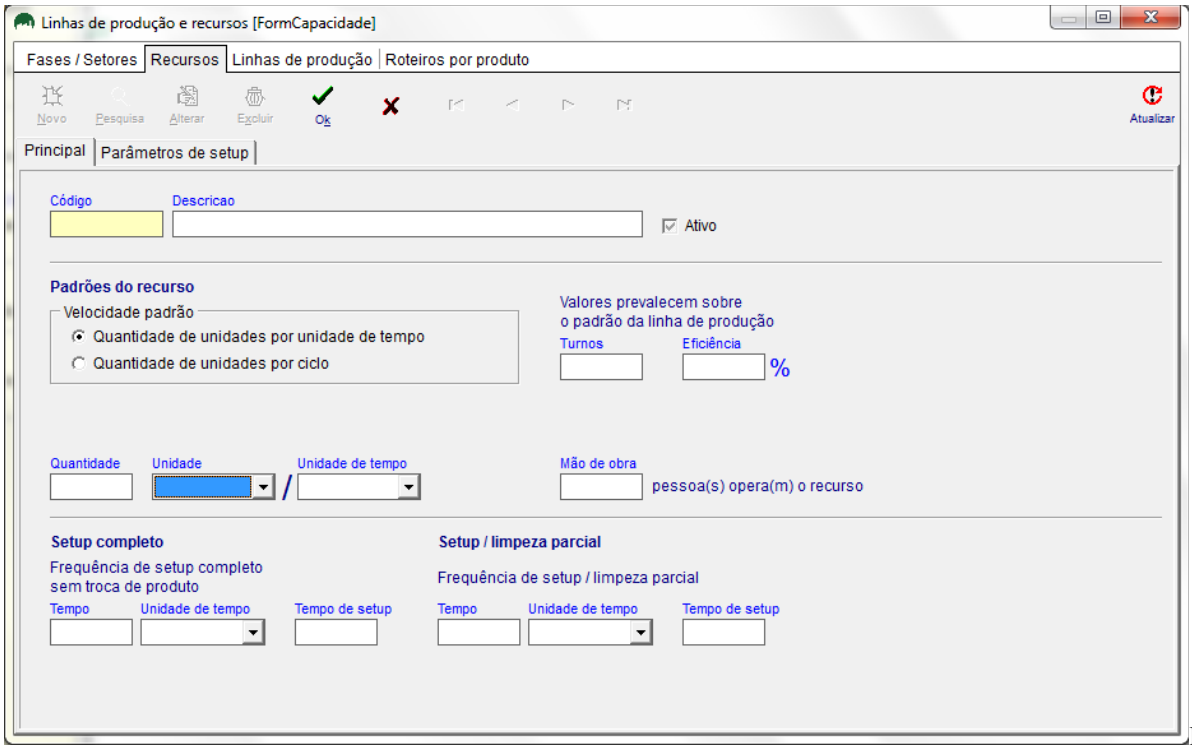
4.2.3.6 Caso de teste 2: Cadastros / Linhas de produção e recursos / Recursos

Identificação	[FormCapacidade]			
Itens a testar	Cadastros / Linhas de produção e recursos / Recursos			
Entrada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**
	Código			
	Descrição	Nulo	Recurso 1	Recurso 2
	Turnos	Nulo	-2	1,5
	Eficiência	Nulo	-10	1,5
	Quantidade	Nulo	-10	1,5
	Unidade	Nulo	Lote	Unidade
	Unidade de Tempo	Nulo	Dia	Hora
	Mão de obra	Nulo	-10	1,5
	Tempo	Nulo	-10	1,5
	Unidade de tempo	Nulo	Dia	Hora
	Tempo de Setup	Nulo	-10	99:99:99
Saída esperada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**
	Código	Núm. sequencial	Núm. sequencial	Núm. sequencial
	Descrição	Nulo	Recurso 1	Recurso 2
	Turnos	Nulo	Alerta: inválido	1,5
	Eficiência	Nulo	-10	1,5
	Quantidade	Nulo	Alerta: inválido	1,5
	Unidade	Nulo	Lote	Unidade
	Unidade de Tempo	Nulo	Dia	Hora
	Mão de obra	Nulo	Alerta: inválido	Alerta: inválido
	Tempo	Nulo	Alerta: inválido	Alerta: inválido
	Unidade de tempo	Nulo	Dia	Hora
	Tempo de Setup	Nulo	Alerta: inválido	Alerta: inválido

4.2.3.6.1 Ambiente

A interface do *software* testada no caso de teste 2 está representada na Figura 11.

Figura 11 – Caso de teste 2 - Ambiente de teste



Font
e: EFX Tecnologia, MiniPCP (2012)

4.2.3.6.2 Relatório

Testes de inclusão de registro
O campo código é de preenchimento automático. Não havendo outro campo obrigatório, a entrada com valores nulos, não foi considerada como inválida.

Os campos do tipo tempo (tempo de setup) reconheceram todos os valores inválidos, exibindo a mensagem (Figura 12) ou em caso de número decimais ou caracteres alfabéticos, não permitiu a digitação.

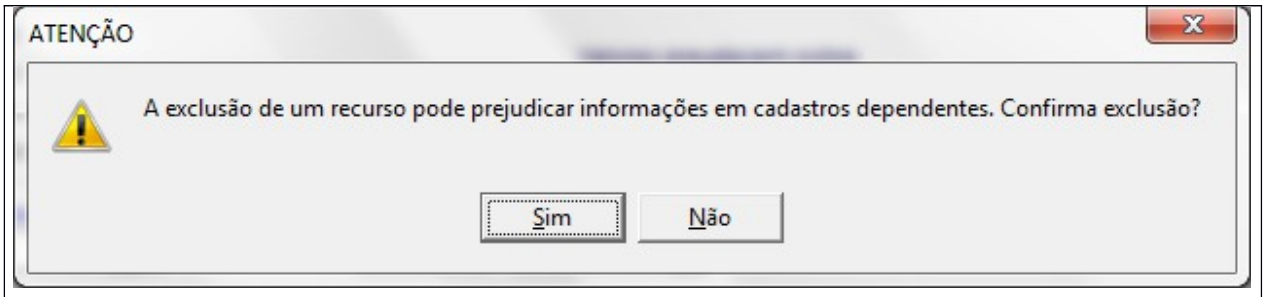
Figura 12 – Caso de teste 2 - Mensagem de erro 1

Fonte: EFX Tecnologia, MiniPCP (2012)

Teste de inclusão de registro duplicado
Não se aplica (campo ‘código’ não editável)

Teste de exclusão
Resultado: | | Conforme. Mensagem exibida na Figura 13.

Figura 13 – Caso de teste 2 - Mensagem de advertência 1



Fonte: EFX Tecnologia, MiniPCP (2012)

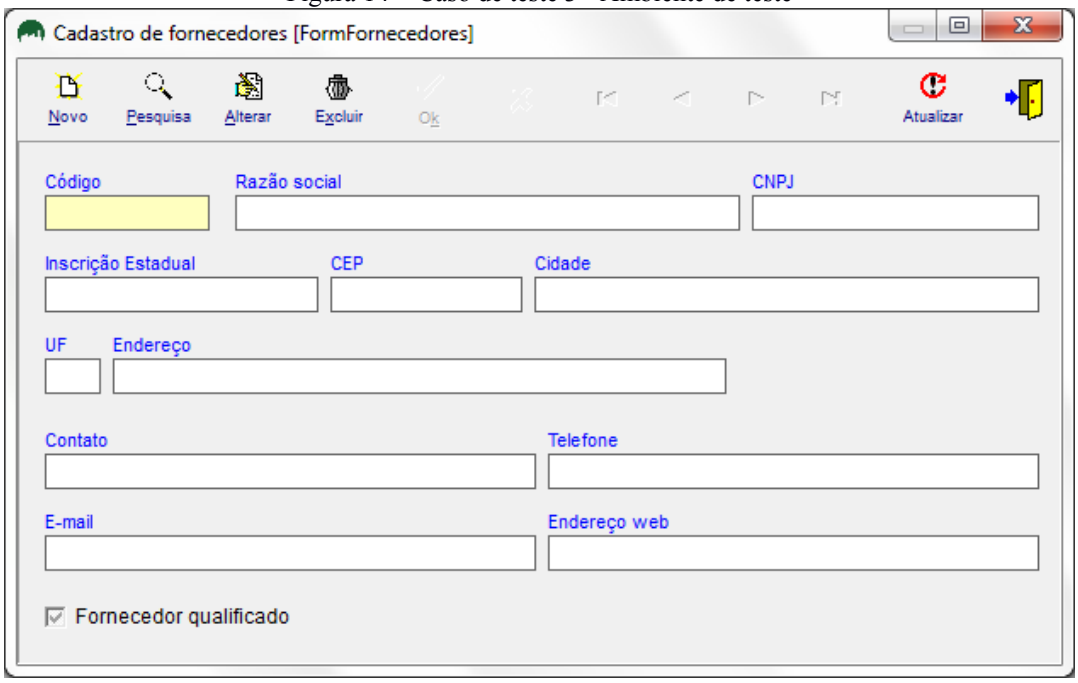
4.2.3.7 Caso de teste 3: Cadastros / Fornecedores

Identificação	[FormFornecedores]			
Itens a testar	Cadastros / Fornecedores			
Entrada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**
	Código	Nulo	Nulo	Nulo
	Razão social	Nulo	Fornecedor 1	Fornecedor 2
	CNPJ	Nulo	23.070.659/0001-10	99.999.999/9999
	Inscrição estadual	Nulo	9999999	9999999
	CEP	Nulo	35.400-000	XXXXXXXXXX
	Cidade	Nulo	Cidade 1	Cidade 2
	UF	Nulo	MG	XXX
	Endereço	Nulo	Endereço 1	Endereço 2
	Contato	Nulo	Fulano	Beltrano
	Telefone	Nulo	553199999999	(55)31-9999-9999
	E-mail	Nulo	forne1@email.com	forne2.com
	Endereço web	Nulo	www.site.com	www
Saída esperada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**
	Código	Número sequencial	Número sequencial	Número sequencial
	Razão social	Nulo	Fornecedor 1	Fornecedor 2
	CNPJ	Nulo	23.070.659/0001-10	Alerta: inválido
	Inscrição estadual	Nulo	9999999	9999999
	CEP	Nulo	35.400-000	Alerta: inválido
	Cidade	Nulo	Cidade 1	Cidade 2
	UF	Nulo	MG	Alerta: inválido
	Endereço	Nulo	Endereço 1	Endereço 2
	Contato	Nulo	Fulano	Beltrano
	Telefone	Nulo	553199999999	(55)31-9999-9999
	E-mail	Nulo	forne1@email.com	Alerta: inválido
	Endereço web	Nulo	www.site.com	www.site.com

4.2.3.7.1 Ambiente

A interface do *software* testada no caso de teste 3 está representada na Figura 14.

Figura 14 – Caso de teste 3 - Ambiente de teste



Fonte: EFX Tecnologia, MiniPCP (2012)

Testes de inclusão de registro O campo código é de preenchimento automático. Não havendo outro campo obrigatório, a entrada com valores

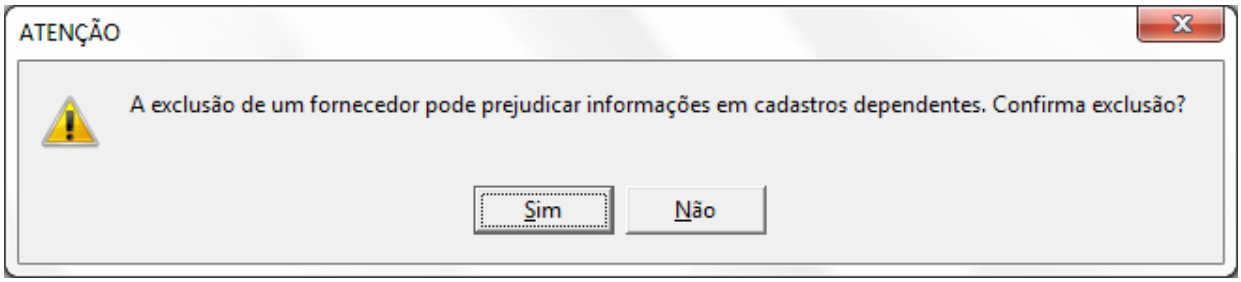
nulos, não foi considerada como inválida.

Os campos ‘CNPJ’ e ‘CEP’ são passíveis de validação por comprimento de chaves e/ou formato padrão.
O campo ‘E-mail’ pode ser parcialmente validado pela presença (ou ausência) do caractere @ (arroba).

Teste de inclusão de registro duplicado
Resultado: | | Não-conforme

Teste de exclusão
Resultado: | | Conforme. Mensagem exibida na Figura 15.

Figura 15 – Caso de teste 3 - Mensagem de advertência 1



Fonte: EFX Tecnologia, MiniPCP (2012)

4.2.3.7.2 Relatório

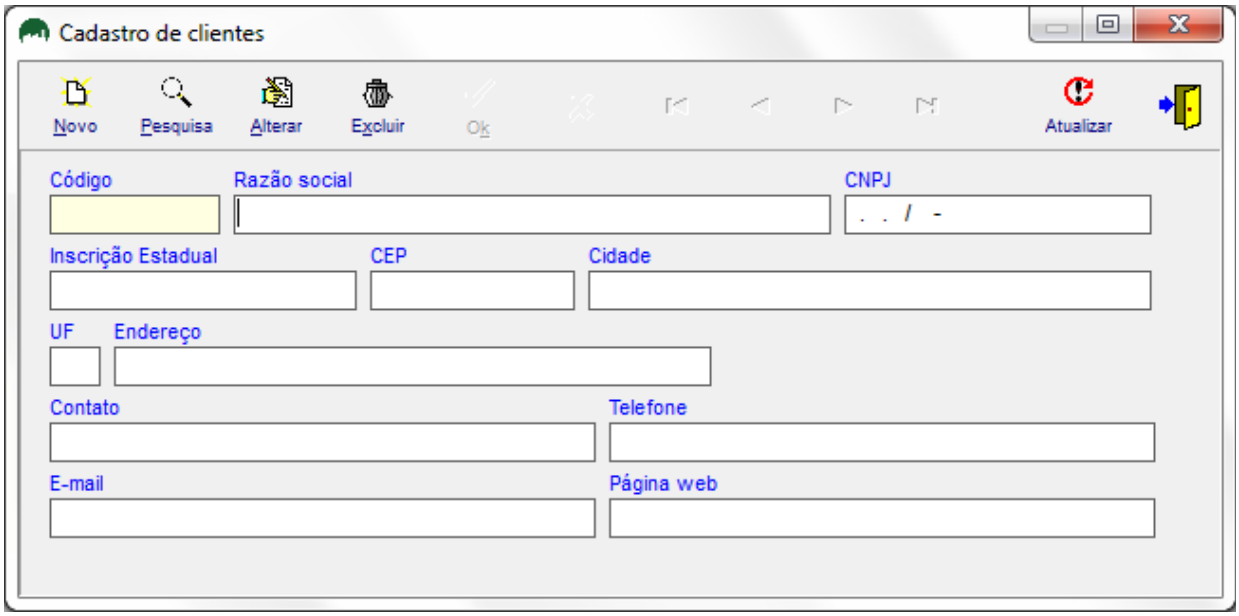
4.2.3.8 Caso de teste 4: Cadastros / Clientes

Identificação	[FormClientes]			
Itens a testar	Cadastros / Clientes			
Entrada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**
	Código	Nulo	Nulo	Nulo
	Razão social	Nulo	Cliente 1	Cliente 2
	CNPJ	Nulo	23.070.659/0001-10	99.999.999/9999
	Inscrição Estadual	Nulo	9999999	9999999
	CEP	Nulo	35.400-000	XXXXXXXXXXX
	Cidade	Nulo	Cidade 1	Cidade 2
	UF	Nulo	MG	XXX
	Endereço	Nulo	Endereço 1	Endereço 2
	Contato	Nulo	Fulano	Beltrano
	Telefone	Nulo	553199999999	(55)31-9999-9999
	E-mail	Nulo	forne1@email.com	forne2.com
	Página web	Nulo	www.site.com	www.site.com
Saída esperada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**
	Código	Número sequencial	Número sequencial	Número sequencial
	Razão social	Nulo	Fornecedor 1	Fornecedor 2
	CNPJ	Nulo	23.070.659/0001-10	Alerta: inválido
	Inscrição Estadual	Nulo	9999999	9999999
	CEP	Nulo	35.400-000	Alerta: inválido
	Cidade	Nulo	Cidade 1	Cidade 2
	UF	Nulo	MG	Alerta: inválido
	Endereço	Nulo	Endereço 1	Endereço 2
	Contato	Nulo	Fulano	Beltrano
	Telefone	Nulo	553199999999	(55)31-9999-9999
	E-mail	Nulo	forne1@email.com	Alerta: inválido
	Página web	Nulo	www.site.com	www.site.com

4.2.3.8.1 Ambiente

A interface do *software* testada no caso de teste 4 está representada na Figura 16.

Figura 16 – Caso de teste 4 - Ambiente de teste



Fonte: EFX Tecnologia, MiniPCP (2012)

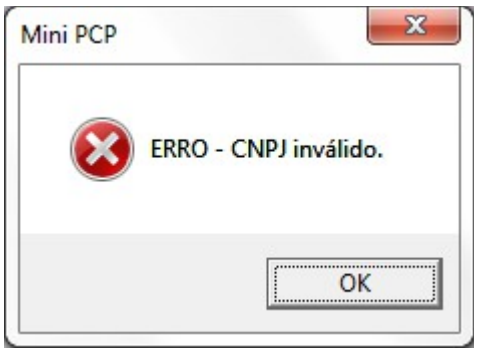
4.2.3.8.2 Relatório

Testes de inclusão de registro
O campo código é de preenchimento automático. Não havendo outro campo obrigatório, a entrada com valores nulos, não foi considerada como inválida.

O campo 'CEP' é passível de validação por comprimento de chaves e/ou formato padrão.
O campo 'E-mail' pode ser parcialmente validado pela presença (ou ausência) do caractere @ (arroba).

O campo 'CNPJ' de cliente identificou corretamente entradas inválidas, exibindo a seguinte mensagem de erro, ilustrada na Figura 17.

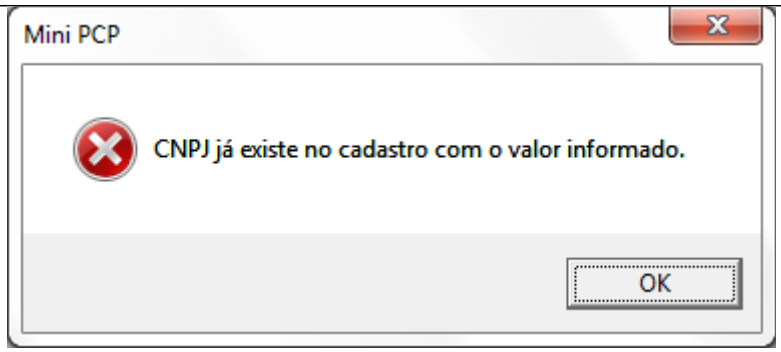
Figura 17 – Caso de teste 4 - Mensagem de erro 1



Fonte: EFX Tecnologia, MiniPCP (2012)

Teste de inclusão de registro duplicado
Resultado: | | Conforme. Mensagem exibida na Figura 18.

Figura 18 – Caso de teste 4 - Mensagem de erro 2



Fonte: EFX Tecnologia, MiniPCP (2012)

Teste de exclusão
Resultado: | | Não-conforme
Cliente com vínculos em outras partes do *software* (como 'Pedidos de venda') foram excluídos, sem restrições.

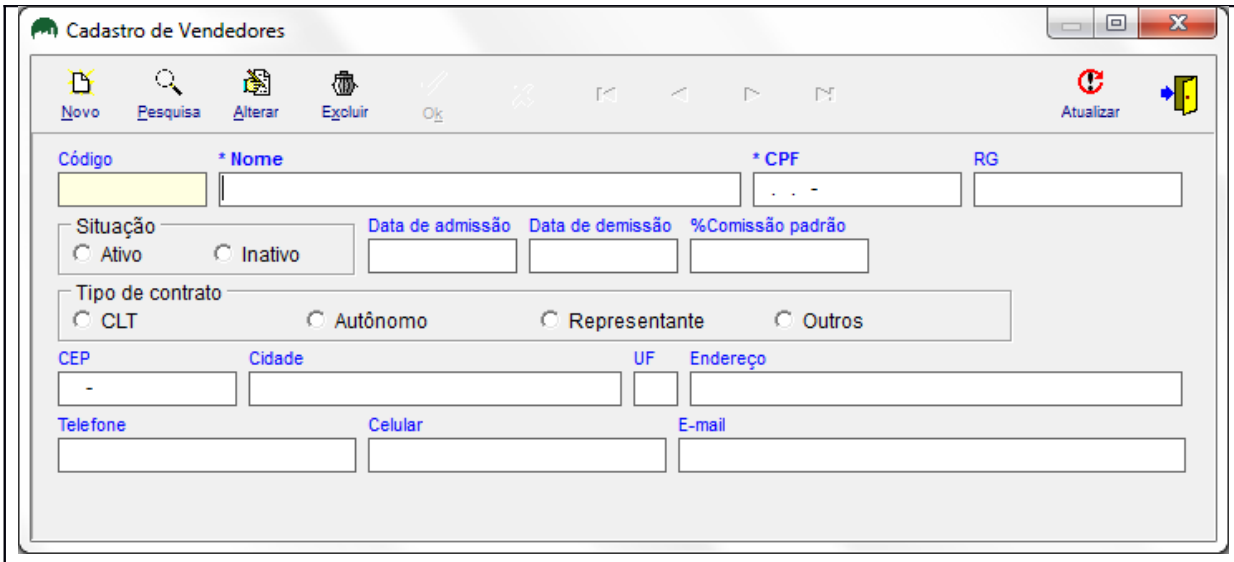
4.2.3.9 Caso de teste 5: Cadastros / Vendedores

Identificação	[FormVendedores]			
Itens a testar	Cadastros / Vendedores			
Entrada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**
	Código	Nulo	Nulo	Nulo
	* Nome	Nulo	Vendedor 1	Vendedor 2
	* CPF	Nulo	000.000.000-00	000.000.000-00
	RG	Nulo	MG12345678	BA12345678910
	Data de admissão	Nulo	24/05/2012	24/05/2012
	Data de demissão	Nulo	20/05/2012	31/02/2012
	% Comissão padrão	Nulo	10	-10
	CEP	Nulo	35400-000	9999999999999999
	Cidade	Nulo	Cidade 1	Cidade 2
	UF	Nulo	MG	XXX
	Endereço	Nulo	Endereço 1	Endereço 2
	Telefone	Nulo	5531333333333	(55)11-3333-3333
	Celular	Nulo	553199999999	(55)11-9999-9999
	E-mail	Nulo	vend1@email.com	vend2.com
Saída esperada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**
	Código	Número sequencial	Número sequencial	Número sequencial
	* Nome	Alerta obrigatório	Vendedor 1	Vendedor 2
	* CPF	Alerta obrigatório	000.000.000-00	Alerta: duplo
	RG	Nulo	MG12345678	BA12345678910
	Data de admissão	Nulo	24/05/2012	24/05/2012
	Data de demissão	Nulo	20/05/2012	Alerta: inválido
	% Comissão padrão	Nulo	10	Alerta: inválido
	CEP	Nulo	35400-000	Alerta: inválido
	Cidade	Nulo	Cidade 1	Cidade 2
	UF	Nulo	MG	XXX
	Endereço	Nulo	Endereço 1	Endereço 2
	Telefone	Nulo	5531333333333	(55)11-3333-3333
	Celular	Nulo	553199999999	(55)11-9999-9999
	E-mail	Nulo	vend1@email.com	Alerta: inválido
	Código	Nulo	Vendedor 1	Vendedor 2

4.2.3.9.1 Ambiente

A interface do *software* testada no caso de teste 5 está representada na Figura 19.

Figura 19 – Caso de teste 5 - Ambiente de teste



Fonte: EFX Tecnologia, MiniPCP (2012)

4.2.3.9.2 Relatório

Testes de inclusão de registro

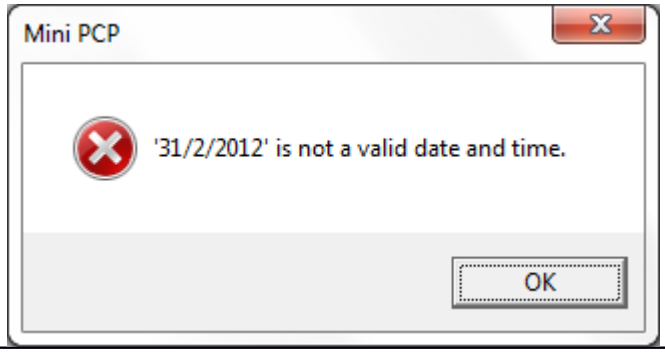
Os campos obrigatórios 'Nome' e 'CPF' foram sinalizados com asterisco (*), porém não há legenda com esta indicação na tela. Foi exibida mensagem (Figura 20) de obrigatoriedade dos mesmos ao tentar incluir o registro destes campos nulos:

Figura 20 – Caso de teste 5 - Mensagem de erro 1

Fonte: EFX Tecnologia, MiniPCP (2012)

Os campos do tipo data (admissão e demissão) reconheceram todos os valores inválidos, exibindo a mensagem similar a da Figura 21.

Figura 21 – Caso de teste 5 - Mensagem de erro



Fonte: EFX Tecnologia, MiniPCP (2012)

Não foi verificada a validade lógica entre ‘data de demissão’(20/05/2012) ser anterior à ‘data de admissão’ (24/05/2012).

Valores negativos foram aceitos, sem restrição, no campo ‘% Comissão padrão’. Esperava-se mensagem de erro de advertência ou de valor inválido para este campo.

O campo ‘CEP’ é passível de validação por comprimento de chaves e/ou formato padrão.
O campo ‘E-mail’ pode ser parcialmente validado pela presença (ou ausência) do caractere @ (arroba).

Teste de inclusão de registro duplicado
Resultado: | | Não-conforme

Teste de exclusão
Resultado: | | Não-conforme
Vendedor com vínculos em outras partes do *software* (como ‘Pedidos de venda’) foram excluídos, sem restrições.

4.2.3.10 Caso de teste 6: Vendas / Pedidos de venda

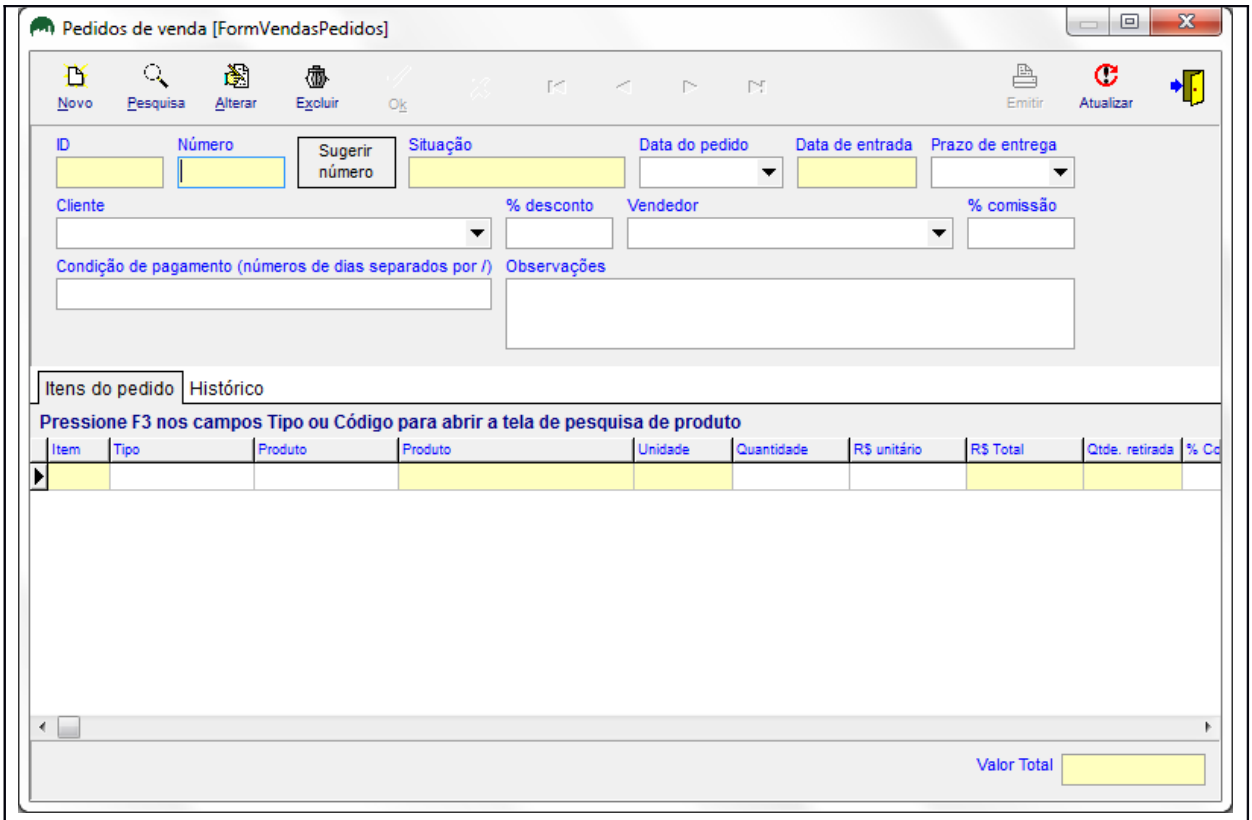
Identificação	[FormVendasPedidos]			
Itens a testar	Vendas / Pedidos de venda			
Entrada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**
	ID	Nulo	Nulo	Nulo
	Número	Nulo	1	1
	Situação	Nulo	Nulo	Nulo
	Data do pedido	Nulo	24/05/2012	31/2/2012
	Data de entrada	Nulo	24/05/2012	24/05/2012
	Prazo de entrega	Nulo	20/05/2012	31/05/2012
	Cliente	Nulo	Cliente 1	Cliente 2
	% desconto	Nulo	10	-10
	Vendedor	Nulo	Vendedor 1	Vendedor 2
	% comissão	Nulo	10	-10
	Condição de pagamento	Nulo	0/30/60	0/30/20
	Observações	Nulo	Obs. 1	Obs. 2

	Campo	Atributo de teste	Atributo de teste	Atributo de teste
Saída esperada	ID	Núm. sequencial	Núm. sequencial	Núm. sequencial
	Número	Alerta: obrigatório	1	Alerta: duplo
	Situação	Aguardando aprovação	Aguardando aprovação	Aguardando aprovação
	Data do pedido	Nulo	24/05/2012	Alerta: inválido
	Data de entrada	Preenc.Automático	Preenc.Automático	Preenc.Automático
	Prazo de entrega	Alerta: obrigatório	Alerta inválido	31/05/2012
	Cliente	Alerta: obrigatório	Cliente 1	Cliente 2
	% desconto	Nulo	10	Alerta: inválido
	Vendedor	Nulo	Vendedor 1	Vendedor 2
	% comissão	Nulo	10	Alerta: inválido
	Condição de pagamento	Nulo	0/30/60	Alerta: inválido
	Observações	Nulo	Obs. 1	Obs. 2

4.2.3.10.1 Ambiente

A interface do *software* testada no caso de teste 6 está representada na Figura 22.

Figura 22 – Caso de teste 6 - Ambiente de teste



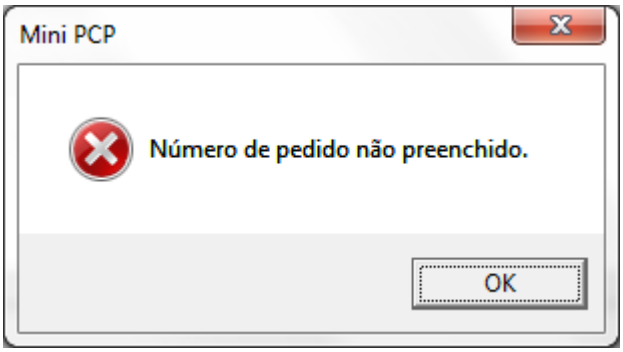
Fonte: EFX Tecnologia, MiniPCP (2012)

4.2.3.10.2 Relatório

Testes de inclusão de registro

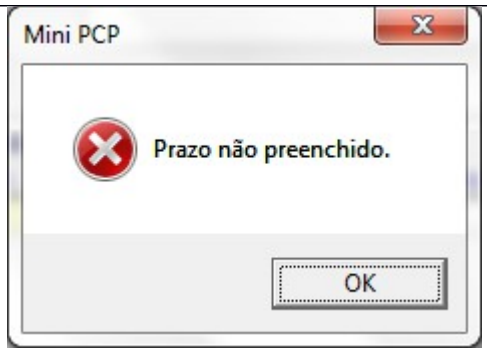
Campos considerados obrigatórios (‘número de pedido’ - Figura 23, ‘prazo de entrega’ - Figura 24, ‘data de emissão do pedido’ - Figura 25 e ‘cliente’ - Figura 26), quando para inclusão foram deixados nulos, tiveram mensagem de erro exibida, porém não estão sinalizados como obrigatórios na interface do *software*.

Figura 23 – Caso de teste 6 - Mensagem de erro 1



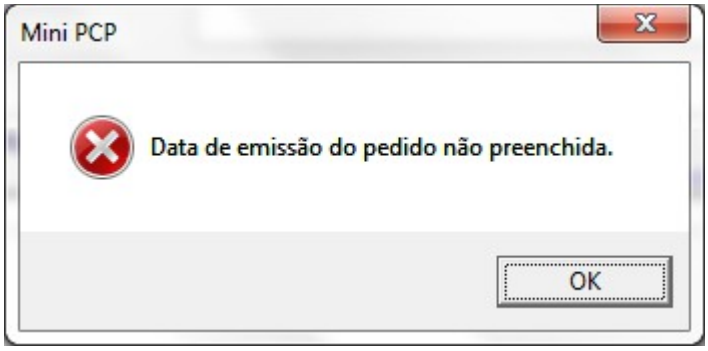
Fonte: EFX Tecnologia, MiniPCP (2012)

Figura 24 – Caso de teste 6 - Mensagem de erro 2



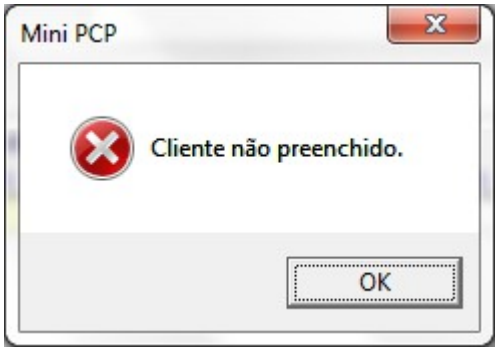
Fonte: EFX Tecnologia, MiniPCP (2012)

Figura 25 – Caso de teste 6 - Mensagem de erro 3



Fonte: EFX Tecnologia, MiniPCP (2012)

Figura 26 – Caso de teste 6 - Mensagem de erro 4



Fonte: EFX Tecnologia, MiniPCP (2012)

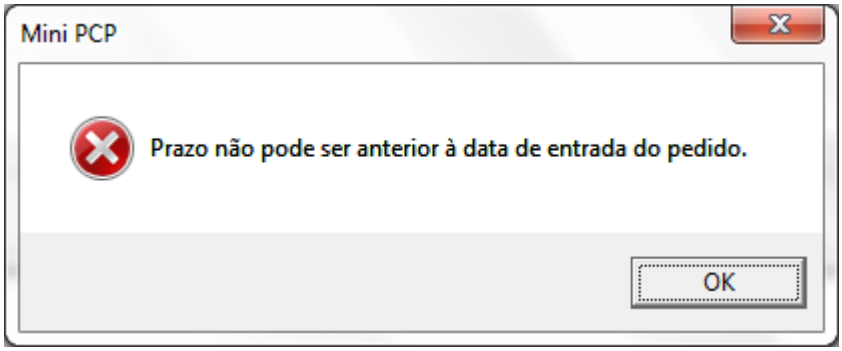
Os campos de data ‘data do pedido’, ‘data de entrada’ e ‘prazo de entrega’ foram devidamente verificados, sendo exibida a seguinte mensagem em caso de erro, mostrada na Figura 27.

Figura 27 – Caso de teste 6 - Mensagem de erro 5

Fonte: EFX Tecnologia, MiniPCP (2012)

Foi verificada a validade lógica entre ‘prazo de entrega’(20/05/2012) ser anterior à ‘data de entrada do pedido’ (24/05/2012), sendo exibida a mensagem da Figura 28.

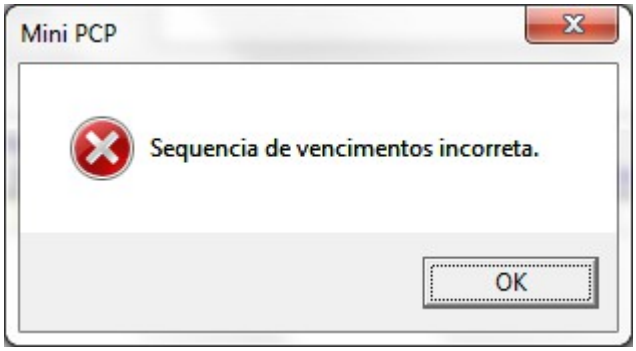
Figura 28 – Caso de teste 6 - Mensagem de erro 6



Fonte: EFX Tecnologia, MiniPCP (2012)

Foi verificada a validade lógica para preenchimento correto do campo ‘condição de pagamento’, sendo exibida para valores inválidos a mensagem de erro da Figura 29.

Figura 29 – Caso de teste 6 - Mensagem de erro 7



Fonte: EFX Tecnologia, MiniPCP (2012)

Valores negativos foram aceitos, sem restrição, no campo ‘% comissão’ e no campo ‘% desconto’. Esperava-se mensagem de erro de advertência ou de valor inválido para este campo.

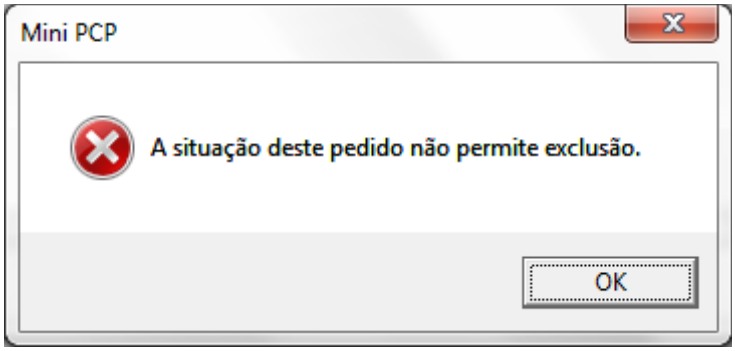
Teste de inclusão de registro duplicado
Resultado: | | Conforme. Mensagem exibida na Figura 30.

Figura 30 – Caso de teste 6 - Mensagem de erro 8

Fonte: EFX Tecnologia, MiniPCP (2012)

Teste de exclusão
Resultado: | | Conforme. Mensagem exibida na Figura 31.

Figura 31 – Caso de teste 6 - Mensagem de erro 9



Fonte: EFX Tecnologia, MiniPCP (2012)

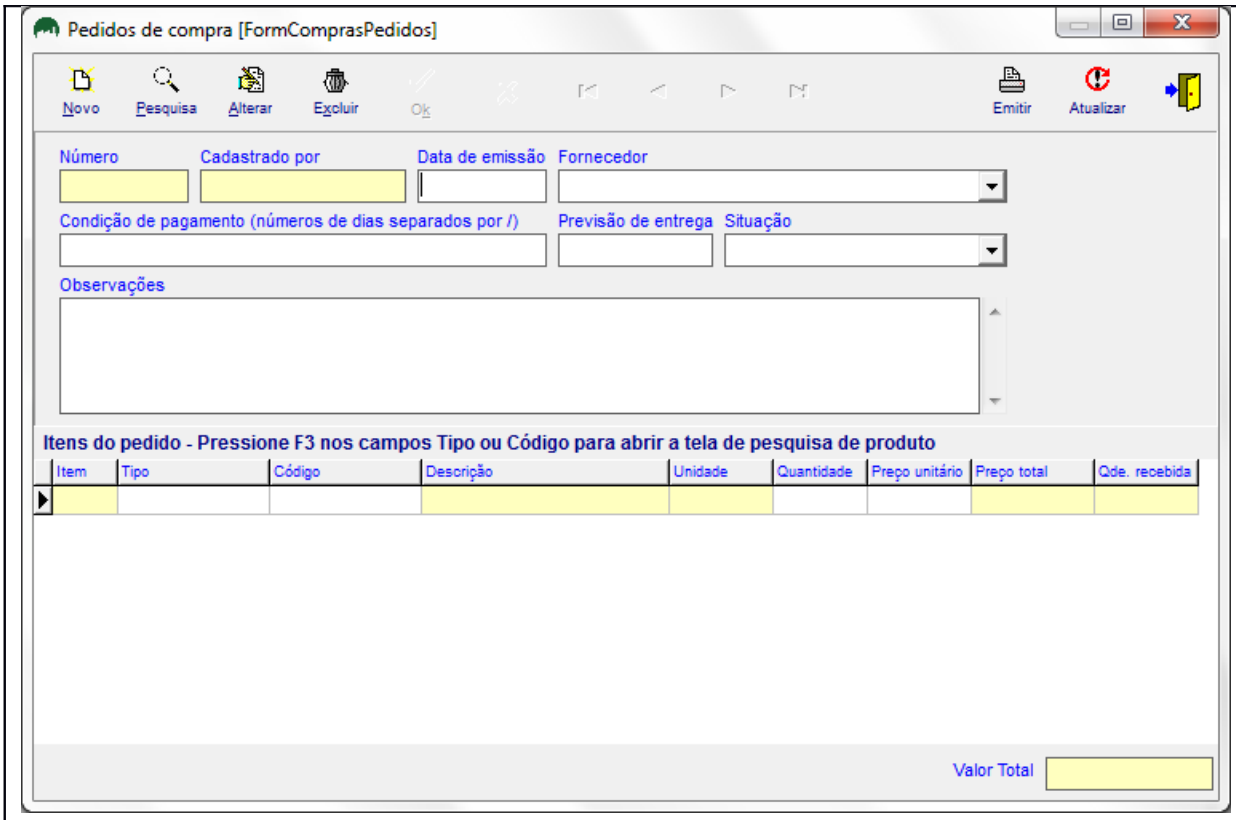
1.1.1.1 Caso de teste 7: Compras / Pedidos de compra

Identificação	[FormComprasPedidos]			
Itens a testar	Vendas / Pedidos de venda			
Entrada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**
	Número	Nulo	Nulo	Nulo
	Cadastrado por	Nulo	Nulo	Nulo
	Data de emissão	Nulo	31/02/2012	x
	Fornecedor	Nulo	Fornecedor 1	Fornecedor 2
	Condição de pagamento	Nulo	0/30/60	0/30/20
	Previsão de entrega	Nulo	31/02/2012	x
	Situação	Nulo	Nulo	Nulo
	Observações	Nulo	Obs. 1	Obs. 2
Saída esperada	**Campo**	**Atributo de teste**	**Atributo de teste**	**Atributo de teste**
	Número	Núm. sequencial	Núm. sequencial	Núm. sequencial
	Cadastrado por	Preenc.Automático	Preenc.Automático	Preenc.Automático
	Data de emissão	Preenc.Automático	Alerta: inválido	Alerta: inválido
	Fornecedor	Alerta: obrigatório	Fornecedor 1	Fornecedor 2
	Condição de pagamento	Nulo	0/30/60	Alerta: inválido
	Previsão de entrega	Alerta: obrigatório	Alerta: inválido	Alerta: inválido
	Situação	Preenc.Automático	Preenc.Automático	Preenc.Automático
	Observações	Nulo	Obs. 1	Obs. 2

4.2.3.10.3 Ambiente

A interface do *software* testada no caso de teste 7 está representada na Figura 32.

Figura 32 – Caso de teste 7 - Ambiente de teste



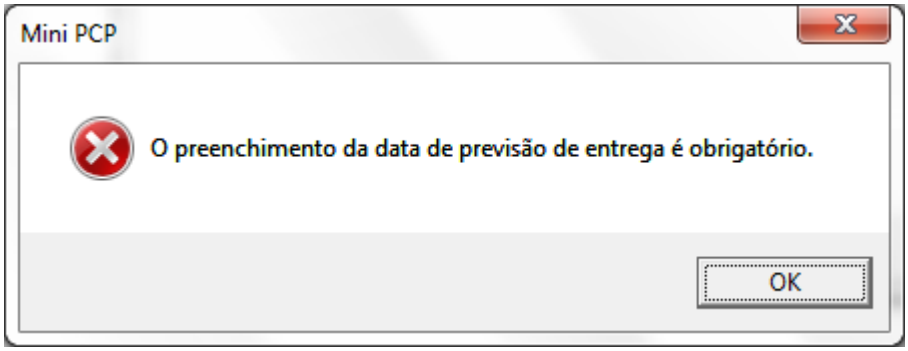
Fonte: EFX Tecnologia, MiniPCP (2012)

4.2.3.10.4 Relatório

Testes de inclusão de registro

O campo 'Previsão de entrega' foi considerado obrigatório (Figura 33), exibindo a mensagem de erro a seguir. Não há indicação da obrigatoriedade na interface do *software*.

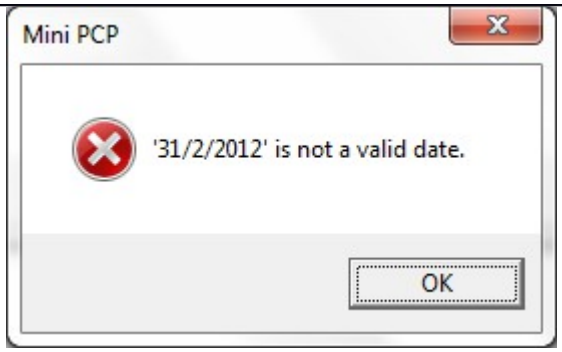
Figura 33 – Caso de teste 7 - Mensagem de erro 1



O campo 'fornecedor' não é considerado como obrigatório.

Os campos de data 'data de emissão' (Figura 34) e 'previsão de entrega' (Figura 35) foram devidamente verificados, sendo exibidas a seguintes mensagens em caso de erro:

Figura 34 – Caso de teste 7 - Mensagem de erro 2



Fonte: EFX Tecnologia, MiniPCP (2012)

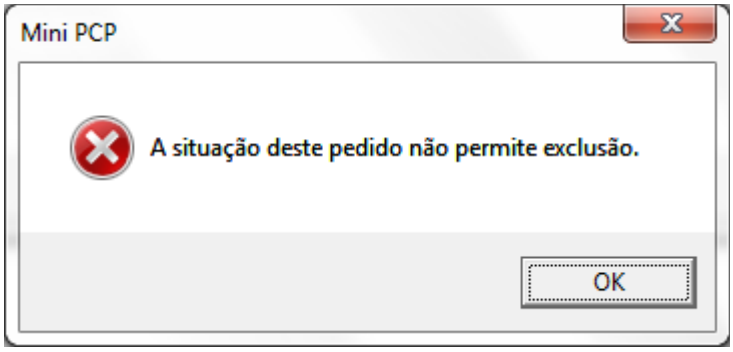
Figura 35 – Caso de teste 7 - Mensagem de erro 3

Fonte: EFX Tecnologia, MiniPCP (2012)

Teste de inclusão de registro duplicado
Não se aplica (campo 'número' da compra não é editável)

Teste de exclusão
Resultado: | | Conforme. Mensagem exibida na Figura 36.

Figura 36 – Caso de teste 7 - Mensagem de erro 4



Fonte: EFX Tecnologia, MiniPCP (2012)

4.2.4 Resultados dos testes

Os diversos casos de testes executados no *software* tiveram como resultados situações que foram consideradas em conformes, parcialmente conformes ou ainda não-conformes, em relação aos parâmetros esperados. Deste modo, a Tabela 5 sintetiza os resultados dos testes aplicados no *software*, separados pelos casos de teste:

Tabela 5 – Síntese dos resultados dos testes

Caso de Teste	Conforme	Parcialmente	Não-conforme
1	32	2	7
2	29	0	5
3	34	2	2
4	35	3	1
5	38	3	6
6	33	3	2
7	23	1	0
Total	224	14	23

Fonte: Elaborada pelo autor

De modo complementar, a Figura 37 representa graficamente os registros obtidos com os testes, em colunas 100% empilhadas, indicando o percentual para cada uma das situações (conformes, parcialmente conformes ou não-conformes) para os vários casos de teste.

Figura 37 – Síntese dos resultados dos testes

Fonte: Elaborada pelo autor

4.3 DISCUSSÃO DOS RESULTADOS

Os testes visaram principalmente os componentes responsáveis pelos cadastros do sistema, que servem como base de dados para as demais funcionalidades existentes no *software*, tais como a Gestão de Materiais (MRP), o Plano Mestre de Produção, Capacidade, Previsão de Vendas, Programação de Produção e o Controle de Estoque, ou ainda para outras funcionalidades que possam vir a ser implementadas.

Os casos de teste revelaram algumas falhas, em termos dos aspectos de funcionalidade, confiabilidade e usabilidade, para a execução adequada do *software*. Estas falhas podem ser consideradas problemas de qualidade.

As principais não-conformidades encontradas com a condução dos testes referem-se à questões de verificação de dados inválidos, que podem provocar transtornos no decorrer da execução de outras funções do *software*.

Por meio do particionamento de equivalência, as classes equivalentes derivadas pelos testes, especialmente as que se referem à entrada de números negativos, mostraram, que não foi devidamente tratada para muitas variáveis, o que pode gerar insconsistências e erros nos cálculos internos do programa.

Uma categoria de não-conformidades encontrada foi com relação às inclusões e exclusões de dados. A falha de inclusão, permitiu que registro fosse cadastrado com duplicidade dos dados, podendo causar insconsistências ao longo do tempo de uso do *software*.

Com relação às exclusões de registros, a falha encontrada foi a do não tratamento da ação de exclusão: para alguns tipos de cadastro foi permitida a exclusão inadvertidamente, sem possibilidade de recuperação imediata pelo usuário, podendo acarretar perda de vínculos com outros elementos do sistema, e perda de informação relevante.

Outra observação acerca resultados foi a inconsistência verificada em algumas interfaces quanto à identificação e tratamento de campos considerados de preenchimento obrigatório, os quais, por vezes, não estavam devidamente sinalizados.

Ainda pode-se ressaltar que os demais testes puderam verificar percorrer caminhos no *software* os quais a princípio não foram encontradas falhas.

5 CONSIDERAÇÕES FINAIS

A evolução das tecnologias de informação precisa ser acompanhada de meios que visem aumentar ou garantir a qualidade que se espera com sua utilização. Particularmente, os *softwares* são produtos de complexo desenvolvimento, dado seu caráter dinâmico e sua intangibilidade. Neste contexto, uma crescente importância vem sendo dada aos conceitos e ao ferramental dos testes de *software*.

A questão problema do trabalho foi contemplada ao verificar que o *software* estudado, através da aplicação de métodos de teste, pode se apropriar de melhorias no seu funcionamento, viabilizadas pela atuação sobre as falhas encontradas, elevando assim, sua qualidade.

Por sua vez, os objetivos específicos seguidos e descritos ao longo do trabalho, permitiram um melhor delineamento de conceitos e das etapas importantes das abordagens para o teste no *software*. Dessa forma, a revisão de literatura investigou a qualidade de *software* e uma relação com diversos métodos de teste. Foi dada ênfase nas características e princípios dos testes caixa-preta e da técnica do particionamento de equivalências.

As etapas de planejamento, elaboração do plano de teste, especificação, execução e análise dos casos de testes permitiram que os testes alcançassem, no *software* estudado, resultados concretos.

Os testes realizados no *software* estudado permitiram encontrar falhas relacionadas à funcionalidade, confiabilidade e usabilidade do sistema, evidenciando assim, que os testes podem ser uma alternativa viável na busca pela melhoria da qualidade no *software*. Para tanto, cabe salientar ainda que os testes não são etapas finais de desenvolvimento, uma vez que ao obter sucesso na busca por falhas, estas devem ainda ser devidamente tratadas pelos programadores/desenvolvedores, ou seja, os testes precisam de *feedback* adequado.

Com a devida atenção ao planejamento, os testes de *software* podem permitir o desenvolvimento do sistema de forma mais robusta, com dispositivos a prova de erros, e aprimoramento ao longo do tempo, criando novas e melhores versões.

O *software* estudado, como sistema dedicado ao planejamento e controle da produção de empresas, sistemas de informação fundamentais na atualidade, precisa de atenção no seu desenvolvimento, pois dele podem depender vários agentes das cadeias produtivas. Testá-lo pode ajudar a minimizar desde problemas que o usuário do sistema em uma indústria se depara no dia-a-dia, até evitar que grandes erros operacionais venham a ocorrer.

Este estudo apresenta suas limitações, pois não propôs uma suíte completa de testes que pudesse esgotar o tema. Outros testes, com diferentes e novas perspectivas e ferramentas podem ser planejados para minimização da ocorrência de erros. Aspectos relacionados à gerência de projeto no desenvolvimento de *software* estão intimamente ligados à atividade de testes, estudos que contemplem variáveis fundamentais como custos associados e tempo despendido para cada atividade são também importantes.

REFERÊNCIAS

ABEPRO. **Associação Brasileira de Engenharia de Produção**, 2008. Disponivel em: <http://www.abepro.org.br/interna.asp?p=399&m=424&s=1&c=362>. Acesso em: Abril 2012.

BARROS FILHO, R.; TUBINO, D. F. **Anais do Encontro Nacional de Engenharia de Produção**, 1998.

BATISTA, E. O. **Sistemas de Informação:** O uso consciente da tecnologia para o gerenciamento. 1. ed. São Paulo: Saraiva, 2006. ISBN 978-85-02-04249-0.

BEAL, A. **Gestão estratégica da informação:** como transformar a informação e a tecnologia da informação em fatores de crescimento e de alto desempenho nas organizações. São Paulo: Atlas, 2008. 137 p. ISBN 978-85-224-3764-1.

CARPANEZ, J. Falha em site vende TVs de plasma e notebooks por R$ 9,90, 2009. Disponivel em: <http://g1.globo.com/Noticias/Tecnologia/0,MUL1160798-6174,00-FALHA+EM+SITE+VENDE+TVS+DE+PLASMA+E+NOTEBOOKS+POR+R.html>. Acesso em: 4 Março 2012.

CHARETTE, R. N. Why *software* fails? **IEEE Spectrum**, set. 2005. 42-49.

DIAS NETO, A. C. Introdução a Teste de *Software*. **Engenharia de *software* magazine**, n. 1, 2007.

EFX TECNOLOGIA. Mini PCP. Disponivel em: <http://www.minipcp.com.br/>. Acesso em: 1 Maio 2012.

FOINA, P. R. **Tecnologia de informação:** planejamento e gestão. 2. ed. São Paulo: Atlas, 2006. 339 p. ISBN 85-224-4372-6.

GOLDRATT, E. M.; SCHRAGENHEIM, E.; PTAK, C. **Necessária, sim, mas não suficiente:** uma estória do mundo empresarial baseada na teoria das restrições. São Paulo: Nobel, 2000. 252 p. ISBN 85-213-1260-1.

GOMES, C. F. S.; RIBEIRO, P. C. C. **Gestão da cadeia de suprimentos integrada à tecnologia da informação**. São Paulo: Thomson, 2004. 360 p. ISBN 85-221-0404-2.

HUMPHREY, W. S. **Managing the *software* process**. [S.l.]: Addison-Wesley, 1989. ISBN 0-201-18095-2.

KALAKOTA, R.; ROBINSON, M. **e-business:** estratégias para alcançar o sucesso no mundo digital. Tradução de Carlos Alberto Picanço Carvalho. 2. ed. Porto Alegre: Bookman, 2002. ISBN 85-7307-875-8.

LASTRES, H. M. M. Ciência e tecnologia na era do conhecimento: um óbvio papel estratégico? **Parcerias Estratégicas**, Brasília, 5, n. 9, out. 2000. 14-21.

LAURINDO, F. J. B.; ROTONDARO, R. G. **Gestão integrada de processos e da tecnologia da informação**. São Paulo: Atlas, 2006. 218 p. ISBN 85-224-4507-9.

MAGELA, R. **Engenharia de *software* aplicada:** fundamentos. Rio de Janeiro: Alta Books, 2006. 218 p. ISBN 857608123-7.

MIGUEL, P. A. C. et al. **Metodologia de pesquisa em engenharia de produção**. 2. ed. Rio de Janeiro: Elsevier: ABEPRO, 2012. ISBN 978-85-352-4891-3.

MOLINARI, L. **Testes de *software*:** produzindo sistemas melhores e mais confiáveis. 4. ed. São Paulo: Érica, 2008a. ISBN 978-85-7194-959-1.

MOLINARI, L. **Testes funcionais de *software***. Florianópolis: Visual Books, 2008b. 224 p. ISBN 978-85-7502-234-4.

MONTEIRO, A. Em reunião com Anac, Gol diz que programa causou erro na escala da tripulação. **Folha.com**, 3 agosto 2010. Acesso em: 12 mar. 2012.

MORAES, R. O.; LAURINDO, F. J. B. Teste de *software* e qualidade de *software*: uma visão geral. **XIX ENEGEP - Encontro Nacional de Engenharia de Produção**, Rio de Janeiro, 1999.

PEDROSO, M. C. Uma metodologia de análise estratégica da tecnologia. **Gestão & Produção**, São Carlos, v. 6, n. 1, p. 61-76, abr. 1999.

PEZZÈ, M.; YOUNG, M. **Teste e análise de *software*:** processos, princípios e técnicas. Tradução de Bernardo Copstein e Flavio Moreira Oliveira. Porto Alegre: Bookman, 2008. 512 p. ISBN 978-85-7780-262-3.

PFLEEGER, S. L. **Engenharia de *software*:** teoria e prática. Tradução de Dino Franklin. 2. ed. São Paulo: Prentice Hall, 2004. ISBN 978-85-87918-4.

PRATES, G. A.; OSPINA, M. T. Tecnologia da informação em pequenas empresas: fatores de êxito, restrições e benefícios. **Revista de Administração Contemporânea**, Curitiba, Junho 2004.

PRESSMAN, R. S. **Engenharia de *software***. São Paulo: Pearson Makron Books, 1995.

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

REZENDE, D. A. **Engenharia de *software* e sistema de informação**. 3. ed. Rio de Janeiro: Brasport, 2005. ISBN 85-7452-215-5.

RIBEIRO, V. G.; ZABADAL, J. R. S. **Pesquisa em computaçao:** uma abordagem metodológica para trabalhos de conclusão de curso e projetos de iniciação científica. Porto Alegre: Editora UniRitter, 2010. 203 p. ISBN 978-85600-47-7. Coleção experiência acadêmica.

ROSINI, A. M.; PALMISANO, A. **Administração de Sistemas de Informação e a Gestão do Conhecimento**. São Paulo: Pioneira Thomson Learning, 2003. ISBN 85-221-0312-7.

SANTOS, N. D. **A era do conhecimento:** os novos desafios para os profissionais de engenharia. [S.l.]: [s.n.], 2004.

SOMMERVILLE, I. **Engenharia de *software***. Tradução de Selma Shin Shimizu Menikoff; Reginaldo Arakaki e Edílson de Andrade Barbosa. 8. ed. São Paulo: Pearson Addison-Wesley, 2007. 549 p. ISBN 978-85-88639-28-7.

STAIR, R. M. **Princípios de Sistemas de Informação:** Uma abordagem gerencial. Tradução de Maria Lúcia Iecker Vieira e Dalton Conde de Alencar. 2. ed. Rio de Janeiro: LTC, 1998.

TONSIG, S. L. **Engenharia de *software*:** análise e projeto de sistemas. 2. ed. Rio de Janeiro: Editora Ciência Moderna Ltda, 2008. ISBN 978-85-7393-653-7.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. Rio de Janeiro: Elsevier, 2008. ISBN 978-85-352-3522-7.

YIN, R. K. **Estudo de caso:** planejamento e métodos. Tradução de Ana Thorell. 4. ed. Porto Alegre: Bookman, 2010. 248 p. ISBN 978-85-7780-655-3.

GLOSSÁRIO

Defeito: ato inconsistente cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta. Por exemplo, uma instrução ou comando incorreto. (IEEE 610, 1990 *apud* Dias Neto, 2007)

Erro: 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 de um programa constitui um erro. (IEEE 610, 1990 *apud* Dias Neto, 2007)

Falha: o comportamento operacional do *software* diferente do esperado pelo usuário. Uma falha pode ter sido causada por diversos erros e alguns erros podem nunca causar uma falha. (IEEE 610, 1990 *apud* Dias Neto, 2007)

Teste do *Software*: Verificação e validação do *software* através de sua execução em um processador. (MAGELA, 2006)

ANEXO A - AUTORIZAÇÃO DE USO DO *SOFTWARE*