ETIERP – Estratégia de Testes para Implantação de Sistemas ERP

Ângela Cristina de Oliveira

Trabalho Final (Mestrado Profissional)

ETIERP – Estratégia de Testes para Implantação de Sistemas ERP

Este exemplar corresponde ao trabalho final devidamente corrigido e defendido por Ângela Cristina de Oliveira e aprovado pela Banca Examinadora.

Campinas, 12 de dezembro de 2003.

Eliane Martins
(Orientadora)

Trabalho final apresentado ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Instituto de Computação
Universidade Estadual de Campinas

ETIERP – Estratégia de Testes para Implantação de Sistemas ERP

Ângela Cristina de Oliveira

Dezembro 2003

Banca Examinadora:

- Profa. Dra. Eliane Martins (Orientadora)
- Profa. Dra. Maria de Fátima Ridolfi Ordini Pires da Silva
C.C.E.U.C – UNICAMP
- Prof. Dr. Ricardo de Oliveira Anido
Instituto de Computação – UNICAMP
- Prof. Dr. Mario Lúcio Côrtes
Instituto de Computação - UNICAMP

“ Existe somente uma idade para a gente ser feliz, somente uma época na vida de cada pessoa em que é possível sonhar e fazer planos e ter energia bastante para realizá-los a despeito de todas as dificuldades e obstáculos.

Uma só idade para a gente se encantar com a vida e viver apaixonadamente e desfrutar tudo com toda intensidade sem medo nem culpa de sentir prazer.

Fase dourada em que a gente pode criar e recriar a vida à nossa própria imagem e semelhança e vestir-se com todas as cores e experimentar todos os sabores e entregar-se a todos os amores sem preconceito nem pudor.

Tempo de entusiasmo e coragem em que todo desafio é mais um convite à luta que a gente enfrenta com toda disposição de tentar algo NOVO, de NOVO e de NOVO, e quantas vezes for preciso.

Essa idade tão fugaz na vida da gente chama-se PRESENTE e tem a duração do instante que passa”.

Mário Quintana

Agradecimentos

Agradeço a,

Deus pela força e perseverança a mim concedidas para a realização deste trabalho.

A minha orientadora Eliane Martins pela paciência e dedicação.

Aos amigos Eduardo Dutra, Professor David Bianchini, Paul Redditt e meu irmão Carlos Eduardo de Oliveira, por terem colaborado e também por acreditarem na realização deste trabalho.

Especialmente ao meu namorado Cássio Lühmann pelo imenso apoio que recebi durante toda a minha trajetória.

Resumo

O teste de software é uma atividade crítica na garantia da qualidade do produto. Os seres humanos são passíveis de erros e a atividade de testes é essencial para diminuir estes possíveis erros que possam ser encontrados.

Com a evolução das ferramentas para desenvolvimento de software, estes vão se tornando cada vez mais complexos e abrangentes. Os sistemas de gerenciamento empresarial, também chamados de sistemas ERP (Enterprise Resource Planning) são bons exemplos. Os sistemas ERP costumam ser genéricos a fim de abranger empresas dos mais variados ramos de negócios, inclusive em diferentes países, conseqüentemente possuem um grande número de parâmetros para permitir tal flexibilidade. Desta forma, testes são necessários a fim de verificar se o sistema está funcionando de acordo com os parâmetros especificados, além de identificar possíveis erros que possam não ter sido identificados e corrigidos pela empresa que desenvolveu o sistema.

O objetivo deste trabalho é apresentar a ETIERP – Estratégia de Testes para Implantação de Sistemas ERP. Esta estratégia propõe a geração de casos de teste de forma estruturada e sistemática para a aplicação na implantação de um sistema ERP, baseando-se em artefatos da UML, como diagrama de casos de uso e diagrama de atividades. Neste trabalho, são apresentados os passos a serem realizados e sua aplicação é mostrada em um exemplo, extraído de um sistema real.

Palavras-chave: testes de software, testes de sistema, teste de sistema ERP, testes baseados em artefatos da UML.

Abstract

The software testing is a critical task in product's quality assurance. The human developers are susceptible to errors and the testing activity is essential to minimize this possible errors.

With the software development tools evolution, software is becoming more complex and providing greater coverage. The business management systems, also known as ERP (Enterprise Resource Planning) systems are a good example. ERP systems tend to be generic systems able to support different company types in different countries and consequently have a huge quantity of parameters to guarantee this flexibility. With this, the test phase is necessary to verify that the system is working according to the settings or even to avoid any serious errors that were not detected and corrected by the supplier.

The goal of this text is to present the ETIERP – Test Strategy for ERP Systems Implementation. This strategy proposes the test cases generation in a structured and systematic way to be apllied in ERP systems implementation basing on UML artifacts, as use case diagrams and activity diagrams. This text will present the steps to be realized and the strategy application that is shown is an example extracted from a real system.

Keywords: software testing, systems testing, ERP systems testing, UML based testing

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Atualmente a tecnologia evolui rapidamente. Surgem constantemente poderosas ferramentas para criação de software e também de variadas e poderosas plataformas de hardware para comportá-los. De posse destas ferramentas e do hardware adequado somos capazes de criar sistemas complexos e abrangentes. O sistema ERP (Enterprise Resource Planning), responsável por controlar todos os processos realizados numa grande empresa, é um bom exemplo.

O teste de software é uma atividade crítica na garantia da qualidade. Os seres humanos são passíveis de erros e a atividade de testes é essencial para diminuir os possíveis erros que possam ser encontrados.

A estratégia ETIERP (Estratégia de Testes para a Implantação de Sistemas ERP), proposta neste trabalho, foi criada sob o ponto de vista da empresa que adquire o sistema ERP. As empresas fornecedoras de sistemas ERP criam este software de forma mais genérica possível a fim de abranger empresas de qualquer ramo de negócios. Para atingir este objetivo as empresas desenvolvedoras se utilizam de muitos parâmetros que são alterados de acordo com o ramo de negócios e processos de trabalho do cliente.

A implantação do sistema ERP é complexa, envolve muitos recursos financeiros, além de muitas pessoas dedicadas integralmente ao projeto. A implantação costuma levar em torno de um ano e a fim de garantir o seu sucesso, entre outros pontos, é necessário que haja uma fase de testes responsável por verificar se as informações estão sendo geradas de acordo com os parâmetros especificados para a empresa em questão, além de identificar possíveis erros que porventura possam não ter sido identificados e corrigidos pela empresa fornecedora.

Supõe-se que a empresa fornecedora já tenha feito testes detalhados da estrutura do sistema, entre outros, portanto são desnecessários testes que dependam do conhecimento da estrutura interna do sistema. Além de desnecessário este tipo de teste não é possível de ser realizado pelo cliente, pois este não recebe a documentação sobre a estrutura do sistema ou mesmo sua especificação. O único material disponível como fonte para a geração dos casos de testes são os manuais do sistema e o conhecimento da equipe responsável pela

implantação do sistema. Desta forma os testes de sistemas são os testes ideais para a situação em questão.

A ETIERP foi baseada em artefatos da UML, como diagramas de casos de uso e diagramas de atividades. Os casos de uso basicamente representam as funcionalidades do sistema e podem ser especificados com facilidade mesmo por uma pessoa que não possua conhecimento da UML (Unified Modeling Language).

A motivação deste trabalho é complementar as metodologias de teste das empresas fornecedoras de sistemas ERP visando minimizar a quantidade de erros ou inconsistências que poderão ser encontradas pela empresa que adquire o software. Além dos erros que já existiam no sistema e inconsistências provocadas pela parametrização errada do sistema, podem ocorrer erros provenientes da customização do sistema, ou seja, acréscimo de novas funcionalidades ao pacote ou mesmo simplificação de determinadas funcionalidades. Normalmente o cliente é voltado para outro ramo de negócios e não possui domínio suficiente das técnicas e metodologias existentes para testes. Além disto, confia que a empresa fornecedora já realizou os testes necessários. Porém, é extremamente necessário que a empresa que adquire o software se previna de possíveis erros e também consiga verificar o resultado da parametrização que foi realizada.

A principal contribuição deste trabalho é organizar a fase de testes de forma estruturada e sistemática a fim de que esta fase não seja comprometida devido à perda de alguma informação importante que porventura pudesse vir a comprometer a qualidade dos testes e conseqüentemente a implantação do sistema.

Este trabalho está estruturado como a seguir. O capítulo 2 explica basicamente o que é um sistema ERP e as fases mais comuns numa implantação. O capítulo 3 aborda conceitos sobre testes: o que são, tipos, técnicas, fases, entre outros. O capítulo 4 aborda especificamente algumas técnicas de testes de sistema e discute quais foram utilizadas na estratégia proposta. O capítulo 5 aborda a ETIERP, proposta deste trabalho e finalmente o capítulo 6 apresenta as conclusões e trabalhos futuros. No Anexo I, são apresentadas todas as descrições dos casos de uso, no Anexo II, constam as tabelas de decisão dos casos de uso e no Anexo III, é realizado o estabelecimento de valores para as variantes de todos os casos de uso utilizados neste trabalho.

Capítulo 2

Sistemas ERP

Este capítulo aborda o sistema ERP. A seção 2.1 esclarece o que é um sistema ERP, a seção 2.2 aborda a estrutura deste tipo de sistema, a seção 2.3 enumera as vantagens e desvantagens do ERP, a seção 2.4 mostra as fases da implantação de um sistema ERP, a seção 2.5 apresenta alguns sistemas ERP existentes no mercado e a seção 2.6 apresenta uma conclusão sobre este capítulo.

2.1 O que é um sistema ERP?

O ERP – *Enterprise Resource Planning* é um conceito administrativo que evoluiu do MRP – Material Requirements Planning e MRPII – Manufacturing Resource Planning. Atualmente a maioria dos sistemas de gestão empresarial são baseados neste conceito e portanto, são chamados de sistemas ERP.

Devido a necessidade de reduzir os níveis de estoque surgiram os primeiros sistemas de MRP. Estes sistemas ofereciam uma visão integrada dos bens baseada no inventário disponível e nos períodos de reabastecimento. Detalhando um pouco mais, o MRP é o método para planejamento de todos os materiais necessários para o atendimento de um pedido de cliente. A mecânica de cálculo do MRP está diretamente ligada a estrutura de produtos, saldos de materiais em estoque, ordens em andamento e políticas de planejamento. Para o atendimento destas necessidades são geradas ordens de compra e fabricação considerando as datas das necessidades. O resultado deste cálculo é um plano de compra e um plano de fabricação para que o pedido do cliente seja atendido no prazo desejado.

Nos anos 80, o MRP evoluiu para MRP-II que tomava como base, além dos bens, outros recursos essenciais à produção, tais como mão-de-obra, máquinas, entre outros. Pode-se dizer que o MRP-II é o método para o planejamento efetivo de todos os recursos de uma indústria. Idealmente envolve o planejamento operacional em unidades, o planejamento financeiro em dinheiro, e possui capacidade de simulação para responder questões do tipo “e se...”. Compreende várias funções integradas: planejamento do negócio, plano de vendas e

operações, plano de produção, programação mestre de produção, planejamento das necessidades de materiais, planejamento das necessidades de capacidade, e bem como os sistemas de suporte à execução, tanto para capacidade quanto para materiais. As saídas desses sistemas são integradas com relatórios financeiros como o plano do negócio, projeção de compras, projeção de faturamento, e projeção de estoques em dinheiro.

Aperfeiçoando ainda mais a solução, foi criado o ERP (Enterprise Resource Planning). Além de permitir a gestão da manufatura, o ERP permitiu controlar toda a empresa, da produção às finanças, integrando e sincronizando todos os departamentos. O ERP identifica e planeja todos os recursos da empresa necessários para comprar, produzir, expedir e controlar os pedidos dos clientes.

O sistema ERP automatiza os processos de uma empresa de forma integrada, conseqüentemente eliminando interfaces complexas e caras entre sistemas não projetados para conversarem, gerando uma base de dados única, sem redundâncias encontradas nos sistemas anteriores.

A figura 2.1 mostra a evolução do conceito ERP e sua abrangência.

Figura 2.1 - Abrangência dos sistemas MRP, MRPII e ERP.

As informações chegam de maneira mais clara, segura e imediata, o que proporciona um controle maior de todo o negócio e, principalmente, de seus pontos vulneráveis: custos, controle fiscal e estoques.

Os sistemas ERP têm se destacado nos últimos anos como ferramenta essencial para a gestão empresarial. Primeiramente grandes empresas adotaram a ferramenta. Geralmente empresas que têm suas matrizes no exterior e que acabam implantando o sistema que se tornará padrão para todas as empresas do grupo, por decisão da matriz. Hoje o mercado destes sistemas já está alcançando empresas de menor porte. Cada vez mais empresas deixam de desenvolver suas próprias aplicações e adquirem ERP's de sólidos fornecedores.

O "bug do milênio", o erro que afetou sistemas antigos despreparados para a chegada do ano 2000, foi um grande propulsor para a disseminação das aplicações ERP, pois muitas empresas preferiram adquirir a solução ERP e se livrar a um só tempo dos computadores de grande porte, com altos custos, e do bug do milênio, a correr o risco de alterar o software existente.

2.2 Estrutura do sistema ERP

Os sistemas ERP são divididos em módulos, sendo que cada módulo abrange uma área administrativa específica: Financeira, Contábil, Materiais e assim por diante.

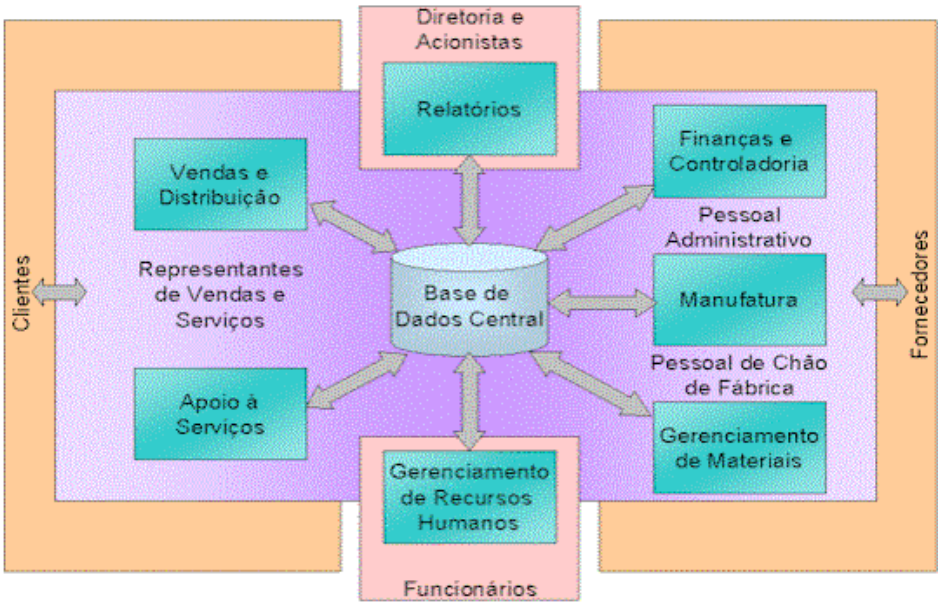

Figura 2.2 - Estrutura típica de funcionamento de um sistema ERP (DAVENPORT, 1998)

A figura 2.2 mostra a estrutura típica de um sistema ERP composta por vários módulos que acessam a mesma base de dados.

Do ponto de vista comercial, a empresa pode adquirir os módulos que forem de seu interesse. Ao se falar de processos verifica-se que estes atravessam vários módulos do ERP. Por exemplo, o processo de custos abrange os módulos de produção, de materiais, financeiro, controladoria e outros.

A implantação destes sistemas é geralmente realizada por uma equipe dividida em módulos, sendo que a integração destes módulos possibilita o fluxo dos processos dentro do sistema. Por exemplo, o módulo de planejamento e controle da produção requisita materiais ao estoque para fabricação e também disponibiliza o produto fabricado no estoque (módulo Estoque). Quando é realizada alguma compra de um produto qualquer (módulo de compras), este produto irá abastecer o estoque (módulo de Estoque). No recebimento do produto comprado são gerados os títulos a pagar (módulo de Contas a Pagar), o lançamento contábil (módulo de contabilidade) e o registro fiscal (módulo Fiscal). Na venda de determinado produto (módulo Faturamento), o produto é baixado do estoque (módulo Estoque). Na emissão da fatura referente à venda (módulo Faturamento) é gerado o título a receber (módulo Contas a Receber), o lançamento contábil (módulo de Contabilidade) e o registro fiscal (módulo Fiscal).

2.3 Vantagens e desvantagens do sistema ERP

Entre as vantagens de um sistema ERP estão a agilidade dos processos e também o fato de que as informações são colocadas no sistema somente uma única vez eliminando a repetição de tarefas. Por exemplo, no instante que um pedido é emitido imediatamente altera-se o planejamento de vendas, aciona-se a fábrica, compra-se a matéria-prima necessária e já é possível dar ao cliente uma boa estimativa do prazo de entrega. O fechamento contábil do mês, fase em que relatórios ficavam pelo menos 20 dias sendo passados de um departamento a outro, praticamente deixa de existir.

As operações manuais são eliminadas e conseqüentemente existe a redução de horas extras e também há a possibilidade de redução do quadro de funcionários. Quando uma empresa utiliza diferentes sistemas coexistindo através de uma interface ou não, as operações manuais são na maioria das vezes necessárias, pois existe a necessidade de se realizar uma constante conferência (conciliação ou compatibilização de informações entre sistemas) e ajustes (correção) de informações nestes sistemas.

Todos os módulos do ERP acessam uma única base de dados, o que evita a redundância. Desta forma as informações são consistentes e a tomada de decisão na empresa

também é beneficiada devido a esta base de dados comum. Conseqüentemente obtém-se a melhor satisfação dos clientes internos e externos. Além disto, existem as vantagens geradas pelo próprio conceito administrativo ERP, como redução de estoque, maior controle do processo produtivo, entre outros.

Entre as desvantagens dos grandes sistemas ERP, como o R/3 da SAP, é que se espera de um negócio mais disciplina do que a maioria dos negócios tem. O problema com o software é que ele é flexível demais. São mais de 60.000 transações, 15.000 tabelas de dados, com quase 2 milhões de telas possíveis. Ele foi feito para acomodar todo tipo de processo, todo tipo de negócio, da refinação de petróleo à revenda de automóveis usados. Isso gera dois problemas. O primeiro é encontrar, nesse emaranhado, a configuração que melhor se adapta a cada empresa. Depois, uma vez instalado o programa, é preciso que os dados sejam fornecidos corretamente por quem os usa, senão os erros também se propagam em tempo real.

Outro problema que tem afetado desde o início os grandes softwares de ERP, geralmente internacionais, é a adaptação às leis e práticas de negócios brasileiras, conhecida no jargão do ramo como localização ou tropicalização. Como se trata de um programa de escopo geral, especificidades locais são seu ponto fraco. Em alguns casos são adquiridos outros sistemas que complementam a funcionalidade não existente no ERP.

Um outro fator, que dependendo dos processos de trabalho da empresa, pode ser considerado como uma vantagem ou como uma desvantagem é o fato de que o ERP exige que a empresa se adapte ao sistema, ou seja, os sistemas ERP levam as empresas a modificar seus processos para se adequar aos descritos em seus módulos. O problema é maior ainda quando as empresas não têm seus processos mapeados. No entanto, empresas que possuem bons processos de negócios não irão ser beneficiadas com adaptações aos modelos do sistema. Já aquelas que possuem processos ultrapassados, com mau funcionamento, terão um grande benefício com tal adaptação. Este fato pode ser amenizado com as customizações. A customização é uma forma de manutenção adaptativa, que geralmente visa incrementar funcionalidades aos pacotes ou mesmo simplificar determinada funcionalidade. Entretanto, customizações não são fortemente recomendadas, uma vez que torna significativamente difícil a manutenção do sistema e a atualização de versões.

A própria integração gerada pelo sistema também pode ser vista como uma vantagem ou como uma desvantagem, pois apesar da integração ser benéfica, ela faz com que qualquer dado colocado no sistema por uma área específica tenha imediato impacto em todas as demais áreas em que esse dado será utilizado de alguma forma.

Cada empresa deve definir e avaliar quais serão os benefícios trazidos pelo uso do ERP, que está relacionado ao tipo de negócio da empresa, à situação atual de seus processos e sistemas.

2.4 Implantação de um ERP

Um projeto de implementação de um sistema de gestão empresarial é bastante complexo. Afeta a organização como um todo, inclusive nas operações do dia-a-dia. Para muitas empresas talvez esse seja o maior projeto em que já se envolveram.

Os projetos são na maior parte dos casos de longa duração. Em média costumam levar um ano. A implantação pode ser conduzida através de várias metodologias elaboradas pelas diversas consultorias atuantes neste campo. Entretanto as metodologias existentes são significativamente similares sendo divididas em fases que proporcionam os mesmos resultados.

Seguem abaixo as fases mais comuns na implantação do sistema ERP [1].

2.4.1 Planejamento do Projeto

A fase inicial é o planejamento do projeto quando são definidos o sistema ERP e módulos que serão implantados, a consultoria externa a ser contratada, a equipe interna , a estratégia de implantação e o cronograma. A equipe interna geralmente é composta por especialistas de cada área da empresa chamados usuários-chave (key-users) e também representantes do setor de tecnologia da informação. Esta equipe deve ser dedicada exclusivamente ao projeto de implantação.

A escolha do produto é um item relevante. Não há soluções idênticas, e empresas têm particularidades. Sendo assim nenhum produto é solução universal, isto é, não existe fornecedor do sistema perfeito, adequado para todos os clientes. Na verdade, avaliar sistemas ERP é mais uma questão de encontrar o menos inapropriado para a empresa.

2.4.2 Treinamento da Equipe

Na fase seguinte deve ocorrer o treinamento da equipe interna no software escolhido.

[1] http://www.kmpress.com.br/01ago02cefet.pdf

2.4.3 Levantamento de Processos

Nesta fase é feito o levantamento de cada processo empresarial atual seguido da alteração dos mesmos, para que se adapte as funcionalidades do sistema.

2.4.4 Configuração e Parametrização

Os sistemas ERP são genéricos e complexos, pois devem se adaptar às diversas variações de empresas existentes. Nesta fase, o ambiente deve ser preparado através de configurações e parâmetros para atender à área de atuação da empresa em questão.

2.4.5 Testes

Neste momento, devem ser realizados os testes do sistema. Inicialmente, testes das funcionalidades dos módulos, testes entre os vários módulos (teste dos processos), testes de interação, caso seja mantido alguma sistema atual da empresa ou adquirido algum sistema complementar.

2.4.6 Treinamento de Usuários

O usuários finais devem ser treinados nos módulos que irão utilizar. Geralmente o material para o treinamento é criado pelos usuários-chave e o treinamento é ministrado pelos mesmos.

2.4.7 Migração de Dados

A conversão ou migração de dados costuma ser realizada no mínimo duas vezes: a primeira antes dos testes do software para que eles possam ser executados e a segunda, dias antes da entrada em produção do novo sistema.

2.4.8 Sistema entra em Produção (Go-live)

Por fim, o sistema entra em produção. Este momento é chamado de “go live”.

Após a finalização da implantação não se pode dizer que o trabalho esteja concluído, pois sempre haverá a necessidade de se corrigir algum erro, atualizar a versão do sistema ou mesmo de se desenvolver alguma customização. Podemos dizer que o sistema estará numa constante fase de manutenção.

Desta forma, mesmo após a implantação do ERP, a atividade de testes (testes de regressão) se torna uma necessidade rotineira a ser executada antes de uma atualização de versão ou quando há alguma modificação no sistema.

2.5 Algumas empresas fornecedoras de sistemas ERP

Seguem abaixo algumas empresas fornecedoras de sistemas ERP:

- SAP: Em 1972, cinco engenheiros de sistemas decidiram abandonar seus empregos e, apostando numa idéia bastante extravagante para a época, abrir uma nova empresa. Surgia a SAP (Systemanalyse und Programmentwicklung.-Sistemas, Aplicações e Produtos para Processamento de Dados). A idéia por trás dessa iniciativa era a criação de uma solução única, totalmente integrada, capaz de automatizar todos os processos inerentes a uma empresa. 29 anos e 10 milhões de usuários depois, a SAP surge como a terceira maior empresa de software do mundo – a primeira em software de gestão empresarial-, empregando mais de 24 mil pessoas em 50 países diferentes e contando com mais de mil parceiros. A subsidiária brasileira, no país desde 1995, compartilha do sucesso do grupo com mais de 400 clientes[2].
- Oracle: A Oracle Corporation (Nasdaq: ORCL) é a maior empresa de software empresarial do mundo, fornecendo produtos para as maiores e mais bem-sucedidas empresas do planeta. Com faturamento anual superior a 10,8 bilhões de dólares, a empresa oferece seus produtos de bancos de dados, ferramentas e aplicativos, bem como serviços relacionados de consultoria, treinamento e suporte. Com matriz em Redwood Shores, na Califórnia, a Oracle é a primeira empresa de software a desenvolver e distribuir software empresarial 100% habilitado para a internet em toda a sua linha de produtos: banco de dados, servidor, aplicativos de negócios empresariais e ferramentas para desenvolvimento de aplicativos e suporte a decisões. O Oracle E-Business Suite é um conjunto completo de aplicativos de negócios que gerencia e automatiza os processos de uma organização[3].
- PeopleSoft: A PeopleSoft é líder mundial em aplicações para e-business. As soluções incluem Customer Relationship Management (CRM), Cadeira de Suprimentos (Supply Chain), Recursos Humanos, Finanças e soluções que auxiliam o completo gerenciamento empresarial. Além das aplicações eBusiness a PeopleSoft fornece consultoria, e treinamento para grandes e médias empresas que desejam soluções de negócios flexíveis e de

[2] http://www.sap.com/brazil/
[3] http://www.oracle.com.br/corporation/

implementação rápida. Fundada em 1987, com sede em Pleasanton, Califórnia, a PeopleSoft conta hoje com mais de 5.200 clientes e mais de 8000 funcionários em cerca de 140 países. A missão da PeopleSoft é fornecer soluções inovadoras que se adaptam às mudanças do mundo dos negócios e demandas das organizações. Atualmente a empresa PeopleSoft adquiriu a JD.Edwards que também comercializa sistemas de gerenciamento empresarial[4].

- A Microsiga Software é uma empresa brasileira especializada no desenvolvimento tecnológico e sistemas de gerenciamento empresarial (ERP/CRM), focando no setor corporativo. A empresa celebrou seu vigésimo aniversário em 2003. A empresa está estruturada em mais de 50 pontos distribuídos entre Brasil, Argentina, Chile, México, Paraguai, Porto Rico e Uruguai. Seu sistema é o Advanced Protheus[5].

2.6 Conclusão

Freqüentemente as empresas têm grandes expectativas em relação aos sistemas ERP. Espera-se que eles melhorem todo o funcionamento do negócio do dia para noite. No entanto, esta é uma percepção errada, e uma das razões é o fato de que o ERP não resolve problemas relacionados a procedimentos, ou seja, um sistema não é capaz de solucionar problemas decorrentes da falta ou do não cumprimento de procedimentos internos. O ERP permite que a empresa padronize seu sistema de informação, além de fornecer grande agilidade e confiabilidade das informações, evitando e eliminando retrabalhos e operações manuais antes amplamente executadas.

Apesar da implementação ser difícil, demorada e cara devido principalmente à extensa parametrização, não há dúvida dos benefícios do ERP nos aspectos antes mencionados. No entanto, para que o projeto tenha o sucesso almejado, deve-se realizar uma boa avaliação na escolha do software, na definição do escopo e da estratégia de implantação, procurando elaborar um plano de implantação de acordo com as necessidades da empresa proporcionando desta forma as vantagens desejadas.

Independente do sistema que usa, uma grande empresa possui necessariamente uma identidade, uma personalidade própria. É claro que o ERP deve ser aplicado, mas o sucesso de uma empresa, além de toda sua competitividade, vem também de sua personalidade, de sua maneira de participar do mercado. O sistema existe para incrementar a empresa e a razão de ser do negócio jamais será satisfazer os requisitos de um sistema.

---

[4] http://www.peoplesoft.com.br/br/pt/about/index.jsp
[5] http://www.microsiga.com.br

Sem dúvida, a integração da empresa é a arma que ajudará na sua sobrevivência, mas não necessariamente o que a fará vencedora. A competição entre as empresas que desenvolvem o ERP com certeza tratará deste assunto, oferecendo maneiras individualizadas de manter a personalidade do negócio. A total integração também é um assunto teórico pois, por maiores que sejam os esforços neste sentido, é bem remota a possibilidade de que um sistema consiga suprir em 100% as necessidades de uma empresa.

Capítulo 3

Testes de Software

Este capítulo aborda conceitos, fases e técnicas de teste de software. A seção 3.1 apresenta alguns conceitos sobre testes de software, a seção 3.2 enumera as fases de um processo de testes, a seção 3.3 apresenta algumas estratégias de testes, a seção 3.4 aborda os métodos de testes e a seção 3.5 aborda algumas técnicas de testes de caixa preta e a seção 3.6 faz um resumo do capítulo.

3.1 Alguns conceitos sobre testes de software

3.1.1 O que são testes?

Testes são um conjunto de técnicas que são aplicadas ao software, geralmente em etapas finais do processo da engenharia de software, com o objetivo de diminuir as chances de que o software contenha erros. Não é possível afirmar que após passar por uma série de testes o software não conterá erros, porém é possível minimizar estes erros.

3.1.2 Porque testes são necessários?

Segundo Sommerville, a atividade de testes de um programa tem dois objetivos: o primeiro é comprovar que o programa está de acordo com sua especificação e o segundo é exercitar o programa de tal maneira que os erros fiquem expostos [Sommerville,92-426].

Como seres humanos estamos sujeitos a erros. É errado afirmar que um software não contém nenhum erro, pois com certeza eles estarão lá. A execução de testes consegue verificar se o software está funcionando de acordo com sua especificação, porém se o projeto foi especificado de forma errada e construído de acordo com esta especificação, o teste não conseguirá reverter esta situação, ou seja, o processo de testes não irá influenciar na qualidade do software.

3.1.3 Quem realiza os testes?

A análise e projeto do software podem ser vistas como tarefas construtivas, sob o ponto de vista psicológico. Porém, a atividade de testes, sob o ponto de vista do desenvolvedor, pode ser considerada destrutiva, pois os casos de teste visam encontrar erros, ou seja, "demolir" o software que está sendo desenvolvido. A fim de evitar o conflito de interesses recomenda-se que o desenvolvedor seja responsável por testar as unidades individuais (módulos) do programa, se certificando que cada uma execute a função para a qual foi projetada. Em muitos casos, o desenvolvedor pode também executar o teste de integração, que é a etapa de teste dedicada ao teste da estrutura completa do programa. Somente depois que a arquitetura do programa foi totalmente testada é que uma equipe de testes independente deve ser envolvida. O desenvolvedor e a equipe de testes trabalham próximos, enquanto os testes são conduzidos pela equipe de testes, o desenvolvedor deve estar disponível para corrigir os erros que estão sendo descobertos [Pressman, 92-634].

3.1.4 O que se pode concluir com o resultado dos testes?

Segundo Pressman, se erros graves que necessitam da alteração do projeto forem encontrados com regularidade, a qualidade do software e confiabilidade são suspeitos e testes adicionais são indicados. Se por outro lado, o software aparenta estar funcionando adequadamente e os erros encontrados são fáceis de serem corrigidos, pode se concluir uma das duas alternativas: (1) a qualidade e confiabilidade são aceitáveis ou (2) os testes são inadequados para descobrir erros graves! Finalmente, se os testes não descobrirem erros, existem poucas dúvidas de que a configuração de testes não esteja adequada e de que os erros estão escondidos no software [Pressman, 92-598].

3.2 Fases de um processo de testes

Segundo Sommerville, exceto para pequenos programas, o sistema não deve ser testado como uma unidade. Grandes sistemas são compostos por subsistemas e estes subsistemas por módulos, que por sua vez são compostos por procedimentos e funções. Seguem abaixo as fases mais utilizadas em um processo de testes [Sommerville, 92-376]:

3.2.1 Teste de Unidade (Unit Testing) – cada componente é testado individualmente para a certificação de que elas estejam operando corretamente.

3.2.2 Teste de Módulo (Module Testing) –

Um módulo é uma coleção de componentes dependentes, como uma classe de objetos, um tipo de dados abstrato, uma coleção de procedimentos e funções. Um módulo encapsula componentes relacionados, conseqüentemente podem e devem ser testados sem outros módulos do sistema.

3.2.3 Teste de subsistema (Sub-system Testing) –

Envolve o teste de coleções de módulos que serão integrados num subsistema. Um dos erros mais comuns nos grandes sistemas são os erros de interface entre subsistemas, portanto o teste de subsistema deve exercitar rigorosamente estas interfaces.

3.2.4 Teste de Sistema (System Testing) –

Os subsistemas são integrados para formarem o sistema inteiro. Nesta fase, o processo de testes se preocupa em encontrar erros que normalmente resultam de interações antecipadas entre subsistemas e componentes. Também se preocupa com a validação do sistema nos seus requisitos funcionais e não-funcionais.

3.2.5 Teste de Aceitação (Acceptance Testing) –

Este é o estágio final do processo de testes, antes que o sistema seja aceito para uso operacional. Esta fase envolve testar o sistema com dados fornecidos pelo usuário (proprietário) do sistema. O teste de aceitação freqüentemente revela erros e omissões de definição dos requisitos do sistema.

3.2.6 Testes de Regressão (Regression Testing) –

O teste de regressão é utilizado quando ocorrem atualizações em um sistema, como correções em componentes ou incorporação de novos componentes, mesmo que o componente em questão tenha sido testado e aprovado e esteja sendo atualizado/incorporado em um sistema previamente testado e aprovado, podemos ainda assim obter erros, pois um componente novo ou modificado pode falhar quando usado com componentes não modificados [Binder, 00-756]. A alteração de Componentes ou a inclusão de novos componentes é uma grande fonte de problemas na implantação de um sistema ERP.

Depois que um erro é descoberto, ele deve ser corrigido e o sistema testado novamente. A princípio, todos os testes deveriam ser repetidos depois de alguma correção, porém o custo é muito alto. Na prática, quando uma alteração é feita, é necessário somente executar uma sub-parte de todo o teste para certificar-se que o componente modificado e suas dependências não possuem novos erros gerados pela correção.

3.3 Estratégias de testes

3.3.1 Estratégia Top-down

A estratégia top-down testa o alto nível do sistema antes de testar seus componentes detalhadamente. O programa é representado por um componente abstrato com sub-componentes representados por *stubs*. Os programas *stubs* podem ser implementados como uma versão simplificada de um componente requerido. O *stub* simplesmente solicita que seja introduzido um valor adequado ou simula a ação de um componente. Depois que os componentes de alto nível são testados, seus sub-componentes são implementados e testados da mesma forma. Este processo continua recursivamente até que os componentes de baixo nível sejam implementados. A vantagem desta estratégia é que o sistema funcionando fica disponível num estágio inicial de desenvolvimento. O processo de validação pode começar antes. Os erros de projeto podem ser detectados num estágio inicial do processo de testes. A detecção dos erros mais cedo significa evitar que a fase de implementação, e eventualmente a de projeto, tenham que ser refeitas. Quanto às desvantagens, se o componente é muito complicado, se torna impraticável produzir um *stub* que o simule perfeitamente. Esta estratégia não é apropriada para sistemas orientados a objetos, pois coleções de objetos não são usualmente integrados numa severa forma hierárquica.

3.3.2 Estratégia Bottom-up

A estratégia bottom-up é o contrário da top-down. O teste começa com os módulos de baixo nível na hierarquia e vai subindo na hierarquia até que o módulo final seja testado.

As vantagens da estratégia bottom-up são as desvantagens da top-down e vice-versa. Quando o teste bottom-up é utilizado, “test drivers” devem ser desenvolvidos a fim de exercitar os componentes de baixo nível. Estes “test drivers” simulam o ambiente do componente sendo testado.

Erros na arquitetura costumam ser descobertos depois que muito do sistema já foi testado. As correções destas falhas podem envolver escrever e testar novamente os módulos de baixo nível do sistema. Devido a este problema, a estratégia de testes bottom-up foi criticada pelos proponentes do desenvolvimento top-down. Porém, o teste bottom-up dos componentes críticos de baixo nível do sistema, é sempre necessário.

3.4 Métodos de testes

3.4.1 Teste estrutural ou teste de caixa branca

Os testes são derivados do conhecimento da estrutura do programa e implementação. O engenheiro de software pode analisar o código e usar seu conhecimento sobre a estrutura do componente para derivar dados de teste. A vantagem do teste estrutural é que os casos de teste podem ser derivados sistematicamente e a abrangência dos testes pode ser medida [Sommerville, 92-432]. Os testes de caixa branca devem garantir que todos os caminhos independentes dentro de um módulo tenham sido exercitados pelo menos uma vez, todas as decisões lógicas para valores falsos ou verdadeiros tenham sido exercitadas, executem todos os laços em suas fronteiras e dentro de seus limites operacionais e exercitem as estruturas de dados internas para garantir sua validade.

3.4.2 Teste funcional ou teste de caixa preta

Os testes são derivados da especificação do programa. Este tipo de teste é chamado de teste de caixa preta, pois o componente é uma caixa preta cujo comportamento só pode ser determinado pela análise das entradas e saídas relacionadas. É uma abordagem complementar para o teste de caixa branca. Este tipo de teste tende a ser aplicado nas últimas etapas da atividade de testes [Sommervile, 92- 428].

A seção 3.5 deste capítulo aborda somente técnicas de testes de caixa preta. A razão para tal é que a estratégia apresentada neste trabalho se utiliza somente de testes de caixa preta, pois testes da estrutura do sistema ou caixa branca são desnecessários ou impossíveis de se realizar pela empresa cliente, como abordado na Introdução – Capítulo I.

3.5 Técnicas de testes de caixa preta

3.5.1 Particionamento de equivalência

A partição de equivalência é uma técnica que determina quais classes de dados de entrada tem propriedades comuns. O programa deve se comportar da mesma maneira para todos os elementos de determinada partição de equivalência. Existem partições de equivalência para entradas e saídas.

A partição de equivalência pode ser identificada pela especificação do programa ou documentação do usuário e pela experiência do testador em prever quais classes de entradas de dados são possíveis de detectar erros. Como exemplo, se a especificação de uma entrada

diz que algum valor de entrada deve ser um inteiro de 5 dígitos, isto é, entre 10.000 e 99.999, a partição de equivalência pode ser valores menores que 10.000, valores entre 10.000 e 99.999, e valores maiores que 99.999.

A técnica de partição de equivalência é útil para seleção de instâncias de cada possível entrada para teste. Entretanto, mesmo quando um programa funciona adequadamente para entradas individuais de testes, combinações destas entradas podem detectar erros no programa. A partição de equivalência não ajuda na seleção destas combinações. Entretanto, testadores experientes costumam desenvolver seu próprio conjunto de combinações que possam descobrir o erro [Sommerville, 92-428].

3.5.2 Análise de valor limite (BVA - Bondary Value Analysis)

Esta técnica leva a escolha de casos de teste que põem à prova os valores fronteiriços, onde costumam ocorrer o maior número de erros. A análise do valor limite completa a técnica de particionamento de equivalência. A BVA também deriva casos de testes do domínio de saída. Devem ser exercitados os valores máximos e mínimos, os valores logo abaixo deles e logo acima respectivamente, relatórios de saída que produzam o número máximo de entradas permitidas numa tabela, casos de teste que exercitem o número máximo de entradas de um array, entre outros.

3.5.3 Técnicas de grafo de causa e efeito

O ponto fraco das técnicas de teste de valor limite e de classes de equivalência é a dificuldade de explorar combinações de valores de entrada. O maior problema em se combinar valores de entrada é que a quantidade de combinações costuma ser enorme e por essa razão, se não dispusermos de uma maneira sistemática de selecionar as combinações, provavelmente obteremos um conjunto pobre de casos de teste. Já a técnica de testes baseada em grafos de causa e efeito é uma técnica que auxilia na seleção sistemática de casos de teste oferecendo uma representação concisa das condições lógicas e das ações correspondentes. Enfim, é uma representação formal para a qual a especificação em linguagem natural é convertida. A partir do grafo de causa-efeito uma tabela de decisão[6] pode ser desenvolvida. Dados de caso de teste são selecionados, de forma que cada regra da tabela seja exercitada.

---

[6] Tabela de decisão: Uma tabela de decisão é composta por duas partes. A seção de condições que lista as condições e combinações de condições. Variáveis de decisão são entradas ou fatores de ambiente referenciadas na seção de condições. Uma condição expressa um relacionamento entre variáveis de decisão e devem ser "verdadeiro" ou "falso". Cada combinação de condições e ações é uma variante. Quanto todas as condições individuais numa variante são verdadeiras, então a ação correspondente deverá ser produzida. A seção da ação lista respostas que deverão ser produzidas quando combinações de condições são verdadeiras. Qualquer

Para a construção do grafo, devem ser executados os passos a seguir:

1. A especificação é dividida em unidades ou módulos manipuláveis (será desenhado um grafo para cada unidade ou módulo distinto);

2. As causas e os efeitos de cada módulo da especificação são identificados. Uma *causa* é uma condição de entrada ou uma classe de equivalência sobre variáveis de entrada. Um *efeito* é uma condição de saída ou uma transformação no estado do sistema (exemplos: alteração de um registro em um arquivo, exibição de uma mensagem). Para cada causa e para cada efeito é atribuído um identificador inteiro único.

3. O conteúdo semântico da especificação é analisado e transformado em um grafo booleano ligando as causas e os efeitos (o grafo de causa e efeito).

4. O grafo é anotado com as restrições que descrevem as combinações de causas/efeitos que são impossíveis em virtude de restrições sintáticas ou semânticas.

5. Percorrendo metodicamente o grafo, o mesmo é convertido em uma tabela de decisão de entradas limitadas. Cada coluna na tabela irá representar um caso de testes.

6. Convertem-se as colunas da tabela em casos de teste.

O próximo passo é a geração da tabela de decisão. O procedimento utilizado é o seguinte:

1. Selecione um efeito que deseja testar;

2. Percorrendo o grafo do efeito para a causa, encontre todas as combinações de causas (eliminando as inibidas pelas restrições) que fazem o efeito ter o valor 1;

3. Crie uma coluna na tabela de decisão para cada combinação de causas.

3.5.4 Teste de recuperação

É um teste que provoca falhas no software (falhas no disco, falhas de interface, memória insuficiente) a fim de avaliar se a recuperação do sistema é adequadamente executada. Se a recuperação for automática (realizada pelo próprio sistema), a reinicialização, recuperação de dados e reinicio são avaliados. Se a recuperação exigir intervenção humana, o tempo médio até o reparo também é avaliado para determinar se ele se encontra dentro de limites aceitáveis. É desejável que o sistema seja capaz de se recuperar rapidamente e com um mínimo de intervenção humana [Inthurn, 01-62].

---

número de ação resultante pode ser especificado e qualquer combinação de ação pode ser associado com qualquer variante [Binder, 00-124].

3.5.5 Teste de segurança

O teste de segurança tenta verificar se todos os mecanismos de proteção embutidos num sistema o protegerão, de fato, de acessos indevidos. O teste de segurança tem por objetivo garantir que o sistema se comporte adequadamente perante acessos indevidos ou ilegais ao sistema. Para testar a segurança, os testadores devem tentar penetrar no sistema durante a recuperação do mesmo, após uma falha, obter senhas ilegalmente, após a inserção de uma falha intencional, entre outros [Inthurn, 01-63].

3.5.6 Teste de estresse

O teste de estresse avalia como o software se comporta perante volumes anormais, desta forma, o sistema é carregado com volumes não usuais com intenção de pará-lo. Neste momento é monitorada a perda de desempenho do sistema e a sua suscetibilidade à falhas durante estas cargas. Se o sistema pára como resultado de uma alta carga, este deverá passar por mais alguns testes de recuperação. O testador deve utilizar o sistema com recursos em quantidade, freqüência e volume anormais, tais como procuras excessivas de dados em disco, aumento excessivo de índice de dados, abertura de muitas janelas e utilização de arquivos com formato não compatível aos esperados pelo programa, entre outros [Inthurn, 01-64].

3.5.7 Teste de desempenho

Para sistemas de tempo real, um software que ofereça as funções exigidas, mas não atinja os requisitos de desempenho, é inaceitável. O teste de desempenho é, às vezes, combinado com teste de estresse e freqüentemente exige instrumentação de hardware e de software, ou seja, muitas vezes é necessário medir a utilização dos recursos rigorosamente.

O teste de desempenho envolve o monitoramento e o registro dos níveis de desempenho durante cargas de estresse regulares, baixas e altas. Ele testa o conjunto de recursos utilizados a cerca das condições descritas e serve como base para se efetuar uma previsão dos recursos adicionais necessários no futuro. É importante notar que os objetivos do teste de desempenho devem ter sido desenvolvidos durante o estágio de planejamento e o teste de desempenho visa garantir que estes objetivos estão sendo atingidos. Entretanto, estes testes podem ser executados em estágios iniciais da produção a fim de comparar o atual uso para os cálculos previstos [Inthurn, 01-64].

3.6 Resumo

O capítulo anterior abordou o sistema ERP. Neste capítulo foram apresentados os conceitos básicos de testes de software, as fases de um processo de testes, as estratégias top-down e bottom-up, os métodos de testes caixa-preta e caixa-branca e algumas técnicas do método de testes de caixa-preta. A razão para a apresentação de somente técnicas de teste funcional ou de caixa-preta, é que a estratégia ETIERP, proposta neste trabalho, se utiliza somente de técnicas de teste de caixa-preta, pois a empresa que adquire o sistema ERP, não dispõe do código-fonte ou documentação sobre a estrutura do sistema.

Nos capítulos seguintes serão apresentados alguns testes de sistema, a estratégia ETIERP e a conclusão e trabalhos futuros.

Capítulo 4

Testes de Sistema

Os testes de sistema envolvem os testes dos aspectos funcionais e não-funcionais do sistema. Neste capítulo são apresentadas algumas técnicas estudadas, baseadas em casos de uso, para testes dos aspectos funcionais do sistema. A razão para tal é que a estratégia proposta neste trabalho se utiliza somente das técnicas que testam o aspecto funcional, se baseando em algumas das técnicas apresentadas neste capítulo. Na seção 4.1 é apresentado o teste de caso de uso estendido, a seção 4.2 aborda o teste de cobertura em CRUD, a seção 4.3 aborda o teste baseado em perfil, a seção 4.4 aborda a metodologia TOTEM, a seção 4.5 apresenta a estratégia ETACS e finalmente a seção 4.6 apresenta a conclusão do capítulo, abordando o que foi utilizado deste capítulo na estratégia proposta.

4.1 Teste de caso de uso estendido

Os casos de uso podem ser aplicados para captar o comportamento pretendido do sistema que está sendo desenvolvido, sem ser necessário especificar como esse comportamento é implementado. Os casos de uso fornecem uma maneira para os desenvolvedores chegarem a uma compreensão comum com os usuários finais do sistema e com os especialistas do domínio. Além disto os casos de uso servem para ajudar a validar a arquitetura e para verificar o sistema à medida que ele evolui durante o desenvolvimento [Jacobson et al, 00-217]. O caso de uso é um dos itens da UML (Unified Modeling Language). A UML é uma linguagem padrão para elaboração da estrutura de projetos de software e pode ser empregada para visualização, especificação, construção e documentação dos artefatos que façam uso de sistemas complexos de software [Jacobson et al, 00-13]. A UML nasceu da unificação dos métodos Booch (de Grady Booch), OOSE (Object-Oriented Software Engineering, de Ivar Jacobson) e OMT (Object Modeling Technique, de James Rumbaugh).

Os casos de uso como definidos na UML, OOSE e outras metodologias orientadas a objetos, são necessários mas não suficientes para o projeto de teste de sistema. Nós devemos determinar os seguintes itens adicionais:

- O domínio de cada variável que participa do caso de uso;
- Os relacionamentos de entrada/saída requeridos entre variáveis de caso de uso;
- A freqüência relativa de cada caso de uso;
- Dependências seqüenciais entre os casos de uso.

Os casos de uso estendidos fornecem esta informação adicional. Variáveis operacionais são analisadas para construir uma relação operacional. Freqüências relativas são desenvolvidas pela construção de um perfil operacional. Esta abordagem permite a geração sistemática de casos de testes [Binder, 00-281]. Os casos de uso estendidos podem ser aplicados em qualquer escopo para o qual casos de uso possam ser escritos.

É possível desenvolver casos de teste imaginando idéias específicas para um caso de uso e os resultados esperados correspondentes. Normalmente, um grande número de idéias podem ser enumeradas, porém testes eficientes e efetivos requerem que sejam modeladas restrições e relacionamentos, além de geração de casos de testes suficientes para que se tenha certeza de que estas restrições e relacionamentos tenham sido exercitados.

Na tabela 4.1, são descritos alguns casos de usos, atores[7] e cenários para o caixa automático:

Caso de uso	Ator	Possíveis combinações de entrada/saída
Sessão estabelecida	Cliente do banco	(1) Senha errada informada uma vez; senha correta informada; exibir menu. (2) Senha correta; o banco está fora de serviço. Exibir "tente mais tarde". (3) Senha correta; conta do cliente está encerrada. Exibir "contate seu banco". (4) Cartão roubado é inserido; senha correta informada. Cartão retido.
Retirada de dinheiro	Cliente do banco	(1) $50 requeridos; conta aberta; saldo $1,234.56; $50 liberados. (2) $100 requeridos; conta encerrada. (3) $155.39 requeridos; conta aberta; saldo $150; Exibir "saldo insuficiente".
Reposição de dinheiro	Operador do caixa automático	(1) Caixa automático aberto; caixa não contém dinheiro; $15.000 são adicionados. (2) Caixa automático aberto; caixa contém dinheiro.

Tabela 4.1 – Alguns casos de uso e cenários para o caixa automático

[7] Ator: um ator representa um conjunto coerente de papéis que os usuários de casos de uso desempenham quando interagem com estes casos de uso. Tipicamente, um ator representa um papel que um ser humano, dispositivo de hardware ou até outro sistema desempenha com o sistema em questão[Jacobson et al, 00-221].

Um caso de uso especifica um conjunto de respostas que serão produzidas para combinações específicas de entrada externa e estados do sistema. A implementação típica de um caso de uso é colaboração: uma corrente de mensagens entre objetos e componentes. Nós devemos exercitar todas as colaborações que levem o caso de uso a falhar. Exercitar as combinações de condições e os valores limites das variáveis operacionais é a estratégia mais eficiente para revelar a falha.

De posse dos casos de uso estendidos, os casos de teste são desenvolvidos em quatro passos, mostrados abaixo.

4.1.1 Identificar as variáveis operacionais

Variáveis operacionais são fatores que variam de um cenário para outro e determinam diferentes e significantes respostas do sistema. Elas incluem:

- Entradas e saídas explícitas;
- Condições do ambiente que resultam em comportamento diferente e significante do ator;
- Abstrações do estado do sistema (fora de serviço, pronto, entre outros).

Como exemplo, no caso de uso “Sessão estabelecida”, quatro variáveis determinam a resposta do caixa automático para o cliente que insere o cartão e informa a senha: o código inserido no cartão, a senha informada pelo usuário, a resposta do banco e a condição da conta bancária do cliente.

4.1.2 Definir os domínios das variáveis operacionais

Os domínios são desenvolvidos pela definição do conjunto dos valores válidos e inválidos para cada variável. Por exemplo, o domínio de um código bancário é quatro dígitos que varia de 0000 a 9999.

4.1.3 Desenvolvimento da relação operacional

Relacionamentos entre as variáveis operacionais que determinam classes distintas de resposta do sistema, são representadas em uma tabela de decisão. Na tabela de decisão, quando todas as condições numa linha são verdadeiras, a ação esperada deve ser produzida. Cada linha na tabela de decisão é uma variante. Somente uma variante deve ser verdadeira para qualquer possível conjunto de variáveis operacionais.

Variáveis operacionais					Resultado esperado	
Variante	**Código do cartão**	**Código informado**	**Resposta do banco**	**Situação da conta do cliente**	**Mensagem**	**Ação do cartão**
1	Inválido	Não importa	Não importa	Não importa	Insira o cartão do banco	Liberado
2	Válido	Código correspondente	Banco responde	Encerrada	Contate seu banco	Liberado
3	Válido	Código correspondente	Banco responde	Aberta	Selecione uma transação	Nenhuma
4	Válido	Código correspondente	Banco não responde	Não importa	Por favor tente mais tarde	Liberado
5	Válido	Não correspondente	Não importa	Não importa	Informe a senha novamente	Nenhuma
6	Revogado	Não importa	Banco responde	Não importa	Cartão inválido	Retido
7	Revogado	Não importa	Banco não responde	Não importa	Cartão inválido	Liberado

Tabela 4.2 – Relação operacional para o caso de uso sessão estabelecida

Na tabela 4.2, a primeira linha indica que quando um código de cartão está inválido, o cartão é imediatamente liberado e a mensagem “insira o cartão” é exibida.

4.1.4 Desenvolver casos de testes

O objetivo do passo 4 é estabelecer valores que testem o sistemas em situações normais e também em situações de erro. Para isto, cada variante deve ser feita verdadeira ao menos uma vez e falsa ao menos uma vez. Este passo requer ao menos dois casos de teste para cada variante: (1) um caso de teste verdadeiro num conjunto de valores que satisfaçam todas as condições numa variante, e (2) um caso de teste falso – alteração nos valores que façam com que ao menos uma condição torne-se falsa. Freqüentemente, o caso falso para uma variante será qualificado como caso verdadeiro para outra variante. A intenção do passo 4 é testar o sistema

Variáveis operacionais					Resultado esperado	
Variante	**Código do cartão**	**Código informado**	**Resposta do banco**	**Situação da conta do cliente**	**Mensagem**	**Ação do cartão**
1	Inválido	Não importa	Não importa	Não importa	Insira o cartão do banco	Liberado
1V	%@#*				Insira o cartão do banco	Liberado
1F	Qualquer teste verdadeiro para as variantes 2 ou 3					
2	Válido	Código correspondente	Banco responde	Encerrada	Contate seu banco	Liberado
2V	1234	1234	Banco responde	conta encerrada	Contate seu banco	Liberado
2F	Qualquer testes verdadeiro para as variantes 1 ou 3					
3	Válido	Código correspondente	Banco responde	Aberta	Selecione uma transação	Nenhuma
3V	1234	1234	Banco responde	conta aberta	Selecione uma transação	Nenhuma
3F	Qualquer teste verdadeiro para as variantes 1 ou 2					

Tabela 4.3 – Conjunto mínimo de teste para o caso de uso sessão estabelecida

A tabela 4.3 mostra o conjunto mínimo de testes somente para as variantes de 1 a 3. A variante 1V é composta por valores que fazem com que a variante 1 seja verdadeira e o 1F é composta por valores que fazem com que a variante 1 seja falsa.

4.1.5 Desvantagens

Uma das desvantagens deste tipo de teste é a possibilidade de ocorrerem divergências de opinião em relação ao nível de abstração apropriado e grau de especificação para um caso de uso. O projetista de testes deve resolver as ambigüidades. Muitos detalhes ou detalhes insuficientes irão criar trabalho extra para o testador.

Os casos de uso não são utilizados tipicamente para especificar desempenho, tolerância a falha, entre outros.

A UML define “extends” e “includes”[8] para os casos de uso. Um caso de uso composto deve ser ajustado para produzir um modelo testável das dependências entre casos de uso. É necessário que o ajuste a fim de realizar o teste de alto nível exercite toda a dependência de baixo nível.

4.1.6 Vantagens

Os casos de uso estão incorporados em quase todas metodologias OOA/D (Object-Oriented Analysis/Design) e são muito utilizados. Eles podem ser desenvolvidos por analistas e testadores que não tem experiência em programação orientada a objetos e são freqüentemente a única documentação requerida disponível.

Os casos de uso refletem o ponto de vista do usuário/cliente. Estes focam nas capacidades que irão determinar o sucesso ou a falha do sistema. Em contraste, desenvolvedores freqüentemente focam numa estratégia técnica específica.

Os casos de uso estendidos provêm uma forma sistemática para o desenvolvimento das informações necessárias para o plano de testes. Se realizado de forma subjetiva e não sistemática, o plano de testes poderá perder relacionamentos necessários. Um caso de uso estendido torna explícito todos os relacionamentos que um caso de uso deve implementar. A narrativa arbitrária suficiente para casos de uso, freqüentemente resulta em modelos ambíguos, incompletos e inconsistentes. Freqüentemente, desenvolvimento de casos de uso estendidos é suficiente para encontrar erros e omissões.

4.2 Teste de cobertura em CRUD

O teste em CRUD (Create, Read, Update e Delete, ou seja Criar, Ler, Atualizar e Excluir) auxilia na verificação de que todas as operações básicas sejam exercitadas, para cada problema de domínio de objeto, no sistema em teste. Conjunto de testes desenvolvidos de casos de uso individuais e diagramas de seqüência não podem garantir que todos os problemas de domínio de classes no sistema sendo testado possam ser alcançados. Um problema de domínio de classe é a representação de uma entidade externa ou conceito que é necessário para um modelo de implementação independente do sistema. Por exemplo,

[8] Extends e Includes são relacionamentos de dependência existentes entre os casos de uso. Extend especifica que o caso de uso de destino estende o comportamento de origem. Include especifica que o caso de uso de origem incorpora explicitamente o comportamento de outro caso de uso em um local especificado pela origem.

cliente, pedido, produto são problemas de domínio de classes num sistema de processamento de pedidos. Para cada classe, o sistema deve estar apto a criar, ler, atualizar e excluir os objetos das classes. As operações básicas são tipicamente requeridas para qualquer problema de domínio de classes. Devido aos casos de uso serem necessariamente parciais, deve-se analisar todas as ações básicas sobre todos os casos de uso para identificar alguma omissão do sistema. Suponha que para alguma classe, uma ação básica (criar, ler, atualizar, excluir) não esteja sendo realizada por algum caso de uso do sistema. Será impossível para o sistema realizar alguma operação básica para esta classe. Algumas classes não devem propositalmente realizar todas as operações básicas. Por exemplo, classes que representam informações estáticas ou mantidas por outros sistemas devem ser limitadas a leitura.

4.2.1 Procedimento para o desenvolvimento da matriz CRUD

Seguem abaixo, os passos necessários para o desenvolvimento da matriz CRUD:

- Identificar todos os casos de uso do sistema.
- Identificar o problema de domínio de classes.
- Preparar uma matriz de casos de uso x classes, como mostrado na tabela 4.4. Para cada caso de uso, determinar a ação realizada no objeto de cada classe: criar, ler, atualizar, excluir.
- Analisar a matriz. Depois de todas as ações dos casos de uso serem transcritas, verifique as ações para cada classe. Cada classe deve ter ao menos uma das operações básicas. Se não existir, desenvolva um teste para alcançar a ação.

	Classe 1				**Classe 2**					**Classe 3**			
	C	**R**	**U**	**D**	**C**	**R**	**U**	**D**	**...**	**C**	**R**	**U**	**D**
Caso de uso 1	X	X		X		X							X
Caso de uso 2			X				X					X	
...													
Caso de uso n	X				X			X		X	X		

Tabela 4.4 – Matriz de casos de uso x classes

Se o conjunto de testes não cobrir todas as ações para cada classe, desenvolva casos de teste como indicados na tabela 4.5. Esta análise freqüentemente revela omissões no projeto e por isto deve ser utilizada para validar modelos de sistemas também.

Cobertura de ação básica do teste de caso de uso	Comportamento requerido para uma ação básica		
	Requerido	**Não requerido**	**Proibido**
Executado, aprovado	Validado	Surpresa	Falha
Executado, ocorreu exceção esperada	Falha	Validado	Validado
Não executado	Casos de testes incompletos: adicionar teste de capacidade	Casos de testes incompletos: adicionar teste de exceção	Casos de testes incompletos: adicionar teste de exceção

Tabela 4.5 – Interpretação da análise da cobertura da matriz CRUD

4.3 Teste baseado em perfil

A intenção do teste baseado em perfil é alocar o orçamento para teste para cada caso de uso de acordo com sua relativa freqüência. Se você estiver planejando desenvolver um modelo de caso de uso estendido, é necessário responder algumas questões:

- Quantos testes devem ser selecionados para cada variante?
- Quantos destes testes devem ser combinados?

O perfil operacional oferece uma técnica sistemática e quantitativa para responder estas questões. Os casos de uso são avaliados de acordo com sua freqüência relativa e recursos de testes são alocados de acordo com seu perfil. Casos de teste são então desenvolvidos para cada caso de uso até que o orçamento esteja esgotado.

Os casos de uso mais prováveis que falhem são os casos de uso utilizados mais freqüentemente. O perfil operacional lista todos os casos de uso do sistema e sua freqüência relativa. Por exemplo, suponha que o sistema tenha dois casos de uso: atualização e pesquisa. Na média 90.000 pesquisas e 10.000 atualizações são realizadas por dia. As freqüências relativas são 0,9 e 0,1 para pesquisa e atualização respectivamente.

Alocação de esforços: com o perfil operacional em mãos, nós podemos fazer uma alocação racional dos esforços do teste.

Suponha que nosso orçamento total para o teste do sistema do software ATM seja 1000 horas e assuma por exemplo, que na média o teste leva:

- 1 hora para planejamento, configuração e executar um caso de teste;
- revela um erro 5% do tempo;
- 4 horas para corrigir uma falha.

Quantos casos de teste nós podemos executar? Se temos T como número total de testes, então

T + (0.05 x 4T) = 1000

1.2T = 1000

T = 833

Desta forma nós conseguimos executar aproximadamente 833 casos de testes.

4.3.1 Procedimento

Seguem abaixo os passos para o desenvolvimento do teste baseado em perfil:

- Prepare o perfil operacional: Avalie os casos de uso e operações de acordo com sua freqüência relativa. Então prepare o perfil operacional: estime a freqüência relativa com que cada ator executa o caso de uso a ser testado.
- Estime sua produtividade para teste (testing productivity).
- Estabeleça seu orçamento para testar o sistema.
- Alocar o tempo de teste (ou número de testes) para cada caso de uso.
- Gerar casos de testes para cada caso de uso.
- Escolha uma estratégia para combinar os casos de testes. Uma forma simplista é randomizar a combinação.

Uma aplicação efetiva do teste baseado em perfil requer bom julgamento. Suponha que um caso de uso seja raramente executado, porém devido a sua importância, é necessário ter certeza de que será executado corretamente. Isto pode ser coberto por perfil operacional separado, para casos de uso críticos. Esforços de teste podem ser alocados de acordo com o risco ou freqüência relativa.

Suponha que uma alta freqüência do caso de uso tenha uma implementação trivial. Para propósito de teste, este caso de uso pode ser mesclado com outro caso de uso.

Suponha que seu perfil seja dominado por muitos casos de uso de baixa freqüência. Novamente, para propósitos de teste, alguns destes casos de uso podem ser mesclados com outros casos de uso.

4.3.2 Desvantagens

Uma desvantagem do teste baseado na freqüência é que pode ser difícil ou impossível obter ou estimar um perfil operacional preciso. Além disto, o esforço requerido para desenvolver o perfil é tipicamente lento.

4.3.3 Vantagens

Uma vantagem é que o orçamento para testes é freqüentemente restrito, desta forma a alocação baseada no perfil faz o melhor uso dos recursos disponíveis para testes.

4.4 A metodologia TOTEM

A metodologia TOTEM, cujos autores são Lionel Briand e Yvan Labiche, da Universidade de Carleton, se concentra nos testes funcionais do sistema. O objetivo da metodologia é suportar a derivação de requisitos de teste que serão transformados em casos de teste. Os requisitos de teste de sistema são derivados dos artefatos de análise da UML, tais como casos de uso, seus correspondentes diagramas de atividades, de seqüência e de colaboração, diagrama de classes e possivelmente expressões de linguagem de restrição de objetos (OCL – Object Constraint Language) entre todos estes artefatos. Enfim, a TOTEM provê uma metodologia sistemática a fim de executar as atividades mostradas na figura 4.1, além de automatizar a metodologia o máximo possível.

A figura 4.1 resume as etapas da metodologia TOTEM. Este trabalho se concentra nas etapas A2, A3 e A5, neste momento. Os requisitos de teste serão derivados de diferentes artefatos (de A2 a A6). Então estes requisitos são combinados num conjunto, evitando redundância. A etapa A4 é uma importante contribuição para os requisitos de testes e será tratada em trabalhos futuros. Sua ausência não afetará a validade e a utilidade do que está sendo apresentado neste trabalho. As etapas A7 e A8 se preocupam em gerar casos de teste e código para oráculos.

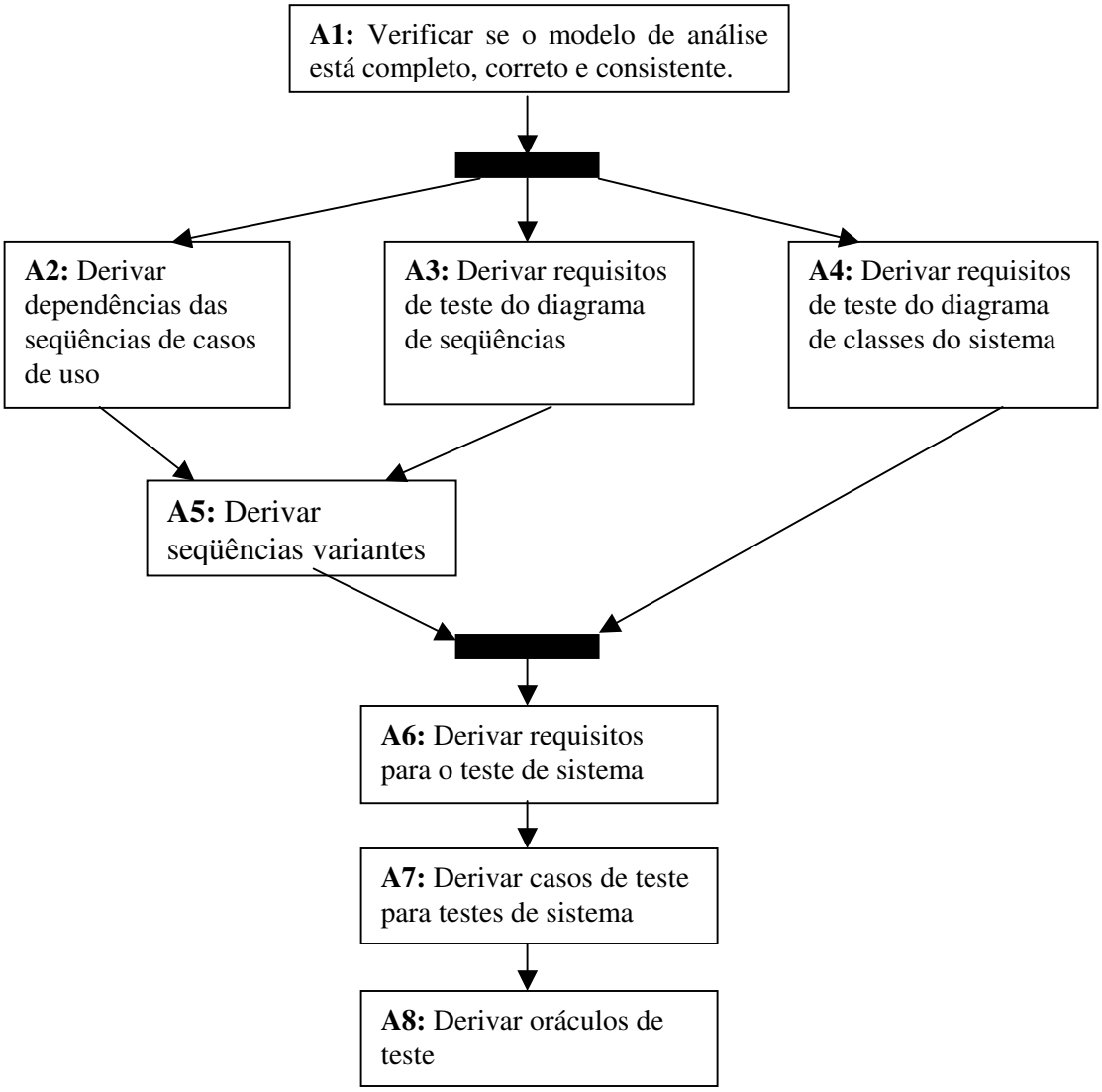

Figura 4.1 – Visão geral da metodologia TOTEM

4.4.1 Derivação das seqüências de casos de uso

A figura 4.2, mostra os passos necessários para que a derivação das seqüências de casos de uso possam ser derivadas e testadas.

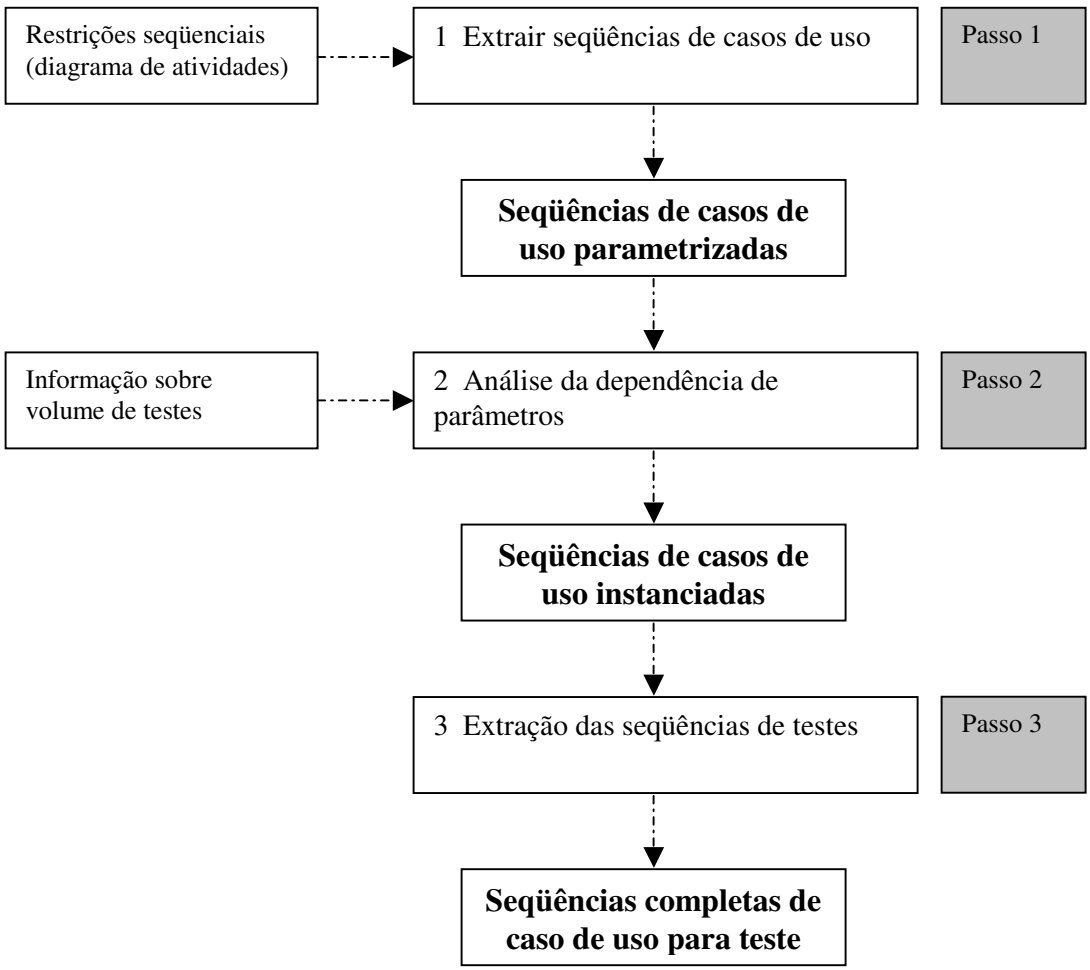

Figura 4.2 – Passos para a extração das seqüências de casos de uso para testes

As seqüências de casos de uso parametrizadas são derivadas do diagrama de atividades. Com a informação provida pelo testador sobre volume de testes que se pretende realizar, estas seqüências são instanciadas com valores simbólicos e combinadas a fim de obter a seqüência final de seqüências de casos de uso instanciados para serem testados.

4.4.1.1 Extrair seqüências de casos de uso

Os casos de uso são a primeira boa fonte para derivação de requisitos de teste de sistemas, afinal eles representam o alto nível das funcionalidades providas pelo sistema para o usuário. A figura 4.3 apresenta o Diagrama de casos de uso para um sistema de biblioteca.

Normalmente os casos de uso não são independentes. Além das dependências de <<include>> e <<extends>> também podem haver dependências seqüenciais, que se originam da lógica de processos de negócios que o sistema suporta. Quando se planeja

realizar testes de casos de uso, é necessário identificar possíveis seqüências nas execuções de casos de uso.

A representação das dependências seqüenciais entre os casos de uso é feita através do diagrama de atividades criado para cada ator do sistema. No diagrama, os vértices são casos de uso e as setas são dependências seqüenciais entre os casos de uso: uma seta entre dois casos de uso (do final para a ponta) significa que o caso de uso do final da seta deve ser executado antes do caso de uso para o qual aponta a seta. Mas isto não significa que necessariamente o caso de uso para o qual a seta aponta deva ser executado. Em algumas situações vários casos de uso podem ser executados de forma independente (sem dependência seqüencial) para que só então outro caso de uso possa ser executado. Para modelar este caso é utilizado o símbolo de junção. Para ser mais preciso, os vértices do nosso diagrama de atividades são casos de uso estendidos. Especificados explicitamente ou não, casos de uso tem parâmetros que determinam o comportamento que os casos de uso podem exibir, assim como valores de saída (resultados da sua execução). Parâmetros efetivos dos casos de uso são representados no diagrama de atividades, entre parênteses. A razão de ter o parâmetro efetivo neste contexto é mostrar as dependências entre parâmetros durante a execução de um caminho, no diagrama de atividades. Os casos de uso de diferentes raias, sem uma ligação, poderão ser executados de forma independente, ou seja qualquer um poderá ser executado antes do outro.

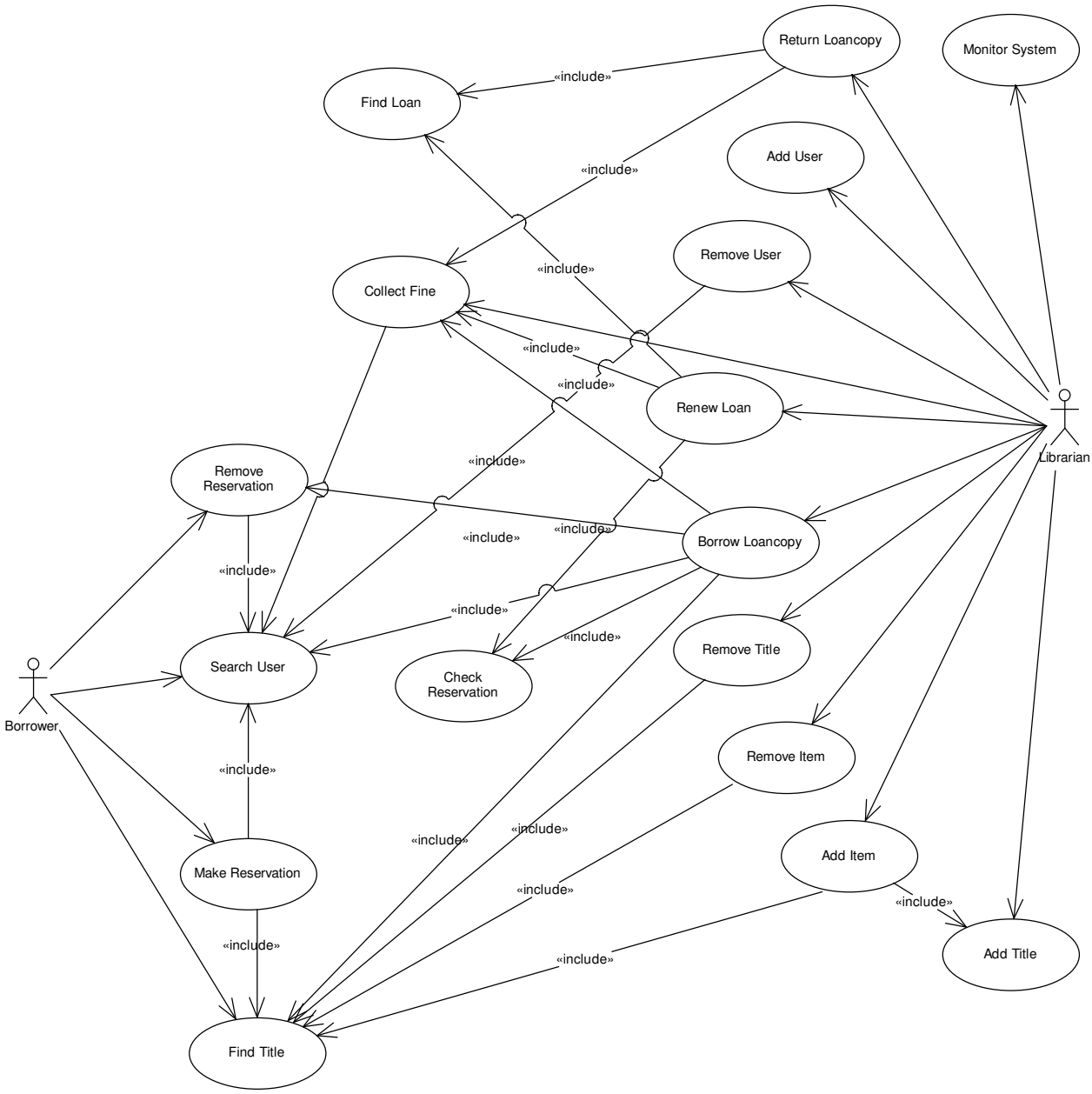

Figura 4.3 – Diagrama de casos de uso do sistema da biblioteca (Library System)

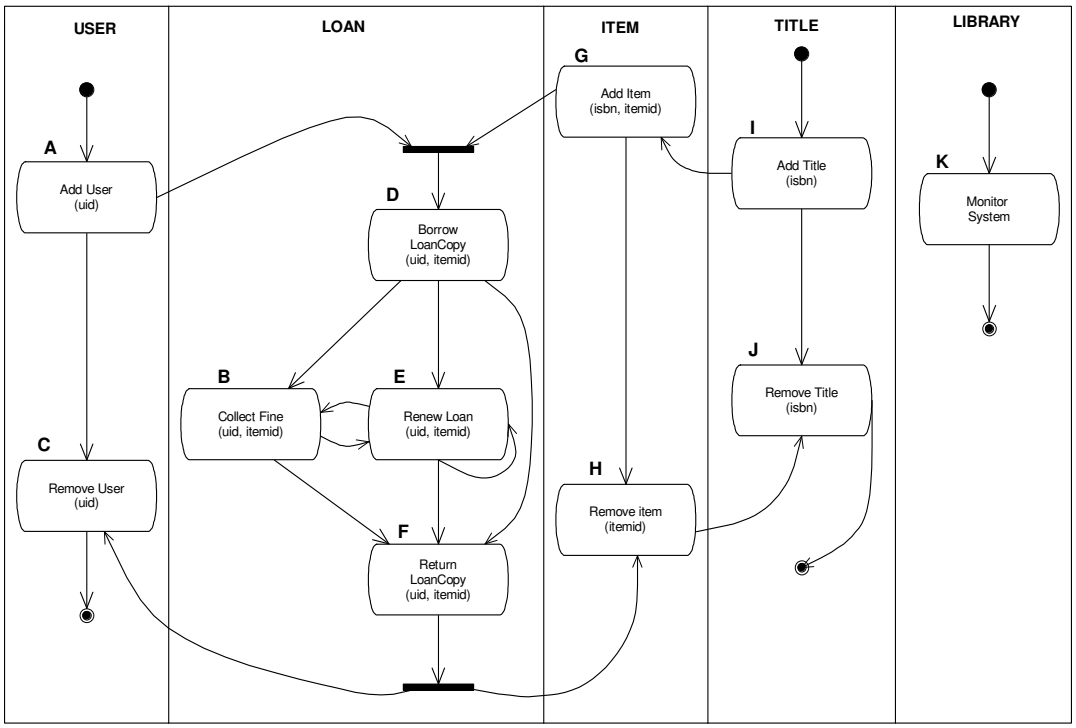

Figura 4.4 – Restrição de casos de uso seqüenciais para o ator bibliotecária (Diagrama de atividades)

A figura 4.4 apresenta o diagrama de atividades para o diagrama de casos de uso apresentado para figura 4.3.

Analisando os casos de uso A e C no diagrama, as setas indicam que necessariamente o A necessita ser executado antes que o C possa ser executado, mas não obrigatoriamente o C deve ser executado.

Os símbolos de junção que unem o A e o G significam que necessariamente os dois casos de uso tenham que ser executados para que o D seja executado depois, mas não é necessária uma ordem específica entre o A e o G. Os casos de uso são agrupados em raias de natação, de acordo com suas responsabilidades em termos de manipulação de objetos. Os casos de uso de diferentes raias, sem flechas de ligação, são independentes e podem ocorrer independentemente em qualquer ordem. Esta situação é modelada pelo conceito de intercalação (símbolo: ‖), que será abordado no passo 3 - extração das seqüências de testes.

Existe um infinito número de caminhos derivados do loop entre “colect fine” (multa na devolução) e “renewloan” (renovação do empréstimo). Se estabelecermos que o empréstimo poderá ser renovado apenas duas vezes, o número de caminhos entre “borrowloancopy” (empréstimo do exemplar) e “returnloancopy” (retorno do exemplar) será igual a 14. No caminho AddTitle.AddItem.RemoveItem.RemoveTitle, o parâmetro *isbn*

para o AddItem deve ser idêntico ao *isbn* do AddTitle. Tais dependências entre os valores dos parâmetros são necessárias para identificar o fluxo de dados entre as execuções dos casos de uso.

Para que o sistema de biblioteca seja testado em situações mais realísticas é necessário ter mais que um item por título e mais que um empréstimo por item. Também será necessário proceder com muitos usuários, itens, empréstimos a fim de testar a escalabilidade do sistema. Como conseqüência, as seqüências de casos de uso devem ser instanciadas várias vezes, com diferentes valores. O número de vezes que os parâmetros devem ser instanciados é provido pelo testador, que além disto deve indicar restrições ou dependência de loops.

Seqüências de caso de uso parametrizadas retiradas do diagrama de atividades:

- A(uid).C(uid)
- I(title).J(title)
- I(title).G(title,item).H(item).J(title)
- (A(uid) ‖ I(title).G(title.item) ‖. D (uid.item). F (uid,item).(C(uid) ‖ H(item).J(title))

Note que os casos de uso B e E (Loop) estão sendo desconsiderados para a exemplificação da metodologia.

4.4.1.2 Análise da dependência de parâmetros

Este passo requer a informação sobre volume de testes:

Seqüências e casos de uso parametrizadas	- A(uid).C(uid) - I(title).J(title) - I(title).G(title,item).H(item).J(title) - (A(uid) ‖ I(title).G(title.item) ‖. D (uid.item). F (uid,item).(C(uid) ‖ H(item).J(title))
Instâncias parametrizadas (valores simbólicos)	- usuários (2): u1, u2 - títulos (3) : t1, t2 e t3 - itens (1 por título) : (t1,i11), (t1,i12), (t2,i21), (t2,i22), (t3,i31), (t3,i32) - empréstimo (1 por usuário e item): pares (u1,i22) e (u2,i32)

Tabela 4.6 – Requisitos para a derivação de seqüências de casos de uso instanciados

A segunda linha da tabela acima corresponde a informação referente ao volume de testes que se pretende realizar. Esta quantidade de valores que serão utilizados devem ser especificados pelo testador. A instanciação seleciona uma seqüência parametrizada por vez, substituindo os parâmetros atuais por valores simbólicos e iniciando novamente da primeira

seqüência, se valores simbólicos ainda tiverem de ser substituídos, a fim de cobrir todas as seqüências.

Após realizar a substituição, oito seqüências serão encontradas:

Seq1: A(u1).C(u1)

Seq2: I(t1).J(t1)

Seq3: I(t2).G(t2.i21).H(i21).J(t2)

Seq4: I(t3).A(u2).G(t3,i32).D(u2,i32).F(u2,i32).C(u2).H(i32).J(t3)

Seq5: I(t1).G(t1,i11).H(i11).J(t1)

Seq6: I(t1).G(t1,i12).H(i12).J(t1)

Seq7: A(u1).I(t2).G(t2,i22).D(u1,i22).F(u1,i22).H(i22).C(u1).J(t2)

Seq8: I(t3).G(t3,i31).H(i31).J(t3)

Algumas seqüências (inclusive com os mesmos valores) estão inclusas em outras. Como exemplo, as seqüências 1 e 2 estão contidas nas seqüências 5 e 7, respectivamente. A fim de diminuir as seqüências obtidas, é necessário manter somente as seqüências que não estão inclusas em outras. Desta forma, as seqüências finais obtidas são: {Seq3, Seq4, Seq5, Seq6, Seq7, Seq8}.

4.4.1.3 Extração das seqüências de testes

As seqüências de casos de uso instanciados são combinadas a fim de produzir completas seqüências de casos de uso para serem testadas. A cada vez, duas seqüências são combinadas. A combinação das seqüências de casos de uso instanciados deve ser realizada cuidadosamente preservando as dependências dos casos de uso e evitando a duplicação dos casos de uso instanciados. Isto pode ser formalizado pelo conceito de intercalação, onde ‖ é o símbolo utilizado para a intercalação e deve ser implementado de tal forma a derivar as possíveis seqüências. Como exemplo, as seqüências Seq1 e Seq2 podem ser combinadas da seguinte forma:

AddTitle(title1).((AddItem(title1,item1).RemoveItem(Item1)) ‖
(AddItem(title1,item2).RemoveItem(item2)).RemoveTitle(title1)

O exemplo abaixo pode ser utilizado como regra geral para a combinação dos casos de uso:

Prefixo1 . X . Meio1 . Y . Sufixo1

Prefixo2 . X . Meio1 . Y . Sufixo2 , onde x e y representam casos de uso em comum e Prefixo, Meio e Sufixo representam qualquer subseqüência, que não tem elementos em

comum. Entende-se por elementos em comum, mesmos casos de uso com mesmos valores para os parâmetros. Resultado da combinação:

(Prefixo1 ‖ Prefixo2). X . (Meio1 ‖ Meio2) . Y . (Sufixo1 ‖ Sufixo2)

Seguem as combinações de seqüências para o exemplo da biblioteca, porém será mostrado somente um resultado para cada combinação.

Inicialmente as seqüências 3 e 4 são combinadas e produzem 495 intercalações, sendo que uma delas inclui a seqüência abaixo:

S: I(t3).A(u2).I(t2).G(t3,i32).G(t2,i21).D(u2,i32).H(i21).F(u2,i32).C(u2).H(i32).J(t2).J(t3)

Após, as seqüências 5 e S são combinadas e produzem 1820 intercalação, incluindo:

S^1: I(t3).I(t1).G(t1,i11).A(u2).I(t2).G(t3,i32).H(i11).G(t2,i21).D(u2,i32).H(i21).J(t1).
F(u2,i32).C(u2).H(i32).J(t2).J(t3)

As seqüências 6 e S^1 produzem 45 intercalações, incluindo:

$S^{2:}$ I(t3).I(t1).G(t1,i11).G(t1,i12).A(u2).I(t2).G(t3,i32).H(i11).G(t2,i21).
D(u2,i32).H(i12).H(i21).J(t1).F(u2,i32).C(u2).H(i32).J(t2).J(t3)

As seqüências 7 e S^2 produzem 18018 intercalações, incluindo:

$S^{3:}$ I(t3).I(t1).A(u1).G(t1,i11).G(t1,i12).A(u2).I(t2).G(t2,i22).G(t3,i32).H(i11).D(u1,i22).
G(t2,i21).F(u1,i22).D(u2,i32).H(i12).H(i21).J(t1).H(i22).C(u1).F(u2,i32).C(u2).H(i32).
J(t2).j(t3)

O último passo, mostrado abaixo, produzirá as seqüências completas de casos de uso para serem testadas. A combinação das seqüências 8 e $S^{3,}$ produzem 276 intercalações, incluindo:

S^4: I(t3).I(t1).G(t3,i31).A(u1).G(t1,i11).G(t1,i12).A(u2).I(t2).G(t2,i22).G(t3,i32).H(i11).
D(u1,i22).G(t2,i21).H(i31).F(u1,i22).D(u2,i32).H(i12).H(i21).J(t1).H(i22).C(u1).
F(u2,i32).C(u2).H(i32).J(t2).J(t3)

Na etapa A7 da figura 4.1, os valores simbólicos terão que ser substituídos pelos valores verdadeiros. Porém isto está fora do escopo deste trabalho, mas fará parte de trabalhos futuros.

As etapas A3 e A4 apresentadas a seguir não serão apresentadas detalhadamente, pois não serão utilizadas na estratégia proposta neste trabalho.

4.4.2 A3: Derivar requisitos de teste do diagrama de seqüências

Na etapa A3 da metodologia TOTEM são identificadas as seqüências de operações para serem testadas, assim como suas pré-condições e oráculos.

Estas seqüências de operações são derivadas a partir de seqüências de cenários, que por sua vez, são derivadas a partir dos modelos de interação referentes a um caso de uso, como diagramas de colaboração ou de seqüência e também de informações do diagrama de classes. Estes modelos descrevem interações alternativas dos objetos, cada uma delas realizando um possível cenário do caso de uso.

Os oráculos de teste, mencionados acima, são responsáveis por descrever o resultado esperado para determinado caso de teste. Segundo Sommerville, um oráculo é um programa que, dado um conjunto de dados de teste, pode prever qual deveria ser a saída do programa testado, se não houvesse defeitos na parte do sistema testado por estes dados [Sommerville, 92-446].

Após a obtenção das seqüências de operações, suas pré-condições e oráculos, tudo isto poderá ser formalizado numa tabela de decisão que será utilizada para formalizar o conjunto de requisitos de teste, que fará parte do plano de testes.

4.4.3 A5: Derivar seqüências variantes

Se nós assumirmos que cada caso de uso tem uma tabela de decisão, nós precisaremos nos aprofundar e dividir a seqüência de operações para serem testadas sobre uma seqüência de caso de uso completa. Em outras palavras, nós necessitamos ir da seqüência do caso de uso para as seqüências variantes de caso de uso, usando tabelas de decisão de caso de uso. Supondo que uma seqüência de casos de uso composta por 3 casos de uso: A.B.C, tendo respectivamente o número de variantes |A|, |B| e |C|, o número máximo de seqüências variantes deverá ser |A|*|B|*|C|. Uma variante corresponde a uma possível condição para realização de um caminho para um dos termos do produto na expressão regular do diagrama de interação. Uma variante pode requerer muitos casos de teste, como símbolos de interação podem estar presentes no termo do produto correspondente, cada um requer muitas seqüências de operações para serem tratadas.

4.5 A ETACS – Estratégia de Teste de software para Ambiente Cliente-Servidor

Em um ambiente cliente-servidor é necessário testar todas as camadas, verificar o desempenho e confiabilidade quando executadas em rede ou sob carga maior de informação.

A partir dos casos de uso devem-se extrair casos de testes e depois avaliá-los, de acordo com o critério e técnica e assim selecionar os casos de testes mais adequados. Cada critério e tipo de teste determina como são gerados os casos de teste.

Segue abaixo, uma visão global da ETACS:

- Iniciação/Etapa Inicial: Levantar uma estimativa de custos com hardware, software e pessoas envolvidas, bem como a viabilidade da solução do problema, apresentar testes genéricos com base na descrição geral do sistema.
- Elaboração/Levantamento de requisitos e definição da solução com modelos:

- Para cada caso de uso verificar coeficiente de prioridade.
- Com base na prioridade definir os tipos de testes.
- Fazer ou refazer o cronograma do projeto prevendo tempo para preparar e executar os testes que foram definidos.
- Rever o plano de trabalho com relação a estimativa de custos.
- Elaborar casos de teste para testes de sistema e regressão.
- Definir medidas de cobertura para os testes.
- Documentar.

- Contrução/Análise/Projeto e Implementação:

- Elaborar casos de teste para executar testes de integração e de unidades.
- Estabelecer medidas de cobertura.
- À medida que módulos ou componentes forem ficando prontos colocam-se em execução os casos de teste do projeto de casos de teste e se necessário geram-se mais para alcançar a cobertura determinada.
- Testes de integração são realizados.
- Na execução dos testes deve-se isolar as camadas e validar a arquitetura buscando melhor desempenho nas primeiras iterações.
- Documentar.

- Contrução/Análise/Projeto e Implementação

- Execução dos testes de sistema, se houver manutenção, refazer os testes (teste de regressão).
- Nos casos de prioridade alta, testar camada do servidor separadamente, cria-se um stub para acessá-lo com funcionalidade limitada e depois integra-o ao sistema e analisa-se o impacto.
- Testes de validação ou aceitação.
- Documentar.

- Transição/Implantação/Entrega - Releases

- Execução de testes de aceitação, inicialização do sistema, configurações de máquina e testes do instalador
- Teste beta

- Documentar
  - Manutenção

- Executar novamente todos os testes (teste de regressão), teste de unidade, integração e se necessário também de sistema, dependendo da prioridade e das mudanças.

A priorização dos casos de uso, conforme mencionado na fase de Elaboração/Levantamento de requisitos e definição da solução com modelos, deve ser realizada a partir da análise dos itens apresentados na tabela 4.7 e utilizando-se os critérios apresentados na tabela 4.8.

Item	Descrição abreviada	Pesos
1	Efeitos de uma falha no caso de uso	3
2	Causas de uma falha no caso de uso	3
3	Probabilidade do caso de uso falhar	3
4	Número de acessos a este caso de uso	2
5	Perfil dos usuários que utilizarão o caso de uso	1
6	Contrato com fornecedor deste caso de uso	3

Tabela 4.7 – Descrição resumida dos itens a serem avaliados nos riscos e seus pesos

Definição	Valor
Crítico (Cr)	5
Prioritário (Pr)	4
Regular (Re)	3
Pouca Importância (Po)	2
Exceção de Qualidade (Ex)	1

Tabela 4.8 – Valores para atribuição da prioridade

Segue abaixo, um exemplo de como os casos de uso devem ser avaliados:

Descrição do caso de uso	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
Caso de Uso 1	Crítico	Prioritário	Nulo	Pouca Importância	Regular	Exceção de Qualidade
Caso de Uso 2	Regular	Pouca Importância	Prioritário	Crítico	Regular	Regular
Caso de Uso 3	Prioritário	Regular	Pouca Importância	Regular	Exceção de Qualidade	Regular
Caso de Uso 4	Regular	Pouca Importância	Exceção de Qualidade	Pouca Importância	Crítico	Exceção de Qualidade

Tabela 4.9 – Exemplo da tabela com conceitos para itens

Substituição dos critérios estabelecidos para cada item por seus pesos:

Descrição do caso de uso	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
Caso de Uso 1	Cr=5	Pr=4	Nulo	Po=2	Re=3	Ex=1
Caso de Uso 2	Re=3	Po=2	Pr=4	Cr=5	Re=3	Re=3
Caso de Uso 3	Pr=4	Re=3	Po=2	Re=3	Ex=1	Re=3
Caso de Uso 4	Re=3	Po=2	Ex=1	Po=2	Cr=5	Ex=1

Tabela 4.10 – Tabela intermediária de exemplo dos conceitos x itens

Na tabela 4.10, a prioridade é definida de acordo com o seguinte coeficiente:

Cf= ∑ ((média absoluta dos item_1,item_2, item_3)*3 + item_4*2+item_5*1+item_6*3)

Estes coeficientes também são utilizados pela ferramenta da Rational [57].

Classificação final das prioridades:

Cf >= 30 {Alta}

20 <=Cf<30 {Média}

Cf < 20 {Baixa}

Após a execução dos cálculos, podemos definir a prioridade no exemplo utilizado (Tabela 4.11), de acordo com a classificação descrita acima:

Descrição do Caso de Uso	Item 1,2,3 (x 3)	Item 4 (x 2)	Item 5 (x 1)	Item 6 (x 3)	Somatório dos Valores obtidos	Prioridade
Caso de Uso 2	9,00	10,00	3,00	9,00	31,00	Alta
Caso de Uso 3	9,00	6,00	1,00	9,00	25,00	Média
Caso de Uso 1	13,50	4,00	3,00	3,00	23,50	Média
Caso de Uso 4	6,00	4,00	5,00	3,00	18,00	Baixa

Tabela 4.11 – Tabela final de exemplo das prioridades

Na tabela 4.12 é apresentada uma sugestão de testes que deverão ser realizados, de acordo com a priorização definida para cada caso de uso:

Níveis de teste	Prioridade Alta	Prioridade Média	Prioridade Baixa
Teste de unidade	Teste estrutural Cobertura de Condição(2) Critério Todos Ramos(2) Teste funcional Análise do valor limite(2) Teste baseado em erro Critério Análise de mutantes(2)	Teste estrutural Cobertura de Condição(2) Critério Todos Ramos(2) Teste funcional Particionamento de equivalência(2)	Teste funcional Baseados em caso de uso (cenários)(2)
Teste de integração	Análise do valor limite(2) Teste de segurança(1,2,3) Teste de integridade(2,3)	Particionamento de equivalência (2)	Não necessário
Teste de sistema	Teste de carga(2,3) Teste de estresse(2,3) Teste de segurança(2,3) Teste de integridade(2,3) Teste de volume(3) Teste de recuperação de cópia de segurança(3) Teste de desempenho(1,2,3)	Teste de carga(2,3) Teste de segurança (2,3) Teste de integridade(2,3) Teste de desempenho(1,2,3)	Teste de segurança(2,3) Teste de integridade(2,3)
Teste de aceitação	Teste de usabilidade(1) Teste de ícones(1)	Teste de usabilidade(1)	Não necessário
Teste de regressão	Refazer todos os teste com casos de usos que foram documentados para o módulo que foi alterado	Refazer teste de unidade e sistema para verificação do impacto das alterações	Refazer algum teste de sistema

Tabela 4.12 – Sugestão de testes e critérios de acordo com a prioridade

4.6 Conclusão

Neste capítulo foram apresentadas algumas técnicas para realização de testes de sistemas: teste de caso de uso estendido, teste de cobertura em CRUD, teste baseado em perfil, a TOTEM e a ETACS.

A estratégia ETIERP, proposta neste trabalho, foi baseada em parte da metodologia TOTEM, apresentada na sessão 4.4 e no teste de caso de uso estendido, apresentado na sessão 4.1.

A etapa A2- Derivação das seqüências de casos de uso, foi a contribuição da TOTEM para este trabalho. O motivo para a utilização somente da etapa A2, é que como a ETIERP foi desenvolvida para a empresa que adquire o ERP, não seria possível utilizar informações do diagrama de classes, por exemplo, que é uma das informações importantes para a execução da etapa A4 da TOTEM. Já para a execução da etapa A2 são necessários basicamente os diagramas de casos de uso e os diagramas de atividades. Apesar da documentação sobre a especificação do sistema não ser aberta ao cliente que adquire o ERP, estes diagramas podem facilmente ser criados com a ajuda dos usuários-chave, consultores e analistas de sistemas, enfim toda a equipe envolvida na implantação do sistema e que conseqüentemente participou do treinamento inicial no software. Além disto, pode-se contar com os manuais do sistema. A fim de gerar os cenários e valores reais para os casos de teste, a ETIERP se baseia nos testes de caso de uso estendidos.

A TOTEM também aborda formas para se automatizar algumas etapas da metodologia, porém isto não foi abordado neste capítulo, por não fazer parte do escopo da ETIERP, neste momento. Isto será considerado como trabalho futuro.

Este capítulo também aborda o teste de cobertura em CRUD, na sessão 4.2, o teste baseado em perfil, na sessão 4.3 e a estratégia ETACS, na sessão 4.5. Apesar destas técnicas não serem utilizadas na ETIERP, também são interessantes para complemento deste trabalho. O teste em CRUD facilita o teste das operações básicas e o teste de caso de uso por perfil e a ETACS podem ser utilizados para priorizar o teste de determinados casos de uso, quando o cronograma e ou orçamento disponíveis para os testes forem restritos.

Capítulo 5

ETIERP – Estratégia de Testes para Implantação de Sistemas ERP

Este capítulo apresenta a estratégia proposta para geração de casos de teste de forma estruturada e sistemática para aplicação na implantação de um sistema ERP. É importante esclarecer que este trabalho foi desenvolvido pensando-se na empresa que irá adquirir o sistema ERP para implantação e não na empresa fornecedora do software. Quando o cliente adquire um sistema ERP, geralmente ele não possui a especificação do sistema ou código-fonte, normalmente apenas recebe os manuais de utilização do sistema. Desta forma, os testes ficam restritos aos testes de caixa preta, mais especificamente aos testes de sistema. Os testes de sistema se preocupam em testar todo o sistema baseado em suas especificações. Envolvem muitas atividades, como testes dos aspectos funcionais e não-funcionais do sistema, como desempenho, segurança, entre outros. Nos concentraremos nos testes funcionais, apesar dos testes referentes aos aspectos não-funcionais serem altamente recomendados, como complemento deste trabalho.

Os sistemas ERP são sistemas grandes e complexos que resultam num grande número de casos de testes, portanto uma possível automação de parte da estratégia que será apresentada neste capítulo deve ser levada em consideração. A automatização da estratégia não é abordada neste trabalho e será considerada como trabalho futuro.

A ETIERP (Estratégia de Testes para Implantação de Sistemas ERP) está baseada em parte da metodologia TOTEM, que visa a derivação de requisitos para a geração de testes funcionais e em parte da abordagem de Robert V. Binder, sobre testes de casos de uso estendidos. A seção 5.1 apresenta um paralelo entre as fases de implantação de um sistema ERP (abordadas no capítulo 2) e as etapas da estratégia proposta neste capítulo, a seção 5.2 aborda a estratégia em questão e a seção 5.3 faz um resumo da ETIERP.

5.1 ETIERP ao longo das fases de implantação do sistema ERP

A figura 5.1, representa um paralelo entre as fases de implantação de um sistema ERP e as etapas da estratégia proposta. Juntamente com a fase de planejamento do projeto deve ser iniciada a etapa de planejamento dos testes. Note que logo após a terceira fase da implantação do ERP, quando os processos empresariais são levantados, as outras etapas da ETIERP já poderão ser iniciadas. A fase 4 da implantação poderá ser executada paralelamente com a ETIERP. Este paralelismo proposto pode prolongar o tempo de implantação do sistema ERP e conseqüentemente encarecê-lo, pois pode ser necessário o auxílio dos consultores externos para a execução de algumas etapas da estratégia proposta. Após a conclusão da etapa 9 da ETIERP, a fase 5 da implantação, referente aos Testes, poderá ser realizada e logo após a etapa 10 da ETIERP poderá ser executada.

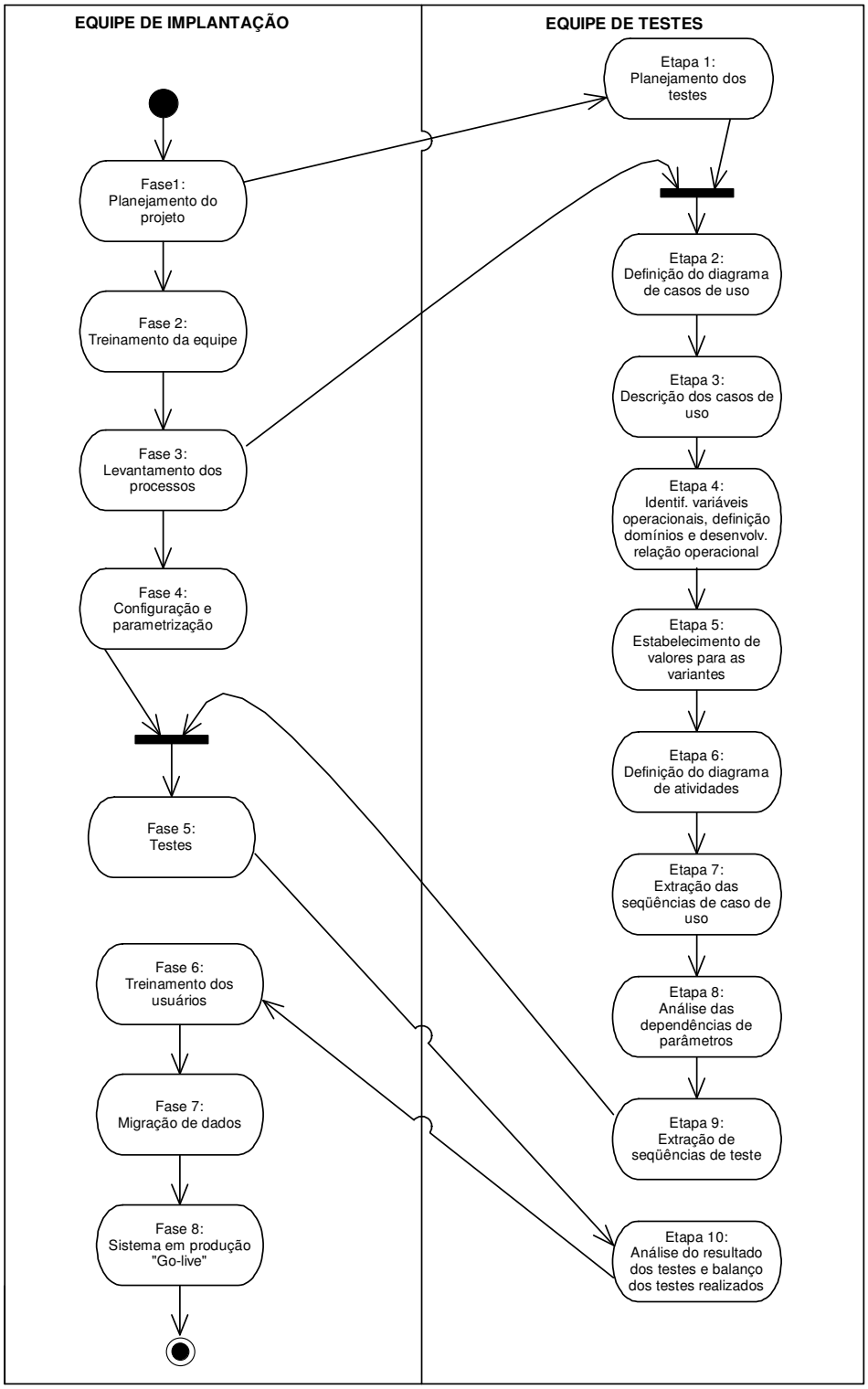

Figura 5.1 - Fases da implantação de um sistema ERP X Etapas da ETIERP

5.2 Apresentação da ETIERP

Na figura 5.2 estão representadas as etapas da ETIERP que devem ser seguidas para a aplicação da estratégia proposta.

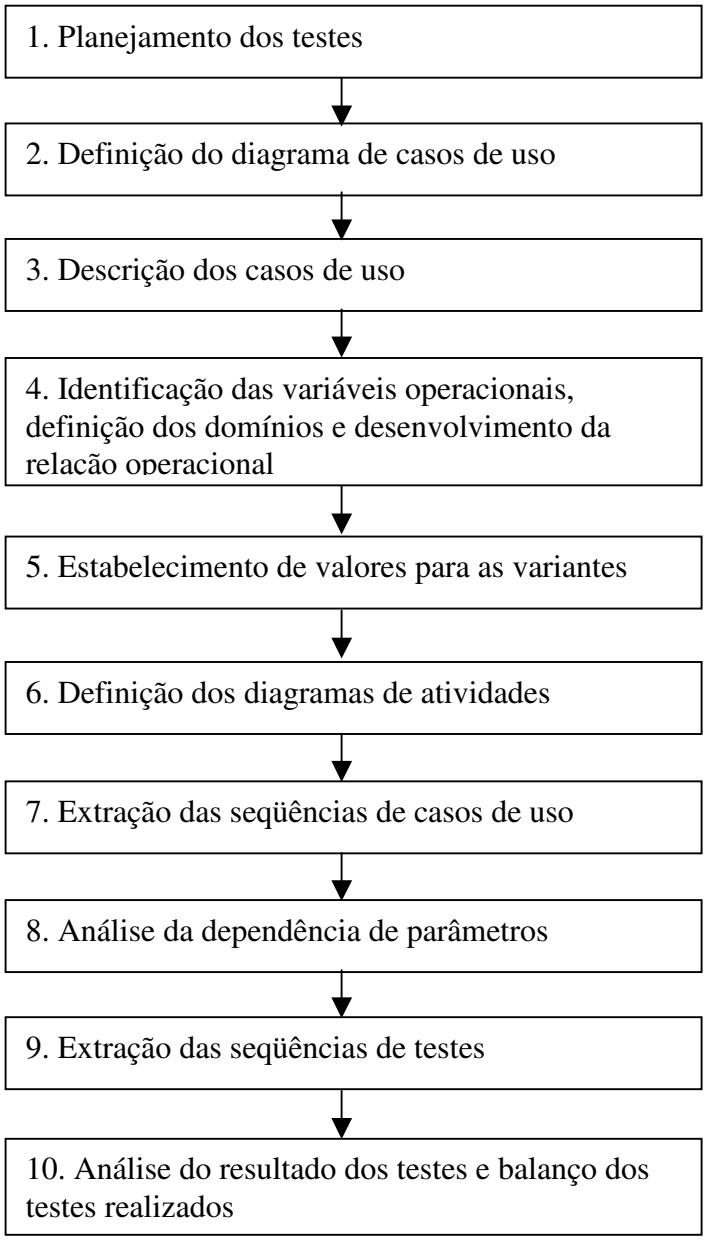

Figura 5.2 Representação das etapas da metodologia

Como abordado na metodologia TOTEM, no contexto da orientação a objetos, os requisitos para testes de sistema serão derivados de artefatos de análise da UML, como diagramas de casos de uso e diagramas de atividades. Na aquisição de um sistema ERP não se pode contar com a especificação do sistema, porém pode-se utilizar manuais, a orientação de consultores envolvidos na implantação e o conhecimento dos "usuários-chave", para a definição dos casos de uso.

5.3 Descrição das etapas da metodologia

5.3.1 Planejamento dos testes

A fase inicial da implantação do ERP é o planejamento do projeto, quando são definidos o sistema ERP e módulos que serão implantados, a consultoria externa a ser contratada, a equipe interna, a estratégia de implantação e o cronograma. Juntamente com a fase inicial do planejamento do projeto, a etapa 1 da ETIERP, referente ao planejamento dos testes deve ser iniciada. Nesta etapa, deverá ser definido o cronograma de testes, que será uma atividade integrante do cronograma de implantação do sistema ERP. Caso o cronograma de implantação sofra alguma alteração, isto poderá gerar a necessidade de alteração também no cronograma de testes e vice-versa. Nesta etapa também deverá ser definida a equipe de testes, que poderá ser composta apenas pelo pessoal especializado da área de informática ou também pelos chamados usuários-chave, que são os especialistas e representantes de cada área da empresa.

5.3.2 Definição do diagrama de casos de uso

Nesta etapa deve ser definido o diagrama de casos de uso referente ao sistema ERP.

Como os casos de uso são muitos, pois o ERP é um sistema genérico, criado para abranger diferentes empresas e até diferentes países, devem ser definidos somente aqueles casos de uso que serão utilizados pela empresa que estará implantando-o.

Os sistemas ERP são divididos comercialmente em módulos: Faturamento, Compras, Contas a Pagar, Contas a Receber, Contabilidade, Fiscal, entre outros. O módulo de compras por exemplo, envolve vários departamentos da empresa: todos os departamentos podem incluir solicitações de compra no sistema. Logo após a inclusão da solicitação, o departamento de compras se encarrega de efetivar a compra. Quando o produto é entregue à empresa, o funcionário do departamento fiscal é responsável por registrar o recebimento físico e fiscal do produto. Todas estas ações são possíveis de serem realizadas pelo módulo

de compras. Os atores do diagrama de casos de uso deverão ser estes módulos que compõem o sistema ERP.

Os casos de uso representarão as funcionalidades do sistema: incluir clientes, alterar clientes, gerar nota fiscal, baixar título a pagar, entre outros.

Na figura 5.3, é mostrado um diagrama de casos de uso representando o sistema ERP. Para exemplificar foram utilizados somente dois atores: Compras e Contas a Pagar.

No diagrama, podemos observar a integração do sistema. Incluir recebimento, um dos casos de uso do ator Compras, gera automaticamente o título a pagar do ator Contas a Pagar. O diagrama mostra também que o título a pagar poderia ser gerado diretamente pelo ator Contas a Pagar. Cada caso de uso apresentado no diagrama será explicado detalhadamente na etapa 3 da estratégia.

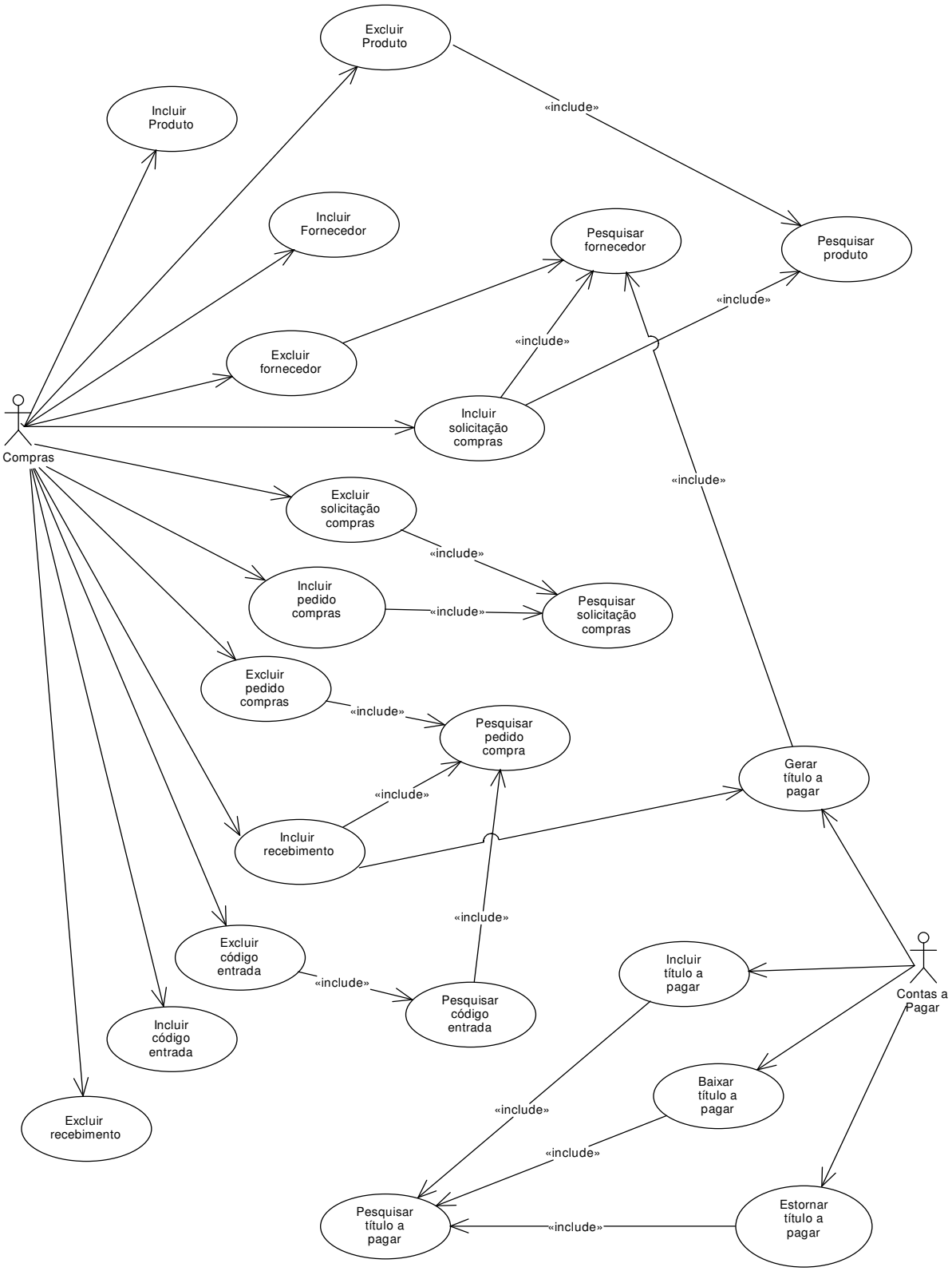

Figura 5.3. Diagrama de casos de uso de parte do sistema ERP.

5.3.3 Descrição dos casos de uso

A descrição dos casos de uso mostrados no diagrama facilitará a compreensão dos mesmos e também fornecerá dados que auxiliarão na elaboração dos casos de testes e conferência dos resultados obtidos após a aplicação dos testes. A descrição de todos os casos de uso, consta no Anexo I deste trabalho.

Seguem abaixo, exemplos do modelo de descrição adotado neste trabalho:

Nome do caso de uso: Incluir produto
Ator: Compras
Descrição: Um solicitante, que pode ser qualquer funcionário da empresa, necessita comprar um novo produto ainda não cadastrado no sistema.
Pré_condições: Produto ainda não incluso no sistema.
*Pós_condições:*Produto incluso com sucesso.
Fluxo básico: Código do produto informado corretamente. Produto incluso com sucesso.
Fluxo alternativo: Código do produto inválido. Não foi possível incluir o produto.

Nome do caso de uso: Excluir produto
Ator: Compras
Descrição: Excluir informações sobre o produto.
Pré_condições: Produto já incluso no sistema e ainda não utilizado em nenhuma outra operação.
*Pós_condições:*Produto excluído com sucesso.
Fluxo básico: Código do produto informado corretamente. Produto excluído com sucesso.
Fluxo alternativo: Código do produto inválido. Não foi possível excluir o produto.

Nome do caso de uso: Incluir fornecedor
Ator: Compras / Contas a pagar
Descrição: Incluir informações sobre um novo fornecedor.
Pré_condições: Fornecedor ainda não incluso no sistema.
*Pós_condições:*Fornecedor incluso com sucesso.
Fluxo básico: Código do fornecedor e código do estado informados corretamente. Fornecedor incluso com sucesso.
Fluxo alternativo: - Código do fornecedor inválido. Não foi possível incluir o fornecedor.
- Código do fornecedor correto. Código do estado inválido. Não foi possível incluir o fornecedor.

Nos exemplos apresentados, a descrição deve esclarecer o que significa determinado caso de uso. As pré-condições especificam quais são as condições para que o caso de uso em questão possa ser executado e as pós-condições especificam quais são as condições do sistema depois que o caso de uso foi executado. O fluxo básico[9] e o fluxo alternativo[10] correspondem aos cenários do caso de uso. Eles relacionam as possíveis situações que podem ocorrer na execução do caso de uso, porém o fluxo básico especifica o fluxo mais comum e o fluxo alternativo especifica situações que não são tão comuns.

5.3.4 Identificação das variáveis operacionais, definição dos domínios das variáveis operacionais e desenvolvimento da relação operacional

As etapas 3 e 4 desta estratégia, são baseadas no teste de caso de uso estendido, de Robert V. Binder, abordadas na seção 4.1 do capítulo anterior. Estas etapas facilitarão o levantamento dos parâmetros e seus valores, que serão utilizados em outras etapas da ETIERP, além de fornecer informações que poderão ser utilizadas no plano de testes.

O teste de caso de uso estendido aborda a criação de casos de teste baseados no levantamento de algumas idéias específicas sobre o caso de uso e o resultado correspondente esperado. Estas idéias podem corresponder aos cenários do caso de uso, mas também podem incluir fatores do ambiente, como exemplo, hora do dia.

Nesta etapa, os passos descritos abaixo serão aplicados a todos os casos de uso definidos na etapa 2.

- Identificação das variáveis operacionais: os casos de uso reagem de acordo com a combinação de entradas externas e estados do sistema. As variáveis operacionais representam todas estas situações que poderão produzir diferentes respostas dos casos de uso.
- Definição dos domínios das variáveis operacionais: Devem ser exercitadas todas as colaborações e valores limites que ajudem o caso de testes a detectar a presença de uma falha.
- Desenvolvimento da relação operacional: Os relacionamentos operacionais, que determinam classes distintas de resposta do sistema, são representadas em uma tabela de

[9] Fluxo básico ou curso básico é o fluxo esperado de eventos. É o mais importante fluxo de eventos, que dará o melhor entendimento do caso de uso [Jacobson et al, 95 – 331].

[10] Fluxo alternativo ou curso alternativo cobre todos os outros fluxos de eventos, os casos não comuns do caso de uso. São variantes do fluxo básico de eventos e erros que podem ocorrer. Normalmente, um caso de uso tem somente um curso básico, mas vários cursos alternativos [Jacobson et al,95-331].

decisão: quando todas as condições numa linha são verdadeiras a ação esperada é produzida. Cada linha x coluna na tabela de decisão é uma variante do caso de uso.

Abaixo seguem 3 exemplos de como deverá ser criada a relação operacional para cada caso de uso do sistema. Na tabela devem ser especificadas as variantes e o oráculo de testes.[11]

Caso de uso: Incluir produto

Variáveis operacionais		Resultado esperado	
Variante	Prod_id	Ação	Mensagem
1	Inválido	Produto não incluso no sistema	"Código de produto inválido"
2	Válido	Produto incluso no sistema	"Inclusão do produto efetuada com sucesso"

Tabela 5.1. Definição da relação operacional para o caso de uso : Incluir Produto

A tabela 5.1 foi exemplificada com duas variantes. Neste caso, ambas são determinadas pela variação do parâmetro identificador do produto (prod-id). Quando o identificador do produto é inválido, a ação esperada é que o produto não seja cadastrado no sistema e a mensagem exibida seja: "Código de produto inválido".

Estas tabelas devem ser criadas com o auxílio das informações da descrição do caso de uso, especialmente do fluxo básico e alternativo.

Caso de uso: Excluir produto

Variáveis operacionais		Resultado esperado	
Variante	Prod_id	Ação	Mensagem
1	Inválido	Produto não excluído do sistema	"Código de produto inválido"
2	Válido	Produto excluído do sistema	"Exclusão do produto efetuada com sucesso"

Tabela 5.2. Definição da relação operacional para o caso de uso : Excluir Produto

[11] Oráculo de testes: são responsáveis por descrever o resultado esperado para determinado caso de teste. Segundo Sommerville, um oráculo é um programa, que dado um conjunto de dados de teste, pode prever qual deveria ser a saída do programa testado, se não houvesse defeitos na parte do sistema testado por estes dados [Sommerville, 92-446]

Caso de uso: Incluir fornecedor

Variáveis operacionais			Resultado esperado	
Variante	Forn_id	Forn_uf	Ação	Mensagem
1	Inválido	Não importa	Fornecedor não incluso no sistema	"Código de fornecedor inválido"
2	Válido	Inválido	Fornecedor não incluso no sistema	"Código do estado inválido"
3	Válido	Válido	Fornecedor incluso no sistema	"Inclusão do fornecedor efetuada com sucesso"

Tabela 5.3. Definição da relação operacional para o caso de uso : Incluir fornecedor

O levantamento completo das variáveis operacionais e variantes consta no anexo II deste trabalho.

5.3.5 Estabelecimento de valores para as variantes

Após a definição das variáveis operacionais na etapa anterior serão definidos os valores reais que provocarão as diferentes reações do sistema.

Para cada variante deve ser definido pelo menos um valor verdadeiro e outro falso que exercite as diferentes situações.

Estes valores devem ser estabelecidos de acordo com a necessidade do usuário. Como exemplo, utilizaremos dois casos (situações) específicos que necessitarão ser contemplados pelo sistema e desta forma, todos os valores definidos para os parâmetros (variáveis operacionais) levarão em consideração estes casos que deverão ser testados:

Situações

1. Importação de produtos (fornecedor fora do Brasil e CFOP - Código Fiscal de Operação 3101);
2. Compra no mercado nacional (fornecedor nacional e CFOP - Código Fiscal de Operação 1101).

As tabelas abaixo são criadas com base nas tabelas mostradas na etapa anterior. Para cada variante foram criadas as variantes V (Verdadeira) e F (Falsa) que contêm os valores que deverão ser utilizados nos casos de testes.

Caso de uso: Incluir produto

Variáveis operacionais		Resultado esperado	
Variante	Prod_id	Ação	Mensagem
1	Inválido	Produto não incluso no sistema	"Código de produto inválido"
1V	%@#		
1F	Qualquer teste verdadeiro para a variante 2		
2	Válido	Produto incluso no sistema	"Inclusão do produto efetuada com sucesso"
2V	P00001		
2F	Qualquer teste verdadeiro para a variante 1		

Tabela 5.4. Definição dos valores para o caso de uso : Incluir produto

Caso de uso: Excluir produto

Variáveis operacionais		Resultado esperado	
Variante	Prod_id	Ação	Mensagem
1	Inválido	Produto não excluído do sistema	"Código de produto inválido"
1V	%@#		
1F	Qualquer teste verdadeiro para a variante 2		
2	Válido	Produto excluído do sistema	"Exclusão do produto efetuada com sucesso"
2V	P00001		
2F	Qualquer teste verdadeiro para a variante 1		

Tabela 5.5. Definição dos valores para o caso de uso : Excluir produto

Caso de uso: Incluir fornecedor

Variáveis operacionais			Resultado esperado	
Variante	Forn_id	Forn_uf	Ação	Mensagem
1	Inválido	Não importa	Fornecedor não incluso no sistema	"Código de fornecedor inválido"
1V	%@#			
1F	Qualquer teste verdadeiro para as variantes 2 e 3			
2	Válido	Inválido	Fornecedor não incluso no sistema	"Código do estado inválido"
2V	BR0001	%@		
2F	Qualquer teste verdadeiro para as variantes 1 e 3			
3	Válido	Válido	Fornecedor incluso no sistema	"Inclusão do fornecedor efetuada com sucesso"
3V	BR0001	SP		
3V	EX0002	EX		
3F	Qualquer teste verdadeiro para a variante 1 e 2			

Tabela 5.6. Definição dos valores para o caso de uso : Incluir fornecedor

A tabela 5.6 apresenta duas variantes verdadeiras que são necessárias para que ambas as situações enumeradas possam ser testadas: compra com um fornecedor nacional e com fornecedor estrangeiro.

O estabelecimento de valores para todos os casos de uso utilizados na estratégia, constam no anexo III.

5.3.6 Definição dos diagramas de atividades

Como abordado no capítulo 4, na apresentação da metodologia TOTEM, o casos de uso representam para o usuário, as funcionalidades de alto nível do sistema. Mas normalmente, eles não são independentes. Além das dependências <<extend>> e <<include>> existem também as dependências seqüenciais, derivadas da lógica de processos de negócios, ou seja, um pedido de vendas tem que ser incluso antes que uma nota fiscal possa ser gerada. Nós representamos as dependências seqüenciais entre os casos de uso através do diagrama de atividades. Esta representação facilita a identificação e a visualização das dependências seqüenciais, pois o diagrama é fácil de ser interpretado.

Um diagrama de atividades específico deverá ser criado para cada ator do sistema. Os títulos das raias representam os cadastros do sistema e os casos de uso representam as funções realizadas durante o ciclo de vida destes cadastros.

Veja na figura 5.4, a dependência seqüencial entre Incluir produto e Remover Produto. A representação mostra que a inclusão do produto deve ser realizada antes da remoção do mesmo, mas não necessariamente a remoção deve ser executada. Um símbolo de junção, que aparece logo acima do caso de uso Incluir solicitação, indica que o caso de uso Incluir produto e Incluir fornecedor devem ser obrigatoriamente executados antes do caso de uso Incluir solicitação, porém não é necessária uma seqüência lógica entre os casos de uso Incluir produto e Incluir fornecedor. Note também que os parâmetros dos casos de uso mostrados no diagrama de atividades são as variáveis operacionais definidas na etapa 4.

Diagrama de atividades

ATOR: COMPRAS

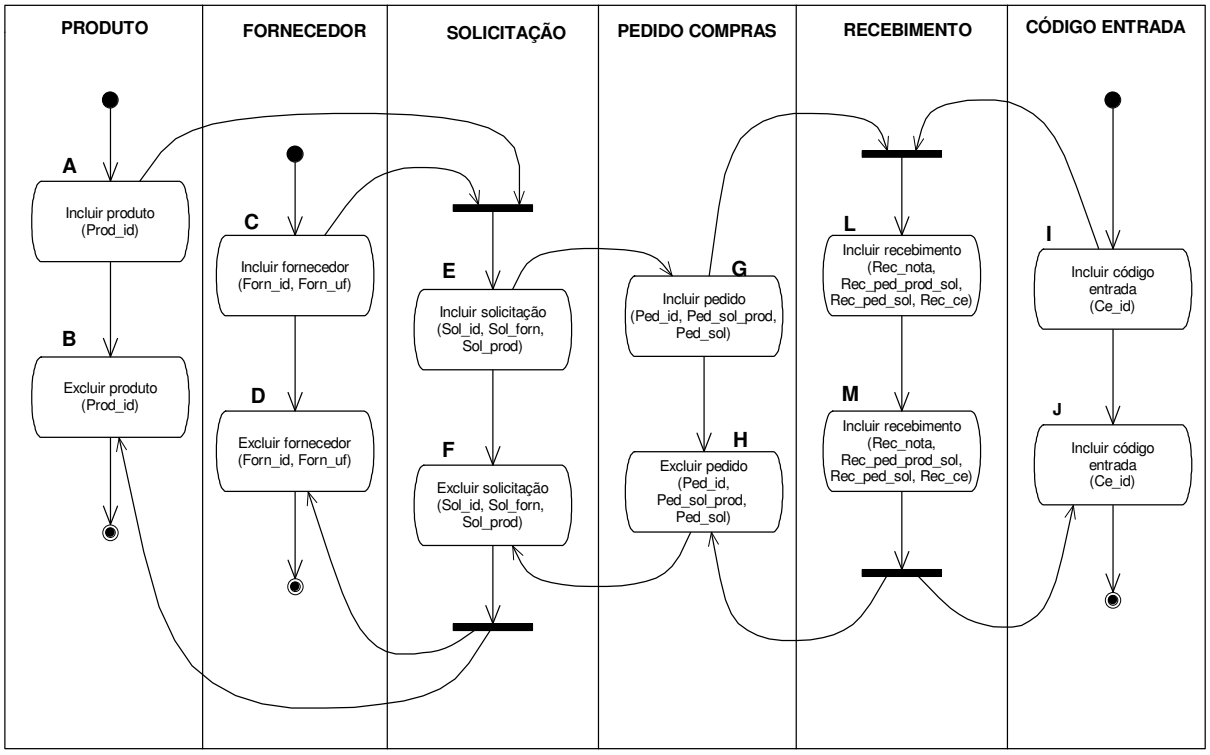

Figura 5.4. Diagrama de atividades do ator Compras

5.3.7 Extração das seqüências de casos de uso

As seqüências de caso de uso devem ser derivadas do diagrama de atividades. Estas seqüências significam os prováveis caminhos na execução das funcionalidades do sistema e devem ser utilizadas na geração de casos de teste.

O diagrama de atividades pode especificar um número infinito de caminhos se as seqüências forem geradas de forma automatizada. Por exemplo, quando existe um loop entre dois casos de uso e o número máximo de vezes que ele pode ser executado é muito alto, nós podemos utilizar a estratégia de testes de laços, utilizada para testar código-fonte. Nesta estratégia é requerido, quando possível, executar cada laço uma vez, um número representativo ou uma média maior que 1. Além disto, se quisermos testar nosso sistema em situações mais realísticas, nós iremos precisar de mais de um item por pedido de compras, por exemplo.

Possíveis seqüências de casos de uso, extraída do diagrama de atividades do ator Compras:

- A (Prod_id).B (Prod_id)
- C (Forn_id, Forn_uf).D (Forn_id, Forn_uf)
- I (Ce_id).J (Ce_id)
- A(Prod_id)||C(Forn_id, Forn_uf)).E(Sol_id, Sol_prod, Sol_forn).G((Ped_id, Ped_sol_prod, Ped_sol) || I (Ce_id)).L (Rec_nota, Rec_ped_prod_sol, Rec_ped_sol, Rec_ce).M (Rec_nota, Rec_ped_prod_sol, Rec_ped_sol, Rec_ce).(J(Ce_id) || H (Ped_id, Ped_sol_prod, Ped_sol)).F (Sol_id, Sol_prod, Sol_forn).(D (Forn_id, Forn_uf) || B (Prod_id))

5.3.8 Análise da dependência de parâmetros

Nesta etapa, os parâmetros das seqüências de casos de uso serão substituídos por valores reais de teste, ou seja os casos de uso serão instanciados[12]. A substituição deverá ser feita de forma que todos os valores levantados possam ser testados. Os valores que utilizamos nesta etapa, são os valores definidos na etapa 4.

Os valores enumerados na tabela 5.7 são valores referentes as situações válidas da etapa 5. As situações inválidas não são consideradas, pois para que a seqüência de caso de uso possa ser executada, todos os parâmetros devem ser válidos porque a execução de um caso de uso depende do sucesso da execução do caso de uso antecessor.

[12] Instância de caso de uso: caso de uso com valores especificados para os parâmetros de casos de uso e para o estado do sistema em teste [Binder, 00-277].

Seqüências de casos de uso parametrizadas	- A (Prod_id).B (Prod_id) - C (Forn_id, Forn_uf).D (Forn_id, Forn_uf) - I (Ce_id).J (Ce_id) - A(Prod_id) ‖ C(Forn_id, Forn_uf)).E(Sol_id, Sol_prod, Sol_forn). G((Ped_id, Ped_sol_prod, Ped_sol) ‖ I (Ce_id)). L (Rec_nota, Rec_ped_prod_sol, Rec_ped_sol, Rec_ce). M (Rec_nota, Rec_ped_prod_sol, Rec_ped_sol, Rec_ce). (J(Ce_id) ‖ H (Ped_id, Ped_sol_prod, Ped_sol)). F (Sol_id, Sol_prod, Sol_forn).(D (Forn_id, Forn_uf) ‖ B (Prod_id))
Instâncias de parâmetros (valores válidos, utilizados na etapa 5)	- Produto {(P00001)} - Fornecedor {(BR0001,SP) (EX0002,EX)} - Solicitação {(Sol1,P00001,BR0001-SP) (Sol2,P00001,EX0002-EX)} - Pedido {(Ped1,P00001,Sol1) (Ped2, P00001, Sol2)} - Código entrada {(1101,3101)} - Recebimento {(000001,P00001,Ped1,1101) (000002,P00001,Ped2,3101)

Tabela 5.7. Requisitos para a derivação de seqüências de casos de uso instanciados.

Substituição dos parâmetros pelos valores reais (ator Compras):
Seq 1. A(P00001).B(P00001)
Seq 2. C(BR0001-SP).D(BR0001-SP)
Seq 3. I(1101).J(1101)
Seq 4. A(P00001).C(EX0002-EX).E(SOL2.P00001,EX0002-EX).G(PED2,P00001, SOL2).I(3101).L(00002,P00001,PED2,3101).M(00002,P00001,PED2,3101).J(3101). H(PED2,P00001,SOL2),F.(SOL2.P00001,EX0002-EX).D(EX0002-EX).B(P00001)
Seq 5. C(BR0002-SP).A(P00001).E(SOL1.P00001,BR0001-SP).I(1101).G(PED1,P00001, SOL1).L(00001,P00001,PED1,1101).M(000011,P00001,PED11,1101).H(PED1,P00001,SO L1).J(1101),F.(SOL1.P00001,BR0001-SP).B(P00001).D(BR0001-SP)

A instanciação seleciona uma seqüência parametrizada por vez, substituindo os parâmetros pelos valores e iniciando novamente da primeira seqüência se ainda existirem valores que não foram utilizados, a fim de cobrir todas as seqüências.

Note que a seqüência 1 está contida na quarta e na quinta seqüência e as seqüências 2 e 3 estão contidas na quinta seqüência, logo, as seqüências menores devem ser descartadas. Desta forma, deverão ser mantidas somente as seqüências 4 e 5.

5.3.9 Extração das seqüências de teste

Nesta etapa, as seqüências deverão ser combinadas para produzirem as seqüências de casos de uso completas para serem testadas. Esta combinação (intercalação) vai gerar um grande número de seqüências, porém na prática somente um número razoável de seqüências devem ser testadas.

A cada vez, duas seqüências finais obtidas na etapa 7 devem ser combinadas. O resultado da combinação de duas seqüências deve ser combinado com a próxima seqüência e assim sucessivamente.

Segue abaixo o exemplo que pode ser utilizado como regra geral para a combinação dos casos de uso, apresentado na abordagem sobre a metodologia TOTEM, no capítulo 4, seção 4.4.1.3:

Prefixo1 . X . Meio1 . Y . Sufixo1

Prefixo2 . X . Meio1 . Y . Sufixo2 , onde x e y representam casos de uso em comum e Prefixo, Meio e Sufixo representam qualquer subseqüência que não tem elementos em comum. Entende-se por elementos em comum, mesmos casos de uso com mesmos valores para os parâmetros.

Resultado da combinação:
(Prefixo1 || Prefixo2). X . (Meio1 || Meio2) . Y . (Sufixo1 || Sufixo2)

Na etapa anterior, as seqüências finais obtidas foram as Seq 4 e Seq 5. Estas seqüências devem ser combinadas cuidadosamente, preservando as dependências dos casos de uso e evitando a duplicidade dos casos de uso.

Seq 4. A(P00001).C(EX0002-EX).E(SOL2.P00001,EX0002-EX).G(PED2,P00001, SOL2).I(3101).L(00002,P00001,PED2,3101).M(00002,P00001,PED2,3101).J(3101). H(PED2,P00001,SOL2),F.(SOL2.P00001,EX0002-EX).D(EX0002-EX).B(P00001)

Seq 5. C(BR0002-SP).A(P00001).E(SOL1.P00001,BR0001-SP).I(1101).G(PED1,P00001, SOL1).L(00001,P00001,PED1,1101).M(000011,P00001,PED11,1101).H(PED1,P00001,SO L1).J(1101),F.(SOL1.P00001,BR0001-SP).B(P00001).D(BR0001-SP)

Segue abaixo a seqüência S1, que é uma das possíveis seqüências completas extraídas entre a combinação das seqüências 4 e 5. A seqüência S1 deve ser utilizada nos testes.

S1. A(P00001).C(BR0001-SP).C(EX0002-EX).E(SOL1.P00001-SP).E(SOL2.P00001, EX0002-EX).I(1101).G(PED2.P00001.SOL2).G(PED1,P00001,SOL1).I(3101). L(000001,P00001,PED1,1101).L(000002,P00001,PED2,3101).M(000001,P00001,PED1, 1101).M(000002,P00001,PED2).H(PED1,P00001,SOL1).J(3101).J(1101).H(PED2, P00001,SOL2).F(SOL1,P00001,BR0001-SP).F(SOL2,P00001,EX0002-EX).B(P00001). D(EX0002-EX).D(BR0001-SP).B(P00001)

5.3.10 Análise dos resultados dos testes e balanço dos testes realizados.

Após a realização dos testes, o resultado deve ser analisado. Isto foi aprovado ou não? Estas análises resultam em relatórios de teste. Um relatório de teste contém um resumo do teste aplicado e é também o relatório final de teste. O resumo deve ser breve e conter conclusões, os recursos gastos e se o teste foi aprovado ou rejeitado. O resultado de cada subteste é também mostrado neste relatório com o resultado, recursos gastos e a ação tomada, caso ela exista [Jacobson, 95-337].

Na tabela 5.10 é apresentado um exemplo simplificado de relatório de incidentes de testes.

Relatório de Incidentes de testes	
Identificador	REL-ETIERP01
Contexto	Caso de teste de inclusão de solicitação de compra – ETIERP01, executando o procedimento PROD-ETIERP01
Descrição	Resultado esperado: Resultado obtido:
Impacto	

Tabela 5.8. Relatório de incidentes de testes do caso de teste inclusão da solicitação de compras

5.4 Resumo da estratégia proposta

A etapa 1 da estratégia proposta, aborda o planejamento de testes, quando são estabelecidos os módulos a serem testados, o cronograma e a equipe de testes.

As etapas 2 – definição do diagrama de casos de uso, 3 – descrição dos casos de uso, 6 – definição do diagrama de atividades, 7 – extração das seqüências de casos de uso, 8 – análise da dependência de parâmetros e 9 – extração das seqüências de testes foram baseadas na metodologia TOTEM.

A etapa 4 – identificação das variáveis operacionais, definição de domínios e desenvolvimento da relação operacional e a etapa 5 – estabelecimento de valores para as variantes, foram baseadas na abordagem de Robert V. Binder, sobre testes de caso de uso estendidos.

O motivo para a não utilização de toda a metodologia TOTEM é que ela se utiliza de diagramas de classes e de seqüências a fim de extrair informações para a criação de cenários. O cliente do sistema ERP normalmente não dispõe destas informações.

A TOTEM trabalha com valores simbólicos que mais tarde deverão ser substituídos pelos valores verdadeiros, apesar deste ponto não ser abordado no trabalho e fazer parte de trabalhos futuros. A ETIERP foi criada de forma que os valores reais já possam ser utilizados desde o princípio, conseqüentemente reduzindo o tempo gasto na execução da mesma.

Uma limitação da estratégia apresentada e também da TOTEM, é que ela desconsidera as situações inválidas. Por exemplo, o Prod-id (o parâmetro identificador do produto) inválido não poderia ser utilizado em seqüências grandes, pois caso contrário, a execução das seqüências de casos de uso seriam interrompidas. Porém estas situações inválidas devem ser consideradas para os casos de teste, pois caso contrário o sistema não será testado em situação de erro.

Finalmente, a etapa 11 abordam a análise dos resultados dos testes e balanço dos testes realizados.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho propôs uma estratégia de testes para ser empregada na situação da implantação de um sistema ERP.

A estratégia se preocupa com os testes funcionais, pois supõe-se que testes baseados na estrutura do sistema, entre outros, já foram realizados pela empresa fornecedora do software. A empresa que adquire o sistema não possuirá a especificação do sistema, mas possuirá manuais e acesso ao conhecimento de usuários-chave, consultores e analistas de sistemas envolvidos na implantação. Estas fontes serão importantes na geração dos modelos da UML (diagramas de casos de uso e de atividades), que serão utilizados na estratégia.

A ETIERP foi baseada na metodologia TOTEM – para geração de requisitos para testes e também na abordagem sobre casos de uso estendidos. A estratégia proposta procura selecionar a parte mais interessante da TOTEM para a situação da implantação de um sistema ERP e completá-la com os testes de caso de uso estendidos para a geração de cenários e valores, a fim de torná-la o mais prática possível e aplicável.

A fim de facilitar a utilização da ETIERP a automatização de certas etapas deve ser fortemente considerada. Não é possível automatizar a criação de diagramas de atividades, porém é possível fazê-lo para a geração de seqüência de casos de uso, assim como a combinação das seqüências. A metodologia TOTEM foi construída de forma a permitir a automatização. Conseqüentemente, a ETIERP também possui grande potencial para automatização, porém isto não faz parte do escopo deste trabalho.

Uma das limitações da ETIERP é o fato de a mesma não tratar situações inválidas. As situações válidas e inválidas são levantadas na etapa 4 da estratégia proposta. Porém as situações inválidas não são consideradas nas seqüências de casos de uso, pois para que uma seqüência possa ser concluída totalmente, é necessário que todas as situações sejam válidas, pois a execução de um caso de uso depende do sucesso da execução do caso de uso antecessor. O teste para as situações inválidas deve ser considerado para complemento deste trabalho, a fim de que o sistema também seja testado numa situação de erro. Outra limitação é que, neste primeiro instante, a estratégia visa a preparação dos testes separados por

módulos (divisão para comercialização do software). Para que os testes sejam completos é necessário testar também o sistema como um todo.

No capítulo 2, foram abordadas informações sobre o sistema ERP. No capítulo 3 foram passados importantes conceitos sobre testes de software. No capítulo 4 foram apresentadas algumas metodologias para testes de sistemas. No capítulo 5, a ETIERP, proposta de trabalho, foi apresentada. Finalmente, este capítulo apresenta a conclusão do trabalho, contribuições, dificuldades encontradas e trabalhos futuros.

5.1. Contribuições

A estratégia contribui para que a fase de testes, geralmente negligenciada, principalmente numa situação de implantação de sistema ERP, seja realizada de forma metódica e sistemática a fim de que esta fase não seja comprometida devido à perda de alguma informação importante, que porventura pudesse vir a comprometer a qualidade dos testes e conseqüentemente a implantação do sistema.

5.2. Dificuldades encontradas

Como o sistema ERP é um sistema de grande abrangência, torna-se difícil executar todas as etapas da estratégia manualmente. A automatização de parte da estratégia facilitaria muito a aplicação da mesma.

5.3. Trabalhos futuros

Durante a realização deste trabalho foram identificados os seguintes trabalhos futuros:

- Automatização da estratégia, a fim de facilitar a aplicação da mesma.
- Criação da etapa referente ao teste do sistema como um todo. Como já abordado neste capítulo, a ETIERP visa a realização dos testes de casos de uso agrupados por módulos (divisão comercial do ERP). É necessário que seja acrescentada na estratégia uma parte referente ao teste de todos os módulos adquiridos de forma integrada, ou seja, o teste do sistema ERP como um todo.
- Realização de testes de sistema visando aspectos não-funcionais. A ETIERP se concentra no teste dos aspectos funcionais do sistema. É necessário acrescentar no

trabalho, os testes referentes aos aspectos não-funcionais também, como testes de sobrecarga, de recuperação, entre outros.

Anexo I

Descrição dos casos de uso

Nome do caso de uso: Incluir produto
Ator: Compras
Descrição: Um solicitante, que pode ser qualquer funcionário da empresa, necessita comprar um novo produto, ainda não cadastrado no sistema.
Pré_condições: Produto ainda não incluso no sistema.
*Pós_condições:*Produto incluso com sucesso.
Fluxo básico: Código do produto informado corretamente. Produto incluso com sucesso.
Fluxo alternativo: Código do produto inválido. Não foi possível incluir o produto.

Nome do caso de uso: Excluir produto
Ator: Compras
Descrição: Excluir informações sobre o produto.
Pré_condições: Produto já incluso no sistema e ainda não utilizado em nenhuma outra operação.
*Pós_condições:*Produto excluído com sucesso.
Fluxo básico: Código do produto informado corretamente. Produto excluído com sucesso.
Fluxo alternativo: Código do produto inválido. Não foi possível excluir o produto.

Nome do caso de uso: Incluir fornecedor
Ator: Compras / Contas a pagar
Descrição: Incluir informações sobre um novo fornecedor.
Pré_condições: Fornecedor ainda não incluso no sistema.
*Pós_condições:*Fornecedor incluso com sucesso.
Fluxo básico: Código do fornecedor e código do estado informados corretamente. Fornecedor incluso com sucesso.
Fluxo alternativo: - Código do fornecedor inválido. Não foi possível incluir o fornecedor.
- Código do fornecedor correto. Código do estado inválido. Não foi possível incluir o fornecedor.

Nome do caso de uso: Excluir fornecedor
Ator: Compras / Contas a Pagar
Descrição: Excluir informações sobre o fornecedor.
Pré_condições: Fornecedor já incluso no sistema e ainda não utilizado em nenhuma operação.
*Pós_condições:*Fornecedor excluído com sucesso.

Fluxo básico: Código do fornecedor informado corretamente. Fornecedor excluído com sucesso.
Fluxo alternativo: - Código do fornecedor inválido. Não foi possível excluir o fornecedor.

Nome do caso de uso: Incluir solicitação
Ator: Compras
Descrição: Incluir uma nova solicitação de compras no sistema. Quando qualquer funcionário da empresa deseja comprar determinado produto, ele deve criar uma nova solicitação de compras.
Pré_condições: Produto e fornecedor previamente inclusos no sistema.
*Pós_condições:*Solicitação inclusa com sucesso.
Fluxo básico: Código do produto e código do fornecedor informados corretamente. Solicitação inclusa com sucesso.
Fluxo alternativo: - Código do produto inválido. Não foi possível incluir a solicitação.
- Código do fornecedor inválido. Não foi possível incluir a solicitação.

Nome do caso de uso: Excluir solicitação
Ator: Compras
Descrição: Excluir solicitação de compras do sistema.
Pré_condições: Solicitação previamente inclusa no sistema e ainda não associada a nenhum pedido de compras.
*Pós_condições:*Solcitação excluída com sucesso.
Fluxo básico: Código da solicitação informado corretamente. Solicitação inclusa com sucesso.
Fluxo alternativo: - Código da solicitação inválido. Não foi possível excluir a solicitação.

Nome do caso de uso: Incluir pedido
Ator: Compras
Descrição: Incluir um novo pedido de compras no sistema. O comprador cria um novo pedido de compras no sistema a partir de uma solicitação de compras criada por algum funcionário da empresa. Quando o pedido é criado, a solicitação é encerrada.
Pré_condições: Solicitação de compras já inclusa no sistema e ainda em aberto.
*Pós_condições:*Pedido incluído com sucesso e solicitação encerrada.
Fluxo básico: Código do produto e código da solicitação informados corretamente. Pedido incluso com sucesso.
Fluxo alternativo: - Código do produto inválido. Não foi possível incluir o pedido.
- Código da solicitação inválida. Não foi possível incluir o pedido.

Nome do caso de uso: Excluir pedido
Ator: Compras
Descrição: Excluir pedido de compras do sistema. Quando o pedido é excluído, a solicitação de compras é reaberta.
Pré_condições: Pedido de compras já incluso no sistema e ainda não realizado nenhum recebimento para este pedido.

*Pós_condições:*Pedido excluído com sucesso e solicitação reaberta.
Fluxo básico: Código do pedido informado corretamente. Pedido excluído com sucesso.
Fluxo alternativo: - Código do pedido inválido. Não foi possível excluir o pedido.

Nome do caso de uso: Incluir tipo de entrada
Ator: Compras
Descrição: Incluir um novo tipo de entrada. Este tipo de entrada definirá informações como código fiscal de operação, quais impostos serão calculados, entre outros.
Pré_condições: Tipo de entrada ainda não incluso no sistema.
Pós_condições: Tipo de entrada incluso com sucesso.
Fluxo básico: Código do tipo de entrada informado corretamente. Tipo de entrada incluso com sucesso.
Fluxo alternativo: - Código do tipo de entrada inválido. Não foi possível incluir tipo de entrada.

Nome do caso de uso: Excluir tipo de entrada
Ator: Compras
Descrição: Excluir tipo de entrada.
Pré_condições: Tipo de entrada já incluso no sistema.
Pós_condições: Tipo de entrada excluído com sucesso.
Fluxo básico: Código do tipo de entrada informado corretamente. Tipo de entrada excluído com sucesso.
Fluxo alternativo: - Código do tipo de entrada inválido. Não foi possível excluir o tipo de entrada.

Nome do caso de uso: Incluir recebimento
Ator: Compras
Descrição: Incluir um novo recebimento de material. No recebimento do material na empresa, os dados da nota fiscal enviada pelo fornecedor, devem ser digitados no sistema e um pedido compras é associado. Quando o recebimento é incluso, o pedido de compras é encerrado.
Pré_condições: Pedido de compras e tipo de entrada já cadastrados no sistema e recebimento ainda não cadastrado no sistema..
Pós_condições: Recebimento cadastrado com sucesso.
Fluxo básico: Número da nota fiscal, produto, pedido e código do tipo de entrada informados corretamente. Recebimento incluso com sucesso.
Fluxo alternativo: - Número da nota fiscal inválido. Não foi possível incluir o recebimento.
- Código do produto inválido. Não foi possível incluir o recebimento.
- Código do pedido inválido. Não foi possível incluir o recebimento.
-Código do tipo de entrada inválido. Não foi possível incluir o recebimento.

Nome do caso de uso: Excluir recebimento
Ator: Compras

Descrição: Excluir recebimento de material. Quando o recebimento é excluído, o pedido de compras é reaberto.
Pré_condições: Recebimento já incluso no sistema.
Pós_condições: Recebimento excluído com sucesso.
Fluxo básico: Número da nota fiscal informado corretamente. Recebimento excluído com sucesso.
Fluxo alternativo: - Número da nota fiscal inválido. Não foi possível excluir o recebimento.

Nome do caso de uso: Incluir título a pagar
Ator: Contas a pagar
Descrição: Através deste caso de uso, é registrado um título a pagar para determinado fornecedor. Este caso de uso pode ser incluso manualmente ou pode ser gerado automaticamente pelo sistema, quando é realizado o recebimento de determinado produto.
Pré_condições: Fornecedor já incluso no sistema.
Pós_condições: Título a pagar incluso com sucesso.
Fluxo básico: Número do título e código do fornecedor informados corretamente. Título a pagar incluso com sucesso.
Fluxo alternativo: - Número do título a pagar inválido. Não foi possível incluir o título a pagar. - Código do fornecedor inválido. Não foi possível incluir o título a pagar.

Nome do caso de uso: Excluir título a pagar
Ator: Contas a pagar
Descrição: Exluir do sistema determinado título a pagar.
Pré_condições: Título a pagar já registrado no sistema e ainda não baixado.
Pós_condições: Título a pagar excluído com sucesso.
Fluxo básico: Número do título informado corretamente. Título a pagar excluído com sucesso.
Fluxo alternativo: - Número do título a pagar inválido. Não foi possível excluir o título a pagar.

Nome do caso de uso: Baixar título a pagar
Ator: Contas a pagar
Descrição: Através deste caso de uso, o título a pagar de determinado fornecedor poderá ser encerrado.
Pré_condições : Título a pagar ainda em aberto.
Pós_condições: Título a pagar encerrado com sucesso.
Fluxo básico: Número do título informado corretamente. Título a pagar encerrado com sucesso.
Fluxo alternativo: - Número do título a pagar inválido. Não foi possível encerrado o título a pagar.

Nome do caso de uso: Estornar título a pagar
Ator: Contas a pagar

Descrição: Através deste caso de uso, um título a pagar já encerrado poderá ser estornado. No estorno, é gerado um título com crédito para o fornecedor.
Pré_condições: Título a pagar já baixado.
Pós_condições: Estorno realizado com sucesso.
Fluxo básico: Número do título informado corretamente. Título a pagar estornado com sucesso.
Fluxo alternativo: - Número do título a pagar inválido. Não foi possível estornar o título a pagar.

Anexo II

Identificação das variáveis operacionais, definição dos domínios e desenvolvimento da relação operacional

Caso de uso: Incluir produto

Variáveis operacionais		Resultado esperado	
Variante	Prod_id	Ação	Mensagem
1	Inválido	Produto não incluso no sistema	"Código de produto inválido"
2	Válido	Produto incluso no sistema	"Inclusão do produto efetuada com sucesso"

Caso de uso: Excluir produto

Variáveis operacionais		Resultado esperado	
Variante	Prod_id	Ação	Mensagem
1	Inválido	Produto não excluído do sistema	"Código de produto inválido"
2	Válido	Produto excluído do sistema	"Exclusão do produto efetuada com sucesso"

Caso de uso: Incluir fornecedor

Variáveis operacionais			Resultado esperado	
Variante	Forn_id	Forn_uf	Ação	Mensagem
1	Inválido	Não importa	Fornecedor não incluso no sistema	"Código de fornecedor inválido"
2	Válido	Inválido	Fornecedor não incluso no sistema	"Código do estado inválido"
3	Válido	Válido	Fornecedor incluso no sistema	"Inclusão do fornecedor efetuada com sucesso"

Caso de uso: Excluir fornecedor

Variáveis operacionais		Resultado esperado	
Variante	Forn_id	Ação	Mensagem
1	Inválido	Fornecedor não excluído do sistema	"Código de fornecedor inválido"
2	Válido	Fornecedor excluído do sistema	"Exclusão do fornecedor efetuada com sucesso"

Caso de uso: Incluir solicitação

Variante	Sol_id	Sol_prod	Sol_forn	Ação	Mensagem
1	Seqüencial	Inválido	Não importa	Solicitação não inclusa no sistema	"Código do produto não cadastrado"
2	Seqüencial	Válido	Inválido	Solicitação não inclusa no sistema	"Código do fornecedor não cadastrado"
3	Seqüencial	Válido	Válido	Solicitação inclusa no sistema	"Inclusão da solicitação efetuada com sucesso"

Caso de uso: Excluir solicitação

Variáveis operacionais		Resultado esperado	
Variante	Sol_id	Ação	Mensagem
1	Inválido	Solicitação não excluída do sistema	"Código da solicitação não cadastrado"
2	Válido	Solicitação excluída do sistema	"Solicitação excluída com sucesso"

Caso de uso: Incluir pedido

Variáveis operacionais				Resultado esperado	
Variante	Ped_id	Ped_sol_prod	Ped_sol	Ação	Mensagem
1	Seqüencial	Inválido	Não importa	Pedido não incluso no sistema	"Código do produto não cadastrado"
2	Seqüencial	Válido	Inválido	Pedido não incluso no sistema	"Código do solicitação não cadastrado"
3	Seqüencial	Válido	Válido	Pedido incluso no sistema	"Inclusão do pedido efetuada com sucesso"

Caso de uso: Excluir pedido

Variáveis operacionais		Resultado esperado	
Variante	Ped_id	Ação	Mensagem
1	Inválido	Pedido não excluído do sistema	“Código do pedido não cadastrado”
2	Válido	Pedido excluído do sistema	“Pedido excluído com sucesso”

Caso de uso: Incluir código de entrada

Variáveis operacionais		Resultado esperado	
Variante	Ce_id	Ação	Mensagem
1	Inválido	Código de entrada não incluso no sistema	“Código de entrada inválido”
2	Válido	Código de entrada incluso no sistema	“Código de entrada cadastrado com sucesso”

Caso de uso: Excluir código de entrada

Variáveis operacionais		Resultado esperado	
Variante	Ce_id	Ação	Mensagem
1	Inválido	Código de entrada não excluído do sistema	“Código de entrada inválido”
2	Válido	Código de entrada excluído do sistema	“Código de entrada excluído com sucesso”

Caso de uso: Incluir recebimento

Variáveis operacionais					Resultado esperado	
Variante	Rec_nota	Rec_ped_prod_sol	Rec_ped_sol	Rec_ce	Ação	Mensagem
1	Inválido	Não importa	Não importa	Não importa	Recebimento não incluso no sistema	"Nota fiscal inválida"
2	Válido	Inválido	Não importa	Não importa	Recebimento não incluso no sistema	"Produto não cadastrado"
3	Válido	Válido	Inválido	Não importa	Recebimento não incluso no sistema	"Pedido não cadastrado"
4	Válido	Válido	Válido	Inválido	Recebimento não incluso no sistema	"Código de entrada não cadastrado"
5	Válido	Válido	Válido	Válido	Recebimento incluso no sistema	"Recebimento incluso com sucesso"

Caso de uso: Excluir recebimento

Variáveis operacionais		Resultado esperado	
Variante	Rec_nota	Ação	Mensagem
1	Inválido	Recebimento não excluído do sistema	"Nota fiscal inválida"
2	Válido	Recebimento excluído do sistema	"Recebimento excluído com sucesso"

Caso de uso: Incluir título a pagar

Variáveis operacionais			Resultado esperado	
Variante	Título_id	Título_Forn	Ação	Mensagem
1	Inválido	Não importa	Título a pagar não incluso no sistema	"Código do título inválido"
2	Válido	Inválido	Título a pagar não incluso no sistema	"Código do fornecedor inválido"
3	Válido	Válido	Título a pagar incluso no sistema	"Título a pagar incluso com sucesso"

Caso de uso: Excluir título a pagar

Variáveis operacionais		Resultado esperado	
Variante	Título_id	Ação	Mensagem
1	Inválido	Título a pagar não excluído do sistema	“Código do título inválido”
2	Válido	Título a pagar excluído do sistema	“Título excluído com sucesso”

Caso de uso: Baixar título a pagar

Variáveis operacionais		Resultado esperado	
Variante	Título_id	Ação	Mensagem
1	Inválido	Título a pagar não encerrado no sistema	“Código do título inválido”
2	Válido	Título a pagar encerrado no sistema	“Título encerrado com sucesso”

Caso de uso: Estornar título a pagar

Variáveis operacionais		Resultado esperado	
Variante	Título_id	Ação	Mensagem
1	Inválido	Título a pagar não estornado no sistema	“Código do título inválido”
2	Válido	Título a pagar estornado no sistema	“Título estornado com sucesso”

Anexo III

Estabelecimento de valores para as variantes

Caso de uso: Incluir produto

Variáveis operacionais		Resultado esperado	
Variante	Prod_id	Ação	Mensagem
1	Inválido	Produto não incluso no sistema	"Código de produto inválido"
1V	%@#		
1F	Qualquer teste verdadeiro para a variante 2		
2	Válido	Produto incluso no sistema	"Inclusão do produto efetuada com sucesso"
2V	P00001		
2F	Qualquer teste verdadeiro para a variante 1		

Caso de uso: Excluir produto

Variáveis operacionais		Resultado esperado	
Variante	Prod_id	Ação	Mensagem
1	Inválido	Produto não excluído do sistema	"Código de produto inválido"
1V	%@#		
1F	Qualquer teste verdadeiro para a variante 2		
2	Válido	Produto excluído do sistema	"Exclusão do produto efetuada com sucesso"
2V	P00001		
2F	Qualquer teste verdadeiro para a variante 1		

Caso de uso: Incluir fornecedor

Variáveis operacionais			Resultado esperado	
Variante	Forn_id	Forn_uf	Ação	Mensagem
1	Inválido	Não importa	Fornecedor não incluso no sistema	"Código de fornecedor inválido"
1V	%@#			
1F	Qualquer teste verdadeiro para as variantes 2 e 3			
2	Válido	Inválido	Fornecedor não incluso no sistema	"Código do estado inválido"
2V	BR0001	%@		
2F	Qualquer teste verdadeiro para as variantes 1 e 3			
3	Válido	Válido	Fornecedor incluso no sistema	"Inclusão do fornecedor efetuada com sucesso"
3V	BR0001	SP		
3V	EX0002	EX		
3F	Qualquer teste verdadeiro para a variante 1 e 2			

Caso de uso: Excluir fornecedor

Variáveis operacionais		Resultado esperado	
Variante	Forn_id	Ação	Mensagem
1	Inválido	Fornecedor não excluído no sistema	"Código do fornecedor inválido"
1V	%@#		
1F	Qualquer teste verdadeiro para a variante 2		
2	Válido	Fornecedor excluído do sistema	"Exclusão do fornecedor efetuada com sucesso"
2V	BR0001		
2V	EX0002		
2F	Qualquer teste verdadeiro para a variante 1		

Caso de uso: Incluir solicitação

Variáveis operacionais				Resultado esperado	
Variante	Sol_id	Sol_prod	Sol_forn	Ação	Mensagem
1	Seqüencial	Inválido	Não importa	Solicitação não inclusa no sistema	"Código do produto não cadastrado"
1V		#@%			
1F	Qualquer teste verdadeiro para as variantes 2 e 3				
2	Seqüencial	Válido	Inválido	Solicitação não inclusa no sistema	"Código do fornecedor não cadastrado"
2V		P00001	#@%		
2F	Qualquer teste verdadeiro para as variantes 1 e 3				
3	Seqüencial	Válido	Válido	Solicitação inclusa no sistema	"Inclusão da solicitação efetuada com sucesso"
3V	Sol1	P00001	BR0001		
3V	Sol2	P00001	EX0002		
3F	Qualquer teste verdadeiro para as variantes 1 e 2				

Caso de uso: Excluir solicitação

Variáveis operacionais		Resultado esperado	
Variante	Sol_id	Ação	Mensagem
1	Inválido	Solicitação não excluída do sistema	"Código da solicitação não cadastrado"
1V	#@%		
1F	Qualquer teste verdadeiro para a variante 2		
2	Válido	Solicitação excluída do sistema	"Exclusão da solicitação efetuada com sucesso"
2V	Sol1		
2V	Sol2		
2F	Qualquer teste verdadeiro para a variante 1		

Caso de uso: Incluir pedido

Variáveis operacionais				Resultado esperado	
Variante	Ped_id	Ped_sol_prod	Ped_sol	Ação	Mensagem
1	Seqüencial	Inválido	Não importa	Pedido não incluso no sistema	"Código do produto não cadastrado"
1V		%@#			
1F	Qualquer teste verdadeiro para as variantes 2 e 3				
2	Seqüencial	Válido	Inválido	Pedido não incluso no sistema	"Código da solicitação não cadastrada"
2V		P00001	%@#		
2F	Qualquer teste verdadeiro para as variantes 1 e 3				
3	Seqüencial	Válido	Válido	Pedido incluso no sistema	"Inclusão do pedido efetuada com sucesso"
3V	Ped1	P00001	Sol1		
3V	Ped2	P00001	Sol2		
3F	Qualquer teste verdadeiro para as variantes 1 e 2				

Caso de uso: Excluir pedido

Variáveis operacionais		Resultado esperado	
Variante	Ped_id	Ação	Mensagem
1	Inválido	Pedido não excluído do sistema	"Código do pedido não cadastrado"
1V	%@#		
1F	Qualquer teste verdadeiro para a variante 2		
2	Válido	Pedido excluído do sistema	"Exclusão do pedido efetuada com sucesso"
2V	Ped1		
2V	Ped2		
2F	Qualquer teste verdadeiro para a variante 1		

Caso de uso: Incluir código de entrada

Variáveis operacionais		Resultado esperado	
Variante	Ce_id	Ação	Mensagem
1	Inválido	Código de entrada não incluso no sistema	“Código de entrada inválido”
1V	%@#		
1F	Qualquer teste verdadeiro para a variante 2		
2	Válido	Código de entrada incluso no sistema	“Código do entrada cadastrado com sucesso”
2V	1101		
2V	3101		
2F	Qualquer teste verdadeiro para a variante 1		

Caso de uso: Excluir código de entrada

Variáveis operacionais		Resultado esperado	
Variante	Ce_id	Ação	Mensagem
1	Inválido	Código de entrada não excluído do sistema	“Código de entrada inválido”
1V	%@#		
1F	Qualquer teste verdadeiro para a variante 2		
2	Válido	Código de entrada excluído do sistema	“Código do entrada excluído com sucesso”
2V	1101		
2V	3101		
2F	Qualquer teste verdadeiro para a variante 1		

Caso de uso: Incluir recebimento

Variáveis operacionais					Resultado esperado	
Variante	Rec_nota	Rec_ped_prod_sol	Rec_ped_sol	Rec_ce	Ação	Mensagem
1	Inválido	Não importa	Não importa	Não importa	Inclusão não permitida	"Número de nota fiscal inválido"
1V	%@#					
1F	Qualquer teste verdadeiro para as variantes 2, 3, 4 ou 5					
2	Válido	Inválido	Não importa	Não importa	Inclusão não permitida	"Produto não cadastrado no pedido"
2V	000001	#@%				
2F	Qualquer teste verdadeiro para as variantes 1, 3, 4 ou 5					
3	Válido	Válido	Inválido	Não importa	Inclusão não permitida	"Pedido não cadastrado"
3V	000001	P00001	#@%			
3F	Qualquer teste verdadeiro para as variantes 1, 2, 4 ou 5					
4	Válido	Válido	Válido	Inválido	Inclusão não permitida	"Código de entrada não cadastrado"
4V	000001	P00001	Ped1	#@%		
4F	Qualquer teste verdadeiro para as variantes 1, 2, 3 ou 5					
5	Válido	Válido	Válido	Válido	Inclusão permitida	"Recebimento incluído com sucesso"
5V	000001	P00001	Ped1	1101		
5V	000002	P00001	Ped2	3101		
5F	Qualquer teste verdadeiro para as variantes 1, 2, 3 ou 4					

Caso de uso: Excluir recebimento

Variáveis operacionais		Resultado esperado	
Variante	Rec_nota	Ação	Mensagem
1	Inválido	Recebimento não excluído do sistema	"Número de nota fiscal inválido"
1V	%@#		
1F	Qualquer teste verdadeiro para a variantes 2		
2	Válido	Recebimento excluído do sistema	"Recebimento excluído com sucesso"
2V	000001		
2V	000002		
2F	Qualquer teste verdadeiro para a variante 1		

Caso de uso: Incluir título a pagar

Variáveis operacionais			Resultado esperado	
Variante	Título_id	Título_Forn	Ação	Mensagem
1	Inválido	Não importa	Título a pagar não incluso no sistema	"Código do título inválido"
1V	%@#			
1F	Qualquer teste verdadeiro para as variantes 2 e 3			
2	Válido	Inválido	Título a pagar não incluso no sistema	"Código do fornecedor inválido"
2V	T00001	%@		
2F	Qualquer teste verdadeiro para as variantes 1 e 3			
3	Válido	Válido	Título a pagar incluso no sistema	"Título a pagar incluso com sucesso"
3V	T00001	BR0001		
3V	T00002	EX0001		
3F	Qualquer teste verdadeiro para as variantes 1 e 2			

Caso de uso: Excluir título a pagar

Variáveis operacionais		Resultado esperado	
Variante	Rec_nota	Ação	Mensagem
1	Inválido	Título a pagar não excluído do sistema	“Código do título inválido”
1V	%@#		
1F	Qualquer teste verdadeiro para a variantes 2		
2	Válido	Título a pagar excluído do sistema	“Título excluído com sucesso”
2V	T00001		
2V	T00002		
2F	Qualquer teste verdadeiro para a variante 1		

Caso de uso: Baixar título a pagar

Variáveis operacionais		Resultado esperado	
Variante	Rec_nota	Ação	Mensagem
1	Inválido	Título a pagar não encerrado no sistema	“Código do título inválido”
1V	%@#		
1F	Qualquer teste verdadeiro para a variantes 2		
2	Válido	Título a pagar encerrado no sistema	“Título encerrado com sucesso”
2V	T00001		
2V	T00002		
2F	Qualquer teste verdadeiro para a variante 1		

Caso de uso: Estornar título a pagar

Variáveis operacionais		Resultado esperado	
Variante	Rec_nota	Ação	Mensagem
1	Inválido	Título a pagar não estornado	“Código do título inválido”
1V	%@#		
1F	Qualquer teste verdadeiro para a variantes 2		
2	Válido	Título a pagar estornado	“Título estornado com sucesso”
2V	T00001		
2V	T00002		
2F	Qualquer teste verdadeiro para a variante 1		

Bibliografia

[Bin00] Robert V. Binder. Testing Object-Oriented Systems:models, patterns and tools. Addison-Wesley Inc., 2000.

[B9la,02] Lionel Briand, Yvan Labiche. A UML Based Approach to System Testing. Carleton University. June, 2002. (http://www.sce.carleton.ca/Squall/Totem/).

[BRJ00] G. Booch, J. Rumbaugh, I. Jacobson. UML, Guia do Usuário. Editora Campus Ltda, 2000

[Int01] Cândida Inthurn. Qualidade e Teste de Software. Visual Books, 2001.

[Jac95] I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard. Object-Oriented Software Engineering – A Use Case Driven Approach. ACM Press, 1995.

[Pre92] Roger S. Pressman. Software Engineering: a pratictioner's approach – Third Edition. McGraw-Hill Inc.,1992.

[Som92] Ian Sommerville. Software Engineering – 4th. Edition. Addison-Wesley Inc., 2000.

[Vol,01] Lisiane Maes Volpi, Uma Estratégia de Teste de Software para Ambiente Cliente-Servidor. Universidade Federal do Paraná – Mestrado em Informática, Agosto, 2001.

[Oliv00] Jair F. Oliveira, Sistemas de Informação – Um enfoque gerencial inserido no contexto empresarial e tecnológico. São Paulo: Érica, 2000.

[Tau98] C. Taurion - Sistemas de Gestão Empresarial: a Solução Final?, Developers, Rio de Janeiro, ano 2, n. 20, pp. 10-11, abril 1998.

[Bell98] A. Belloquim: a Nova - ERP: a Nova Solução Definitiva Para Todos os Problemas,

Developers, Rio de Janeiro, ano 2, n. 20, pp. 38-41, abril 1998.