Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Simone Tejo Salgado Beato

Casos de Teste de Uso: uma contribuição para o desenvolvimento de *software* dirigido por testes

São Paulo

2008

Lombada

Simone Tejo Salgado Beato

Casos de Teste de Uso: uma contribuição para o desenvolvimento de *software* dirigido por testes

Simone Tejo Salgado Beato

Casos de Teste de Uso: uma contribuição para o desenvolvimento de *software* dirigido por testes

Dissertação apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, para obtenção do título de Mestre em Engenharia da Computação.

Área de concentração: Engenharia de *Software*

Orientador: Dr. José Eduardo Zindel Deboni

São Paulo

2008

Ficha Catalográfica
Elaborada pelo Departamento de Acervo e Informação Tecnológica – DAIT
do Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT

B369c **Beato, Simone Tejo Salgado**
Caso de teste de uso: uma contribuição para o desenvolvimento de software dirigido por testes. / Simone Tejo Salgado Beato. São Paulo, 2008.
131p.

Dissertação (Mestrado em Engenharia de Computação) - Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Área de concentração: Engenharia de Software.

Orientador: Prof. Dr. José Eduardo Zindel Deboni

1. Engenharia de software 2. Processo unificado 3. Desenvolvimento dirigido por testes 4. Tese I. Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Coordenadoria de Ensino Tecnológico II. Título

08-149 CDU 004.415.53(043)

Dedico este trabalho aos meus grandes amores e inspiradores: Guilherme, meu amado filho, Roberto, meu marido, e aos meus pais Célia e José Mário.

AGRADECIMENTOS

Ao Prof. Dr. José Eduardo Zindel Deboni, por sua sabedoria, conselhos e incentivo durante todo o processo de orientação e desenvolvimento deste trabalho, o que me proporcionou uma visão mais ampla do tema.

Ao Programa de Mestrado Profissional em Engenharia da Computação do Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, representado por todos os seus profissionais, professores e funcionários que me proporcionaram uma excelente experiência de aprendizagem.

Aos membros da Banca de Avaliação, Profª. Drª. Márcia Ito e Prof. Dr. Fábio Kon pelas importantes sugestões para a melhoria deste estudo, na ocasião da qualificação.

A Roberto, meu querido marido, pois sem seu incentivo, paciência, apoio e amor, talvez tudo isso não fosse possível.

A Guilherme, meu amado filho, que mesmo com apenas dois anos me incentivou com seu adorável sorriso e sua contagiante alegria.

Aos meus familiares, especialmente meus pais Célia e José Mário, por serem a minha base e por sempre apoiarem todas as minhas ações, decisões e por acreditarem em mim.

A Deus por ter me guiado e me dado forças, principalmente nos momentos mais difíceis deste trabalho.

A todos aqueles que, direta ou indiretamente, contribuíram para o desenvolvimento deste trabalho.

RESUMO

O desenvolvimento de um *software* envolve uma série de atividades em que a possibilidade de inserção de erros é bastante alta. Alguns desses erros podem ser identificados e corrigidos se o *software* contiver um conjunto de testes que garanta que as funcionalidades capturadas na fase de especificação sejam, de fato, testadas no produto final. Dessa forma, os casos de testes podem ser utilizados desde as fases iniciais do desenvolvimento do *software*, passando a ser um artefato formal para validação e verificação dos requisitos, o que reduz a possibilidade de equívocos durante a especificação. Este trabalho tem o objetivo de incorporar a atividade de teste na fase de especificação do sistema, substituindo a especificação de requisitos do *software* baseada em casos de uso, por um novo artefato baseado em casos de teste. Tal artefato concentra-se na integração do modelo de casos de uso com os casos de testes e tem a finalidade de substituir o detalhamento dos casos de uso pelos casos de teste, criando o conceito de especificação dirigida por casos de teste. Esse artefato estimula o uso da técnica de desenvolvimento dirigido por testes (do inglês *Test-driven development*), no processo unificado (UP), assim como na programação extrema (XP). Para validar a técnica e o artefato propostos, apresentam-se dois casos de aplicação e uma pesquisa com um grupo de analistas e desenvolvedores de sistemas, com o intuito de coletar impressões sobre o impacto desse artefato, dentro do processo de desenvolvimento.

Palavras-chave: Processo Unificado; Programação Extrema; Desenvolvimento dirigido por testes; Casos de Teste; Modelagem dirigida por testes; Modelo baseado no Teste Extremo.

ABSTRACT

Use Test Cases: a contribution to test-driven development

Software development encompasses such an amount of activities that the possibility of error insertion is very high. Some of these errors can be identified and corrected if the software contains a set of tests assuring that those functionalities detected during the specification phase are tested in the final product. Hence, the tests cases can be applied since the early stages of software development, acting as a formal artifact for requisite verification and validation, decreasing the possibilities of misinterpretations during software specification. This work aims the incorporation of the test activity into the software specification phase, using a new artifact based on test cases. This artifact is based on the integration of use case model with test cases and proposes to replace the use case detailing by test cases, creating the concept of test case driven specification. The proposed artifact stimulates the usage of test-driven development (TDD) technique, into the Unified Process as well as into Extreme Programming (XP). In order to validate the proposed technique and the artifact, two application cases are presented and a survey with system analysts and developers is carried out in order to collect their opinions about the impact of the proposed artifact during software development.

Keywords: Unified Process ; Extreme Programming; Test-Driven Development; Test Cases; Test-Driven Modeling; Model-Based Extreme Testing;.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURA E SIGLAS

ACM	**Association for Computing Machinery**
ANSI	**American National Standards Institute**
CETIP	**Câmara de Custódia e Liquidação**
CTU	**Caso de Teste de Uso**
EDCT	**Especificação Dirigida por Casos de Teste**
IEEE	**Institute of Electrical and Electronics Engineers**
MDD	**Model-driven Development**
MSC	**Message Sequence Charts**
MXT	**Model-Based Extreme Testing**
NDF	**Non Deliverable Forward**
OOSE	**Object-Oriented Software Engineering**
SDL	**Specification and Description Language**
TDD	**Test-driven Development**
TDM	**Test-driven Modeling**
UML	**Unified Modeling Language**
UP	**Unified Process**
XP	**Extreme Programming**

SUMÁRIO

CAPÍTULO 1
INTRODUÇÃO

Muito tem se pesquisado a respeito dos métodos ágeis para desenvolvimento de *software*, com destaque especial para a programação extrema (do inglês *Extreme Programming* - XP), proposto por Kent Beck, em que um dos objetivos é reduzir o custo das mudanças de requisitos durante o processo de desenvolvimento (BECK, 1999). Uma das práticas mais valorizadas pelo XP é o teste, que deve ser criado antes mesmo de o programa ser escrito, ou seja, é o teste quem orienta o desenvolvimento. Tal prática deu origem ao chamado Desenvolvimento dirigido por testes (do inglês *Test-Driven Development* – TDD) (BECK, 2003).

Nos métodos convencionais de desenvolvimento de *software*, o teste é realizado, em geral, ao final de cada fase. Parte-se do pressuposto de que os testes somente são úteis, ou possíveis de serem executados, com o sistema já desenvolvido.

1.1 Motivação

A construção de um *software* envolve uma série de atividades em que as possibilidades para inserir erros humanos são grandes (DEUTSCH, 1979 apud PRESSMAN, 2001). Por isso, de todas as etapas do processo de desenvolvimento, a atividade de teste deve ser realizada com uma atenção especial.

Nos vários modelos de desenvolvimento de sistemas, como no modelo cascata ou nos modelos iterativos, entre eles o Processo Unificado (também conhecido como *Unified Process* – UP), a elaboração do plano de teste, com os casos de teste funcionais, ocorre em etapa posterior à especificação. Em geral, esse plano é utilizado somente no final do desenvolvimento do sistema.

No entanto, mais de 50% dos defeitos encontrados em um *software* são decorrentes da fase de especificação (PATTON, 2001) e, de acordo com Robertson (2006), esse percentual aumentou para 60% em 2006.

A importância de se investir em teste de *software* vem crescendo nos últimos anos. Muitas empresas já descobriram os benefícios de realizar a atividade de teste mais cedo e,

com isso, estão contratando e treinando programadores e analistas de teste para executarem os testes funcionais (PATTON, 2001).

Qualquer complexidade em um *software* deve ser testada porque código sem teste é uma fonte potencial de falhas. Uma aplicação bem testada tem grande chance de sucesso (BECK, 2003).

De acordo com Myers (2004):

> Testing has been an out-of-vogue subject – it was true when this book was first published and, unfortunately, it is still true today. Today there are books and articles about software testing, meaning that, at least, the topic has more visibility than it did when this book was first published. But testing remains among “dark arts” of software development (MYERS, 2004, p. xiii).

Diante deste panorama e considerando o estado atual do mercado de desenvolvimento de *software*, a motivação deste trabalho é de antecipar a atividade de teste para o início do processo de desenvolvimento de *software*, incentivando e divulgando a atividade de teste, essencial em qualquer processo de desenvolvimento de *software* e não apenas como a etapa final do processo. Para tanto, este trabalho propõe adaptações em alguns modelos de desenvolvimento de *software*.

1.2 Objetivo

O objetivo desse trabalho é incorporar a atividade de teste já na fase de especificação do sistema, substituindo a especificação de requisitos do *software* baseada em casos de uso, por um novo artefato baseado em casos de teste. Este artefato se baseia na integração do modelo de casos de uso com os casos de testes e tem a finalidade de substituir o detalhamento dos casos de uso pelos casos de teste, criando o conceito de especificação dirigida por casos de teste.

1.3 Resultados Esperados e Contribuições

Este trabalho gera um artefato contendo a especificação de requisitos de um *software* no formato de casos de teste. Este artefato pode ser utilizado independente do modelo de desenvolvimento de sistemas.

A contribuição que este estudo dá para a área de Engenharia de *Software* é a da possibilidade de substituir a especificação de requisitos do *software* pelos casos de teste, podendo ser utilizado pelos analistas de sistemas independentes do modelo de desenvolvimento de software que eles utilizam.

Propõe-se a simplificação do processo de desenvolvimento de sistemas por meio da unificação das atividades de especificação de requisitos do *software* e da preparação do plano de testes, produzindo um único artefato a ser utilizado durante o desenvolvimento do sistema, com economia de tempo e de recursos e disponibilizando aos desenvolvedores de sistema, já nas fases iniciais, um conjunto de casos de teste para serem usados durante todo o processo de desenvolvimento, em consonância com o desenvolvimento dirigido por testes (TDD).

1.4 Metodologia de Trabalho

Analisa-se a bibliografia referente à programação extrema (XP), ao desenvolvimento dirigido por testes (TDD), à modelagem dirigida por testes (TDM) e ao processo unificado (UP), além de outros conceitos importantes relacionados à atividade de teste.

Este trabalho parte principalmente do modelo proposto por Kent Beck – desenvolvimento dirigido por testes, em que os casos de teste são criados antes mesmo de o código ser escrito (BECK, 2003). Estuda-se, também, o modelo de desenvolvimento de sistemas do UP, com o objetivo de compreender a dinâmica dos testes nesse modelo, uma vez que o UP utiliza os casos de uso como artefato para especificação de requisitos funcionais.

Descreve-se então, passo a passo, como elaborar o artefato que contém os requisitos do sistema no formato de casos de teste, semelhante ao que foi desenvolvido por Kent Beck no TDD.

Finalmente, valida-se a proposta deste trabalho por meio de dois casos de aplicação, sendo um didático e um prático, em que é produzido um artefato contendo os casos de teste de cada caso de aplicação. A partir desse artefato, realiza-se uma pesquisa com um grupo de desenvolvedores e de analistas de sistemas para avaliar a técnica proposta e, com base nos resultados obtidos, analisa-se essa técnica em relação aos modelos UP e XP.

1.5 Organização do Trabalho

O trabalho está organizado em cinco capítulos. A seguir, apresenta-se um breve resumo do conteúdo destes capítulos.

No Capítulo 1, é apresentada uma introdução do trabalho, em que são tratados: a descrição do problema a ser resolvido, a motivação, o objetivo, a metodologia utilizada para solucionar o problema, bem como os resultados e contribuições deste trabalho.

No Capítulo 2, que trata da revisão dos conceitos fundamentais, são apresentados os principais conceitos que são utilizados durante o trabalho: o processo unificado, a programação extrema, o desenvolvimento e a modelagem dirigidos por testes, o modelo baseado no teste extremo, a *Unified Modeling Language* (UML), casos de teste e alguns tipos de teste. Também são apresentados alguns trabalhos desenvolvidos em relação a esses assuntos e que são úteis para o desenvolvimento deste trabalho.

No Capítulo 3 – Especificação Dirigida por Casos de Teste (EDCT), descreve-se o funcionamento da técnica proposta, e cada etapa é detalhada para a criação do artefato que contém os requisitos do sistema em forma de casos de teste. Finalmente, avalia-se a adoção dessa técnica em relação aos outros modelos de desenvolvimento de sistemas.

Já no Capítulo 4 – Exemplos de Aplicação e Validação da Técnica EDCT, são apresentados dois casos de aplicação para validar a técnica EDCT, sendo um caso com o Jogo da Forca, e o outro, a contratação do produto Termo de Moeda que é negociado no mercado financeiro brasileiro. Apresenta-se o modelo do artefato EDCT para os casos de aplicação, o qual contém os dados de entrada, os resultados esperados e as condições necessárias para a execução dos casos de teste. Por último, realiza-se uma pesquisa com desenvolvedores e com analistas de sistemas e analisam-se os resultados dessa pesquisa.

Por último, no Capítulo 5 – Conclusão, em que é apresentado um resumo deste trabalho, ressaltando-se os objetivos alcançados, as dificuldades para o seu desenvolvimento e as contribuições para a área de Teste. Também são apresentadas algumas sugestões para futuras pesquisas relacionadas ao tema – especificação dirigida por casos de teste.

CAPÍTULO 2
REVISÃO DOS CONCEITOS FUNDAMENTAIS

2.1 Introdução

Este capítulo resume os principais conceitos que servem de alicerce para o desenvolvimento deste trabalho. Dentre os conceitos abordados, cita-se: o Processo Unificado (UP) (JACOBSON, 1999) e a Programação Extrema (XP) (BECK, 1999), com destaque especial para a prática de teste presente no XP. Outros modelos também estudados neste trabalho, porém ainda pouco utilizados no mercado de desenvolvimento de sistemas, são: a Modelagem dirigida por Teste (TDM) (ZHANG, 2004) e o modelo de teste extremo (MXT) (HOWDEN, 2002). Também se apresenta o conceito de casos de uso relacionados à Linguagem de Modelagem Unificada (UML) e, com relação à atividade de teste, são detalhados os conceitos de casos de teste e tipos de teste.

Como primeiro item deste capítulo, aborda-se a UML. Essa linguagem utiliza o conceito de casos de uso para descrever os requisitos do sistema (JACOBSON, 1999). Ainda em relação a UML, descreve-se resumidamente o conceito dos diagramas de seqüência, de atividade e de estado que são mencionados nos modelos TDD, TDM e MXT. Os demais conceitos da UML não são abordados por estarem fora do escopo deste trabalho.

O Processo Unificado é um processo de engenharia de *software* que utiliza as melhores práticas do desenvolvimento de *software* de forma a ser apropriado para um grande número de projetos de *software* e de organizações de desenvolvimento. Faz uso de técnicas de orientação a objetos e da UML como principal notação dos vários modelos que são construídos durante o desenvolvimento. O UP também é um processo flexível que permite às empresas de *software* o adaptar às suas necessidades específicas (JACOBSON, 1999). O UP tem como objetivo ser o processo para qualquer tipo de projeto de *software* e, por isso, é unificado. Assim qualquer melhoria no processo de desenvolvimento de *software* deve partir desse processo.

A Programação Extrema (XP) é um modelo de desenvolvimento de *software* chamado de ágil porque é baseado na simplicidade e na comunicação informal. Visa diminuir o esforço do desenvolvimento do *software* e da documentação (BECK, 1999). Dentre as práticas do XP,

destaca-se o Desenvolvimento Dirigido por Teste (também conhecido como *Test-first Design*, *Test-first Programming* ou *Test-driven Design* (GEORGE, 2003)), uma técnica que ganhou notoriedade com o XP. A idéia principal do TDD é programar casos de teste e implementar apenas o código necessário para esse teste passar, com isto o teste orienta o desenvolvimento do *software* (BECK, 2003).

Merece destaque também a Modelagem Dirigida por Teste (TDM), um processo de desenvolvimento de *software* baseado no *Model-driven Development* (MDD), em que os casos de teste são gerados a partir dos diagramas de seqüência de mensagens (ZHANG, 2004).

Os modelos TDD e TDM são dirigidos por teste e ambos podem ser usados por desenvolvedores que já utilizam o XP, por isso faz-se necessário introduzir os principais conceitos do XP para, em seguida, descrever o funcionamento do TDD e do TDM.

Encerra-se, este capítulo, com referências ao modelo baseado no teste extremo (MXT), que também utiliza conceitos do XP. Pode-se citar que um dos objetivos desse modelo é ser utilizado como especificação de um programa (HOWDEN, 2002).

Embora os modelos TDM e MXT sejam pouco praticados no mercado de desenvolvimento de *software* e não tenham a mesma notoriedade que o TDD, faz-se necessário descrever seu funcionamento, pois ambos reutilizam algum artefato para criar os casos de teste.

A partir da análise dos citados modelos, em especial UP e XP, a proposta deste trabalho é apresentar uma técnica de especificação de requisitos dirigida por testes, por esta razão é preciso compreender o funcionamento desses modelos, bem como suas restrições e similaridades.

2.2 UML: Unified Modeling Language

A *Unified Modeling Language* (UML) é uma linguagem padrão de modelagem de *software* – uma linguagem para visualização, especificação, construção e documentação dos artefatos de um *software*. É considerada uma das linguagens mais expressivas para

modelagem de sistemas. Por meio de seus diagramas, é possível representar sistemas de *softwares* sob diversas perspectivas de visualização (RUMBAUGH, 1998).

Basicamente, a UML permite que os desenvolvedores visualizem o produto do seu trabalho em diagramas e modelos padronizados, simplificando a comunicação entre os membros de um projeto e seus clientes, por apresentar um vocabulário de fácil entendimento (RUMBAUGH, 1998). Os diversos diagramas da UML fornecem múltiplas visões do sistema a ser modelado, analisando-o e modelando-o sob diversos aspectos, permitindo que cada diagrama complemente os outros. Cada diagrama da UML analisa o sistema, ou parte dele, sob um determinado ponto de vista.

2.2.1 Diagramas da UML

Apresenta-se uma breve descrição do diagrama de seqüência, do diagrama de atividade e do diagrama de estado, por serem utilizados nos modelos de desenvolvimento de *software* TDD (BECK, 2003), TDM (ZHANG, 2004) e MXT (HOWDEN, 2002). Também se descreve o diagrama de casos de uso por este ser utilizado nos exemplos de aplicação da proposta deste trabalho. Os outros diagramas da UML estão fora do escopo deste trabalho, entretanto podem ser consultados em RUMBAUGH (1998) .

O diagrama de seqüência representa a seqüência de mensagens passadas entre objetos. Descreve a maneira como os grupos de objetos colaboram em algum comportamento ao longo do tempo e pode ser construído a partir do diagrama de Casos de Uso. Ele registra o comportamento de um único caso de uso e exibe os objetos e mensagens passadas entre esses objetos no caso de uso (RUMBAUGH, 1998). Pode-se analisar a seqüência de eventos que é executada nos cenários principais dos casos de uso, para identificar as operações necessárias para completar cada processo (DEBONI, 2003). Na Figura 1, é ilustrado um exemplo do diagrama de seqüência para o Jogo da Forca.

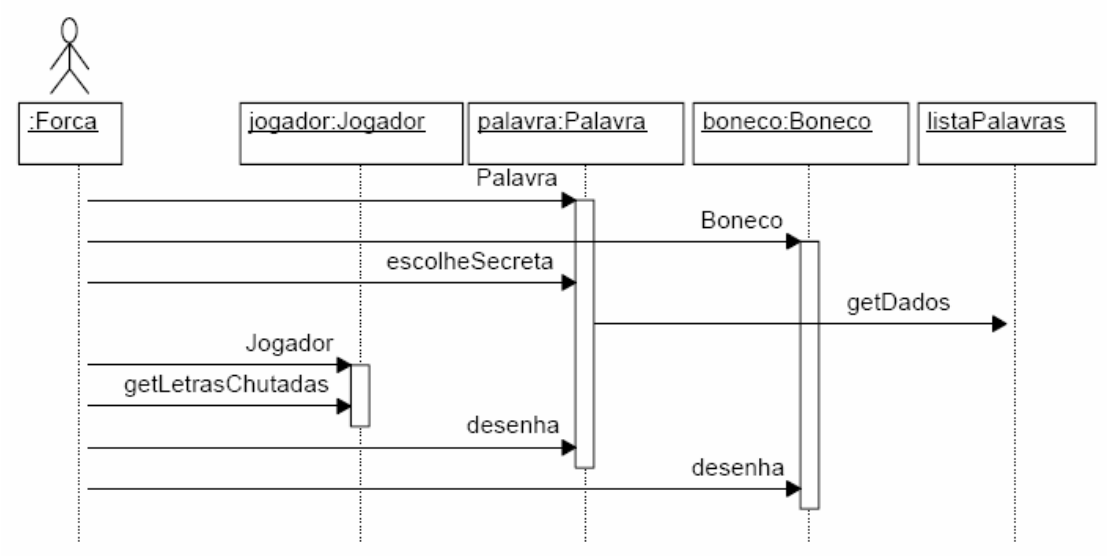

Figura 1 – Exemplo de Diagrama de Seqüência.
Fonte: Deboni (2003)

O diagrama de atividade representa os fluxos conduzidos por processamentos. É essencialmente um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra. Comumente isso envolve a modelagem das etapas seqüências em um processo computacional (RUMBAUGH, 1998). Na Figura 2, encontra-se ilustrado um exemplo do diagrama de atividade para o Jogo da Forca.

O diagrama de estado mostra os possíveis estados de um objeto e as transações responsáveis por suas mudanças. Um objeto apresenta um comportamento e um estado, e o estado depende da atividade na qual ele está processando (RUMBAUGH, 1998). Na Figura 3, está ilustrado um exemplo do diagrama de estado para o Jogo da Forca.

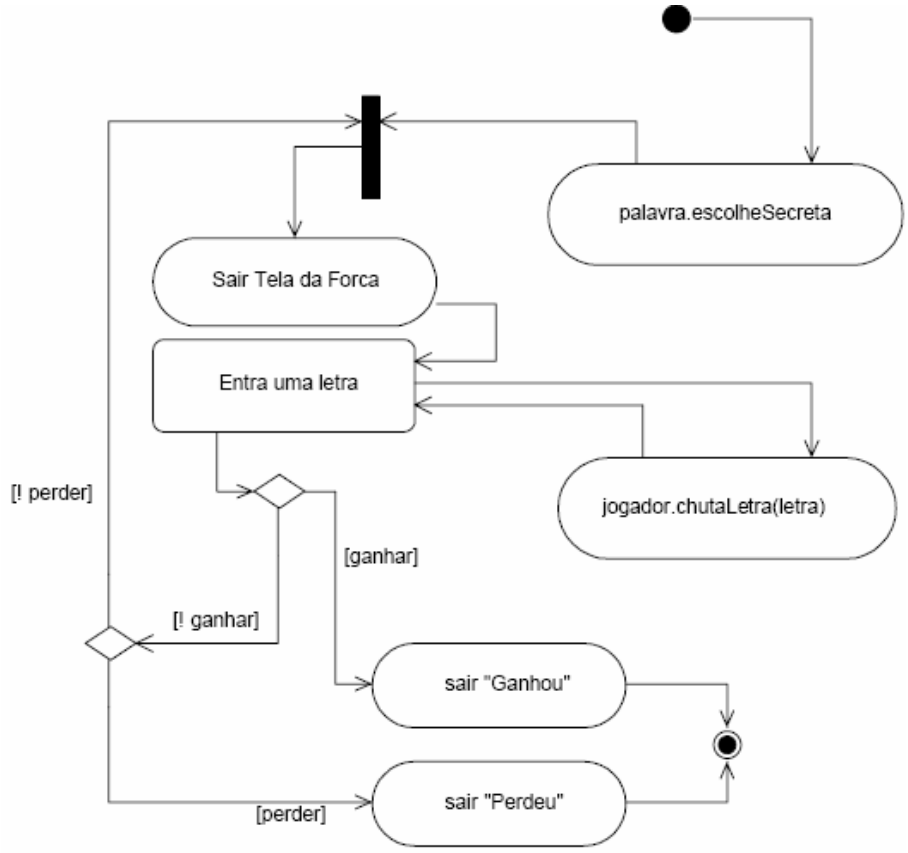

Figura 2 – Exemplo de Diagrama de Atividade.
Fonte: Deboni (2003)

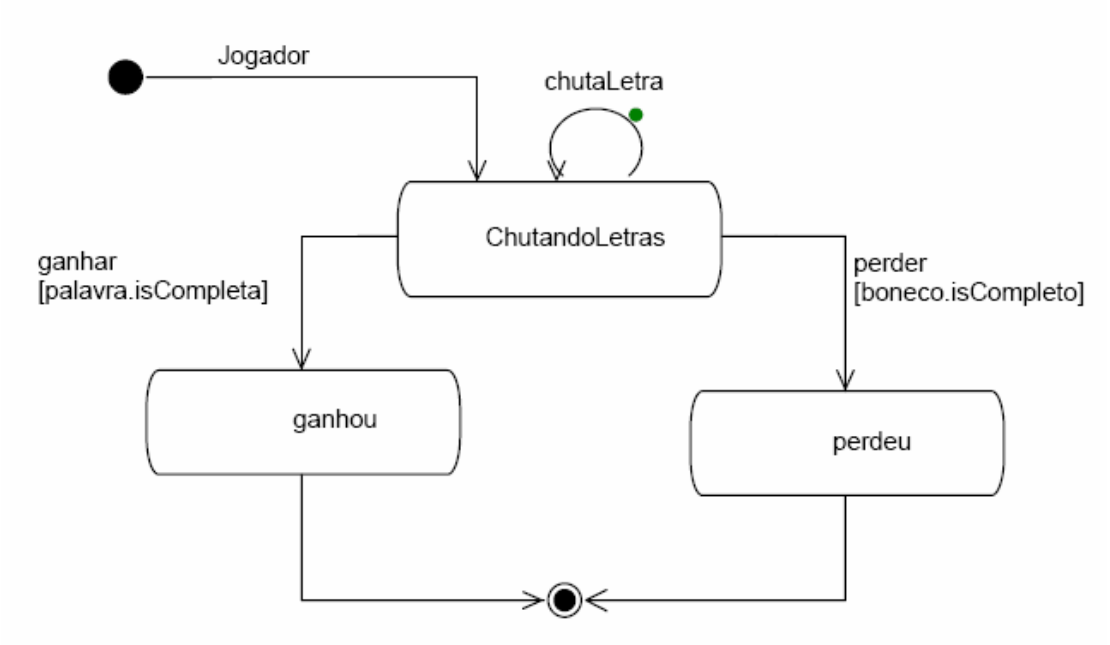

Figura 3 – Exemplo de Diagrama de Estado.
Fonte: Deboni (2003)

O diagrama de casos de uso descreve a funcionalidade proposta para o novo sistema, exibe a associação entre os atores e os casos de uso (RUMBAUGH, 1998) e serve para nortear a criação dos demais diagramas da UML (DEBONI, 2003). Na Figura 4, é apresentado um exemplo do diagrama de casos de uso para o Jogo da Forca.

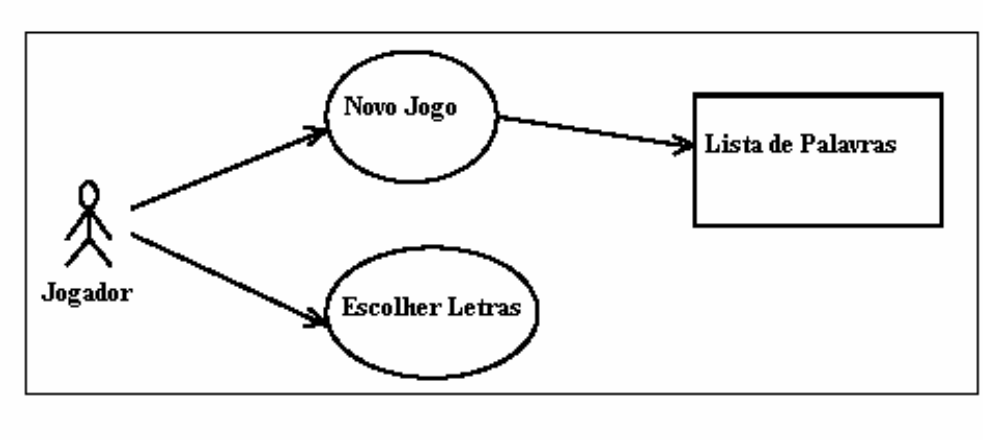

Figura 4 – Exemplo de Diagrama de Casos de Uso.
Fonte: Deboni (2003)

2.2.2 Casos de uso

Ivar Jacobson propôs em sua metodologia de desenvolvimento de sistemas orientados a objetos, OOSE, a utilização de casos de uso (JACOBSON, 1992). Posteriormente, os casos de uso foram incorporados à UML, o que torna seu uso uma prática freqüente na identificação de requisitos de sistemas, por uma larga comunidade de desenvolvedores (RUMBAUGH, 1998).

Os casos de uso analisam o sistema sob o ponto de visto do usuário que, na UML, é representado pelo ator. Um ator representa um conjunto coerente de papéis que os usuários de casos de uso desempenham quando interagem com esses casos de uso. Tipicamente, um ator representa um papel que um ser humano, um dispositivo de *hardware* ou até outro sistema, desempenha com o sistema (RUMBAUGH, 1998).

Pode-se dizer que os casos de uso não são apenas uma ferramenta para especificar os requisitos de um sistema; eles também orientam o projeto, a implementação e o teste, ou seja, eles guiam o processo de desenvolvimento (RUMBAUGH, 1998). Todos os casos de uso juntos compõem o modelo de casos de uso, o qual descreve todas as funcionalidades de um sistema.

Existem várias razões para explicar porque os casos de uso se tornaram populares. Dentre elas, pode-se destacar o fato de os casos de uso proporem um significado sistemático e intuitivo na captura dos requisitos funcionais com ênfase no valor agregado ao usuário e de orientarem todo o processo de desenvolvimento, uma vez que muitas atividades como análise, projeto e teste, são realizadas a partir dos casos de uso. As atividades de projeto e teste também podem ser planejadas e coordenadas em termos de casos de uso. Por último, os casos de uso são fáceis de implementar e de usar, ou seja são intuitivos. Os clientes não precisam aprender uma notação complexa, pois a linguagem utilizada para descrevê-los é simples, o que facilita no momento de propor mudanças (JACOBSON, 1999).

Os analistas de teste validam os componentes do *software* para assegurar que os casos de uso sejam implementados corretamente (JACOBSON, 1999). Ao final, os casos de uso são a fonte para que os analistas de teste construam os casos de teste, visto que cada caso de uso resulta em um conjunto de casos de teste (JACOBSON, 2003). Os fluxos básicos e alternativos representam a parte mais importante de um caso de uso e devem ser considerados na geração dos casos de teste (HEUMANN, 2001).

Os analistas de sistemas também podem levar em consideração os casos de uso provenientes de mau uso de uma funcionalidade do sistema para documentar e analisar os cenários. Um caso de uso proveniente de mau uso é simplesmente um caso de uso do ponto de vista de um ator hostil que engloba os fluxos alternativos e as exceções. A vantagem de analisar cenários com esses casos de uso é o aumento da segurança do sistema, mitigando as ameaças (ALEXANDER, 2003).

Os casos de uso provenientes de mau uso também podem ser usados para solucionar problemas de usabilidade, quando, por exemplo, um usuário iniciante se torna uma ameaça ao sistema. Ou ainda, tais casos de uso podem claramente identificar exceções e falhas, em que cada uma deve ser verificada e testada. O hábito de pensar em cenários provenientes da má utilização de uma funcionalidade é uma habilidade essencial para os engenheiros de teste (ALEXANDER, 2003).

Quando empregados em conjunto, os casos de uso provenientes ou não de mau uso, auxiliam o projeto do sistema, tanto na parte de definição de requisitos funcionais e não

funcionais, bem como na geração de casos de teste (ALEXANDER, 2003), visto que eles são a base para descrever os requisitos funcionais e não funcionais. Os requisitos não funcionais representam as qualidades que um produto deve ter, como ser atrativo, usável, rápido, confiável e seguro. Os requisitos não funcionais são usados, por exemplo, para especificar tempos de resposta ou para limitar a precisão de um cálculo (ROBERTSON, 2006).

Os casos de uso são importantes artefatos para especificar as funcionalidades de um sistema e também são muito úteis para derivar o conjunto de casos de teste (COCKBURN, 2001). Sendo assim, o perfeito entendimento deles representam o ponto de partida para o desenvolvimento deste trabalho.

2.3 Teste de Software

Nesta seção, são resumidos os conceitos relacionados à atividade de teste relevantes para o desenvolvimento deste trabalho, são eles: casos de teste e tipos de teste. Dentre os tipos de testes, podem-se citar: teste de aceitação, teste de sistema, teste funcional, teste de integração, teste de unidade, teste de instalação e teste de regressão.

2.3.1 Casos de teste

O caso de teste é um artefato produzido no processo de desenvolvimento do sistema com a finalidade de testar outros artefatos do *software*. Os casos de teste podem ser elaborados para identificar defeitos nas estruturas internas do *software*, por meio de situações que exercitem adequadamente todas as estruturas utilizadas na codificação; ou ainda, testar a especificação para garantir que os requisitos do *software* sejam plenamente atendidos.

De acordo com Myers (2004), um bom caso de teste é aquele que tem alta probabilidade de encontrar erro, e um caso de teste bem sucedido é aquele que encontra um erro ainda desconhecido.

Diante do fato de que, geralmente, é difícil testar um programa completamente, o conjunto de casos de teste deve ser um conjunto suficientemente representativo e com boa probabilidade de encontrar os defeitos que o *software* apresenta (MYERS, 2004). Uma

seleção imprópria dos casos de teste pode resultar em: testar exageradamente a aplicação, testá-la insuficientemente ou ainda testar os itens errados.

Heumann (2001) recomenda que os desenvolvedores de sistema comecem a criar casos de teste assim que os casos de uso estiverem disponíveis, antes mesmo de escrever qualquer linha de código. Com isso, as equipes de teste podem começar a verificar o sistema mais cedo no ciclo de vida do desenvolvimento, o que lhes permite identificar e reparar defeitos que seriam mais caros se fossem descobertos mais tarde. Além do mais, assegura que o sistema funciona adequadamente. Esta recomendação de escrever casos de teste antes de escrever código representa um dos pontos de partida para o desenvolvimento deste trabalho, cuja proposta é antecipar a atividade de teste.

De acordo com o padrão ANSI/IEEE 829, que define o conteúdo e o formato dos documentos do processo de teste, um caso de teste deve ter a seguinte estrutura (IEEE, 1983):

- Identificador: número ou nome único que identifica o caso de teste;
- Item testado: indica o item a ser testado, por exemplo qual o módulo, sistema ou unidade;
- Especificação de entrada: lista todos os valores de entrada e condições necessárias para a execução de um caso de teste;
- Especificação de saída: descreve qual o resultado esperado com a execução do caso de teste;
- Necessidades de ambiente: identifica as necessidades de ambiente, como *hardware*, *software* ou ferramentas de teste para a execução do caso de teste;
- Requisitos especiais: identifica alguma condição especial que deve ser atendida para executar/realizar o teste;
- Dependências com outros casos de teste: identifica se um caso de teste é dependente de outro, ou se este caso de teste pode afetar outro.

2.3.2 Tipos de teste

Existem vários tipos de teste e cada um abrange uma área específica do *software* e do seu processo de desenvolvimento. Um teste de *software* durante o desenvolvimento tem

muitas variações, que vão desde um simples teste de unidade até um teste completo da aplicação (MYERS, 2004).

Baseado em Myers (2004), segue uma descrição resumida de alguns tipos de teste. Os demais tipos de teste de *software* podem ser consultados em Myers (2004) e não são abordados por estarem fora do escopo deste trabalho.

O objetivo do teste de aceitação é comparar o programa com os requisitos do sistema e com as necessidades dos usuários finais (MYERS, 2004). Este tipo de teste é geralmente realizado por um grupo restrito de usuários finais do sistema, que testam se o sistema está de acordo com os requisitos solicitados, ou seja, esse teste é aplicado em todos os requisitos do sistema (PATTON, 2001). O resultado dele determina se o usuário final (cliente) aceita ou não o sistema.

O teste funcional, também conhecido como teste de caixa preta, tem como objetivo validar as funcionalidades do sistema, sem se preocupar com a estrutura do código fonte. O principal objetivo é expor os erros do sistema em relação à especificação, em vez de demonstrar apenas que o programa é igual à especificação (MYERS, 2004).

Para realizar o teste funcional, a especificação do sistema é analisada para derivar o conjunto de casos de teste. Esse conjunto deve abranger, por exemplo: as classes de equivalência (dados equivalentes), as condições limites, os dados de entrada inválidos ou inesperados e os limites de equivalência que marcam pontos ou regiões de transição entre classes de equivalência (MYERS, 2004).

As classes de equivalência são identificadas a partir das condições de entrada, que geralmente são uma sentença ou frase na especificação, e tais classes são divididas em dois grupos ou mais grupos, que podem ser classes de equivalência com dados válidos e classes de equivalência com dados inválidos (MYERS, 2004). Como é impossível testar todas as possibilidades de teste em um sistema, deve-se fazer uso da classe de equivalência.

Outro tipo de teste é o teste de unidade, cujo objetivo é testar individualmente cada componente do *software* ou coleção de componentes, ou seja, focaliza a verificação na menor unidade de projeto do *software*, testando a lógica do programa (MYERS, 2004).

Uma das motivações para realizar testes de unidade é que tal conduta facilita a atividade de depurar, uma vez que, quando um erro é encontrado, sabe-se em que módulo específico esse erro está (MYERS, 2004).

Por último, descreve-se o teste de regressão, que tem como objetivo executar novamente algum subconjunto de testes que já foram conduzidos para garantir que as modificações não propagaram efeitos colaterais indesejáveis, pois cada vez que um novo módulo é adicionado como parte do teste de integração, o *software* se modifica. Esse teste deve ser aplicado em todos os programas do sistema que sofrem alteração (MYERS, 2004).

Os tipos de teste se diferenciam em função dos itens do sistema a serem testados. Enquanto um teste funcional testa as funcionalidades do sistema, o teste de unidade testa o código fonte, com o objetivo de validar a estrutura lógica que compõe o sistema. Os tipos de teste se diferenciam também quanto ao artefato testável. Assim, alguns testes podem ser integrados com casos de uso, como: teste de aceitação, teste funcional e teste de unidade, por testarem as funcionalidades do sistema descritas em um caso de uso. O teste de regressão também pode ser integrado aos casos de uso, pois avalia os efeitos colaterais das modificações em um programa.

2.4 Processo Unificado

O processo unificado (UP) é um processo de desenvolvimento de *software* composto por um conjunto de atividades necessárias para transformar os requisitos do usuário em um sistema. Na Figura 5, encontra-se ilustrado esse processo de desenvolvimento de *software* (JACOBSON, 1999).

O UP é mais do que um simples processo, ele é uma arquitetura genérica que pode ser aplicada a uma grande classe de sistemas, de diferentes áreas de aplicação, para diferentes tipos de empresas e para diferentes tamanhos de projeto (JACOBSON, 1999). O UP utiliza os conceitos da UML na construção dos artefatos do sistema e é um processo de desenvolvimento de *software* iterativo, centrado na arquitetura e guiado por casos de uso. Abrange o ciclo de vida do projeto e orienta a equipe nas atividades de gerência de projeto, bem como nas atividades técnicas. O objetivo do UP é assegurar um desenvolvimento de

software de alta qualidade que atenda às necessidades do usuário final, de acordo com cronograma e orçamento pré-estabelecidos (ERIC, 2004).

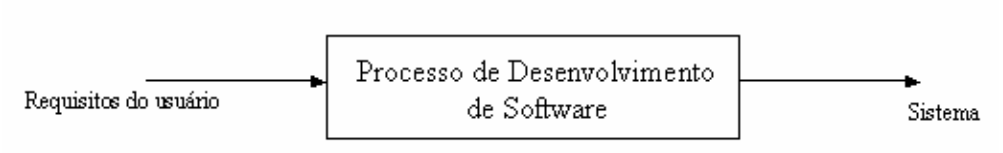

Figura 5 – Processo de Desenvolvimento de Software.
Fonte: Adaptada de Jacobson (1999)

O processo do UP pode ser descrito em duas dimensões, como ilustrado na Figura 6, em que o eixo horizontal representa o tempo, mostra o aspecto dinâmico do processo, e é expresso em termos de ciclos, fases, iterações e marcos. O eixo vertical, que representa o aspecto estático do processo, é descrito em termos de atividades, artefatos, papéis e fluxos de trabalho (KRUCHTEN, 2003).

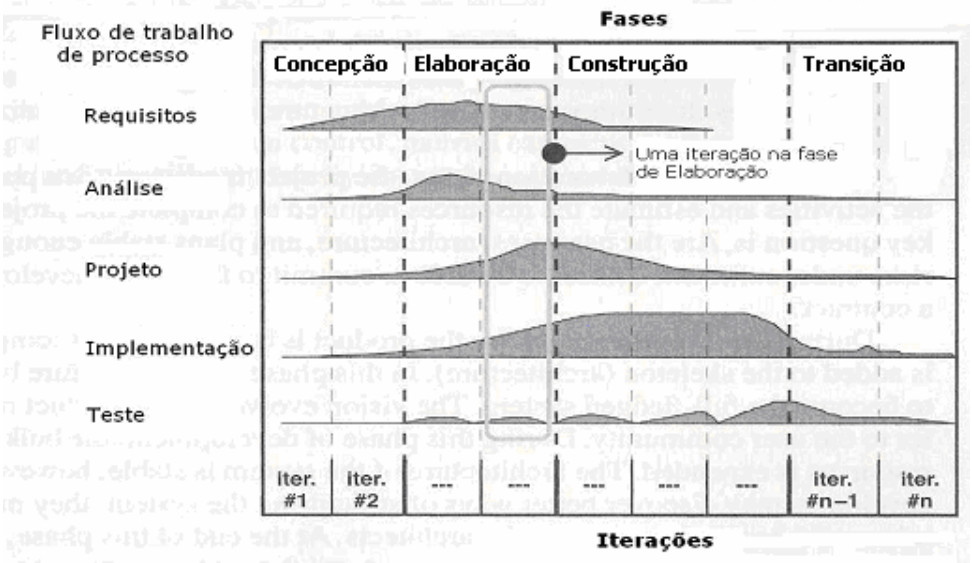

Figura 6 – Dimensões do UP.
Fonte: Adaptada de Jacobson (1999)

Existem cinco fluxos de trabalho no UP ao longo das cinco fases. A seguir é apresentada uma descrição resumida de cada fluxo de trabalho, proposta por Jacobson (1999).

O fluxo de trabalho Requisitos identifica o que o sistema deve e o que não deve fazer e quais requisitos priorizar. Os requisitos são determinados em reuniões com os clientes, refinando-os após cada reunião, de forma que a equipe de requisitos tenha a mesma visão do

sistema que os clientes têm. Nesse processo são descobertos e documentados os atores e os casos de uso.

O fluxo Análise tem como objetivo refinar e estruturar os requisitos capturados em seu fluxo em casos de uso, com o propósito de alcançar um entendimento mais preciso sobre eles. Também se gera o modelo de casos de uso.

Já o fluxo de trabalho Projeto utiliza os casos de uso para identificar o conjunto de objetos que são subseqüentemente refinados em classes, subsistemas e pacotes, e para definir a arquitetura do sistema.

No fluxo Implementação, define-se o código em termo de implementação de subsistemas que são organizados em camadas. Também são implementados as classes e os objetos em termo de componentes, testados os componentes desenvolvidos como unidades e integrados em um executável, por um implementador individual ou uma equipe.

Por último, tem-se o fluxo de trabalho Teste, cujo propósito principal é avaliar a qualidade do produto. Para tanto, esse fluxo de trabalho envolve as seguintes práticas: encontrar e documentar as falhas do *software*, verificar a iteração e a integração dos componentes, verificar se todos os requisitos foram implementados corretamente, e identificar e assegurar que todas as falhas encontradas são tratadas antes de o *software* ser disponibilizado para o usuário final. O teste não é uma atividade isolada e deve ocorrer durante todo o ciclo de vida do projeto. Ele depende da fase em que se encontra o projeto, por exemplo: na fase de concepção, tem-se o plano de teste inicial e o teste do protótipo, na fase de elaboração, há o teste inicial da arquitetura, na fase de construção, encontra-se o teste significativo para cada construção e na fase de transição, acontecem o teste de regressão e o teste dos itens corrigidos (ERIC, 2004).

2.4.1 Estrutura do UP

Cada fase do UP pode ser dividida em iterações. Uma iteração é um ciclo completo de desenvolvimento que resulta em uma versão executável do produto (PRIESTLEY, 2000). De acordo com Kruchten (KRUCHTEN, 2003) e conforme ilustrado na Figura 7, o UP é

representado usando-se os principais elementos de modelagem: papéis, atividades, artefatos e fluxos de trabalho.

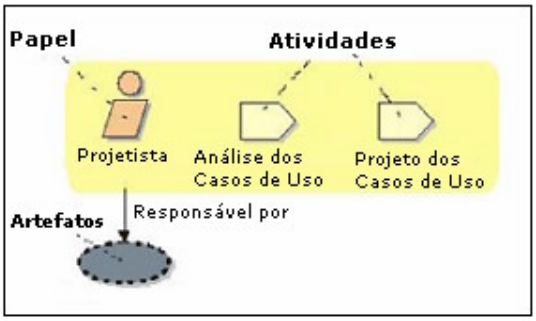

Figura 7 – Principais elementos de modelagem do UP.
Fonte: Adaptado de Kruchten (2003)

A seguir descrevem-se, resumidamente, os principais elementos de modelagem do UP, que são:

- **Papel** (ou perfil): define o comportamento e as responsabilidades de um indivíduo ou de um grupo de indivíduos que trabalham juntos como um time. Um indivíduo pode representar diferentes papéis o que é uma importante distinção, porque é natural pensar em um papel como um indivíduo ou time, mas, no processo unificado, ele pode exercer vários papéis (KRUCHTEN, 2003);

- **Atividade:** é uma unidade de trabalho que um indivíduo executa quando está exercendo um determinado papel e produz um resultado importante para o contexto do projeto. Cada atividade pode ser dividida em passos. Alguns exemplos de atividades são: planejar uma iteração - realizada pelo papel Gerente de Projeto, ou encontrar casos de uso e atores - realizada pelo papel Analista de Sistemas (KRUCHTEN, 2003);

- **Artefato:** é um pedaço de informação que é produzido, modificado ou utilizado em um processo. Os artefatos são produtos de um projeto e são utilizados como entrada de atividades, mas que também são produzidos como saída. Os artefatos podem ter várias formas, como: um modelo de caso de uso ou um modelo de projeto; um elemento de um modelo, como uma classe, um caso de uso e um subsistema; um documento, como

um caso de negócio, um glossário e a visão; o código fonte e os executáveis (KRUCHTEN, 2003);

- **Fluxo de Trabalho**: a enumeração de atividades, papéis e artefatos não constituem um processo. É necessário saber a seqüência do desenvolvimento das atividades para que possam ser produzidos os artefatos de valor para o projeto. Um fluxo de trabalho é uma seqüência de atividades que são executadas para a produção de um resultado valioso para o projeto (KRUCHTEN, 2003).

2.4.2 Fases e iterações do UP

O ciclo de vida do *software* é dividido em fases, cada fase trabalha na geração de um artefato e cada uma apresenta um propósito específico. O UP divide o ciclo de desenvolvimento em quatro fases consecutivas. Cada fase é composta por uma ou mais iterações, contendo o levantamento de requisitos, a análise, o projeto, a implementação e o teste. Tal ciclo resulta em uma versão executável do produto, que cresce, aos poucos, de uma iteração a outra, até compor o sistema final (JACOBSON, 1999; KRUCHTEN, 2003; ERIC, 2004).

Cada fase é concluída com um marco bem definido, que é um ponto no tempo em que certas decisões devem ser tomadas, motivo pelo qual os objetivos devem ter êxito. Na Figura 8, estão ilustradas as fases do UP e seus principais marcos. O foco deste trabalho é em relação às fases de concepção, elaboração e construção. O detalhamento da fase de transição está fora do escopo deste trabalho.

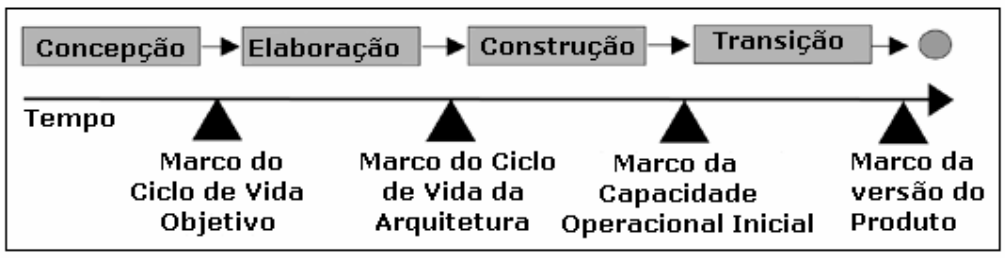

Figura 8 – Fases e principais marcos do UP.
Fonte: Adaptado de Kruchten (2003)

2.4.2.1 Fase de concepção

Nesta fase, definem-se os casos de negócio e delimita-se o escopo do projeto. Para executá-la são identificadas as entidades externas com as quais o sistema interage (atores). Tal procedimento envolve conhecer todos os casos de uso e descrever os mais significativos. Os casos de negócio incluem critérios de sucesso, avaliação de risco, estimativa dos recursos necessários e planejamento da fase mostrando os principais marcos. Conforme ilustrado na Figura 9, observa-se que a maior parte do trabalho realizado na fase de concepção ocorre durante o fluxo de trabalho de requisitos, já que o objetivo principal dela é a definição do escopo do sistema. O tamanho dos retângulos ilustra apenas quais fluxos de trabalho requerem mais e menos atenção nessa fase.

Como resultados da fase de concepção, são produzidos os artefatos: a visão geral dos requisitos do projeto, as características e as principais restrições dele, o modelo de caso de uso inicial, o glossário inicial do projeto; os casos de negócio iniciam, o que inclui o contexto do negócio, os critérios de sucesso e a previsão financeira, a avaliação de risco inicial, o plano do projeto, mostrando as fases e as iterações, o modelo de negócio (se necessário) e um ou vários protótipos (KRUCHTEN, 2003).

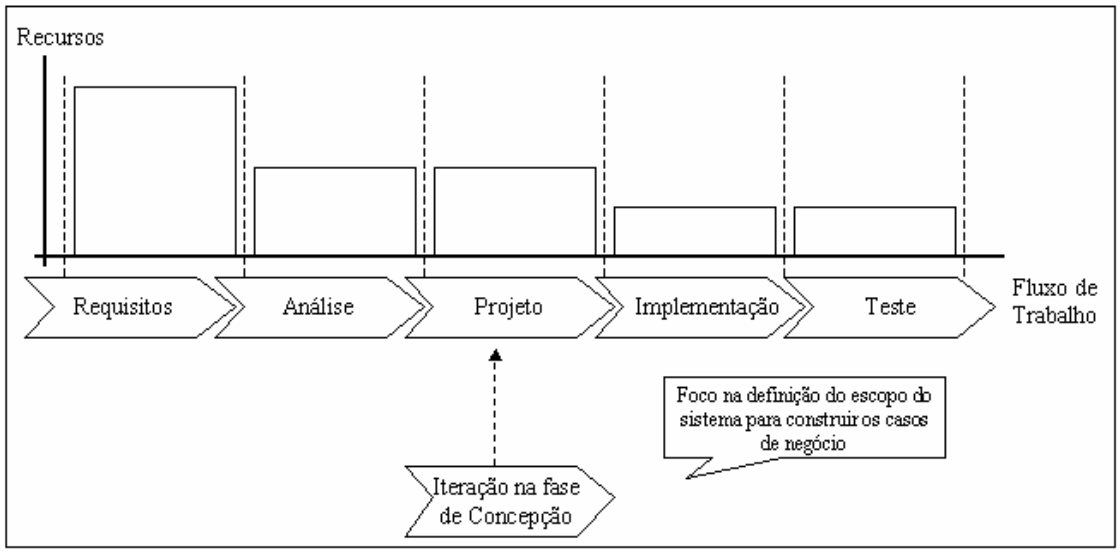

Figura 9 – Recursos x Fluxo de Trabalho da Fase de Concepção.
Fonte: Adaptada de Jacobson (1999)

Conforme ilustrado na Figura 8, ao final da fase de concepção tem-se o primeiro marco do projeto e os objetivos do ciclo de vida (KRUCHTEN, 2003).

2.4.2.2 Fase de elaboração

O propósito desta fase é analisar o domínio do problema, determinar o alicerce da arquitetura, desenvolver o plano do projeto e eliminar os altos riscos dele, sempre com ênfase na arquitetura do sistema. Tal arquitetura baseia-se nos artefatos produzidos na fase de concepção e, conforme a Figura 10, a maior parte do trabalho realizado na fase de elaboração ocorre na captura de requisitos, de análise e de projeto (JACOBSON, 1999).

As decisões de arquitetura devem ser tomadas a partir da compreensão de todo o sistema, considerando o escopo, as principais funcionalidades e os requisitos não funcionais, como os requisitos de desempenho (KRUCHTEN, 2003).

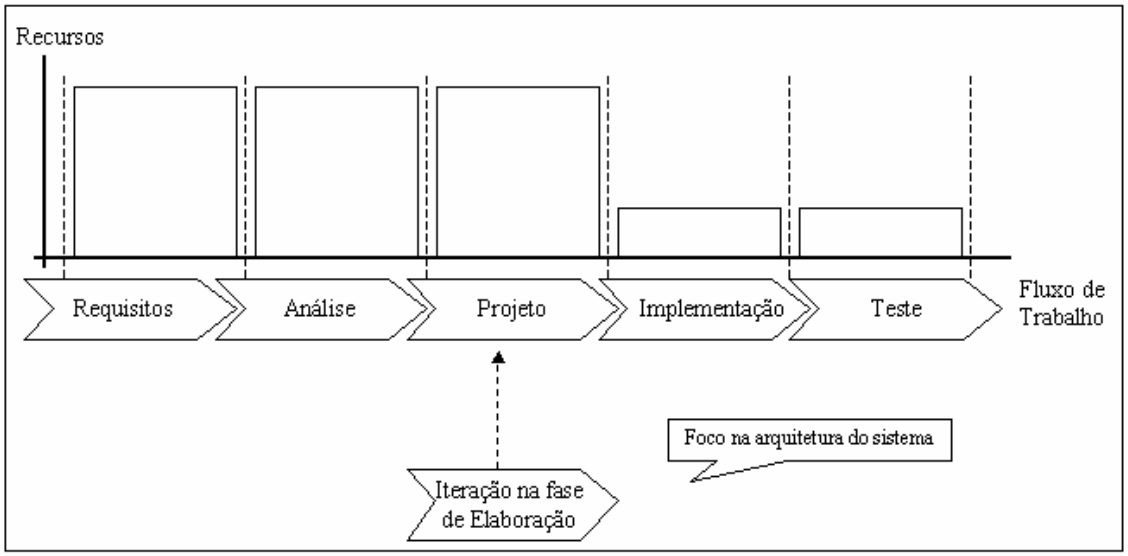

Figura 10 – Recursos x Fluxo de Trabalho da Fase de Elaboração.
Fonte: Adaptada de Jacobson (1999)

Na fase de elaboração, um protótipo executável da arquitetura é construído com uma ou mais iterações, dependendo do escopo, do tamanho, do risco e da inovação do projeto. Esse esforço deve, pelo menos, chamar a atenção para os casos de uso mais críticos identificados na fase de concepção, o qual tipicamente expõe os maiores riscos técnicos do projeto.

Como resultados da fase de elaboração, podem ser produzidos os seguintes artefatos:

- Modelo de casos de uso em que todos os casos de uso e os atores devem ser identificados;
- Requisitos adicionais que capturam os requisitos não funcionais e qualquer requisito que não está associado com um caso de uso específico;
- Descrição da arquitetura de *software*;
- Protótipo executável;
- Revisão da lista de riscos e revisão dos casos de negócios;
- Plano de desenvolvimento para todo o projeto, incluindo um plano de projeto detalhado que mostre as iterações e critérios de evolução para cada iteração;
- Manual de usuário preliminar (opcional) (KRUCHTEN, 2003).

Ao final dessa fase, ocorre o segundo importante marco do projeto: o marco de ciclo de vida da arquitetura, conforme é mostrado na Figura 8. Nesse momento, detalham-se os objetivos do sistema e o escopo, a escolha da arquitetura e a decisão dos principais riscos (KRUCHTEN, 2003).

2.4.2.3 Fase de construção

Durante a fase de construção, os componentes e as características das aplicações são desenvolvidas e integradas em um produto, e as funcionalidades são testadas. A fase de construção é, de certo modo, o processo de manufatura em que é dada ênfase nos recursos de gerenciamento e nas operações de controle para otimizar o custo, o prazo e a qualidade. Atividades em paralelo podem acelerar significativamente a disponibilidade de versões de desenvolvimento e também podem incrementar a complexidade da gerência dos recursos e sincronizar o fluxo (JACOBSON, 1999).

O resultado da fase de construção é o produto pronto para ser disponibilizado para o usuário final. É composto, no mínimo, por: produto integrado e adequado para as plataformas, manual do usuário e descrição da versão atual (JACOBSON, 1999).

2.5 XP: Programação Extrema

Beck define o XP como uma metodologia leve para equipes de projeto de pequeno a médio porte e com grande habilidade para se adaptar às mudanças de requisitos. Os quatro valores fundamentais do XP são: comunicação, simplicidade, retorno e coragem. Essa metodologia enfatiza o trabalho em equipe e o envolvimento dos clientes durante todo o processo de desenvolvimento de *software* (BECK, 1999).

Zuser (2005) afirma que, para pequenos projetos, o XP oferece um bom caminho para produzir *software* de alta qualidade. O vigoroso envolvimento do cliente, o foco principal em testes e a tentativa para reduzir os esforços de projetos dão suporte aos objetivos do XP.

O XP se baseia em um processo iterativo rigoroso, demandando o desenvolvimento diário dos componentes, o que limita o tempo necessário para encontrar erros e força os desenvolvedores a corrigir esses erros mais rapidamente (ZUSER, 2005).

Outro conceito importante do XP são os cartões de histórias (do inglês *User Stories*). Os cartões de histórias são escritos pelos clientes e descrevem, em poucas linhas, as funcionalidades do sistema; os detalhes dessa funcionalidade só são obtidos verbalmente pelos desenvolvedores no momento de sua implementação. São utilizados para que os desenvolvedores possam dar uma estimativa de tempo de desenvolvimento e são descritos utilizando a terminologia dos clientes, sem citar terminologia de tecnologia. Os cartões de histórias também são utilizados na criação dos testes de aceitação, e um ou mais testes de aceitação automatizada devem ser criados para validar se a funcionalidade descrita no cartão de história está devidamente implementada (COHN, 2004).

De acordo com Beck (1999), na primeira edição do seu livro *Extreme Programming Explained: embrace change*, o XP utiliza doze práticas, conforme ilustrado na Figura 11.

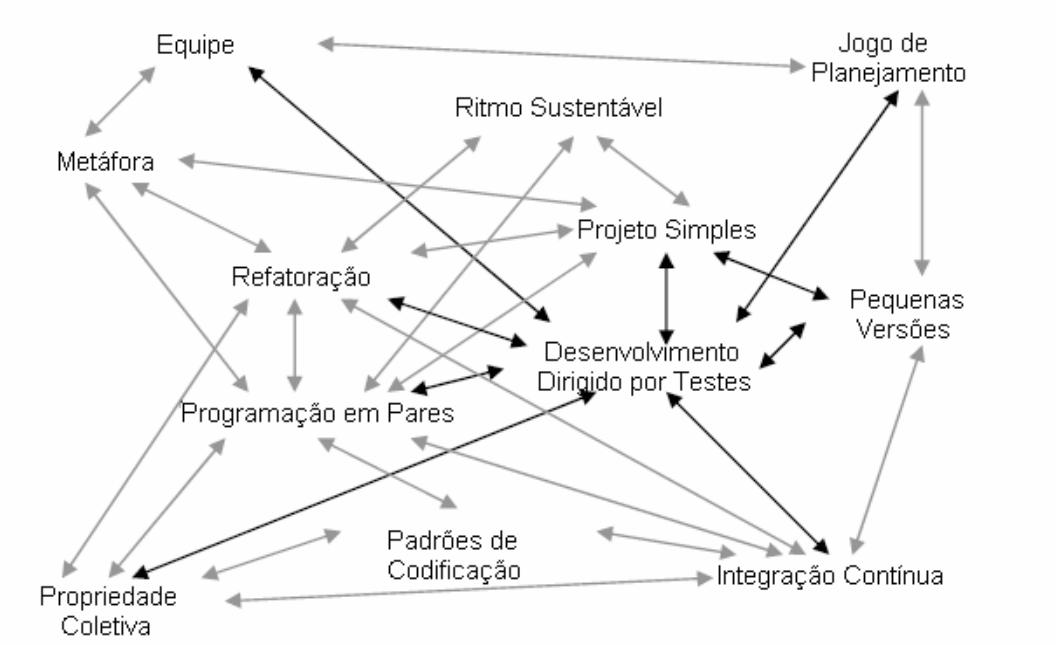

Figura 11 – Práticas do XP.
Fonte: Adaptado de Beck (1999)

A seguir, apresentam-se, resumidamente, as práticas do XP:

- Equipe: deve ser composta pelo Cliente e pelo time de desenvolvimento. O cliente é responsável por fornecer os requisitos do sistema, priorizar as atividades e acompanhar o projeto;

- Jogo de Planejamento: define a estimativa e a priorização de cada tarefa, com o propósito de proporcionar um ótimo retorno (*feedback*) para o cliente. É possível visualizar, semanalmente o progresso do projeto. No XP, inexiste o conceito “90% da atividade A está concluída”, uma atividade está ou não concluída;

- Pequenas Versões: disponibiliza a cada iteração *software* 100% funcional que pode ser utilizado pelo cliente, caso necessário. Essa prática proporciona confiabilidade para o cliente, que recebe o sistema em pequenas partes e não somente ao final do prazo de desenvolvimento;

- Metáfora: as equipes de XP mantêm uma visão compartilhada do sistema, o que proporciona uma ótima comunicação entre as pessoas da equipe. A equipe utiliza uma

padronização de nomes para garantir a compreensão de todos sobre como o sistema funciona e sobre o lugar em que se encontram e se adicionam funcionalidades;

- Integração Contínua: mantém o *software* integrado o tempo todo. O código é integrado e testado depois de algumas horas de desenvolvimento, no máximo a cada dia;

- Propriedade Coletiva: o código apresenta um único dono – a equipe, e, por isso, todos compartilham a responsabilidade das alterações. Essa prática aumenta a qualidade do código e diminui os defeitos, pois toda a equipe está atenta ao código;

- Padrões de Codificação: os padrões de codificação são definidos pela equipe. A utilização desses padrões traz como benefício o fato de o código parecer ter sido gerado por um único desenvolvedor;

- Ritmo Sustentável: prioriza a qualidade de vida da equipe e evita trabalhar com excesso de horas-extras. Se os desenvolvedores trabalham em excesso, ficam cansados, conseqüentemente há queda de produtividade. Sendo assim, o tempo de trabalho ganho com as horas extras é perdido devido à exaustão;

- Projeto Simples: o sistema deve ser projetado do modo mais simples possível, abrangendo apenas as funcionalidades solicitadas pelo cliente, sem se pensar nas funcionalidades futuras.

- Refatoração: o projeto é melhorado continuamente por meio do refinamento de código. A refatoração é uma prática para alterar um *software*, de forma que não se modifique a funcionalidade desenvolvida; o intuito é melhorar sua estrutura interna para gerar programas mais reutilizáveis, mais simples e mais eficientes (FOWLER, 1999). Com isso, tenta-se reduzir as duplicações e dar mais robustez ao código.

  Também se pode dizer que o objetivo da refatoração é reestruturar o código, fazendo pequenas mudanças para melhorá-lo, sem modificar seu comportamento semântico, o que significa que não é excluída nem acrescentada nenhuma funcionalidade durante a refatoração (AMBLER, 2007).

A existência prévia de testes é fundamental para a utilização dessa prática, pois ameniza o receio dos desenvolvedores de efetuar mudanças no código. Por isso, das práticas do XP, a refatoração é a técnica que tem mais afinidade com o desenvolvimento dirigido por testes.

A refatoração permite que os desenvolvedores respondam mais rapidamente às mudanças de requisitos do cliente e às mudanças de tecnologias (SMITH, 2001).

- Programação em pares: pares de programadores produzem código de melhor qualidade, pois todo código é revisado por pelo menos um programador, o que o torna mais livre de defeitos (MULLER 2003), não despende significativamente mais tempo que o código produzido por apenas uma pessoa e é mais prazeroso para os programadores (SMITH, 2001). Os integrantes da equipe exercem todas as funções técnicas no projeto, como: análise, projeto, teste e programação. Tipicamente, um par é dividido da seguinte forma: uma pessoa produz código ou casos de testes, enquanto a outra pessoa revisa, sugere melhorias e pensa em novas funcionalidades.

- Desenvolvimento dirigido por testes: o desenvolvimento é dirigido por testes, ou seja, escrevem-se os casos de testes antes mesmo do código. Com essa prática, a equipe produz código com 100% de cobertura dos testes. Faz-se necessário detalhar essa técnica, em virtude da necessidade de compreender seu funcionamento, pois ela é fundamental para o desenvolvimento deste trabalho .

Do ponto de visto econômico do projeto, as práticas mais importantes do XP são programação em pares e desenvolvimento dirigido por testes, pois elas podem reduzir a densidade de defeitos do código (MÜLHER, 2003).

Kent Beck propõe uma nova divisão das práticas do XP, na segunda edição do seu livro *Extreme Programming Explained: embrace change,* que muda de 12 para 24 práticas, o que não implica ter que utilizar as 24 práticas, que podem ser adaptadas para cada equipe (BECK, 2004).

De acordo com a segunda edição do XP, as práticas passam a ser divididas em: práticas primárias e práticas corolárias. As práticas primárias podem ser adotadas imediatamente, de forma segura, para melhorar o desenvolvimento do *software*. Já as práticas corolárias tendem a ser mais difíceis de adotar sem antes utilizar as práticas primárias (BECK, 2004). O desenvolvimento dirigido por teste (TDD) é uma prática primária. Os detalhes das novas práticas do XP podem ser consultados em BECK (2004).

2.6 Desenvolvimento Dirigido por Testes

O Desenvolvimento Dirigido por Testes (*Test-driven Development*) é uma prática eficaz para reduzir erros, colaborar com a documentação e conduzir o projeto e a análise de maneira ágil e simples (BECK, 2003).

O TDD utiliza o teste funcional (caixa preta) de forma semelhante a um teste de unidade, abrangendo as atividades de criação de testes, execução e depuração de erros. Ambler e Beck classificam o TDD como uma técnica de análise e projeto (AMBLER, 2003) (BECK, 2003), porém o ponto fundamental dessa técnica é o teste. O TDD é uma técnica de análise porque obriga o desenvolvedor a definir explicitamente as situações para as quais ele planejou que o código vai funcionar. E é uma técnica de projeto porque o programador acaba definindo o projeto físico da interface em virtude de o teste ser criado para verificar o funcionamento dessa interface.

Também se pode conceituar o TDD como uma disciplina de projeto e de programação em que cada nova linha de código é escrita para responder a um teste do programador, escrito no instante exatamente anterior ao da codificação (JEFFRIES, 2007).

Segundo Chaplin (2001 apud GEORGE, 2003), uma regra importante do TDD é: “Se o programador não pode escrever teste para o que ele deve codificar, então não deveria nem pensar em código” ou ainda, de acordo com Jeffries (2007) “Nunca escreva uma linha de código exceto quando necessário para que o teste atual passe”.

De acordo com Martin (2007), os praticantes de TDD devem seguir três regras: não escrever código a menos que um teste de unidade falhe; não escrever mais de um teste de

unidade que seja suficiente para falhar, e não escrever mais código do que é necessário para que o teste de unidade passe.

Seguir essas regras do TDD nem sempre faz sentido, pois, em alguns casos, é necessário escrever um teste muito longo ou escrever código extra, ou, ainda, escrever o teste depois que tiver escrito o código que faz esse teste passar. O objetivo não é a perfeita adesão às regras, mas sim diminuir o intervalo entre escrever teste e produzir código para uma questão de minutos (MARTIN, 2007).

Com o TDD, os desenvolvedores de sistema projetam um conjunto de testes enquanto constroem a aplicação, o que proporciona segurança para todo o sistema e oferece confiança razoável de que nenhuma parte do código está quebrada. Como resultado, o TDD ajuda a aliviar o receio de mudanças no código, pois se pode verificar, quase que instantaneamente, se, após as alterações, os testes continuam passando (MARTIN, 2007). Aliviados, os programadores trabalham mais relaxados e menos estressados, o que permite dar mais ênfase à qualidade (JEFFRIES, 2007).

No TDD, assim como no UP, os casos de uso são, em geral, a base para os testes e para a codificação (JACOBSON, 1999) (BECK, 2003). A partir dos casos de teste, o programador escreve o código de teste e acrescenta novo código à medida que o teste passa.

O primeiro passo do TDD é criar rapidamente um teste, contendo a estrutura mais simples possível para que o código possa ser compilado. Em seguida, executa-se o teste; se o teste falhar, modifica-se o código para que o teste funcione. A idéia é acrescentar código, aos poucos, até que o teste passe. Por último, deve-se fazer a refatoração do código para eliminar as possíveis duplicações e deixar o código o mais simples possível (AMBLER, 2003). Na Figura 12, está ilustrado o ritmo do TDD.

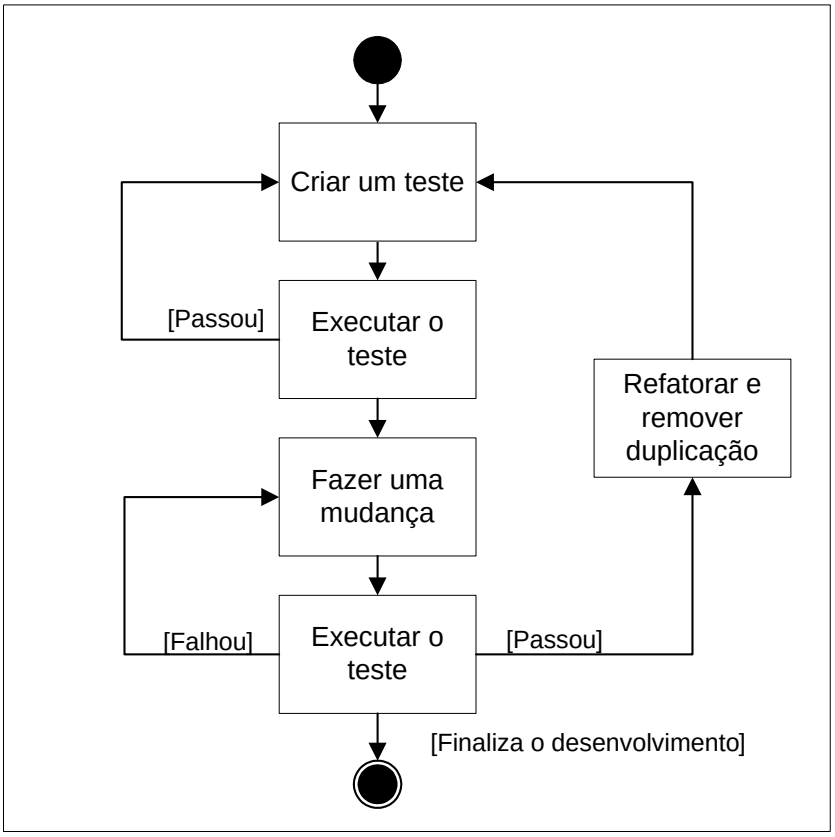

Figura 12 – Diagrama de Atividades TDD.
Fonte: Adaptado de Ambler (2003)

Ao observar os passos do TDD, verifica-se que ele preconiza a idéia de se escrever código de teste aos poucos, em pequenas etapas. Dessa maneira, o código funcional também é gerado aos poucos. Outro ponto importante na criação dos testes é focar no baixo acoplamento entre eles, a fim de se evitar que, ao falhar um teste, outros inúmeros testes falhem (BECK, 2003). Portanto, deve-se construir códigos de teste independentes, com o intuito de evitar a propagação de erros, o que é um conceito fundamental e que deve ser aplicado independente do modelo de desenvolvimento de sistemas.

O TDD possibilita limpar código confuso, sem o receio de que a limpeza danifique alguma parte do programa. A utilização do TDD também significa que se pode modificar o comportamento de uma parte do sistema sem o risco de que esta alteração gere impacto em outras partes. O TDD sugere sempre deixar o código em um estado melhor do que aquele em que foi encontrado (MARTIN, 2007).

De acordo com Beck e Myers, é impossível testar todo o código exaustivamente (BECK, 2003) (MYERS, 2004). Por isso, o importante é que os testes passem confiança ao desenvolvedor de que o código criado atende aos pontos mais importantes da funcionalidade, para que erros ocasionados em futuras refatorações possam ser facilmente identificados e corrigidos (BECK, 2003).

A agilidade no desenvolvimento do *software*, segundo essa técnica, depende da complexidade do teste e da experiência do programador. De acordo com a sua experiência, ele pode codificar rapidamente o teste para atender à determinada funcionalidade ou pode adotar uma solução não ótima para o teste passar, como, por exemplo, forçar o retorno de uma constante no programa.

A técnica TDD tem atraído a atenção de uma grande parte da comunidade de desenvolvimento de *software*, tanto que, nos últimos anos, foram escritos muitos livros a seu respeito, os quais abrangem vários aspectos dessa técnica, como ferramentas específicas e experiências de projetos. Atualmente existem ferramentas de TDD para a maioria das linguagens de programação, desde C++ até *Visual Basic* (JEFFRIES, 2007).

George e Williams (GEORGE, 2003) realizaram um dos primeiros experimentos com TDD, composto por 12 pares de programadores, sendo 6 pares para cada método (TDD e modelo cascata). Ambos os grupos desenvolveram um pequeno programa em Java. Os resultados obtidos neste experimento foram:

- Os desenvolvedores que utilizaram o TDD produziram código com mais qualidade, com ganho de 18% em testes de caixa-preta em termos de redução de defeitos (GEORGE, 2003);
- A equipe do TDD gastou 16% a mais de tempo no desenvolvimento. Entretanto, deixaram como subproduto os testes de regressão, enquanto os desenvolvedores do método convencional não completaram seus testes (GEORGE, 2003).

Uma das vantagens do TDD é aumentar a confiança do desenvolvedor e conseqüentemente do cliente, sobre a qualidade do sistema que está sendo criado (GEORGE, 2003). Segundo Cornett (2002 apud GEORGE, 2003), o padrão de cobertura de código usado na indústria de *software* está entre 80% e 90%, apesar de o ideal ser uma cobertura de código

de 100%. Entretanto, com o uso do TDD os desenvolvedores aumentam essa cobertura, variando de 92% a 98%, dependendo do tipo de cobertura (método, instrução de código ou desvio). Ainda de acordo com o experimento de George e Williams, podem-se identificar outras vantagens dessa técnica, como: gerar código com maior qualidade, facilitar a realização de testes de regressão, permitir a refatoração sem receio e, ainda, os testes podem ser usados como documentação técnica do sistema (GEORGE, 2003).

O TDD utiliza ferramentas da família xUnit, que é uma família de ferramentas para o desenvolvimento de testes de unidade, disponíveis para várias linguagens de programação. Kent Beck é o criador do conceito da ferramenta xUnit, com sua primeira implementação no início dos anos noventa, usando a linguagem de programação Smalltalk, que ficou conhecida como SUnit (BECK, 1994). Entretanto esse conceito ganhou mais destaque com a linguagem de programação Java, por meio da ferramenta JUnit criada por Erich Gamma e Kent Beck (WICK, 2005).

Os pesquisadores concordam que o TDD encoraja o foco nas tarefas e a cobertura de teste, embora o simples fato de realizar mais testes não signifique necessariamente que a qualidade do *software* seja melhor. Considerando-se teste como uma grande amostra de potenciais comportamentos, mais testes significam mais exemplos completos. Cada teste pode encontrar um problema importante que nenhum outro teste identificou, testes são proveitosos, especialmente quando a execução deles é fácil e barata (JEFFRIES, 2007).

A grande diferença do TDD em relação às outras práticas de teste de *software* é que os testes são elaborados antes mesmo de se escrever o código do programa, dessa maneira a necessidade de o teste passar ajuda a definir a forma de implementar a solução do programa. Por isso, diz-se que, no TDD, o desenvolvimento é dirigido por testes. Nas outras práticas de teste, os programadores primeiro codificam as funcionalidades do sistema, para posteriormente realizarem os testes de unidade.

Alguns desenvolvedores consideram que o TDD é contra a intuição, pois os ciclos extremamente curtos entre escrever um teste e fazê-lo passar são contra a prática de escrever todos os módulos do sistema e testá-los manualmente posteriormente. O TDD altera a forma como os desenvolvedores trabalham, minuto a minuto, o que afeta profundamente o resultado do trabalho (MARTIN, 2007).

Utilizando-se das práticas do TDD, escrevem-se dúzias de teste por dia, centenas de testes por mês e milhares de testes por ano. A experiência mostra que esses testes cobrem mais de 90% do código produzido. Eles devem ser guardados, devem ser fáceis de executar e essa execução não deve demorar mais do que alguns minutos (MARTIN, 2007).

De acordo com Spolsky (2000 apud MARTIN, 2007), a atividade de depurar é difícil de estimar e pode demorar de 100 a 200% do tempo de codificação. Independente da precisão desses dados, o fato é que a depuração é uma incógnita no cronograma de um projeto de *software*. TDD reduz esta incógnita, pois os defeitos são encontrados mais precocemente (MARTIN, 2007).

Na tabela 1, é apresentado um resumo de alguns estudos realizados com o TDD, o que reflete o estado atual da pesquisa dessa técnica e sumariza o impacto em relação à produtividade e à qualidade observadas na indústria e nos trabalhos acadêmicos (JEFFRIES, 2007). A tabela é composta pelas seguintes colunas:

- Autores: indica os autores que realizaram o estudo;
- Tipo: indica o tipo de estudo realizado, como: estudo de caso, experimento controlado, entre outros;
- Tempo de desenvolvimento analisado: indica o tempo de desenvolvimento analisado durante o estudo;
- Organização: indica a organização em que o estudo foi realizado, seja na indústria ou no meio acadêmico;
- Tamanho do *software*: indica o tamanho do *software* em que o estudo foi realizado;
- Número de participantes: indica o número de participantes do estudo realizado;
- Linguagem: indica a linguagem de programação utilizada no estudo;
- Efeito em relação à produtividade: indica o efeito da utilização da técnica TDD em relação à produtividade dos programadores;
- Efeito em relação à qualidade: indica o efeito da utilização da técnica TDD em relação à qualidade do *software* produzido.

As colunas *Efeito em Relação à Produtividade* e *Efeito em Relação à Qualidade* são classificadas com cores, em que o amarelo claro indica que há uma deterioração devido à

utilização da técnica TDD, enquanto que a cor verde claro indica que há uma melhora (JEFFRIES, 2007).

Tabela 1 - Sumário dos estudos realizados com o TDD (*).

Autores	Tipo	Tempo de desenv. analisado	Organização	Tam. do *software*	Nº de partic.	Linguagem	Efeito em relação à produtividade	Efeito em relação à qualidade
Sanchez et al.	Estudo de caso	5 anos	IBM	Médio	9-17	Java	Aumento de 19% de esforço.	40%†
Bhat and Nagappan	Estudo de caso	4 meses	Microsoft	Pequeno	6	C/C++	Aumento de 25-35% de esforço.	62% †
	Estudo de caso	7 meses	Microsoft	Médio	5-8	C++/C#	Aumento de 15% de esforço.	76% †
Melis et al.	Simulação	49 dias		Médio	4	Smalltalk	Aumento de 17% de esforço.	36% de redução na densidade de defeito residual.
Mann	Estudo de caso	8 meses	PetroSleuth	Médio	4-7	C#	Não analisado.	81% § dos clientes e desenvolvedores perceberam a aumento de qualidade.
George and Williams	Experimento quase controlado	4.75 horas	John Deere, Role Model *Software*, Ericsson	Muito pequeno	24	Java	Aumento de 16% de esforço.	18% #

Autores	Tipo	Tempo de desenv. analisado	Organização	Tam. do *software*	Nº de partic.	Linguagem	Efeito em relação a produtividade	Efeito em relação a qualidade
Erdogmus et al.	Experimento controlado	13 horas	Politecnico di Torino	Muito pequeno	24	Java	Aumento de 22% de produtividade.	Sem diferença.
Melnik and Maurer	Vários estudos de caso	4 meses Projetos nos últimos 3 anos	Universidade de Calgary/SAIT Polytechnic	Pequeno	240	Java	Não analisado.	73% dos respondentes percebem que o TDD aumenta a qualidade.
Edwards	Análise de artefatos	2-3 semanas	Virginia Tech	Muito pequeno	118	Java	Aumento de 19% de esforço.	45%
George	Experimento quase controlado	1-3/4 horas	Universidade do Estado na Carolina do Norte	Muito pequeno	138	Java	Aumento de 16% de esforço.	16%
Muller and Hagner	Experimento quase controlado	10 horas	Universidade de Karlsruhe	Muito pequeno	19	Java	Nenhum efeito.	Nenhum efeito, mas melhora o entendimento do programa.
Ynchausti	Estudo de	8,5 horas	Monstor	Pequeno	5	Não se	Aumento de 60-100% de	38-267% †

	caso		Consulting			aplica	esforço.	
Pancur et al	Experimento controlado	4 meses e meio	Universidade de Ljubljana	Muito pequeno	38	Java	Não analisado.	Sem diferença.

Fonte: Jeffries; Melnik, (2007, p.28-29)

(*) Célula verde = indica que houve uma melhora, célula amarela = indica que houve uma deterioração.

† Redução na densidade interna de defeitos.

§ Redução na relação de defeitos externos (não pode ser atribuído somente ao TDD, mas a um conjunto de práticas).

Aumento no percentual de testes de caixa-preta que passaram.

O TDD está se tornando popular independentemente do tamanho e do tipo de projeto de desenvolvimento de *software*. Muitas universidades já inseriram o TDD na grade curricular de seus cursos, inclusive a diretriz do IEEE/ACM de 2004 para os estudantes dos cursos de engenharia de *software*, que cita o TDD como mais uma habilidade desejável (JEFFRIES, 2007).

O TDD é um excelente estilo de desenvolver *software* e pode mudar a maneira como os desenvolvedores pensam a respeito de *software* e do seu desenvolvimento, entretanto a habilidade dos desenvolvedores e a confiança ainda representam um grande papel para assegurar a qualidade do produto final (VODDE, 2007).

2.6.1 Integrando TDD com o XP e com o UP

O TDD pode ser utilizado independentemente do modelo de desenvolvimento de *software*, embora seja mais usado entre os seguidores do XP. Os desenvolvedores que utilizam modelos como o UP, podem usar essa técnica e se beneficiarem com as suas vantagens, como, por exemplo, aumentar a qualidade do código produzido.

A utilização da técnica TDD no modelo XP é simples, por ela já ser uma das práticas desse modelo e já estar integrada ao processo. Entretanto, utilizar o TDD no modelo do UP não é tão evidente, apesar de ser possível construir o código utilizando essa técnica, o que requer algumas mudanças, tanto no processo de desenvolvimento do sistema quanto no comportamento dos desenvolvedores de sistema.

Em relação ao processo do UP, as principais mudanças são na fase de construção, pois os desenvolvedores devem, a partir dos casos de uso elaborados pelos analistas de sistemas, criar os casos de teste antes de escrever qualquer linha de código. Para tanto devem utilizar ferramentas específicas para automação de teste, como da família xUnit.

Já em relação aos desenvolvedores, a principal mudança é na forma de pensar, pois eles devem primeiro pensar nos testes funcionais antes de codificar o sistema, o que não é uma mudança trivial, visto que esses desenvolvedores estão habituados a codificar primeiro para depois criar os testes para o código já pronto.

Conforme mencionado no resumo dos experimentos realizados com o TDD, ilustrados na tabela 1 deste trabalho, em média, a utilização da técnica TDD requer mais tempo de desenvolvimento, o que, em contrapartida, aumenta a qualidade do *software*.

2.7 Outros Modelos Dirigidos por Testes

A seguir são descritos mais dois modelos dirigidos por testes que são: Modelagem Dirigida por Testes e Modelo Baseado no Teste Extremado. Esses modelos ainda são pouco conhecidos e utilizados no desenvolvimento de sistemas, entretanto, para o desenvolvimento deste trabalho, são importantes para exemplificar a disseminação do conceito Dirigido por Testes.

2.7.1 Modelagem dirigida por testes

A Modelagem Dirigida por Testes (TDM, do inglês *Test-driven Modeling*) é uma técnica recente, idealizada por Yuefeng Zhang. Essa técnica envolve a automação de testes por meio da simulação e do uso de modelos executáveis, como se fossem documentos da arquitetura do sistema. No TDM, usam-se os diagramas de seqüência de mensagens, ou *Message Sequence Charts* (MSCs), para a análise do sistema e para a criação dos casos de teste. Desse modo, economiza-se uma quantidade significativa de tempo com o reuso dos MSCs e com a modelagem de diagramas, em comparação com os outros métodos de desenvolvimento de sistemas (ZHANG, 2004).

O diagrama de seqüência de mensagens (MSC) utilizado no TDM é um diagrama de interação do *Specification and Description Language* (SDL), entretanto esse diagrama é semelhante ao diagrama de seqüência da UML. A SDL é uma linguagem para especificação e descrição formal de sistemas de tempo real e é mais utilizada para a definição de sistemas de telecomunicações (ELLSBERGER, 1997).

É sabido que a atividade de testes é importante para o desenvolvimento de qualquer tipo de *software*, porém o desafio está em descobrir quando parar de testar, ou seja, o quanto de teste é o suficiente para garantir que as funcionalidades solicitadas pelo cliente estão desenvolvidas e funcionando de acordo com suas expectativas. A análise de cobertura de

código no TDM também destaca a dificuldade de determinar quando parar de testar, porém o TDM almeja alcançar um alto percentual de cobertura, idealmente 100% (ZHANG, 2004).

Os MSCs consideram o sistema como uma caixa preta, ou seja, apenas indicam a troca de mensagens entre o sistema e o meio ambiente do sistema (atores da UML). A arquitetura do sistema apenas indica quais subsistemas são usados pelo sistema, como eles estão conectados entre si para formar o sistema e quais classes e objetos ativos e passivos são usados e como estão conectados com os subsistemas.

O TDM é dirigido por testes porque inicialmente são criados os MSCs do sistema, para posteriormente definir a arquitetura e projeto do sistema. Os MSCs são automaticamente executados por simulação para verificar se a arquitetura do sistema atende aos seus requisitos. Finalmente, os MSCs são refinados e executados para criar os testes de unidade (ZHANG, 2004).

No TDM, os MSCs são usados como casos de teste, em que um MSC pode referir-se a outros MSCs, construindo uma hierarquia de MSCs. Assim, um caso de teste pode incluir outros casos de teste. É usada a ferramenta de modelagem *Telelogic TAU SDL* para indicar quais MSCs passaram ou falharam no teste. Se a execução de um MSC falhar, então a ferramenta automaticamente indica o ponto exato da falha (ZHANG, 2004).

A intenção do TDM é usar a idéia contida nos processos dirigidos por testes, como acontece no TDD, porém enfatizando o modelo. Observa-se que tais métodos são semelhantes, entretanto o primeiro enfatiza o modelo, enquanto que o segundo enfatiza o código. Na Tabela 2, é exibida a similaridade entre os dois métodos.

Tabela 2 – Similaridades entre TDM e TDD.

Correspondência		TDM	TDD
Atividades correspondentes	Implementação	Modelagem	Codificação
	Teste	Simulação	Execução dos casos de teste junto ao código da aplicação.
Artefatos correspondentes	Casos de teste	MSC (diagramas de seqüência de mensagens)	Implementação dos casos de teste.
	Conjunto de testes	Um conjunto de MSCs	Implementação de um conjunto de casos de teste.
Características correspondentes	Testes controlados	Primeiramente cria as MSCs.	Primeiramente cria casos de testes.
	Automação de teste	Execucão automática das MSCs usando uma ferramenta.	Execução automática dos casos de teste usando uma ferramenta da família xUnit.
	Testes de regressão	Executa todas as MSCs para a iteração atual e para a anterior.	Executa todos os casos de teste implementados para a iteração atual e para a anterior.
	Iterativo	Múltiplas iterações de desenvolvimento.	Múltiplas iterações de desenvolvimento.
	Incremental	Cada iteração implementa mais funcionalidade de sistema no topo da iteração anterior.	Cada iteração implementa mais funcionalidades de sistema no topo da iteração anterior.

Fonte: Zhang, (2004, p.84.)

2.7.2 Modelo baseado no teste extremo

O modelo baseado no teste extremo, ou *Model-Based Extreme Testing* (MXT), é uma alternativa para a atividade de teste dar mais ênfase às propriedades abstratas do sistema. Esse modelo é decorrente do XP e do modelo baseado em teste e pode ser usado para prover critérios de teste completos para o nível de teste do sistema (HOWDEN, 2002).

O objetivo do modelo baseado em teste extremo é usar o diagrama de estado como especificação de um programa. No caso de um programa interativo, os eventos de transição podem corresponder às ações de entrada do usuário, enquanto que os estados podem corresponder às respostas do sistema (HOWDEN, 2002).

O modelo baseado em teste extremo apresenta potencias problemas. Por exemplo, os modelos freqüentemente não estão disponíveis, até mesmo para sistemas que naturalmente apresentam uma especificação técnica. Além do que, os requisitos do modelo baseado em teste são freqüentemente apresentados como caminhos diretos ao modelo que devem ser seguidos pelos testes. Se o modelo é abstrato, então é necessário desenhar o caminho do modelo abstrato em uma seqüência concreta de teste, com ações dos usuários e entrada de dados, o que pode não ser trivial e não ter um processo automatizado (HOWDEN, 2002).

A idéia básica do MXT é rastrear o comportamento do sistema durante o teste, e, então, usar essa investigação para gerar o modelo abstrato do sistema (HOWDEN, 2002). Esse modelo resume a cobertura dos testes funcionais executados até o momento; serve como especificação funcional parcial de um sistema parcial e pode ser usado para propor funcionalidades adicionais que podem ser desempenhadas pelo sistema parcial que ele descreve.

2.8 Conclusão

Este capítulo apresenta vários conceitos que são importantes para o desenvolvimento da técnica que é proposta e apresentada no próximo capítulo.

O Processo Unificado é uma metodologia de desenvolvimento de sistemas centrada na arquitetura, iterativa e guiada por casos de uso. Os casos de uso descrevem as funcionalidades do sistema por meio dos fluxos básicos, alternativos e de exceções.

Já a Programação Extrema é baseada em valores, princípios e práticas e utiliza os cartões de história, escritos pelos clientes, para descrever sucintamente uma funcionalidade. O principal objetivo do cartão de história é estimar o tempo de desenvolvimento de uma determinada funcionalidade do sistema.

Os casos de teste são uns conjuntos de condições ou variáveis utilizados para testar os elementos do software. O teste pode ser realizado para identificar defeitos nas estruturas internas do *software* ou para garantir que as funcionalidades desenvolvidas estejam de acordo com os requisitos do sistema.

Também são detalhados, neste capítulo, os modelos de desenvolvimento de *software* dirigidos por testes, com destaque especial para o TDD, por ser o modelo dirigido por testes mais utilizado entre os desenvolvedores de sistemas. Os outros modelos, TDM e MXT, também são detalhados por dois motivos. Primeiro, para explicitar que existe uma tendência que no desenvolvimento de sistemas os testes sejam executados cada vez mais cedo e, segundo, para descrever o funcionamento desses modelos.

Conclui-se que, para que o desenvolvimento possa ser dirigido por testes, a especificação também deve ser dirigida por testes. Assim, deve-se antecipar a elaboração dos casos de teste, realizando-os na fase de especificação e usando-os ao longo do processo de desenvolvimento do sistema.

CAPÍTULO 3
EESPECIFICAÇÃO DIRIGIDA POR CASOS DE TESTE – EDCT

3.1 Introdução

Neste capítulo, apresenta-se a proposta deste trabalho que é a de substituir a especificação de requisitos de sistema, baseado em casos de uso, pela atividade de elaboração dos casos de teste. Essa proposta chama-se Especificação Dirigida por Casos de Teste (EDCT). Propõe-se uma forma de detalhar os casos de teste de modo que eles possam ser utilizados para desenvolver o sistema, seguindo o modelo proposto no Desenvolvimento Dirigido por Testes (TDD), em que os casos de teste dirigem o desenvolvimento do sistema. A proposta dessa técnica é detalhar os casos de uso no formato de casos de teste, e a esse artefato chama-se caso de teste de uso (CTU).

Nas próximas seções deste capítulo, descreve-se, com detalhes, o conceito do caso de teste de uso e como implementá-lo. Em seguida, apresenta-se uma avaliação do impacto da adoção do EDCT nos modelos UP e XP estudados no Capítulo 2.

3.2 Técnica EDCT

No modelo de desenvolvimento UP, os casos de uso, identificados na fase de concepção, são a base para a criação dos casos de teste, e são esses casos de uso que dirigem o desenvolvimento do sistema. A proposta da técnica EDCT é utilizar os casos de teste de uso (CTU), junção dos casos de uso com os casos de teste, para dirigir o desenvolvimento do sistema. Os CTU são as especificações do sistema no formato de casos de teste.

O uso da técnica EDCT se adapta ao TDD, porque propõe que os casos de teste também dirijam o desenvolvimento do sistema. A idéia é integrar elementos dos modelos UP e XP, em especial do TDD, para desenvolver a técnica EDCT.

Outra característica da técnica EDCT é seguir a tendência do mercado de *software*, a qual propõe trazer a atividade de teste para o início do processo de desenvolvimento de sistemas.

3.3 Introdução do Conceito de Caso de Teste de Uso – CTU

Observa-se que os casos de uso representam uma das formas mais populares de documentação de requisitos, forma bastante conhecida e utilizada entre os analistas de sistemas. A técnica EDCT propõe criar os casos de teste de uso em substituição ao detalhamento dos casos de uso. Como base para criar os casos de teste de uso, utiliza-se a descrição sucinta da funcionalidade que pode ser: o objetivo principal do caso de uso ou o cartão de história do XP. O caso de uso passa a ter o mesmo nível de detalhe do cartão de história do XP, ou seja, passa a descrever, de forma resumida, a funcionalidade a ser desenvolvida.

Um caso de teste deve abranger os requisitos funcionais, assim como os fluxos alternativos e de exceções. Dessa mesma maneira, um caso de teste de uso engloba os requisitos funcionais com seus fluxos alternativos e de exceções, visto que bons casos de teste vão além do tratamento de fluxos básicos.

Um benefício de escrever a especificação em forma de casos de teste de uso (CTU) é que ela pode facilitar a validação pelo usuário final, visto que ele consegue visualizar, a partir dos casos de teste, como o sistema deve funcionar e como esse se comporta nas mais variadas situações, o que reduz a ambigüidade da especificação e torna-a mais tangível.

O fato de se escrever a especificação em forma de casos de teste de uso é também uma maneira de se aproveitar a experiência do usuário final, pois, a partir da especificação em casos de teste fornecidas pelo usuário, cria-se uma especificação de requisitos, que comunica claramente a todos os envolvidos como o sistema deve funcionar. O objetivo dessa especificação é aproveitar o benefício de o usuário especificar o sistema de *software* com exemplos, ou seja, com casos de teste. Entretanto, como ressalva a essa especificação por meio de exemplos, deve-se lembrar que todo exemplo, assim como todo caso de teste, é limitado a alguma funcionalidade. Assim sendo, devem-se buscar critérios de cobertura para abranger todos os requisitos do sistema.

O artefato produzido pela técnica EDCT é dividido em duas partes, uma parte contendo suas informações gerais – cabeçalho do documento – e outra contendo os casos de teste. Na Figura 13, está ilustrado esse artefato.

A parte do cabeçalho é composta por informações referentes ao caso de uso analisado, que contém os seguintes itens:

- Sistema: nome do sistema;
- Caso de Uso: nome do caso de uso do sistema;
- Data da elaboração: data da elaboração do artefato pelo analista responsável;
- Objetivo principal do caso de uso: descrição do principal objetivo do caso de uso a que se refere;
- Elaborado por: identificação do analista responsável pela elaboração do plano de teste;
- Executado por: identificação do analista responsável pela execução do teste;
- Versão: indicação da versão do artefato.

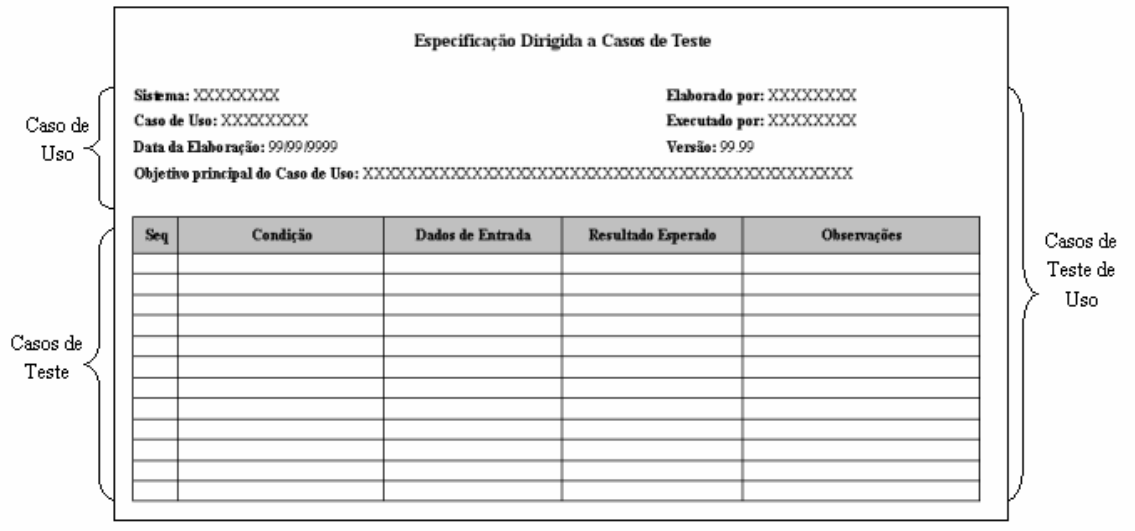

Figura 13 – Artefato da Especificação Dirigida por Casos de Teste.

A outra parte do artefato é apresentada em tabela com informações referentes aos casos de teste, que contém os seguintes itens:

- Seq: é o número seqüencial que identifica o caso de teste;
- Condição: descrição das condições necessárias para a execução do caso de teste especificado;
- Dados de Entrada: descrição dos valores de entrada necessários para a execução do caso de teste;
- Resultado Esperado: descrição do resultado esperado com a execução do caso do teste;
- Observações: descrição das observações, se houver, referentes ao caso de teste. Esse campo é opcional.

O cabeçalho do documento da Figura 13 é um breve resumo do caso de uso, enquanto que a outra parte do artefato, a tabela, é composta pelos casos de teste. O conjunto desse documento forma o caso de teste de uso (CTU).

O artefato produzido pela técnica EDCT abrange os tipos de teste funcional e de unidade e também pode ser utilizado para testes de regressão.

A contribuição desse artefato não está exatamente no seu formato, ou seja, nos seus campos, inclusive porque as colunas da tabela são semelhantes às informações sugeridas pelo padrão ANSI/IEEE 829 para criar casos de teste. A diferença está na proposta de utilização desse artefato que deixa de ser utilizado apenas para testar o sistema e é proposto para que seja utilizado para dirigir o desenvolvimento do sistema.

Ainda neste capítulo detalha-se como implementar os casos de teste de uso da técnica EDCT. Em seguida, também se detalha como incorporar essa técnica nas práticas do XP.

3.4 Implementação dos Casos de Teste de Uso

A proposta da técnica EDCT é, após a identificação dos atores e dos casos de uso, detalhar todos as funcionalidades do sistema em forma de casos de testes.

Na técnica EDCT, deve-se, primeiramente, selecionar um caso de uso dentre os casos de uso identificados e, preferencialmente, iniciar por um caso de uso simples e pequeno.

Escolhido o caso de uso, devem-se criar os casos de teste de uso do fluxo básico e incrementá-los com os casos de teste de uso dos fluxos alternativos e de exceções. Para cada caso de teste, devem-se detalhar as condições necessárias para execução do teste, os dados de entrada e o resultado esperado com a sua execução. O mesmo procedimento pode ser realizado para criar os casos de teste de uso, a partir dos cartões de história do XP.

Quando se finalizam os casos de teste de uso do caso de uso, seleciona-se o próximo caso de uso e repete-se a seqüência anterior até finalizar todos os casos de uso.

Os casos de teste de uso são criados pelo analista de sistemas e utilizados pelo desenvolvedor para codificar o sistema e, posteriormente são utilizados pelo analista de testes ou pelo responsável por executar os testes do sistema. Se o modelo de desenvolvimento utilizado for o XP, os casos de teste de uso são escritos pela mesma pessoa que é responsável pela codificação e pela execução dos testes.

Para cada CTU, pode-se aplicar o diagrama de atividades do TDD, apresentado no capítulo 2 (ver Figura 12). Ou seja, para cada caso de teste de uso criado, pode-se criar código, aos poucos, seguindo o modelo do TDD. Deve-se executar o teste do CTU, se o teste passar cria-se código para outro CTU; se o teste falhar, fazem-se mudanças até que o teste passe.

Ao final do desenvolvimento do sistema, os mesmos casos de teste usados no desenvolvimento são usados na fase de homologação do sistema que podem ser executados pelos mesmos analistas que os criaram, ou por uma equipe de arquitetos de teste, dependendo da metodologia adotada pela empresa. Os casos de teste de uso também podem ser utilizados para testes de aceitação realizados pelos usuários finais.

Como vantagem da técnica EDCT, pode-se citar a redução de artefatos a serem atualizados, pois, se na fase de homologação for encontrado alguma exceção não detalhada nos casos de teste, ou até mesmo alguma funcionalidade nova, basta atualizar os casos de teste para que sirvam como documentação do sistema, bem como posteriormente podem ser utilizados para testes de regressão. Não é necessário atualizar o caso de uso ou o cartão de história, visto que eles descrevem apenas de forma resumida a funcionalidade do sistema.

Uma limitação da técnica EDCT é a dependência em relação ao protótipo do sistema, embora ele não necessite ser interativo, pois a existência de um protótipo do sistema facilita a criação dos casos de teste de uso. De acordo com Gottesdiener (2003), os protótipos desenvolvidos na presença dos usuários, ou pelo menos revisados por eles, trazem mais realidade para os casos de uso. Além do mais, o protótipo descreve os requisitos do sistema do ponto de vista dos atores e servem para melhorar o entendimento dos casos de uso (JACOBSON, 1999).

Associam-se os casos de uso com o início do ciclo de desenvolvimento de *software*, enquanto que os casos de teste estão associados com o final do ciclo. A utilização dos casos de uso para gerar os casos de teste de uso possibilita identificar e reparar defeitos que seriam mais onerosos se fossem consertados mais tarde.

De acordo com Palmer (1997), o rastreamento de requisitos estabelece o relacionamento entre os requisitos e os programas do sistema, com isso podem-se identificar e corrigir erros mais eficientemente, pois sabe-se exatamente qual programa implementa o requisito que apresenta problemas. Os CTU também estabelecem esse relacionamento entre requisitos e programa, o que reduz o tempo e o custo de manutenção de um sistema.

A granularidade dos casos de teste de uso facilita as alterações de requisitos durante o desenvolvimento do sistema, isso porque os CTU são bem detalhados, evitando requisitos dúbios e inconsistentes. Dessa maneira, a técnica EDCT pode diminuir os defeitos decorrentes da alteração de requisitos durante o processo de desenvolvimento.

3.5 Avaliação do Impacto da Adoção do EDCT nos Modelos UP e XP

Neste item, é avaliado o impacto da adoção da técnica EDCT nos modelos UP e XP, exibindo quais atividades, artefatos e fases do processo de desenvolvimento de sistemas podem ser adaptados para utilizar a técnica proposta.

A técnica EDCT é independente de ferramenta, ou seja, não é necessário utilizar nenhuma ferramenta para desenvolver os casos de teste de uso, que podem ser elaborados pelo analista de sistemas. Outro ponto importante é que o EDCT pode ser usado

independentemente do modelo de desenvolvimento de sistemas, com a possibilidade de se adaptar a ele.

A seguir, detalha-se como adaptar a técnica EDCT aos modelos de desenvolvimento de sistemas UP e XP.

3.5.1 Adaptação do EDCT ao UP

Nesta seção, é proposta uma sugestão de adaptação da técnica EDCT ao modelo de desenvolvimento de sistemas UP. Na Figura 14, ilustra-se o fluxo geral do UP, já com as adaptações das atividades para utilizarem os casos de teste de uso, em vez dos casos de uso.

A primeira adaptação do UP ocorre no fluxo de trabalho Requisitos, na atividade de “Detalhamento dos casos de uso”. No UP, essa atividade ocorre após a identificação dos atores e da definição dos casos de uso. A técnica EDCT propõe substituir a atividade de “Detalhamento dos casos de uso” do fluxo de trabalho Requisitos, que é executada pelo papel Analista de Sistemas, pela atividade de “Detalhamento dos casos de teste de uso”. Essa nova atividade, identificada em vermelho na Figura 14, continua no mesmo fluxo de trabalho Requisitos e continua sendo executada pelo papel Analista de Sistemas.

Propõe-se a substituição da atividade de detalhamento dos casos de uso, pois os casos de uso passam a ter apenas uma descrição resumida da funcionalidade a ser desenvolvida. Esses casos de uso são utilizados como base para escrever os casos de teste de uso. A adaptação da técnica EDCT ao modelo UP não elimina a necessidade de criação do diagrama de casos de uso.

A segunda adaptação sugerida no UP é em relação a atividade de “Protótipo da Interface do Usuário”. No UP, essa atividade é realizada após o “Detalhamento dos casos de uso”, porém a sugestão é que ela seja realizada antes da atividade de “Detalhamento dos casos de teste de uso”, pois o protótipo auxilia na elaboração dos casos de teste de uso. A seta verde na Figura 14 indica essa inversão de ordem que continua sendo realizada pelo papel Projetista da interface do usuário.

No fluxo de trabalho Análise, a adaptação sugerida é em relação à atividade “Análise dos casos de uso” que passa a se chamar “Análise dos casos de teste de uso”, pois passa a analisar os CTU em vez dos casos de uso. Na Figura 14, essa adaptação é ilustrada em vermelho.

A outra adaptação sugerida é em relação ao fluxo de trabalho Teste, pois os casos de teste de uso passam a ser subsídios para a atividade de “Planejar o teste”; a seta verde da Figura 14 ilustra essa adaptação. Outra adaptação nesse fluxo é em relação à eliminação da atividade “Projetar o teste”, visto que a elaboração dos casos de teste é antecipada para a atividade de “Detalhamento dos casos de teste de uso”.

No fluxo de trabalho Implementação, a adaptação se dá em relação à atividade de “Implementar as classes” que passa a ser implementada a partir dos casos de teste de uso. Com isso, os fluxos de trabalho Implementação e Teste passam a ser realizados em conjunto, conforme ilustrado na Figura 14 identificado na cor amarela.

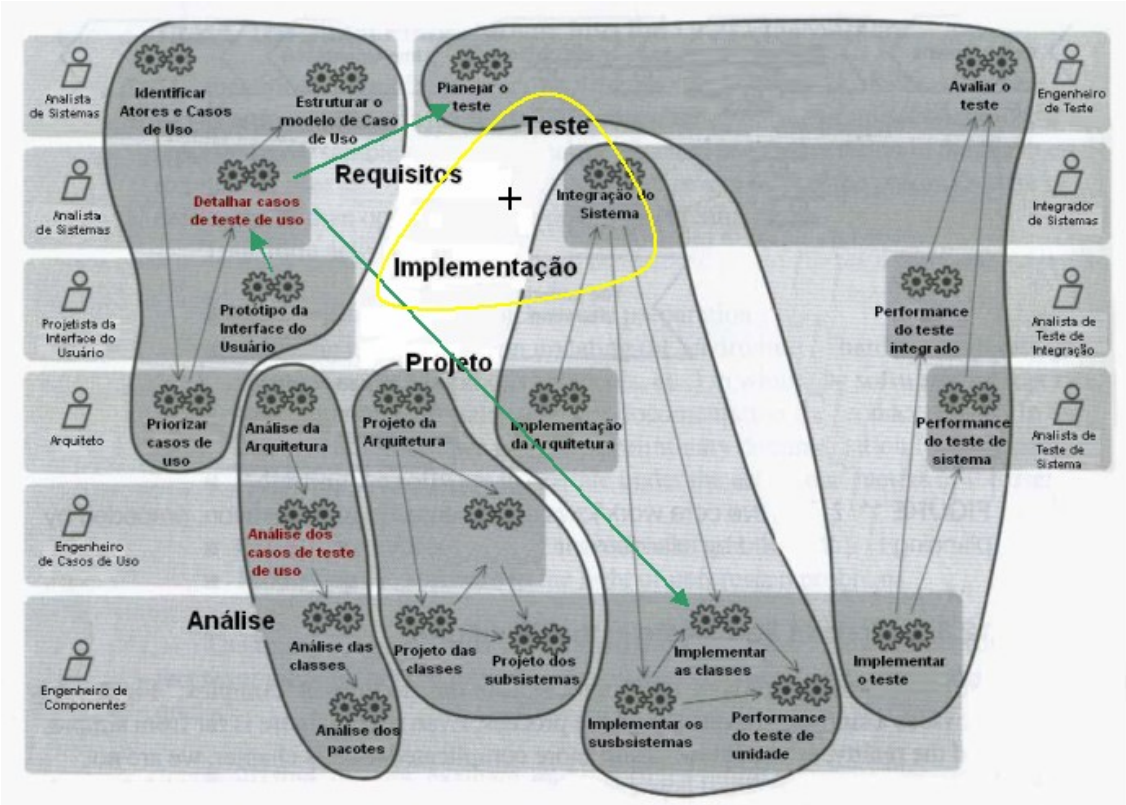

Figura 14 – Fluxo Geral do UP adaptado à técnica EDCT.

Os CTU elaborados na fase de concepção do UP guiam o desenvolvimento do sistema por serem utilizados como especificação do sistema durante a fase de construção, o que significa que o desenvolvedor usa os CTU para codificar o sistema, de acordo com as funcionalidades especificadas nos casos de teste. O CTU também é usado para testar o sistema, podendo até ser utilizado para o teste de aceitação do cliente.

Uma outra sugestão de adaptação no modelo UP é, após a criação dos casos de teste de uso, utilizar a técnica TDD para guiar o desenvolvimento do sistema e, assim, executar os casos de teste de uso antes de criar o próprio código do sistema.

3.5.2 Adaptação do EDCT ao XP

Uma sugestão de integração do CTU com o XP é poder utilizar os casos de teste de uso como especificação a ser entregue para uma fábrica de *software*, ou seja, uma outra empresa desenvolve *software* utilizando a técnica do TDD. O modelo atual do XP pode dificultar a utilização de fábricas de *software*, uma vez que é mais difícil para ela não conhecer o negócio e desenvolvê-lo apenas a partir do cartão de história.

No XP, os casos de teste de uso podem ser elaborados a partir dos cartões de histórias escritos pelos clientes, como uma etapa intermediária antes da codificação dos testes que utiliza a técnica TDD.

Os casos de teste de uso também podem ser adaptados ao modelo de desenvolvimento de sistemas XP, quando se propõe a utilização dos CTU como um artefato para facilitar a elaboração do teste de aceitação, ou até mesmo utilizá-los como o próprio teste de aceitação, uma vez que os CTU descrevem um conjunto de testes de aceitação.

A prática “Jogo de planejamento do XP” também pode ser adaptada à utilização dos casos de teste de uso, pois esses casos podem auxiliar na definição da estimativa de cada tarefa. O fato de os casos de teste de uso serem detalhados permite que a estimativa do planejamento seja mais precisa.

3.6 Conclusão

Neste capítulo, são descritos a técnica EDCT e o modo de implementar os casos de teste de uso. É proposto, ainda, um artefato que contém os casos de teste de uso a serem utilizados durante o desenvolvimento do sistema.

No capítulo, também é apresentada uma avaliação do impacto da adoção da técnica EDCT nos modelos UP e XP estudados no Capítulo 2 e é sugerido um modelo de desenvolvimento de sistemas híbrido que mescla atividades do UP e do XP.

As idéias de adaptação dos modelos UP e XP apresentadas nas seções 3.5.1 e 3.5.2 são sugestões para desenvolvimento de trabalhos futuros, necessários para avaliar os benefícios e os impactos dessas sugestões em projetos de construção de sistemas.

No próximo capítulo, são apresentados dois casos de aplicação da técnica EDCT, sendo um exemplo didático, com o Jogo da Forca, e outro exemplo prático, com a contratação do produto Termo de Moeda. Também são apresentados os casos de teste de uso referentes aos dois casos de aplicação. Finalmente, realiza-se uma pesquisa com um grupo de analistas e de desenvolvedores de sistemas, para avaliar a aplicabilidade da técnica EDCT.

CAPÍTULO 4
EXEMPLOS DE APLICAÇÃO E DE VALIDAÇÃO DA TÉCNICA EDCT

4.1 Introdução

Neste capítulo apresentam-se dois exemplos de aplicação da técnica EDCT. No primeiro exemplo, com o Jogo da Forca, os casos de uso estão detalhados, com seus fluxos básicos, alternativos e de exceções. Em seguida, detalham-se os casos de teste no novo formato do EDCT, ou seja, descreve-se a especificação dessa aplicação no formato de casos de teste de uso (CTU), visando à comparação direta.

Um outro exemplo de aplicação da técnica EDCT também é descrito neste capítulo e trata da contratação do produto Termo de Moeda que é praticado no mercado financeiro brasileiro. Para esse exemplo, o caso de uso é apenas uma descrição sucinta da funcionalidade, semelhante ao cartão de história do XP.

Os exemplos trabalhados servem para a avaliação dos resultados da pesquisa realizada com um grupo de desenvolvedores de sistemas para aferir a viabilidade de aplicação da técnica EDCT.

4.2 Caso de Aplicação: Jogo da Forca

O estudo de caso, Jogo da Forca, que é analisado neste capítulo, é extraído do livro Modelagem Orientada a Objetos com UML (DEBONI, 2003).

“Escolheu-se este jogo como caso de aplicação por ter a vantagem de ser bem conhecido por todos, o que permite que qualquer um possa ser considerado um especialista do assunto, propor e analisar as suas funcionalidades. No Jogo da Forca em computador, o jogador é desafiado a descobrir as letras que formam uma palavra secreta, escolhida automaticamente pelo computador. A cada letra errada, sugerida pelo jogador, ele perde uma parte do corpo: cabeça, tronco e membros desenhados no boneco da forca. Ao se desenhar todas as partes deste boneco, o jogador perde o jogo enforcado. Por outro lado, ao acertar todas as letras e descobrir a palavra secreta, o jogador vence o jogo (DEBONI, 2003)”. Na Figura 15, é ilustrado o esquema típico deste jogo.

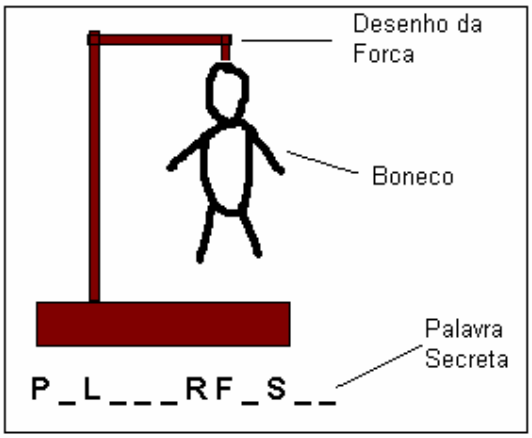

Figura 15 – Esquema típico do Jogo da Forca
Fonte: Deboni (2003).

"O modelo de contexto do Jogo da Forca é representado pelo diagrama de casos de uso e pela descrição textual de cada caso de uso. Para aumentar o detalhamento do modelo, divide-se o sistema em duas atividades: iniciar um novo jogo e escolher letras para adivinhar a palavra secreta. No diagrama de casos de uso do Jogo da Forca, optou-se por representar a lista de palavras como um ator, porque ela pode ser considerada como um elemento imutável, externo ao sistema (DEBONI, 2003)". Na Figura 16, é ilustrado o diagrama de casos de uso do Jogo da Forca.

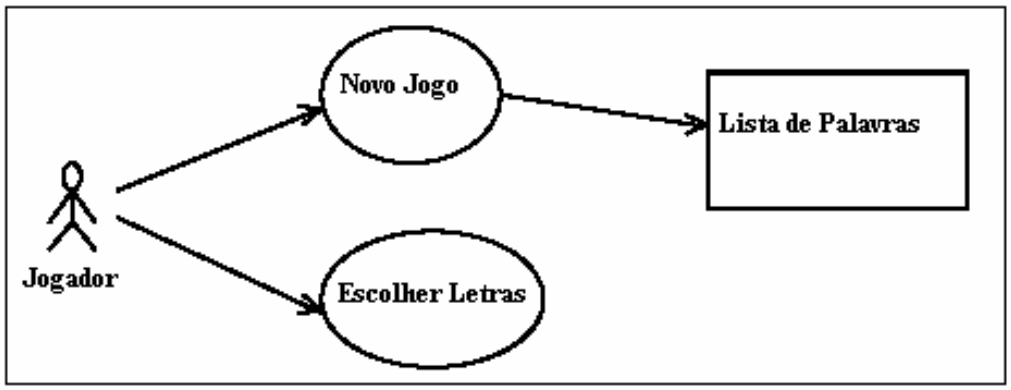

Figura 16 – Diagrama de Casos de Uso do Jogo da Forca.
Fonte: Deboni (2003).

A seguir, descrevem-se os casos de uso do Jogo da Forca extraídos do livro Modelagem Orientada a Objetos com UML (DEBONI, 2003):

Ator: Jogador

Representa os jogadores (usuários) do sistema de Jogo da Forca.

Ator: Lista de Palavras

Representa a lista de palavras armazenadas para alimentar o jogo. É um ator passivo e imutável, que só fornece informações.

Casos de Uso: Novo Jogo

Objetivo Principal

Um novo Jogo da Forca em computador exige que se selecione uma palavra secreta escolhida automaticamente de uma lista de palavras já existente.

Fluxo de Exceção

A lista de palavras não existe ou não pode ser encontrada, o que impede a continuidade do jogo.

Caso de Uso: Escolher Letras

Objetivo Principal

O jogador escolhe letras para tentar acertar a palavra secreta.

Fluxo Alternativo 1

A cada letra errada, o jogador perde uma parte do corpo do boneco. Ao completar a perda de todas as partes do corpo do boneco, o jogador perde a partida.

Fluxo Alternativo 2

A cada letra certa, a palavra vai se desenhando na tela. Ao descobrir todas as letras e descobrir a palavra secreta, o jogador vence a partida.

O detalhamento dos casos de uso é mantido neste exemplo para ilustrar que as funcionalidades especificadas em forma de casos de uso também o podem ser no formato dos casos de teste de uso.

Na Figura 17, é ilustrado o protótipo do Jogo da Forca no computador, que é composto pela forca, pela palavra secreta, pelas letras escolhidas e por um teclado que contém um botão para iniciar um novo jogo e vários botões com as letras do alfabeto que podem ser selecionadas pelo jogador.

Figura 17 - Jogo da Forca no Computador.
Fonte: Deboni (2003).

A seguir, descrevem-se os dois casos de uso do Jogo da Forca no formato de casos de teste de uso, ou seja, os requisitos do sistema Jogo da Forca são detalhados no artefato proposto por este trabalho.

Especificação Dirigida por Casos de Teste

Sistema: Jogo da Forca **Elaborado por:** Simone Beato

Caso de Uso: Novo Jogo **Executado por:** João da Silva

Data da Elaboração: 09/10/2006 **Versão:** 1.00

Objetivo principal do Caso de Uso: Iniciar um novo jogo da forca em computador em que é exigido que se selecione uma palavra secreta escolhida automaticamente de uma lista de palavras já existente.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
1	• A lista de palavras secretas existe; • Palavra secreta escolhida automaticamente = Casa.	Botão selecionado = Novo Jogo.	• Desenhar uma Forca vazia; • Desenha a palavra secreta com 4 espaços indicando as letras.	Palavra de tamanho médio.
2	• A lista de palavras secretas existe; • Palavra secreta escolhida automaticamente = Paralelepípedo.	Botão selecionado = Novo Jogo.	• Desenhar uma Forca vazia; • Desenha a palavra secreta com 14 espaços indicando as letras.	Palavra de tamanho grande.
3	• A lista de palavras secretas existe; • Palavra secreta escolhida automaticamente = Eu.	Botão selecionado = Novo Jogo.	• Desenhar uma Forca vazia; • Desenhar a palavra secreta com 2 espaços indicando as letras.	Palavra de tamanho pequeno.
4	• Lista de palavras secretas vazia.	Botão selecionado = Novo Jogo.	• Msg: Lista de palavras secretas vazia.	Lista de palavras secretas vazia.
5	• Lista de palavras secretas não encontrada.	Botão selecionado = Novo Jogo.	• Msg: Lista de palavras secretas não encontrada.	Lista de palavras secretas não encontrada.

Especificação Dirigida por Casos de Teste

Sistema: Jogo da Forca **Elaborado por:** Simone Beato

Caso de Uso: Escolher letras **Executado por:** João da Silva

Data da Elaboração: 09/10/2006 **Versão:** 1.00

Objetivo principal do Caso de Uso: O jogador escolhe letras para tentar acertar a palavra secreta.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
1	• Palavra secreta = Casa; • Letra selecionada existe na palavra secreta.	Letras selecionadas = A, C e S.	• Escrever letras selecionadas corretamente na palavra; • Escreve letras A, C e S na lista de letras escolhidas; • Msg: Parabéns, você ganhou!; • Desabilitar teclado das letras escolhidas; • Desabilitar teclado das letras, para que o jogador não escolha novas letras com o jogo finalizado.	Palavra acertada, sem nenhuma parte do corpo desenhada na forca.
2	• Palavra secreta = Casa; • Pelo menos uma letra selecionada não existe na palavra secreta.	Letras selecionadas = A, P, C e S.	• Escreve letras selecionadas corretamente na palavra; • Desenha cabeça na forca; • Escreve letras A, P, C e S na lista de letras escolhidas; • Msg: Parabéns, você ganhou!; • Desabilita teclado das letras escolhidas; • Desabilita teclado das letras, para que o jogador não escolha novas letras com o jogo finalizado.	Palavra acertada com pelo menos uma parte do corpo desenhada na forca.

3	• Palavra secreta = Casa; • As letras selecionadas não existem na palavra secreta.	Letras selecionadas = E, P, I, O, N e U.	• Desenha cabeça, tronco, perna esquerda, perna direita, braço esquerdo e braço direito na forca; • Escreve letras E, P, I, O, N e U na lista de letras escolhidas; • Desabilita teclado das letras escolhidas; • Msg: Que pena, você perdeu!; • Desabilita teclado das letras, para que o jogador não escolha novas letras com o jogo finalizado.	Nenhuma letra selecionada existe na palavra.
4	• Palavra secreta = Paralelepípedo; • Algumas letras selecionadas existem na palavra secreta; • Algumas letras selecionadas não existem na palavra secreta.	Letras selecionadas = E, P, I, O, N, L, M, J, A, U e R.	• Escrever letras selecionadas corretamente na palavra; • Desenhar cabeça, tronco, perna esquerda e perna direita; • Escrever letras E, P, I, O, N, L, M, J, A, U e R na lista de letras escolhidas; • Msg: Parabéns, você ganhou!; • Desabilitar teclado das letras escolhidas; • Desabilitar teclado das letras, para que o jogador não escolha novas letras com o jogo finalizado.	Palavra errada, com algumas letras selecionadas que não existem na palavra.
5	• Palavra secreta = Paralelepípedo; • Antes de finalizar o jogo atual, seleciona botão = Novo Jogo.	Letras selecionadas = E, P, I, O, N e L; Botão selecionado = Novo Jogo.	• Escrever letras E, P, I, O, N e L, na lista de letras escolhidas; • Desenhar cabeça na Forca; • Iniciar novo jogo.	Iniciar novo jogo após escolher algumas letras.
6	• Palavra secreta = Avião; • Antes de escolher qualquer letra, seleciona botão = Novo Jogo.	Letras selecionadas = Branco; Botão selecionado = Novo Jogo.	• Iniciar novo jogo.	Iniciar novo jogo sem escolher nenhuma letra.

Analisando a especificação dirigida por casos de teste, observa-se que os casos de teste vão gradativamente desenhando o funcionamento do sistema e complementando os casos de uso, de modo que a mesma informação é apresentada de forma diferente. Essa constatação confirma a possibilidade de que os casos de teste podem ser gerados a partir dos casos de uso e vice-versa.

Analisando-se outros aspectos dos casos de teste de uso, em relação aos casos de uso, observa-se, a partir do caso de aplicação, que os CTU são mais detalhados que os casos de uso, pois os requisitos do sistema estão detalhados com exemplos, o que possibilita uma maior compreensão da funcionalidade do sistema.

Outra comparação que se pode fazer é em relação ao tamanho da especificação nos dois formatos. A especificação em forma de casos de teste de uso é maior que a dos casos de uso, por ser mais completa. Por outro lado, o tempo para elaborar os CTUs é maior que para elaborar os casos de uso, pois, como são exemplos, é necessário detalhá-los, o que é mais demorado do que escrever a especificação de forma genérica, ou seja, escrever os casos de uso.

Já em relação à elaboração dos casos de teste de uso, eles são mais fáceis de elaborar do que os casos de uso, pois aproveita-se o benefício que o usuário especifica com exemplos. Os CTUs também são mais fáceis de serem usados pelos desenvolvedores de sistema do que os casos de uso, pois o desenvolver codifica a funcionalidade, o que é baseado no exemplo descrito no caso de teste de uso.

4.3 Caso de Aplicação: Contratação do produto Termo de Moeda

Termo de Moeda, ou NDF (*Non Deliverable Forward*), é um produto regulamentado pela Câmara de Custódia e Liquidação (CETIP). Em sua forma mais simples, esse produto é negociado pelas instituições financeiras que realizam operações de compra e venda de moeda estrangeira sem entrega física, com preço acordado para uma data futura específica (CETIP, 2007).

O modelo de contexto da contratação do produto Termo de Moeda é representado pelo diagrama de casos de uso e pela descrição textual desse caso de uso. Os outros casos de uso referentes a esse produto estão fora do escopo deste trabalho e, por isso, não são representados e nem detalhados. Na Figura 18, é ilustrado o diagrama de casos de uso da contratação do produto Termo de Moeda.

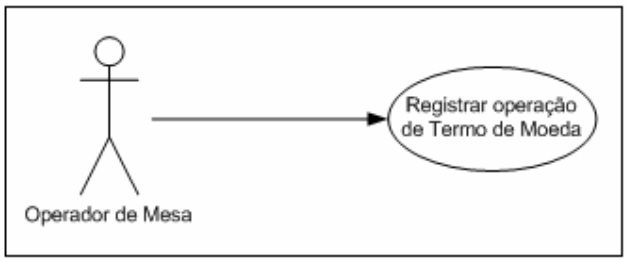

Figura 18 – Diagrama de Casos de Uso da Contratação de Termo de Moeda.

Na Figura 19, apresenta-se ilustrada a tela de registro da contratação do produto de Termo de Moeda no sistema da CETIP.

Baseada nos manuais de Termo de Moeda da CETIP, segue a descrição dos campos da Figura 19 (CETIP, 2007):

- Lançamento do Participante: Código CETIP do participante para quem está sendo lançado o contrato;
- Posição do Participante: Indica a posição assumida pelo participante. Lista pré-definida: Vendedor ou Comprador;
- Contraparte (Conta): Código CETIP da outra parte envolvida no contrato;
- Contrato Global: Indica se o contrato que está sendo registrado é amparado pelo Contrato Global de Derivativos;
- Nº Controle Interno: Número de identificação do lançamento pelo participante ou pela contraparte;
- Valor Base: Valor base do contrato, expresso em quantidades da moeda de referência;
- Data de Operação: Data em que o participante contratou a operação;
- Data de Vencimento: Data de vencimento do contrato, que é livremente pactuada entre os participantes;

- Moeda de Referência: Moeda negociada pelos participantes;

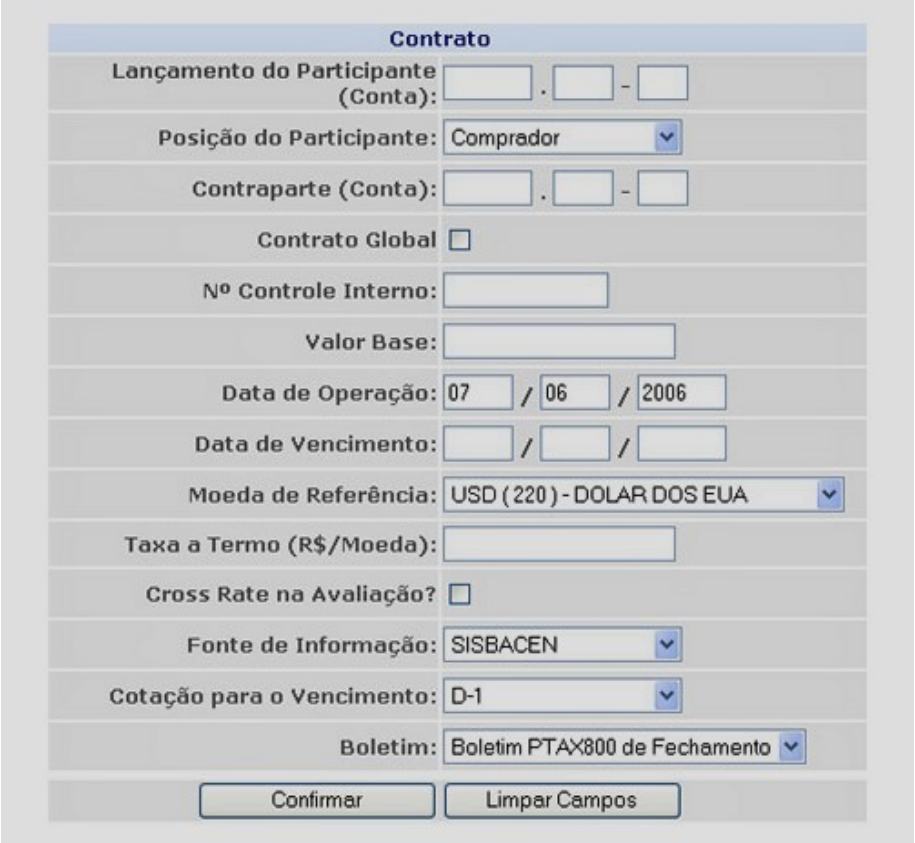

Figura 19 – Registro de uma operação de Termo de Moeda.
Fonte: CETIP.

- Taxa a termo: Valor da taxa de câmbio contratada entre comprador e vendedor;
- Cross Rate na Avaliação: O participante pode ou não optar pela utilização desse campo;
- Fonte de Informação: Indica a fonte a ser utilizada para cotação da moeda. Lista pré-definida: Sisbacen, Feeder, Spot e Sisbacen/Feeder;
- Cotação para o Vencimento: Indica qual a cotação a ser usada no vencimento. Lista pré-definida: D0 (Data atual), D-1 (Data útil anterior à data atual) e D –2 (Data útil de dois dias antes da data atual);
- Boletim: Indica o tipo de boletim de cotação da moeda, que pode ser intermediário ou de fechamento. Lista pré-definida: 11:00h, 12:00h, 15:30h ou 18:00h.

Por ser tratar de um caso de aplicação apenas para exemplificar a utilização da técnica EDCT, o caso de uso da contratação de Termo de Moeda abrange apenas as modalidades

simples desse produto, em que não existem os campos Contrato Global e *Cross Rate* na Avaliação. Os campos Fonte de Informação, Cotação para o Vencimento e Boletim têm, respectivamente, os seguintes valores fixos: SISBACEN, D-1 e Boletim PTAX800 de Fechamento.

Na Tabela 3, está contida a relação dos casos de teste de uso referentes a esse caso de aplicação. Em seguida, são apresentados os cinco primeiros casos de teste de uso desta tabela, no formato proposto pela técnica EDCT. Os demais casos de teste de uso estão detalhados no apêndice B deste trabalho.

Tabela 3 - Relação dos casos de teste de uso – Termo de Moeda.

Seqüencial	Descrição
1	Operação de compra D0
2	Operação de compra D-1
3	Operação de venda D0
4	Operação de venda D-1
5	Operação com data de operação < D-1
6	Valor Base não informado
7	Valor Base igual a zero
8	Valor Base negativo
9	Valor Base excede o valor permitido de R$ 1.000.000.000
10	Valor Base com valor fracionário
11	Moeda de Referência não selecionada
12	Moeda de Referência diferente de Dólar Americano
13	Data de Operação não informada
14	Data de Vencimento é maior que a Data de Operação
15	Data de Vencimento é menor que ou igual à Data de Operação
16	Data de Vencimento é dia útil
17	Data de Vencimento não é dia útil
18	Nº Controle Interno informado como válido
19	Nº Controle Interno não informado
20	Nº Controle Interno informado já existe
21	Nº Controle Interno maior do que o valor permitido

22	Taxa a Termo não informada
23	Quantidade de casas decimais da Taxa a Termo com Moeda de referência = Dólar Americano (6 casas decimais)
24	Quantidade de casas decimais da Taxa a Termo com Moeda de referência = Dólar Americano (mais de 6 casas decimais)
25	Quantidade de casas decimais da Taxa a Termo com Moeda de referência diferente de Dólar Americano (8 casas decimais)
26	Quantidade de casas decimais da Taxa a Termo com Moeda de referência diferente de Dólar Americano (mais de 8 casas decimais)
27	Taxa a Termo com valor negativo
28	Conta do Participante não informada
29	Conta do Participante inválida na CETIP
30	Conta da Contraparte não informada
31	Conta da Contraparte inválida na CETIP
32	Todos os campos preenchidos e botão selecionado = Limpar Campos
33	Alguns campos preenchidos e botão selecionado = Limpar Campos
34	Campos obrigatórios não preenchidos e botão selecionado = Confirmar

Especificação Dirigida por Casos de Teste

Sistema: Contratação de Termo de Moeda **Elaborado por:** Simone Beato

Caso de Uso: Registrar operação de Termo de Moeda **Executado por:** João da Silva

Data da Elaboração: 23/07/2007 **Versão:** 1.00

Objetivo principal do Caso de Uso: Registrar uma operação de compra ou venda do produto Termo de Moeda. Verifica-se a existência das contas CETIP do participante e da contraparte no cadastro da CETIP e, caso as informações estejam corretas, a operação é registrada na CETIP.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
1	• Campos obrigatórios preenchidos; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Comprador; • Contraparte = 39000.000.1; • Nº Controle Interno = 1111; • Valor Base = 5.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Msg: Operação registrada com sucesso.	Operação de compra D0.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
2	• Campos obrigatórios preenchidos; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Comprador; • Contraparte = 39000.000.1; • Nº Controle Interno = 1222; • Valor Base = 5.000; • Data de Operação = 20/07/2007 (Data útil anterior à data atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Operação de compra D-1.
3	• Campos obrigatórios preenchidos; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1333; • Valor Base = 50.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Operação de venda D0.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
4	• Campos obrigatórios preenchidos; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1444; • Valor Base = 50.000; • Data de Operação = 20/07/2007 (Data útil anterior à data atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Operação de venda D-1.
5	• Data de Operação = 19/07/2007; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1449; • Valor Base = 50.000; • Data de Operação = 19/07/2007 (Data de operação < Data útil anterior à data atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Data da operação menor que data útil anterior a data atual.	Operação com data de operação < D-1.

Para este estudo de caso também se observa que os casos de teste de uso são mais completos do que os casos de uso, apesar de este último não estar descrito neste exemplo. No presente caso, os CTUs vão especificando as funcionalidades do sistema, a partir dos casos de teste.

As mesmas comparações apresentadas na seção 4.2 deste trabalho também são aplicáveis a este caso de aplicação, como, por exemplo, o alto nível de detalhamento dos CTUs.

4.4 Validação da Técnica EDCT

Para validar a técnica proposta, elaborou-se um questionário aplicado a um grupo de analistas e de desenvolvedores de sistemas. Na Tabela 4, é ilustrado o conteúdo desse questionário que está dividido em três grupos de perguntas e que visa qualificar a empresa entrevistada e o profissional, bem como avaliar a técnica EDCT. Para cada pergunta usada na pesquisa, apresenta-se o porquê da pergunta, o que define sua pretensão e a hipótese associada a ela.

Tabela 4 – Questionário.

Grupo - Empresa

Pergunta	**Motivo da pergunta**	**Hipótese**
1. A empresa em que você trabalha utiliza algum processo formal de desenvolvimento de *software*?	Verificar se a empresa tem mais interesse na técnica EDCT por utilizar processo formal de desenvolvimento.	Empresas que utilizam processos formais de desenvolvimento podem se interessar mais pela técnica EDCT.
2. A empresa em que você trabalha utiliza UP ou XP como método de desenvolvimento de *software*?	Verificar se a empresa utiliza UP ou XP como método de desenvolvimento de *software*.	Empresas que usam XP já usam uma técnica semelhante e podem ter facilidade em usar a técnica EDCT.

3. Na empresa em que você trabalha, o caso de uso é suficiente para iniciar o desenvolvimento do sistema? Responda apenas se a empresa utilizar o UP.	Verificar se o caso de uso é suficiente para iniciar o desenvolvimento de um sistema.	O caso de uso não é suficiente para iniciar o desenvolvimento do sistema por ser muito genérico, pois, em geral, são necessários exemplos e protótipos para melhorar o entendimento.
4. Na empresa em que você trabalha, é comum reduzir o prazo dos testes em função de atrasos do desenvolvimento?	Verificar a redução de prazo para realizar os testes.	Empresas que reduzem o prazo dos testes podem se beneficiar com a técnica EDCT, pois os casos de teste já foram criados na fase de especificação.
5. Na empresa em que você trabalha, existe uma área específica para elaborar os casos de teste?	Analisar o grau de maturidade da empresa em relação a testes.	Empresas que possuem uma área específica para elaborar os casos de teste vão se interessar mais pela técnica EDCT.
6. Qual a importância que a atividade de teste tem na empresa em que você trabalha? (Escala de 0 a 10)	Verificar se a empresa considera teste como uma etapa importante do processo de desenvolvimento de sistemas.	Empresas que dão importância à atividade de teste têm mais interesse pela técnica EDCT.
7. Assinale a faixa em que se enquadra o número de analista de sistemas, desenvolvedores e analistas de teste na empresa em que você trabalha.	Verificar se o tamanho da empresa favorece a utilização da técnica EDCT.	Empresas grandes se interessam mais que empresas pequenas pela técnica EDCT.
Grupo – Desenvolvedor/Analista de Sistemas		
8. Qual a sua função na empresa em que você trabalha, desenvolvedor ou	Qualificar o profissional.	Desenvolvedores podem ter mais interesse nos CTU, por orientarem o

analista de sistemas?		desenvolvimento.
9. Qual o seu tempo de experiência na função atual?	Qualificar o profissional.	Profissionais com mais tempo de função tendem a ser mais resistentes em relação a mudanças em seu método de trabalho.
Grupo – Técnica EDCT		
10. É possível extrair os requisitos do sistema a partir dos casos de teste de uso?	Verificar se é possível extrair os requisitos do sistema a partir dos casos de teste de uso.	É possível extrair os requisitos a partir dos casos de teste de uso por serem exemplos da funcionalidade do sistema.
11. Comparando os casos de teste de uso com os casos de uso, os requisitos extraídos a partir dos casos de teste de uso são: ( ) Mais completos; ( ) Menos completos; ( ) Iguais; ( ) Não é possível concluir.	Verificar se os casos de teste de uso são mais completos que os casos de uso.	Os casos de teste de uso são mais completos que os casos de uso.
12. Na posição de analista de sistemas, é possível criar os casos de teste de uso com base nas informações resumidas dos casos de uso?	Verificar se a descrição resumida do caso de uso é suficiente para que os analistas de sistemas criem os casos de teste de uso.	A descrição resumida dos casos de uso é suficiente para criar os casos de teste de uso.
13. Comparando a complexidade para criar os casos de teste de uso em relação aos casos de uso, você acha que para criar os casos de teste de uso é: ( ) Mais fácil;	Verificar se é mais fácil criar os casos de teste de uso do que detalhar os casos de uso.	É mais fácil criar os casos de teste de uso do que detalhar os casos de uso.

( ) Mais difícil; ( ) Igual; ( ) Não é possível concluir.		
14. Na posição de desenvolvedor de sistemas, é possível utilizar a técnica EDCT para desenvolver o sistema?	Verificar o grau de dificuldade para desenvolver o sistema a partir dos casos de teste de uso.	A técnica EDCT facilita o desenvolvimento do sistema.
15. Comparando a complexidade para desenvolver o sistema com os casos de teste de uso, em relação a desenvolvê-lo com os casos de uso, você acha que desenvolver o sistema a partir dos casos de teste de uso é: ( ) Mais fácil; ( ) Mais difícil; ( ) Igual; ( ) Não é possível concluir.	Verificar se é mais fácil desenvolver o sistema a partir dos casos de teste de uso.	É mais fácil desenvolver o sistema a partir dos casos de teste de uso.
16. A técnica EDCT incentiva a prática da atividade de testes?	Verificar se a técnica EDCT contribui para incentivar a atividade de testes.	A técnica EDCT incentiva a atividade de testes, pois os casos de teste de uso são usados para testar o sistema e também são usados no desenvolvimento do sistema.
17. Os casos de teste de uso são suficientes para descrever as funcionalidades do sistema?	Verificar a completeza dos casos de teste de uso em relação aos casos de uso.	Os casos de teste de uso podem até ser mais completos que os casos de uso.
18. Os casos de teste de uso são suficientes para testar	Verificar se os casos de teste de uso são suficientes para	Os casos de teste de uso são suficientes para testar os

funcionalmente o sistema?	testar o sistema, ou se é necessário criar outros casos de teste.	requisitos funcionais do sistema.
19. É possível implantar a técnica EDCT no seu processo atual de desenvolvimento de *software*?	Verificar a possibilidade de implantar a técnica EDCT no processo atual de desenvolvimento de *software* utilizado pela empresa em que o entrevistado trabalha,	Se a empresa utilizar UP ou XP, é possível implantar a técnica EDCT.
20. O impacto dos casos de teste de uso na qualidade do *software* produzido: ( ) Melhora a qualidade; ( ) Piora a qualidade; ( ) Nem melhora e nem piora a qualidade; ( ) Não é possível concluir.	Verificar se a utilização da técnica EDCT melhora a qualidade do *software* produzido.	A utilização da técnica EDCT melhora a qualidade do *software* produzido, por enfatizar o teste.
21. O uso da técnica EDCT aumenta o trabalho do Analista de Sistemas?	Verificar se a elaboração dos casos de teste de uso aumenta o trabalho do analista de sistemas.	A elaboração dos casos de teste de uso aumenta o trabalho do analista de sistema.
22. O uso da técnica EDCT facilita o trabalho do Desenvolvedor?	Verificar se por causa dos casos de teste de uso, o trabalho do desenvolvedor é facilitado.	Os casos de teste de uso facilitam o trabalho do desenvolvedor.

4.5 Análise e Discussão dos Resultados da Pesquisa

Esta seção tem como objetivo avaliar os resultados da pesquisa aplicada a um grupo de analistas e de desenvolvedores de sistema. O detalhamento do questionário, enviado a um grupo de profissionais, encontra-se no apêndice B, bem como um resumo da proposta deste trabalho.

O questionário e a pesquisa não têm por objetivo obter uma comparação estatística significativa, mas sim apresentar indícios de aplicabilidade e motivar a análise da técnica proposta. Foram enviados 35 questionários para um grupo de analistas e desenvolvedores de sistemas, de 23 empresas diferentes. Desse total, 13 questionários foram respondidos, e os outros 22 não enviaram nenhum tipo de resposta. Na Tabela 5, é apresentado um resumo do retorno dos questionários.

A primeira pergunta do questionário avalia se as empresas utilizam processos formais de desenvolvimento de *software*, e 10 empresas responderam que utilizam algum processo formal. Das 3 que responderam que não utilizam processo formal de desenvolvimento de *software*, todas utilizam o método de desenvolvimento XP.

Tabela 5 – Retorno dos questionários.

Questionários	Enviados	Respondidos	Não respondidos
Total	35	13	22
Percentual	100%	37%	63%

A segunda pergunta avalia qual é o método de desenvolvimento de *software* utilizado pela empresa e, para esse quesito, 4 empresas entrevistadas utilizam UP, 3 utilizam XP e 6 utilizam outros métodos de desenvolvimento de sistemas, conforme ilustrado no Gráfico 1.

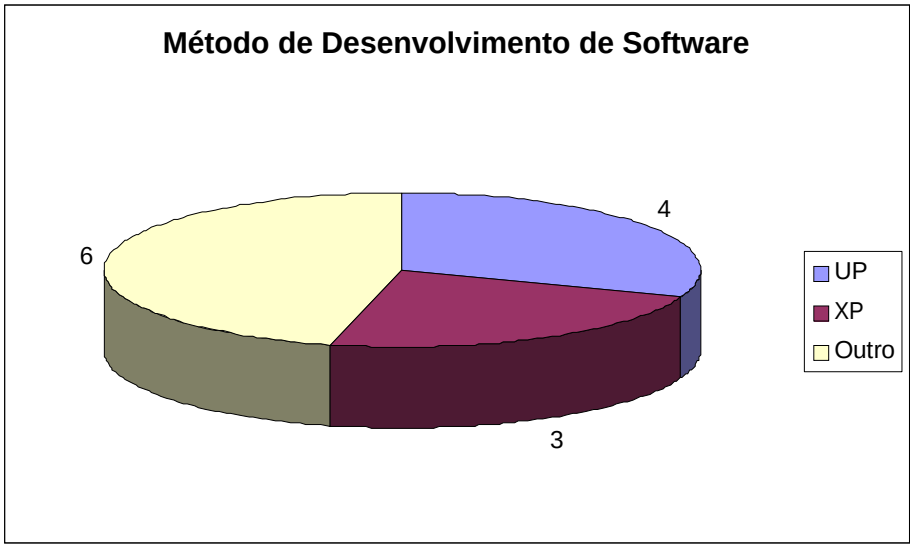

Gráfico 1 – Método de Desenvolvimento de *Software*.

A terceira pergunta foi respondida apenas pelas empresas que utilizam o UP como processo de desenvolvimento de *software* e, destas empresas, 3, das 4 empresas que utilizam UP, responderam que o caso de uso é suficiente para iniciar o desenvolvimento do sistema, contrariando a hipótese apontada na seção 4.4 deste trabalho de que os casos de uso não são suficientes para iniciar o desenvolvimento do sistema.

A quarta pergunta avalia se, na empresa entrevistada, é comum reduzir o prazo dos testes, em função de atrasos na fase de desenvolvimento do sistema, e, destas, 11 responderam que é comum reduzir o prazo dos testes, confirmando a hipótese apontada na seção 4.4 deste trabalho. A técnica EDCT auxilia esse quesito, pois os casos de teste já são criados na fase de especificação e orientam o desenvolvimento do sistema.

A quinta pergunta avalia se as empresas possuem áreas específicas para elaborar os casos de teste e, para esse quesito, a resposta foi bem equilibrada, ou seja, 7 empresas possuem áreas específicas para a elaboração dos casos de teste, contra 6 empresas em que os casos de teste são elaborados pelos próprios analistas de sistemas.

A sexta pergunta avalia a importância que a atividade de teste tem dentro da empresa entrevistada e, destas, 9 atribuíram, em escala de 0 a 10, notas acima de 7, o que demonstra a preocupação das empresas com teste e incentiva propostas que estimulam essa atividade, como, por exemplo, a proposta deste trabalho. No gráfico 2, estão ilustradas todas as notas atribuídas pelos respondentes.

A sétima pergunta avalia o tamanho da empresa em termos de quantidade de funcionários e a distribuição se faz: 7 empresas são de pequeno porte, com até 100 analistas de sistemas, desenvolvedores e analistas de teste; 1 empresa é de médio porte, com 101 a 400 analistas de sistemas, desenvolvedores e analistas de teste; e 5 empresas são de grande porte, com mais de 401 analistas de sistemas, desenvolvedores e analistas de teste. Em relação ao método de desenvolvimento de *software* utilizado e o tamanho da empresa, tem-se: as 3 empresas entrevistadas que utilizam XP são de pequeno porte (até 100 analistas de sistemas, desenvolvedores e analistas de teste) e as 4 empresas entrevistadas que utilizam UP são de grande porte (mais de 401 analistas de sistemas, desenvolvedores e analistas de teste).

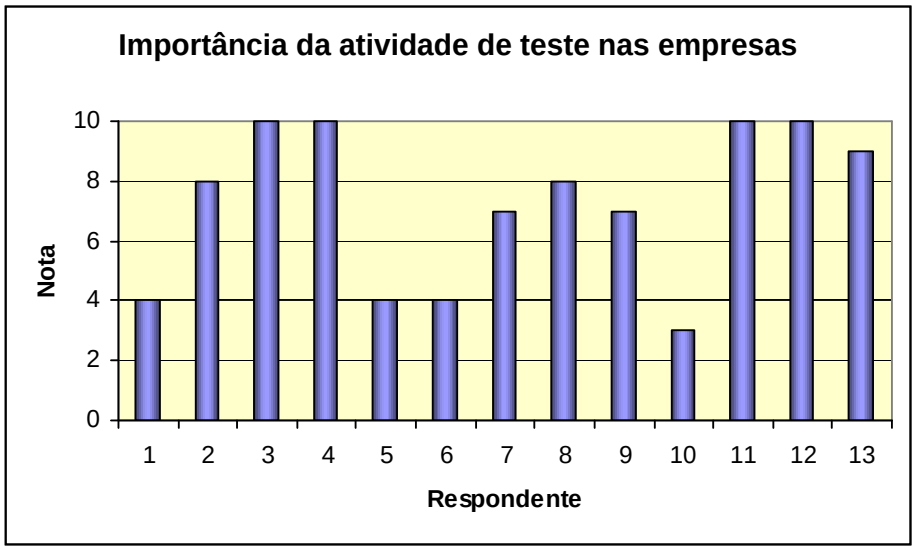

Gráfico 2 – Importância da atividade de teste nas empresas.

A oitava pergunta avalia as funções exercidas pelo entrevistado e 12 dos 13 respondentes exercem a função de analista de sistemas, o que impossibilita a validação da hipótese apresentada na seção 4.4.

A nona pergunta avalia a experiência do entrevistado e 6 dos respondentes têm de 5 a 10 anos de experiência na função atual e 7 têm mais de 10 anos de experiência na função atual, o que demonstra uma elevada maturidade na função e, com isso, uma avaliação mais criteriosa da técnica EDCT. Dos 6, com experiência entre 5 a 10 anos, 5 responderam que é possível implantar a técnica EDCT no seu processo atual de desenvolvimento, enquanto que, dos 7 com mais de 10 anos de experiência, apenas 3 responderam que é possível implantar a técnica EDCT no seu processo atual de desenvolvimento, o que confirma a hipótese apresentada na seção 4.4 de que os profissionais com mais tempo de função podem ser mais resistentes a mudanças de métodos de desenvolvimento.

As próximas perguntas são especifícas da técnica EDCT. A décima pergunta refere-se à possibilidade de extrair os requisitos do sistema, a partir dos casos de teste de uso, e os 13 respondentes afirmaram ser possível extrair os requisitos dos CTUs, confirmando a hipótese apontada na seção 4.4.

A décima primeira pergunta compara os casos de teste de uso com os casos de teste e 10 responderam que os casos de teste de uso são mais completos que os casos de uso e apenas 3 responderam que não é possível concluir, conforme ilustrado no Gráfico 3.

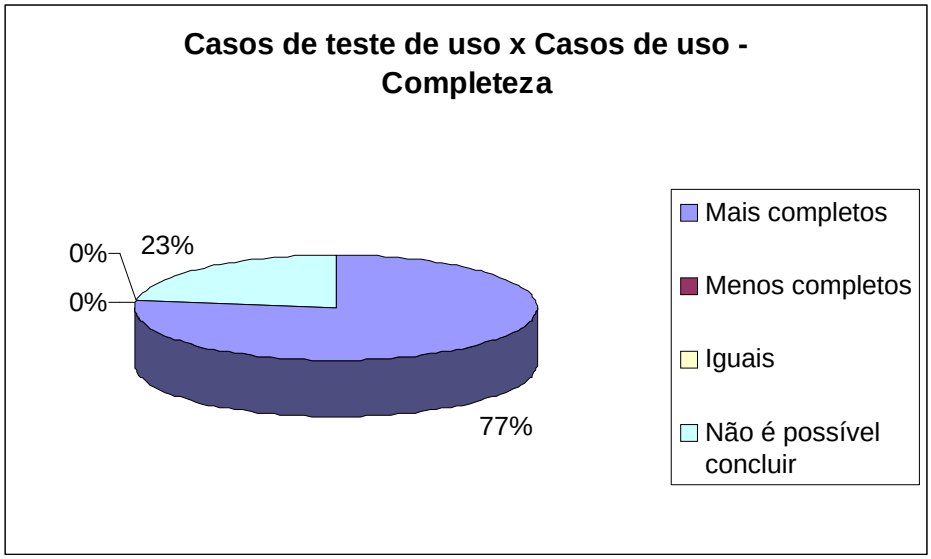

Gráfico 3 – Completeza dos Casos de Teste de Uso x Casos de Uso.

A décima segunda pergunta avalia se, do ponto de visto do analista de sistemas, é possível criar os casos de teste de uso com base nas informações resumidas dos casos de uso e 9 dos 13 respondentes confirmaram que a descrição resumida é suficiente para elaborar os CTU.

A décima terceira pergunta avalia a complexidade para criar os casos de teste de uso em relação à criação dos casos de uso e 4 respondeu que é mais fácil criar os CTU, 5 responderam que é mais difícil e 4 responderam que a complexidade para criar os casos de teste de uso é igual à dos casos de uso, conforme ilustrado no Gráfico 4. As respostas foram, praticamente, distribuídas uniformemente, contrariando a hipótese apresentada na seção 4.4 de que é mais fácil elaborar os CTU do que os casos de uso, o que gera um indício de que talvez exista uma relação entre a habilidade do profissional em elaborar casos de teste e a complexidade para criar os casos de teste de uso.

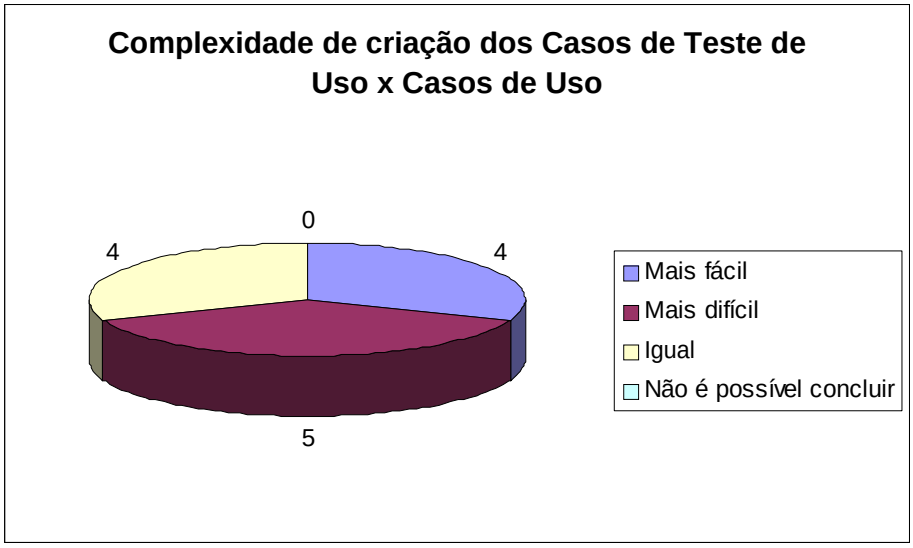

Gráfico 4 – Complexidade de criação dos Casos de Teste de Uso x Casos de Uso.

A décima quarta pergunta avalia se, do ponto de visto do desenvolvedor de sistemas, é possível utilizar os casos de teste de uso para desenvolver o sistema e 12 responderam que é possível desenvolver o sistema a partir da especificação no formato de casos de teste de uso, o que confirma a hipótese apresentada na seção 4.4.

A décima quinta pergunta avalia a complexidade para desenvolver o sistema, a partir dos casos de teste de uso, e 8 responderam que é mais fácil desenvolver o sistema a partir dos casos de teste de uso, 2 responderam que é mais difícil desenvolver a partir dos casos de teste de uso do que a partir dos casos de uso, 2 responderam que a complexidade é igual e 1 respondeu que não é possível concluir. No Gráfico 5, estão ilustradas essas respostas.

A décima sexta pergunta avalia se a técnica EDCT incentiva a prática de testes e 12 responderam que essa técnica incentiva a atividade de testes, confirmando a hipótese apresentada na seção 4.4.

A décima sétima pergunta avalia se os casos de teste de uso são suficientes para descrever as funcionalidades do sistema e 8 responderam que os CTU são suficientes, contra 5 que responderam que os CTU não são suficientes.

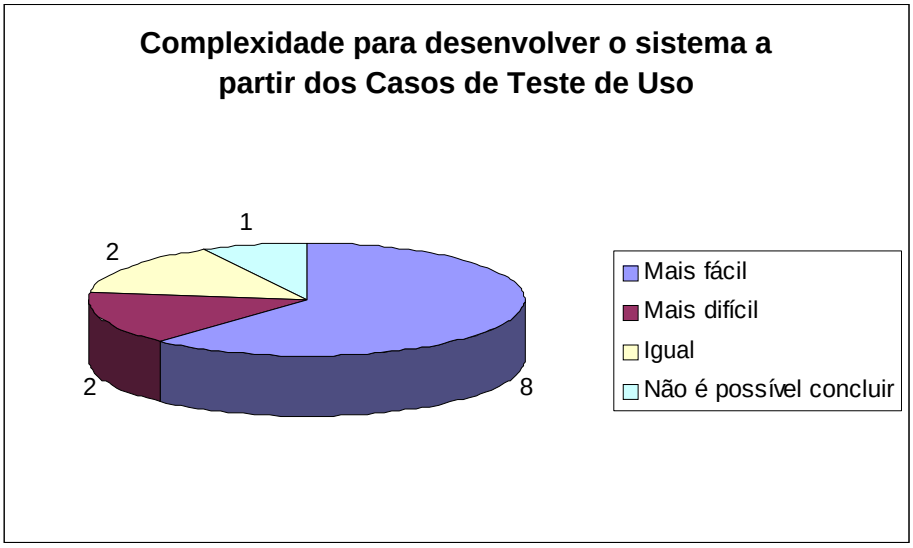

Gráfico 5 – Complexidade para desenvolver o sistema a partir dos CTU.

A décima oitava pergunta avalia se os casos de teste de uso são suficientes para testar funcionalmente o sistema e 11 responderam que são suficientes para testar os requisitos do sistema.

A décima nona pergunta avalia a possibilidade de implantar a técnica EDCT no processo atual de desenvolvimento utilizado pela empresa e 8 responderam que é possível implantá-la no processo atual e 5 responderam que não é possível. Dentre os 8 que responderam que é possível implantá-la no processo atual, todos os respondentes que utilizam UP responderam que é possível implantar a técnica EDCT no processo atual, 2 dos 3 respondentes que utilizam XP também responderam que é possível sua implantação no processo atual e apenas 1 respondente dos que utilizam outros métodos de desenvolvimento responderam que é possível implantar a técnica EDCT no processo atual de desenvolvimento.

A vigésima pergunta avalia o impacto dos casos de teste de uso na qualidade do *software* produzido e 12 responderam que a utilização dos casos de teste de uso melhora a qualidade do *software* produzido e apenas 1 respondeu que não é possível concluir, conforme ilustrado no Gráfico 6.

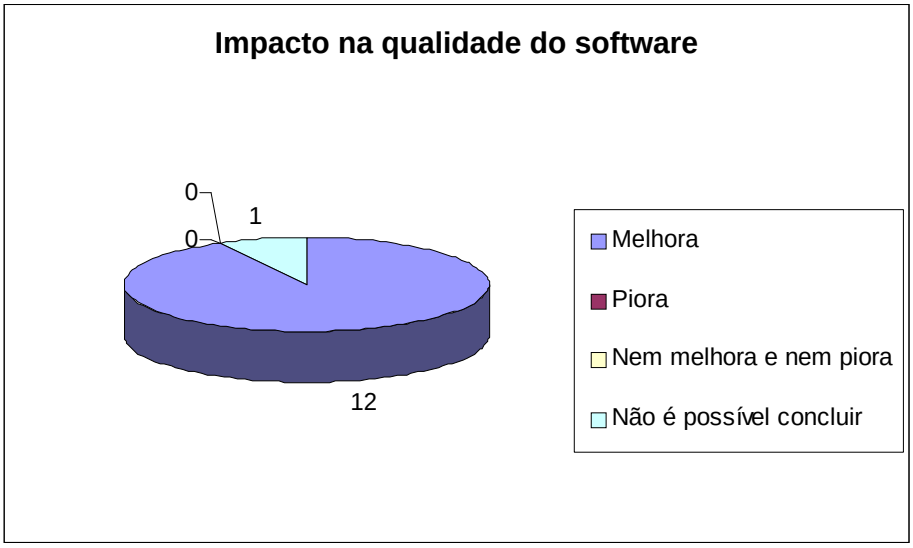

Gráfico 6 – Impacto na qualidade do *software*.

A vigésima primeira pergunta avalia se a utilização da técnica EDCT aumenta o trabalho do analista de sistemas e 9 responderam que aumenta o trabalho do analista de sistemas e 4 responderam que não aumenta o trabalho, confirmando a hipótese apresentada na seção 4.4.

A vigésima segunda, e última, pergunta avalia se a utilização da técnica EDCT facilita o trabalho do desenvolvedor e 11 responderam que facilita o trabalho do desenvolver e apenas 2 responderam que os CTU não facilitam o trabalho do desenvolvedor, o que confirma a hipótese apresentada na seção 4.4 deste trabalho.

4.6 Conclusão

Este capítulo apresenta dois exemplos de aplicação, um do Jogo da Forca e outro da contratação do produto Termo de Moeda, para ilustrar o funcionamento da técnica EDCT. Também são analisados, neste capítulo, os resultados da pesquisa aplicada a um grupo de analistas e de desenvolvedores de sistema.

Os resultados obtidos na pesquisa são satisfatórios, pois das vinte e duas hipóteses apresentadas na seção 4.4, dezessete respostas da pesquisa confirmaram as hipóteses apresentadas na seção citada.

O próximo capítulo apresenta as conclusões e as contribuições deste trabalho, assim como as sugestões para futuras pesquisas relacionadas ao tema Especificação Dirigida por Casos de Testes.

CAPÍTULO 5
CONCLUSÃO

No mercado atual de desenvolvimento de software os índices de defeitos, decorrentes da fase de especificação, alcançam patamares de 60% (ROBERTSON, 2006). Diante dessa realidade pretende-se demonstrar a possibilidade de antecipar a atividade de teste para o início do processo de desenvolvimento de software, incentivando e divulgando tal atividade essencial em qualquer processo de desenvolvimento de software e não apenas como a etapa final do processo.

Sabe-se que a construção de um software envolve uma série de atividades em que as possibilidades para inserir erros humanos são grandes (DEUTSCH, 1979 apud PRESSMAN, 2001). Por isso, de todas as etapas do processo de desenvolvimento, a atividade de teste deve ser realizada com uma atenção especial. Em geral, nos vários modelos de desenvolvimento de sistemas, como no modelo cascata ou nos modelos iterativos, entre eles o Processo Unificado (UP) a elaboração do plano de teste, com os casos de teste funcionais, ocorre em etapa posterior à especificação, no final do desenvolvimento do sistema.

Nesse sentido, este trabalho estuda os modelos de desenvolvimento de sistemas UP e XP, bem como a técnica de desenvolvimento dirigido por testes (TDD). O Processo Unificado (UP) é um método de desenvolvimento de sistemas centrado na arquitetura, iterativo e guiado por casos de uso. Já a Programação Extrema (XP) é baseada em valores, princípios e práticas e utiliza os cartões de história, escritos pelos clientes, para descrever sucintamente uma funcionalidade. No que se refere aos modelos de desenvolvimento de *software* dirigidos por testes, opta-se por dar ênfase ao TDD, por ser o modelo dirigido por testes mais utilizado entre os desenvolvedores de sistemas.

A proposta de substituir a especificação de requisitos de sistema, baseado em casos de uso, pela a atividade de elaboração dos casos de teste, denomina-se Especificação Dirigida por Casos de Teste (EDCT).

Essa técnica, Especificação Dirigida por Casos de Teste (EDCT), propõe a utilização dos casos de teste de uso (CTU), junção dos casos de uso com os casos de teste, para detalhar os requisitos de um sistema. Os casos de teste de uso são elaborados a partir da descrição

resumida da funcionalidade, o que pode ocorrer a partir dos casos de uso – modelo UP, ou a partir dos cartões de história – modelo XP.

Com a utilização da técnica EDCT, é possível executar a atividade de teste no início do processo de desenvolvimento de software, o que permite identificar erros mais cedo, quando a solução é mais simples, e, provavelmente, com redução dos custos de retrabalho. A utilização dessa técnica possibilita diminuir o percentual, expressivo, de 60% dos defeitos normalmente encontrados na fase de especificação (ROBERTSON, 2006), por incentivar a prática de testes por meio dos seus casos de teste de uso.

A técnica EDCT também propõe a valorização da atividade de teste, assim como já ocorre no XP, pois antecipa o teste para a fase de especificação do sistema, o qual dirige o desenvolvimento do sistema ao longo do processo de desenvolvimento de um sistema.

Ressalte-se que a técnica EDCT pode ser utilizada independentemente do modelo de desenvolvimento de sistemas UP ou XP, e este trabalho apresenta as adaptações que se fazem necessárias para utilizá-lo nos modelos citados, adaptações como a eliminação de atividades. Ainda que o XP já pratique o TDD, acredita-se que essa técnica permite que equipes de analistas de sistemas, acostumada com o modelo de casos de uso, integrem-se de modo mais fácil, com equipes de desenvolvedores habituados com o TDD e o XP. O EDCT também pode ser utilizado pelos desenvolvedores que tiverem interesse em migrar do modelo UP para o modelo XP, por estimular o conceito de desenvolvimento dirigido por testes.

Além disso, a adaptação da técnica EDCT ao modelo XP pode facilitar a utilização do conceito de fábricas de *software* para o desenvolvimento do sistema, visto que se pode especificar o sistema com os casos de teste de uso e entregá-los para uma fábrica de *software* desenvolver utilizando o modelo XP.

Para verificar a proposta apresenta-se dois casos de aplicação e uma pesquisa com analistas de sistemas e desenvolvedores para colher impressões a respeito da técnica sugerida.

A pesquisa realizada com um grupo de analistas e desenvolvedores de sistemas, a qual confirma 17 das 21 hipóteses apresentadas neste trabalho. Dentre as hipóteses confirmadas, podem-se citar: a possibilidade de extrair os requisitos do sistema a partir dos casos de teste

de uso e a possibilidade de o desenvolvedor de sistemas utilizar os CTU para desenvolver o sistema. Entretanto o resultado também contraria outras proposições, como a complexidade para criar os casos de teste de uso, em que um percentual um pouco maior analisa como sendo mais complexo criar os CTU do que os casos de uso. No geral, o resultado da pesquisa é satisfatório, pois fornece subsídios para a afirmação de que desenvolvimento dirigido por testes é a nova tendência do mercado de desenvolvimento de *software*.

Por meio de dois casos de aplicação apresentados neste trabalho, observa-se que é possível substituir a especificação dos requisitos pela especificação baseada nos casos de teste de uso, embora sejam necessárias adaptações nos modelos UP e XP. O resultado da pesquisa aponta que todos os pesquisados afirmam que é possível extrair os requisitos a partir dos CTUs, e 8 dos 13 pesquisados afirmam que os CTUs são suficientes para descrever os requisitos do sistema, confirmando a hipótese.

Outra proposta da técnica EDCT é a de reforçar a importância da atividade de teste e para esta proposição, a décima sexta pergunta da pesquisa indica que a técnica EDCT atinge esse objetivo, pois 12 dos 13 pesquisados afirmam que a técnica EDCT incentiva a prática da atividade de teste.

Uma limitação da técnica EDCT é a dependência da criação dos CTU em relação à disponibilidade de um protótipo do sistema, pois se faz necessário ter uma idéia precisa de como o sistema deve ser usado para poder criar os casos de teste. Sem uma definição do protótipo, é mais difícil elaborar os casos de teste de uso.

Outra limitação da técnica EDCT é em relação à habilidade dos analistas de sistemas para criar os casos de teste de uso, visto que, se esses profissionais não estiverem familiarizados com os conceitos e com a prática para elaborar os casos de teste, torna-se mais difícil especificar o sistema com a técnica EDCT do que especificá-lo com casos de uso ou com cartões de histórias.

Acredita-se que é necessária uma mudança de cultura para os analistas de sistemas e desenvolvedores, principalmente para aqueles que não estão habituados a utilizar o TDD, para que eles possam, respectivamente, especificar e desenvolver o sistema a partir de exemplos, ou seja, a partir de casos de teste. Essa mudança não é tão fácil de implementar, podendo

enfrentar resistência por parte desses profissionais, o que foi confirmado com o questionário uma vez que dos 7 profissionais com mais de 10 anos de tempo de função apenas 3 dos pesquisados responderam que é possível implantar a técnica EDCT, contra 4 que responderam que não é possível implementar essa técnica. Enquanto que, analisando-se os 6 profissionais que têm de 5 a 10 anos de experiência, 5 dos 6 responderam que é possível implantar a técnica EDCT, contra, apenas, 1 que respondeu não ser possível implantá-la.

5.1 Contribuições

A técnica EDCT contribui para a simplificação do processo de desenvolvimento de sistemas, por propor a substituição da atividade de especificação de requisitos pela atividade de elaboração dos casos de teste de uso, o que permite diminuir tempo e recursos do processo. Tempo, porque, a nova atividade de elaboração dos casos de teste de uso elimina outras atividades como: detalhamento dos casos de uso e projeto do teste para o modelo UP. Recursos, por facilitar a atividade de elaboração do teste de aceitação do modelo XP.

Há que se ressaltar, outra contribuição, que é disponibilizar para os desenvolvedores de sistemas um conjunto de casos de teste que, além de serem empregados para testar as funcionalidades do sistema, são, acima de tudo, aplicados para especificar o sistema e dirigir o desenvolvedor durante todas as etapas da construção.

5.2 Sugestões para Futuras Pesquisas

Este trabalho apresenta o conceito da técnica Especificação Dirigida por Casos de Testes (EDCT) e, embora existam indícios coletados, durante a pesquisa realizada, de que esta técnica pode trazer benefícios para o desenvolvimento de sistemas, espera-se que trabalhos futuros possam comprová-los com mais eficácia. Assim sendo, como sugestão, estão descritos alguns estudos que podem ser desenvolvidos para ampliar e enriquecer as suas conclusões.

Uma primeira possibilidade é realizar um estudo de caso, ou experimento controlado, utilizando a técnica de Especificação Dirigida por Casos de Teste, para coletar informações mais específicas referentes a, por exemplo, compreensibilidade e completeza dos casos de teste de uso e aumento da qualidade do *software* produzido.

Outra possibilidade é aplicar o questionário, apresentado no apêndice B desta dissertação, para uma amostra significativa de profissionais da área de Tecnologia da Informação e, em seguida, realizar a análise estatística desse resultado.

Outra sugestão é expandir a proposta deste trabalho para atender aos requisitos não funcionais do sistema, ou seja, criar casos de teste de uso para testar, por exemplo, o desempenho do sistema, integrando ao CTU outros tipos de teste, como o teste de desempenho e teste de usabilidade.

Pode-se realizar, ainda, um estudo de caso aplicando a técnica EDCT em um projeto que seja desenvolvido com o modelo XP e que trabalhe os casos de teste de uso como especificação a ser entregue a uma fábrica de *software*, ou seja, os artefatos de especificação (cartões de história + CTU) são entregues para a fábrica de *software*, e essa realiza o desenvolvimento seguindo as práticas do XP.

Finalmente, uma outra proposta para trabalho futuro é desenvolver uma ferramenta para a automação dos casos de teste de uso, semelhante às ferramentas da família xUnit, o que permite facilitar a utilização da técnica EDCT, pois os casos de teste de uso passariam de um artefato em texto para teste executável.

REFERÊNCIAS

ALEXANDER, Ian. Misuse cases: use cases with hostile intent; requirements engineering. In: IEEE JOINT INTERNATIONAL CONFERENCE ON IEEE, 2003. **IEEE Software**, v.20, n.1, p.58-66, jan/feb., 2003.

AMBLER, Scott W. **Agile database techniques**. s.l.: Wiley. Disponível em: http://www.agiledata.org/essays/tdd.html (capítulo 11). Acesso em: 21 de nov. de 2004.

AMBLER, Scott W. Test-driven development of relational databases. **IEEE Software**, v.24, n.3, p.37-43, may/jun., 2007.

BECK, Kent. Simple smaltalk testing: with patterns. **Smalltalk Report,** out., 1994. Disponível em: http://www.xprogramming.com/testfram.htm. Acesso em: 22 de ago. de 2006.

BECK, Kent. **Extreme programming explained:** embrace change. Addison Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1999.

BECK, Kent. **Test-driven development:** by example. Addison Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2003.

BECK, Kent; ANDRES, Cynthia. **Extreme programming explained:** embrace change. 2.ed. s.l.: Addison Wesley Professional, 2004.

CETIP – Câmara de Custódia e Liquidação. **Manual de termo de moeda**. Disponível em: http://www.cetip.com.br/manuais_regulamentos_v06/manuais_dos_sistemas/arquivos/TermodeMoedas/TermoMoedas.htm. Acesso em: 26 de mar. de 2007.

COCKBURN, Alistair. **Writing effective use cases.** s.l.: Addison Wesley Professional, 2001.

COHN, Mike. **User stories applied:** for agile software development. s.l.: Addison Wesley Professional, 2004.

DEBONI, José Eduardo Zindel. **Modelagem orientada a objetos com UML**. s.l.: Editora Futura, 2003.

ELLSBERGER, Jan; HOGREFE, Dieter; SARMA, Amardeo. **SDL:** formal object-oriented language for communicating systems**. s.l:** Pearson Education, 1997

ERIC, Emil; GORBANI, Majid; SELENWALL, Andreas. **The rational unified process**. 2004. Disponível em: http://www.idt.mdh.se/kurser/cd5130/msl/2004lp4/reports/RUP.pdf . Acesso em: 22 de fev. de 2005.

FOWLER, Martin. **Refactoring:** improving the design of existing code. s.l.: Addison Wesley Professional, 1999.

GEORGE, Body; WILLIAMS, Laurie. An initial investigation of test driven development in industry. In: ACM SYMPOSIUM ON APPLIED COMPUTING, march 09-12, 2003. Proceedings on the 2003 ACM symposium on Applied compunting, 2003.

GOTTESDIENER, Ellen. Use cases: best practices. **Rational Software,** 2003. Disponível em: http://www.eg.bucknell.edu/~cs475/F04-S05/useCases.pdf. Acesso em: 30 de out. de 2007

HEUMANN, Jim. Generating test cases from use cases. **The Rational Edge,** jun., 2001.

HOWDEN, W.E.; KIM, Ray.; TRAN, Peter. Model-based extreme testing. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY, ISSRE'02, 13., 2002.

IEEE Standard for Software Test Documentation., mar., 1983.

JACOBSON, Ivar. **Object-oriented software engineering:** a use case driven approach. 2.ed. s.l; Addison-Wesley, 1992.

JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. **The unified software development process.** s.l.: Wesley, 1999.

JACOBSON, Ivar. **Use cases:** yesterday, today and tomorrow. Disponível em: http://www.jaczone.com/papers/use_cases-2002-11-26.pdf. Acesso em: 08 de mai de 2007.

JEFFRIES, Ron. **What is exteme programming**, 2001. Disponível em: http://www.xprogramming.com/xpmag/whatisxp.htm. Acesso em: 19 de nov. de 2004.

JEFFRIES, Ron.; MELNIK, Grigori. TDD: the art of fearless programming. **IEEE Software,** v.24, n..3, p.24-30, may/jun., 2007.

KRUCHTEN, Philippe. **The rational unified process:** an introduction. 3.ed. s.l.: Addison-Wesley Professional, 2003.

MARTIN, Robert C.. Professionalism and test-driven development. **IEEE Software,** v.24, n.3, p.32-36, may/jun., 2007.

MYERS, Glenford J.; SANDLER, Corey. **The art of software testing**. 2.ed. s.l.: John Wiley & Sons,2004.

MÜLHER, Matthis. PADBERG, Frank.. **On the economic evaluation of XP projects**. In: EUROPEAM SOFTWARE ENGINEERING CONFERENCE, 9.; ACM SIGSOFT INTERNATIONAL SYMPOSIUM ON FOUNDATIONS OD SOFTWARE ENGINEERING ACM, 11, 2003. Proceedings of the ACM Sigsoft Software Engineering Notes, v. 28, n. 5, p. 168-177, sep., 2003.

PALMER, J. **Traceability, software requirements engineering**. In: THAYER, R.; DORFMAN, M. (eds.), Software Requirements Engineering, s.l.: IEEE Computer Society Press, p. 364-374, 1997.

PATTON, Ron. **Software testing**. s.l.: Sams Publishing, 2001.

PRESSMAN, Roger S.. **Engenharia de software**.5.ed. Tradução de Mônica Maria G. Travieso. Rio de Janeiro: McGraw-Hill, s.d.

PRIESTLEY, Michael. UTT, Mary. **A unified process for software and documentation development**. In: IEEE PROFESSIONAL COMMUNICATION SOCIETY INTERNATIONAL PROFESSIONAL COMMUNICATION CONFERENCE, 2000; ANNUANL ACM INTERNATIONAL CONFERENCE ON COMPUTER DOCUMENTATION TECNOLOGY AND TEAMWORK IEEE SOFTWARE, 18., 2000, Proceedings of the 18th annual ACM international conference on Computer documentation: technology & teamwork, Cambridge, Massachusetts: ACM, 2000.

ROBERTSON, Suzanne; ROBERTSON, James. **Mastering the requirements process**. 2.ed. s.l.: Addison Wesley Professional, 2006.

RUMBAUGH, James. **Unified modeling languague reference manual**. s.l.: Addison Wesley Professional, 1998.

SMITH, Suzanne; STOECKLIN, Sara. What we can learn from extreme programming. **Journal of Computing Sciences in Colleges,** v.17, 2001.

VODDE, Bas; KOSKELA, Lasse. Learning test-driven development by counting lines. **IEEE Software,** v.24, n.3, p.74-79, may/jun., 2007.

WICK, Michael; STEVENSON, Daniel; WAGNER, Paul. **Using testing and unit across the curriculum**. In: SIGCSE TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION ACM, 36., 2005. Proceedings of the 36th SIGCSE technical symposium on Computer science education, s.l.: ACM, 2005.

ZHANG, Y.. Test-driven modeling for model-driven development. **IEEE Software**, v.21, n.5, p.80-86, sept./oct., 2004.

ZUSER, Wolfgang; HEIL, Stefan.; GRECHENIG, Thomas. Software quality development and assurance in RUP, MSF and XP: a comparative study. In: WORKSHOP ON SOFTWARE QUALITY, 3., 2005. Proceedings of the third workshop on Software quality s.l.: ACM, 2005.

APÊNDICE A

Casos de teste de Uso – Termo de Moeda

Este apêndice contém os outros casos de teste de uso, do caso de aplicação de Termo de Moeda (exemplo prático), que são relacionados na Tabela 3, da seção 4.3 - Caso de aplicação: Contratação do produto Termo de Moeda, deste trabalho e que não estão detalhados na citada seção.

A seguir, apresentam-se os outros casos de teste de uso que complementam a especificação da funcionalidade de registrar uma operação de Termo de Moeda.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
6	• Valor Base = Branco; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1555; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Campo Valor Base é obrigatório.	Valor Base não informado.
7	• Valor Base = 0; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1666; • Valor Base = 0; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Valor Base deve ser maior que zero.	Valor Base igual a zero.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
8	• Valor Base = -100.000; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1667; • Valor Base = -100.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Valor Base não pode ser negativo.	Valor Base negativo.
9	• Valor Base = 1.000.000.001; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1668; • Valor Base = 1.000.000.001; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Valor Base excede o limite permitido.	Valor Base excede o valor permitido de R$ 1.000.000.000

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
10	• Valor Base = 10.000,50; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1669; • Valor Base = 10.000,50; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento;	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Valor Base com valor fracionário.
11	• Moeda de Referência = não selecionada; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Comprador; • Contraparte = 39000.000.1; • Nº Controle Interno = 1777; • Valor Base = 5.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Campo Moeda de Referência é obrigatório.	Moeda de Referência não selecionada.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
12	• Moeda de Referência = EURO; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Comprador; • Contraparte = 39000.000.1; • Nº Controle Interno = 1888; • Valor Base = 5.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Moeda de Referência diferente de Dólar Americano.
13	• Data de Operação = Branco; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Comprador; • Contraparte = 39000.000.1; • Nº Controle Interno = 1999; • Valor Base = 5.000; • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Campo Data de Operação é obrigatório.	Data de Operação não informada.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
14	• Data de Vencimento > Data de Operação; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Comprador; • Contraparte = 39000.000.1; • Nº Controle Interno = 1000; • Valor Base = 250.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,95; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Data de Vencimento é maior que a Data de Operação.
15	• Data de Vencimento <= Data de Operação; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Comprador; • Contraparte = 39000.000.1; • Nº Controle Interno = 4321; • Valor Base = 5.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/07/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Data de Vencimento deve ser maior que a Data de Operação.	Data de Vencimento é menor que ou igual à Data de Operação.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
16	• Data de Vencimento = 24/09/2007; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 2340; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 24/09/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Data de Vencimento é dia útil.
17	• Data de Vencimento = 23/09/2007 (dia **não** útil); • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 2341; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/09/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Data de Vencimento não é dia útil.	Data de Vencimento não é dia útil.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
18	• Nº Controle Interno = 2342; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 2342; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 24/09/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Nº Controle Interno informado como válido.
19	• Nº Controle Interno = Branco; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = Branco; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Campo Nº Controle Interno é obrigatório.	Nº Controle Interno não informado.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
20	• Nº Controle Interno = 1234; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1234; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Nº Controle Interno já existe na CETIP.	Nº Controle Interno informado já existe.
21	• Nº Controle Interno = 10000000000; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 10000000000; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Nº de Controle Interno inválido.	Nº Controle Interno maior do que o valor permitido.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
22	• Taxa a Termo = Branco; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1235; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Campo Taxa a Termo é obrigatório.	Taxa a Termo não informada.
23	• Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 1,987654 (6 casas decimais); • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1236; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 1,987654; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Quantidade de casas decimais da Taxa a Termo com Moeda de referência = Dólar Americano (6 casas decimais).

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
24	• Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 1,9876543 (mais de 6 casas decimais); • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1237; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 1,9876543; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Máximo de 6 casas decimais para a Taxa a Termo quando a moeda de referência é Dólar dos EUA.	Quantidade de casas decimais da Taxa a Termo com Moeda de referência = Dólar Americano (mais de 6 casas decimais).
25	• Moeda de Referência diferente de USD(220) – Dólar dos EUA; • Taxa a Termo = 2,12345678 (8 casas decimais); • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1238; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,12345678; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Quantidade de casas decimais da Taxa a Termo com Moeda de referência diferente de Dólar Americano (8 casas decimais).

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
26	• Moeda de Referência diferente de USD(220) – Dólar dos EUA; • Taxa a Termo = 2,123456789 (mais de 8 casas decimais); • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1239; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,123456789; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Máximo de 8 casas decimais para a Taxa a Termo quando a moeda de referência é diferente de Dólar dos EUA.	Quantidade de casas decimais da Taxa a Termo com Moeda de referência diferente de Dólar Americano (mais de 8 casas decimais).
27	• Taxa a Termo = -2,123; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1240; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = -2,123; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Taxa a termo deve ser positiva.	Taxa a Termo com valor negativo.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
28	• Lançamento do Participante = Branco; • Botão selecionado = Confirmar.	• Lançamento do Participante = Branco; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1240; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,75; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Campo Lançamento do Participante (Conta) é obrigatório.	Conta do Participante não informada.
29	• Lançamento do Participante = 99999.999-9; • Botão selecionado = Confirmar.	• Lançamento do Participante = 99999.999-9; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1241; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,75; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Conta do Participante não cadastrada na CETIP.	Conta do Participante inválida na CETIP

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
30	• Contraparte = Branco; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = Branco; • Nº Controle Interno = 1242; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,75; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Campo Contraparte (Conta) é obrigatório.	Conta da Contraparte não informada.
31	• Contraparte = 99999.999-9; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 99999.999-9; • Nº Controle Interno = 1243; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,75; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Conta da Contraparte não cadastrada na CETIP.	Conta da Contraparte inválida na CETIP.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações

32	• Botão selecionado = Limpar Campos.	• Lançamento do Participante = 37011.000.7;
		• Posição do Participante = Vendedor;
		• Contraparte = 39011.000.7;
		• Nº Controle Interno = 1244;
		• Valor Base = 990.000;
		• Data de Operação = 23/07/2007 (Data Atual);
		• Data de Vencimento = 23/10/2007;
		• Moeda de Referência = EURO;
		• Taxa a Termo = 2,75;
		• Fonte de Informação = SISBACEN;
		• Cotação para o Vencimento = D-1;
		• Boletim = Boletim PTAX800 de Fechamento.
			• Campos iniciados com os valores padrão, são eles:
			  • Posição do Participante = Comprador;
			  • Data de Operação = Data Atual.;
			  • Moeda de Referência = USD(220) – Dólar dos EUA;
			  • Fonte de Informação = SISBACEN;
			  • Cotação para o Vencimento = D-1;
			  • Boletim = Boletim PTAX800 de Fechamento;
			• Os demais campos são iniciados com branco.
				Todos os campos preenchidos e botão selecionado = Limpar Campos.

33	• Botão selecionado = Limpar Campos.	• Lançamento do Participante = Branco;
		• Posição do Participante = Comprador;
		• Contraparte = Branco;
		• Nº Controle Interno = Branco;
		• Valor Base = Branco;
		• Data de Operação = 23/07/2007 (Data Atual);
		• Data de Vencimento = Branco;
		• Moeda de Referência = USD(220) – Dólar dos EUA;
		• Taxa a Termo = Branco;
		• Fonte de Informação = SISBACEN;
		• Cotação para o Vencimento = D-1;
		• Boletim = Boletim PTAX800 de Fechamento.
			• Campos iniciados com os valores padrão, são eles:
			  • Posição do Participante = Comprador;
			  • Data de Operação = Data Atual;
			  • Moeda de Referência = USD(220) – Dólar dos EUA;
			  • Fonte de Informação = SISBACEN;
			  • Cotação para o Vencimento = D-1;
			  • Boletim = Boletim PTAX800 de Fechamento;
			• Os demais campos são iniciados com branco.
				Alguns campos preenchidos e botão selecionado = Limpar Campos.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
34	• Botão selecionado = Confirmar; • Lançamento do Participante = Branco; • Contraparte = Branco; • Nº Controle Interno = Branco; • Valor Base = Branco; • Data de Vencimento = Branco; • Taxa a Termo = Branco;	• Posição do Participante = Comprador; • Data de Operação = 23/07/2007 (Data Atual); • Moeda de Referência = USD(220) – Dólar dos EUA; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Campos obrigatórios não preenchidos.	Campos obrigatórios preenchidos e botão selecionado = Confirmar.

APÊNDICE B

Questionário da Pesquisa

Prezado Colega,

Este questionário é parte do meu trabalho de mestrado, cujo tema é “Caso de Teste de Uso: uma contribuição para o desenvolvimento de *software* dirigido por testes”. Desde já, eu agradeço a sua participação, e me comprometo a enviar uma cópia eletrônica da dissertação, quando ela for aprovada.

A proposta deste trabalho é incorporar testes, já na fase de especificação do sistema, propondo um novo artefato chamado Caso de Teste de Uso (CTU), em substituição aos casos de uso de modo que possam ser utilizados para desenvolver o sistema, seguindo o modelo proposto no Desenvolvimento Dirigido por Testes (TDD).

O objetivo deste questionário é obter a sua opinião a respeito desta técnica.. Para isso, apresentam-se dois casos de aplicação da técnica, sendo um com o Jogo da Forca (exemplo didático) e outro com a contratação do produto Termo de Moeda (exemplo prático), e, para ambos os exemplos, exibe-se o protótipo do sistema. Para o exemplo do Jogo da Forca, os casos de uso estão detalhados, com seus fluxos básicos, alternativos e de exceções. Em seguida, detalham-se os casos de teste, ou seja, descreve-se a especificação dessa aplicação no formato de casos de teste de uso (CTU).

Para o exemplo da contratação de Termo de Moeda, produto este negociado no mercado financeiro brasileiro, apresenta-se o caso de uso apenas como uma descrição sucinta da funcionalidade, semelhante ao cartão de história do XP. Em seguida, são exibidos os casos de teste de uso que descrevem com exemplos a especificação do sistema e que podem ser utilizados para testar funcionalmente o sistema.

Após a análise dos dois exemplos de aplicação, solicita-se que o questionário seja respondido, preferencialmente, por analistas ou desenvolvedores de sistema e que seja enviado para sisalgado@uol.com.br até o dia 02 de novembro de 2007.

Obrigada,
Simone Beato

Exemplo: Jogo da Forca

Descrevem-se os casos de uso do Jogo da Forca extraídos do livro “Modelagem Orientada a Objetos com UML” (DEBONI, 2003):

Ator: Jogador

Representa os jogadores (usuários) do sistema de Jogo da Forca.

Ator: Lista de Palavras

Representa a lista de palavras armazenadas para alimentar o jogo. É um ator passivo e imutável que só fornece informações.

Casos de Uso: Novo Jogo

Objetivo Principal

Um novo Jogo da Forca em computador exige que se selecione uma palavra secreta escolhida automaticamente de uma lista de palavras já existente.

Fluxo de Exceção

A lista de palavras não existe ou não pode ser encontrada, o que impede a continuidade do jogo.

Caso de Uso: Escolher Letras

Objetivo Principal

O jogador escolhe letras para tentar acertar a palavra secreta.

Fluxo Alternativo 1

A cada letra errada, o jogador perde uma parte do corpo do boneco. Ao completar a perda de todas as partes do corpo do boneco, o jogador perde a partida.

Fluxo Alternativo 2

A cada letra certa, a palavra vai se desenhando na tela. Ao descobrir todas as letras e descobrir a palavra secreta, o jogador vence a partida.

Na figura a seguir, é ilustrado o protótipo do Jogo da Forca no computador, que é composto pela forca, pela palavra secreta, pelas letras escolhidas e por um teclado que contém um botão para iniciar um novo jogo e vários botões com as letras do alfabeto que podem ser selecionadas pelo jogador.

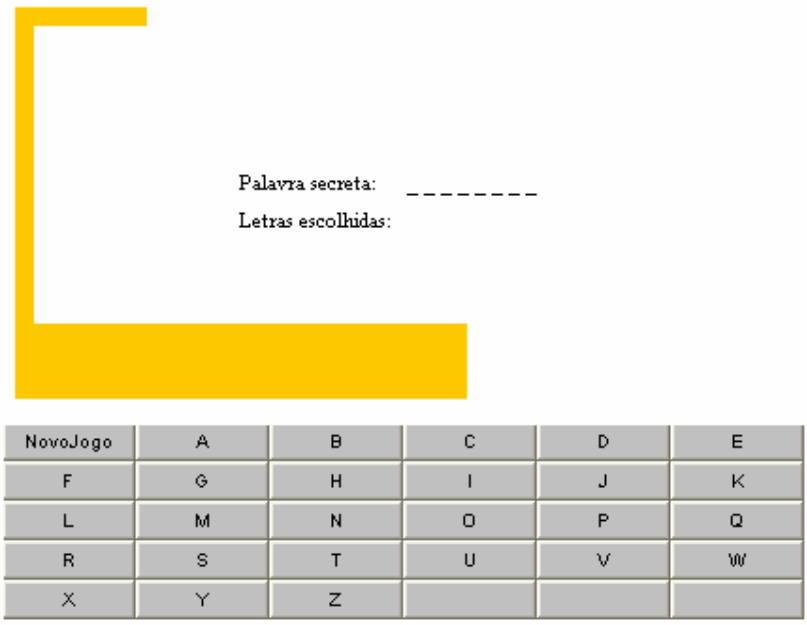

Figura – Jogo da Forca no Computador. (DEBONI, 2003).

A seguir, são exibidos os casos de teste de uso para o exemplo do Jogo da Forca, que estão divididos em dois casos de uso: Novo jogo e Escolher letras.

Especificação Dirigida por Casos de Teste

Sistema: Jogo da Forca **Elaborado por:** Simone Beato

Caso de Uso: Novo Jogo **Executado por:** João da Silva

Data da Elaboração: 09/10/2006 **Versão:** 1.00

Objetivo principal do Caso de Uso: Iniciar um novo jogo da forca em computador em que é exigido que se selecione uma palavra secreta escolhida automaticamente de uma lista de palavras já existente.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
1	• A lista de palavras secretas existe; • Palavra secreta escolhida automaticamente = Casa.	Botão selecionado = Novo Jogo.	• Desenhar uma Forca vazia; • Desenha a palavra secreta com 4 espaços indicando as letras.	Palavra de tamanho médio.
2	• A lista de palavras secretas existe; • Palavra secreta escolhida automaticamente = Paralelepípedo.	Botão selecionado = Novo Jogo.	• Desenhar uma Forca vazia; • Desenha a palavra secreta com 14 espaços indicando as letras.	Palavra de tamanho grande.
3	• A lista de palavras secretas existe; • Palavra secreta escolhida automaticamente = Eu.	Botão selecionado = Novo Jogo.	• Desenhar uma Forca vazia; • Desenhar a palavra secreta com 2 espaços indicando as letras.	Palavra de tamanho pequeno.
4	• Lista de palavras secretas vazia.	Botão selecionado = Novo Jogo.	• Msg: Lista de palavras secretas vazia.	Lista de palavras secretas vazia.
5	• Lista de palavras secretas não encontrada.	Botão selecionado = Novo Jogo.	• Msg: Lista de palavras secretas não encontrada.	Lista de palavras secretas não encontrada.

Especificação Dirigida por Casos de Teste

Sistema: Jogo da Forca **Elaborado por:** Simone Beato

Caso de Uso: Escolher letras **Executado por:** João da Silva

Data da Elaboração: 09/10/2006 **Versão:** 1.00

Objetivo principal do Caso de Uso: O jogador escolhe letras para tentar acertar a palavra secreta.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
1	• Palavra secreta = Casa; • Letra selecionada existe na palavra secreta.	Letras selecionadas = A, C e S.	• Escrever letras selecionadas corretamente na palavra; • Escreve letras A, C e S na lista de letras escolhidas; • Msg: Parabéns, você ganhou!; • Desabilitar teclado das letras escolhidas; • Desabilitar teclado das letras, para que o jogador não escolha novas letras com o jogo finalizado.	Palavra acertada, sem nenhuma parte do corpo desenhada na forca.
2	• Palavra secreta = Casa; • Pelo menos uma letra selecionada não existe na palavra secreta.	Letras selecionadas = A, P, C e S.	• Escreve letras selecionadas corretamente na palavra; • Desenha cabeça na forca; • Escreve letras A, P, C e S na lista de letras escolhidas; • Msg: Parabéns, você ganhou!; • Desabilita teclado das letras escolhidas; • Desabilita teclado das letras, para que o jogador não escolha novas letras com o jogo finalizado.	Palavra acertada com pelo menos uma parte do corpo desenhada na forca.

3	• Palavra secreta = Casa; • As letras selecionadas não existem na palavra secreta.	Letras selecionadas = E, P, I, O, N e U.	• Desenha cabeça, tronco, perna esquerda, perna direita, braço esquerdo e braço direito na forca; • Escreve letras E, P, I, O, N e U na lista de letras escolhidas; • Desabilita teclado das letras escolhidas; • Msg: Que pena, você perdeu!; • Desabilita teclado das letras, para que o jogador não escolha novas letras com o jogo finalizado.	Nenhuma letra selecionada existe na palavra.
4	• Palavra secreta = Paralelepípedo; • Algumas letras selecionadas existem na palavra secreta; • Algumas letras selecionadas não existem na palavra secreta.	Letras selecionadas = E, P, I, O, N, L, M, J, A, U e R.	• Escrever letras selecionadas corretamente na palavra; • Desenhar cabeça, tronco, perna esquerda e perna direita na forca; • Escrever letras E, P, I, O, N, L, M, J, A, U e R na lista de letras escolhidas; • Msg: Parabéns, você ganhou!; • Desabilitar teclado das letras escolhidas; • Desabilitar teclado das letras, para que o jogador não escolha novas letras com o jogo finalizado.	Palavra errada, com algumas letras selecionadas que não existem na palavra.
5	• Palavra secreta = Paralelepípedo; • Antes de finalizar o jogo atual, seleciona botão = Novo Jogo.	Letras selecionadas = E, P, I, O, N e L; Botão selecionado = Novo Jogo.	• Escrever letras E, P, I, O, N e L, na lista de letras escolhidas; • Desenhar cabeça na Forca; • Iniciar novo jogo.	Iniciar novo jogo após escolher algumas letras.
6	• Palavra secreta = Avião; • Antes de escolher qualquer letra, seleciona botão = Novo Jogo.	Letras selecionadas = Branco; Botão selecionado = Novo Jogo.	• Iniciar novo jogo.	Iniciar novo jogo sem escolher nenhuma letra.

Exemplo: Contratação do produto Termo de Moeda

Descreve-se, resumidamente, o caso de uso da contratação do produto Termo de Moeda que tem como objetivo principal permitir que o operador preencha as informações necessárias para registrar a operação de Termo de Moeda. Verifica-se a existência das contas CETIP do participante e da contraparte no cadastro da CETIP e, caso as informações estejam corretas, a operação é registrada na CETIP.

Na figura a seguir, está ilustrada a tela de registro da contratação do produto de Termo de Moeda no sistema da CETIP.

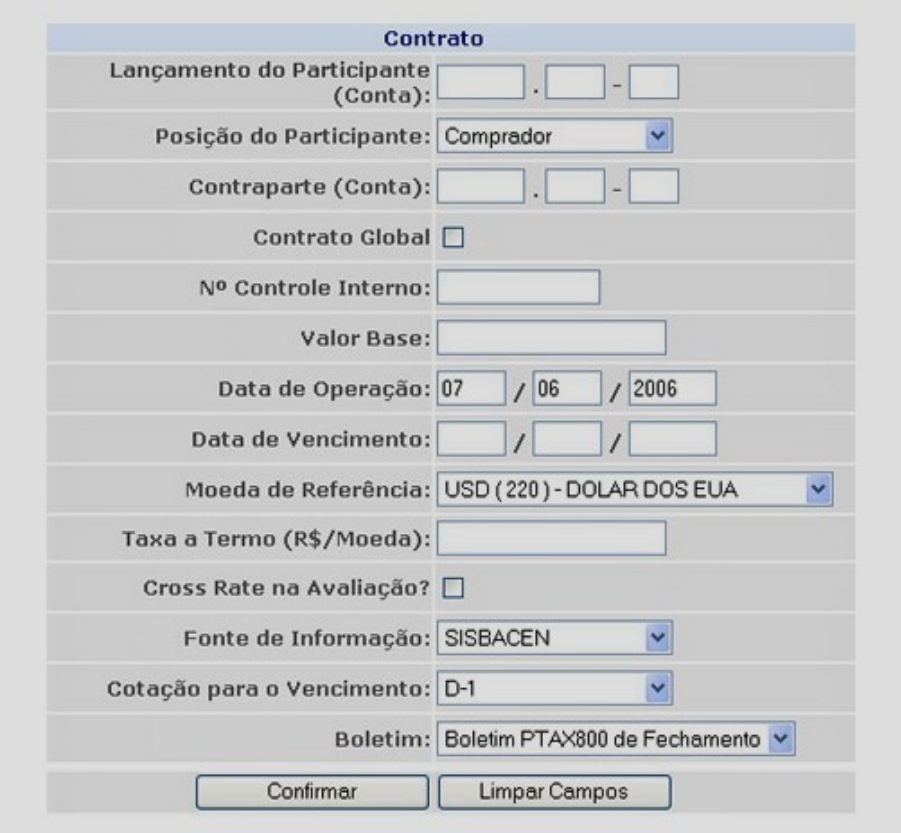

Figura – Registro de uma operação de Termo de Moeda. (Fonte CETIP).

Com base nos manuais de Termo de Moeda da CETIP, segue a descrição dos campos da figura:

- Lançamento do Participante: Código CETIP do participante para quem está sendo lançado o contrato;

- Posição do Participante: Indica a posição assumida pelo participante. Lista pré-definida: Vendedor ou Comprador;
- Contraparte (Conta): Código CETIP da outra parte envolvida no contrato;
- Contrato Global: Indica se o contrato que está sendo registrado é amparado pelo Contrato Global de Derivativos;
- Nº Controle Interno: Número de identificação do lançamento pelo participante ou pela contraparte;
- Valor Base: Valor base do contrato, expresso em quantidades da moeda de referência;
- Data de Operação: Data em que o participante contrata a operação;
- Data de Vencimento: Data de vencimento do contrato, que é livremente pactuada entre os participantes;
- Moeda de Referência: Moeda negociada pelos participantes;
- Taxa a termo: Valor da taxa de câmbio contratada entre comprador e vendedor;
- Cross Rate na Avaliação: O participante pode ou não optar pela utilização desse campo;
- Fonte de Informação: Indica a fonte a ser utilizada para cotação da moeda. Lista pré-definida: Sisbacen, Feeder, Spot e Sisbacen/Feeder;
- Cotação para o Vencimento: Indica qual a cotação a ser usada no vencimento. Lista pré-definida: D0 (Data atual), D-1 (Data útil anterior à data atual) e D –2 (Data útil de dois dias antes da data atual);
- Boletim: Indica o tipo de boletim de cotação da moeda, que pode ser intermediário ou de fechamento. Lista pré-definida: 11:00h, 12:00h, 15:30h ou 18:00h.

Por ser tratar de um caso de aplicação apenas para exemplificar a utilização da técnica EDCT, o caso de uso da contratação do produto Termo de Moeda abrange apenas as modalidades simples desse produto, em que não existem os campos Contrato Global e *Cross Rate* na Avaliação. Os campos Fonte de Informação, Cotação para o Vencimento e Boletim têm, respectivamente, os seguintes valores fixos SISBACEN, D-1 e Boletim PTAX800 de Fechamento.

Especificação Dirigida por Casos de Teste

Sistema: Contratação de Termo de Moeda **Elaborado por:** Simone Beato

Caso de Uso: Registrar operação de Termo de Moeda **Executado por:** João da Silva

Data da Elaboração: 23/07/2007 **Versão:** 1.00

Objetivo principal do Caso de Uso: Registrar uma operação de compra ou venda do produto Termo de Moeda. Verifica-se a existência das contas CETIP do participante e da contraparte no cadastro da CETIP e, caso as informações estejam corretas, a operação é registrada na CETIP.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
1	• Campos obrigatórios preenchidos; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Comprador; • Contraparte = 39000.000.1; • Nº Controle Interno = 1111; • Valor Base = 5.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Msg: Operação registrada com sucesso.	Operação de compra D0.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
2	• Campos obrigatórios preenchidos; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Comprador; • Contraparte = 39000.000.1; • Nº Controle Interno = 1222; • Valor Base = 5.000; • Data de Operação = 20/07/2007 (Data útil anterior à data atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Operação de compra D-1.
3	• Campos obrigatórios preenchidos; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 1333; • Valor Base = 50.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/10/2007; • Moeda de Referência = USD(220) – Dólar dos EUA; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Operação de venda D0.

Seq	Condição	Dados de Entrada	Resultado Esperado	Observações
16	• Data de Vencimento = 24/09/2007; • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 2340; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 24/09/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação registrada na CETIP; • Msg: Operação registrada com sucesso.	Data de Vencimento é dia útil.
17	• Data de Vencimento = 23/09/2007 (dia **não** útil); • Botão selecionado = Confirmar.	• Lançamento do Participante = 37011.000.7; • Posição do Participante = Vendedor; • Contraparte = 39000.000.1; • Nº Controle Interno = 2341; • Valor Base = 990.000; • Data de Operação = 23/07/2007 (Data Atual); • Data de Vencimento = 23/09/2007; • Moeda de Referência = EURO; • Taxa a Termo = 2,50; • Fonte de Informação = SISBACEN; • Cotação para o Vencimento = D-1; • Boletim = Boletim PTAX800 de Fechamento.	• Operação não registrada na CETIP; • Msg: Data de Vencimento não é dia útil.	Data de Vencimento não é dia útil.

QUESTIONÁRIO

Este questionário está dividido em três partes: a primeira refere-se às perguntas relacionadas à empresa em que você trabalha a segunda, às perguntas relacionadas ao profissional entrevistado, e a terceira, às perguntas relacionadas à técnica de especificação dirigida por casos de teste, que utiliza os casos de teste de uso como especificação do sistema.

Parte I - Empresa

Assinale sim ou não, ou escolha uma alternativa para as questões a seguir, que são relacionadas à empresa em que você trabalha.

		Sim	Não
1	A empresa em que você trabalha utiliza algum processo formal de desenvolvimento de *software*?		
2	A empresa em que você trabalha utiliza UP ou XP como método de desenvolvimento de *software*?	( ) UP ( ) XP ( ) Outro método	
3	Na empresa em que você trabalha, o caso de uso é suficiente para iniciar o desenvolvimento do sistema? Responda apenas se a empresa utilizar o UP.		
4	Na empresa em que você trabalha, é comum reduzir o prazo dos testes em função de atrasos do desenvolvimento?		
5	Na empresa em que você trabalha, existe uma área específica para elaborar os casos de teste?		

Informe a escala que corresponde à questão seguinte, sendo que zero significa pouco relevante e dez significa extremamente relevante.

6	Qual a importância que a atividade de teste tem na empresa em que você trabalha?	Escala (0 a 10)
7	Assinale a faixa em que se enquadra o número de analista de sistemas, desenvolvedores e analistas de teste na empresa em que você trabalha?	( ) Até 50 ( ) Entre 51 e 100 ( ) Entre 101 e 200 ( ) Entre 201 e 300 ( ) Entre 301 e 400 ( ) 401 ou mais

Parte II – Desenvolvedor / Analista de Sistemas

Assinale uma alternativa para as questões a seguir que são relacionadas ao profissional entrevistado.

8	Qual a sua função na empresa em que você trabalha, desenvolvedor ou analista de sistemas?	( ) Desenvolvedor ( ) Analista de sistemas
9	Qual o seu tempo de experiência na função atual?	( ) Menos de 1 ano ( ) De 1 a 4 anos ( ) De 5 a 10 anos ( ) Mais de 10 anos

Parte III – Técnica de Especificação Dirigida por Casos de Teste (EDCT)

Assinale sim ou não, ou escolha uma alternativa para as questões seguintes que são relacionadas à técnica de Especificação Dirigida por Casos de Teste.

		Sim	Não
10	É possível extrair os requisitos do sistema a partir dos casos de teste de uso?		
11	Comparando os casos de teste de uso com os casos de uso, os requisitos extraídos a partir dos casos de teste de uso são:	( ) Mais completos ( ) Menos completos ( ) Iguais ( ) Não é possível concluir	
12	Na posição de analista de sistemas, é possível criar os casos de teste de uso com base nas informações resumidas dos casos de uso?		
13	Comparando a complexidade para criar os casos de teste de uso em relação aos casos de uso, você acha que para criar os casos de teste de uso é:	( ) Mais fácil ( ) Mais difícil ( ) Igual ( ) Não é possível concluir	
14	Na posição de desenvolvedor de sistemas, é possível utilizar a técnica EDCT para desenvolver o sistema?		
15	Comparando a complexidade para desenvolver o sistema com os casos de teste de uso, em relação a desenvolvê-lo com os casos de uso, você acha que desenvolver o sistema a partir dos casos de teste de uso é:	( ) Mais fácil ( ) Mais difícil ( ) Igual ( ) Não é possível concluir	
16	A técnica EDCT incentiva a prática da atividade de testes?		
17	Os casos de teste de uso são suficientes para descrever as funcionalidades do sistema?		
18	Os casos de teste de uso são suficientes para testar funcionalmente o sistema?		
19	É possível implantar a técnica EDCT no seu processo atual de desenvolvimento de *software*?		
20	O impacto dos casos de teste de uso na qualidade do *software* produzido:	( ) Melhora a qualidade ( ) Piora a qualidade ( ) Nem melhora e nem piora a qualidade ( ) Não é possível concluir	
21	O uso da técnica EDCT aumenta o trabalho do analista de sistemas?		
22	O uso da técnica EDCT facilita o trabalho do desenvolvedor?		