UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA – ITEC
PROGRAMA DE PÓS-GRADUÇÃO EM ENGENHARIA ELÉTRICA
MESTRADO PROFISSIONAL EM ENGENHARIA ELÉTRICA

PAULO HENRIQUE DE LIMA MACIEL

MODELO PARA ANÁLISE DE NEGOCIAÇÃO EM PROJETOS DE SISTEMAS EM UMA FÁBRICA DE SOFTWARE BASEADO EM PONTOS DE CASO DE USO.

DISSERTAÇÃO DE MESTRADO

BELÉM - PA

2011

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA – ITEC
PROGRAMA DE PÓS-GRADUÇÃO EM ENGENHARIA ELÉTRICA
MESTRADO PROFISSIONAL EM ENGENHARIA ELÉTRICA

MODELO PARA ANÁLISE DE NEGOCIAÇÃO EM PROJETOS DE SISTEMAS EM UMA FÁBRICA DE SOFTWARE BASEADO EM PONTOS DE CASO DE USO.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Instituto de Tecnologia da Universidade Federal do Pará como requisito para obtenção do título de Mestre em Processos Industriais com ênfase em Processos Industriais.

PAULO HENRIQUE DE LIMA MACIEL

ORIENTADOR PROF. DR. ROBERTO CÉLIO LIMÃO DE OLIVEIRA

BELÉM - PA

2011

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA – ITEC
PROGRAMA DE PÓS-GRADUÇÃO EM ENGENHARIA ELÉTRICA
MESTRADO PROFISSIONAL EM ENGENHARIA ELÉTRICA

PAULO HENRIQUE DE LIMA MACIEL

TÍTULO: MODELO PARA ANÁLISE DE NEGOCIAÇÃO EM PROJETOS DE SISTEMAS EM UMA FÁBRICA DE SOFTWARE BASEADO EM PONTOS DE CASO DE USO.

DEFESA DO MESTRADO

Esta Dissertação foi julgada e aprovada para a obtenção do título de Mestre em Processos Industriais na Área de Concentração em Processos Industriais do Programa de Pós-Graduação Stricto Sensu em Engenharia Elétrica da Universidade Federal do Pará – ITEC – UFPA.

Belém-PA, 01 de março de 2011.

_____

Prof. Dr. José Antônio da Silva Souza (UFPA)
Coordenador do CMPPI

BANCA EXAMINADORA

_____ _____

Prof. Dr. Roberto Célio Limão de Oliveira (UFPA) - Orientador

Prof. Dr. Célio Augusto Gomes de Souza (FEQ/UFPA)

_____

Prof. Dr. Emanuel Negrão Macêdo (FEQ/UFPA)

DEDICATÓRIA

À Deus por tudo.

Ao meu pai, Francisco Ponciano Maciel (*In Memoriam*), pelo exemplo que ele foi, que ele é e sempre será.

À minha mãe, Maria Celeny de Lima Maciel, por sua força e dedicação.

À minha esposa, Alessandra Melo Cruz, por estar sempre presente.

Aos meus filhos, Francisco Ponciano Maciel Neto, Cláudio Henrique dos Reis Maciel, Laís Celeny Cruz Maciel e Sônia Beatriz Cruz Maciel, pela inspiração.

Aos meus tios Olinto Ponciano Lima e Estelita Pereira Duarte Ponciano Lima, e aos meus primos Adriano Duarte Ponciano Lima e Altiere Duarte Ponciano Lima, por terem estendido a mão no momento que precisei.

AGRADECIMENTOS

À Universidade Federal do Pará - UFPA.

Ao Instituto de Tecnologia e Educação Galileo da Amazônia – ITEGAM.

Aos Professores M.Sc. Jandecy Cabral Leite e Dr. Roberto Célio Limão de Oliveira, pela condução e orientação com excelência durante o curso.

Ao Professor Dr. Francisco Beraldo Herrera Fernandez, PhD pelo apoio dado na *Conferencia Internacional de Ciencias Computacionales e Informáticas - Cuba* (**CICCI**' 2011).

À minha família.

Aos colegas e professores do curso.

A todos que, direta e indiretamente, contribuíram para realização deste trabalho.

EPÍGRAFE

“Como todas as coisas contribuíram para o seu crescimento, você deveria incluir todas as coisas em seu agradecimento.”

Wallace Wattles (1860-1911)

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE EQUAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

COCOMO: Constructive Cost Model

DAR: Diagrama de Análise Robusta

EV: Eventos do Sistema

FCA: Fatores de Complexidade Ambiental

FCT: Fatores de Complexidade Técnica

IFPUG: International Function Point User Group

ISO: International Organization for Standardization PMBoK - Project Management Body of Knowledge PMI - Project Management Institute

LOC: Linha de Código

PCU: Pontos de Caso de Uso

PCUA: Pontos de Caso de Uso Ajustados

PCUNA: Pontos de Caso de Uso Não Ajustados

PF: Pontos de Função

PMIAM: Project Management Institute do Amazonas BFPUG - Brazilian Function Point User Group

RS: Respostas do Sistema

RUP: Rational Unified Process (ou Processo Unificado Racional)

TPNAA: Total de Peso Não Ajustado dos Atores

TPNAUC: Total de Peso Não Ajustado dos Use Cases

UML: Unified Modeling Language

XP: Extreme Programming

SUMÁRIO

RESUMO

A Modelagem de Sistemas vêm sendo cada vez mais aplicada nos meios de produção para as mais diversas finalidades, incluindo a área de Projeto de Sistemas, com o intuito de definir o número de pessoas na equipe, analisar o esforço, o tamanho do software e os custos totais do projeto. Este trabalho tem por finalidade desenvolver um modelo de apoio à análise baseado em Pontos de Caso de Uso (PCU). Para isso, utiliza-se de vários métodos de pesquisa entre elas a pesquisa exploratória e de laboratório para criar um modelo de apoio para a análise.

Palavras-chaves: Engenharia de Software, Modelagem de Sistemas e Projeto de Sistemas.

ABSTRACT

The systems modeling are being increasingly applied in the means of production for many different purposes, including the area of System Design, in order to set the number of people on the team, analyze the effort, software size and total project costs. This study aims to develop a model to support analysis based on Use Case Points (PCU). For this, we use various research methods including the exploratory research laboratory to create a model of support for analysis.

Keywords: Software Engineering, Systems Modeling and Systems Design.

CAPÍTULO 1

1.1 INTRODUÇÃO

Devidos os constantes avanços tecnológicos, na área de sistemas computacionais, as empresas buscam eficiência e adaptação no mercado através dos serviços e consultorias prestadas pelas fábricas de software. Tais fábricas, atualmente possuem estratégias diversificadas em seus modelos de negócios, pois se em alguns momentos estão desenvolvendo um software específico para algum cliente em outro momento poderão estar desenvolvendo uma aplicação que estará disponível no mercado para ser adquirido por qualquer cliente.

Independente de o software pertencer a um cliente específico ou ao mercado em geral é necessário que tal sistema possua um preço justo e competitivo. Porém quando é um cliente específico além do fator custo outros fatores pesam na negociação, como por exemplo o tempo de desenvolvimento e entrega do produto, a garantia, a qualidade e outros. Por isso o momento da negociação é uma das partes mais importantes do processo de análise.

Atualmente vários modelos de negócios estão disponíveis nos meios corporativos, porém cada cliente possui seu perfil, suas características e suas necessidades, principalmente quando o produto em questão é um software que precisa ser adequado às necessidades do cliente.

Estudos realizados na área de projetos de software mostram que o mercado ainda possui muitas carências, pois segundo PUTINI (*apud* Mertins, 2004) “26% dos projetos de software são bem sucedidos, consequentemente 74% deles são falhos em algum ponto”. De acordo com o autor supracitado os projetos de software que são concluídos com eficácia possuem alguns aspectos em comum e um desses aspectos ocorre quando os “requisitos realmente refletem as necessidades do cliente”.

O desenvolvedor e o cliente devem procurar alcançar, como objetivo, um

ponto comum entre eles durante a fase de negociação. Desta forma o resultado da negociação será satisfatório para ambos, aumentando assim as possibilidades do projeto de software ser concluído com sucesso.

Para evitar complicações durante o processo de negociação o gerente de projetos pode utilizar algumas métricas de tamanho de software para auxiliá-lo, como por exemplo, Pontos de Caso de Uso (PCU), Constructive Cost Model (COCOMO), Pontos de Função (PF) e Linhas de Código (LOC).

1.2 JUSTIFICATIVA DO ESTUDO

Atualmente o fluxo de informações cresce de forma exponencial, devido principalmente pelo avanço da Internet, por isso as empresas vem se atualizando para acompanhar as mudanças do mercado. Essas mudanças mostram a necessidade de gerenciar as informações de forma mais objetiva, pois essas informações podem determinar o fracasso ou sucesso de uma negociação. De acordo com JUNIOR (2007) “o mundo dos negócios é hoje extremamente rápido e competitivo. Decisões oportunas constituem a alma de qualquer negócio. Elas devem ser rápidas e efetivas”.

Segundo PAULA (2007) há dois tipos de negociações, “a competitiva e a comparativa”. A competitiva é identificada como a ganha/perde onde “os resultados de uma parte são prejudicados em detrimento da outra” enquanto a comparativa é classificada como ganha/ganha onde os resultados são “positivos para ambos os lados”.

É importante que o gerente de projetos focalize seus esforços no modelo ganha/ganha, pois de acordo com OGERENTE (2007) a negociação ganha/perde “não trará benefícios de longo prazo para empresa, construção de parcerias, bons relacionamentos e redução de risco”, fazendo com que a parceria não seja duradoura, desta forma o desenvolvedor ou o cliente irão procurar alternativas no futuro.

Na negociação de softwares não é diferente, os gerentes precisam determinar em um tempo aceitável o escopo do projeto, para isso existem métricas de tamanho de softwares que auxiliam o gerente na confecção do mesmo, entre elas a de Pontos de Caso de Uso (PCU) que será explanada e utilizada neste trabalho.

Desta forma, o objetivo principal deste trabalho é mostrar como a métrica de Pontos de Caso de Uso (PCU) é uma importante ferramenta de apoio à negociação de projetos de software, essa métrica visa auxiliar as empresas de software no momento da negociação com os clientes. Através dela o gerente de projetos poderá mensurar o tamanho do projeto em tempo real.

1.3 ESPECIFICAÇÃO DO PROBLEMA

Nem sempre durante uma negociação ambas as partes ficam satisfeitas. Uma alternativa apresentada por um gerente de software na maioria das vezes não é aceita pelo cliente. A falta de um Modelo torna a negociação mais difícil e imprecisa, uma vez que o gerente precisa estimar valores de custo e prazo de forma rápida e segura, fato que não acontece utilizando métodos manuais.

Foi observado que muitos gerentes de software não possuem ferramentas adequadas, tanto para realizar negociação de escopo com seus clientes quanto para aplicar técnicas de estimativa de software. Partindo desse princípio, os gerentes acabam negociando de forma inadequada e utilizando as técnicas de forma manual, que por consequência, geram vários problemas para a empresa, tais como: atraso na entrega do sistema, geração de custos mais elevados, estimativas irreais, entre outros.

Diante deste problema, acredita-se que uma a definição e uso de uma métrica apropriada dará apoio ao gerente, tanto no momento da negociação, quanto na definição dos custos e prazos do software.

1.4 OBJETIVO GERAL

Mostrar a necessidade do uso de métricas durante o processo de negociação em um projetos de sistemas, baseados em Pontos de Caso de Uso (PCU). A escolha dessa métrica deve-se a pouca literatura existente sobre a mesma e também por ser uma métrica com recursos intermediários, ou seja, não é tão exigente, em relação às técnicas utilizadas por outras métricas, porém, também não é um método tão simplificado.

FREIRE (2007) faz em seu artigo uma breve introdução à técnica de estimativa PCU, explica o seu funcionamento e aplica a técnica. HEIMBERG e GRAHL (2007) definem a estimativa de tamanho como uma das métricas de software mais utilizadas, pois a partir da dimensão é possível definir esforço, prazo para o desenvolvimento do projeto. Após isso apresentam um estudo de caso de uma empresa de software onde foi utilizada a técnica PCU.

1.5 OBJETIVO ESPECÍFICO

Desenvolver um modelo, baseado nas métricas dos Pontos de Casos de Uso, para auxiliar os profissionais da área de TI durante o processo de análise de sistemas.

Portanto, esta dissertação visa ratificar, através desde modelo, que a utilização de estimativa de Pontos de Caso de Uso (PCU) auxiliará na negociação de escopo de projetos de software.

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

Esse capítulo apresenta os aspectos mais importantes desse trabalho. Primeiramente, as fábricas de software e o conceito de negociação são explanados junto com suas características e evolução. Depois são apresentadas algumas métricas de tamanho de software destacando a sua importância, funcionalidade e o seu surgimento. Por fim é apresentada a métrica PCU.

2.1 NEGOCIAÇÃO

Com a constante evolução da engenharia de software e as mudanças ocorridas nos processos de desenvolvimento do mesmo, surgiram em meados dos anos 70 as fábricas de software que conforme CASTOR (2007) seria uma “solução para alcançar maior produtividade e menor custo na produção de sistemas de software”.

Na visão da MSW (2007), as fábricas de software vieram para “formalizar todos os processos de desenvolvimento de software trabalhando em uma linha de produção com tarefas definidas para cada profissional”. CASTOR (2007) afirma que as fábricas de software utilizam “técnicas e conceitos provenientes da indústria de manufatura e linhas de produção”, os quais eram baseados em normas da International Organization for Standardization (ISO).

As indústrias em geral possuem processos contínuos, os quais, segundo a tradução do Project Management Body of Knowledge (PMBoK) (2004), não podem ser considerados projetos, pois as características primárias para um projeto são: “tempo determinado, produtos serviços ou resultados exclusivos e elaboração progressiva”. Portanto, um projeto caracteriza-se por um esforço temporário direcionado para a criação de um produto, serviço ou resultado. Tais características não são encontradas nesses processos fabris repetitivos.

Entretanto, AAEN, BOTTCHER e MATHIASSEN (*apud* Castor, 2007) advogam que as fábricas de software possuem uma estratégia que trabalha com dois tipos de serviço: o “aumento de qualidade e produtividade no desenvolvimento e manutenção” de softwares. O primeiro tipo de serviço será comentado no próximo parágrafo, já o segundo tipo é semelhante aquele desenvolvido nas indústrias, pois não possui um fim definido e nem resultados exclusivos, visando apenas o aperfeiçoamento ou correção de um produto já existente.

O primeiro tipo de serviço refere-se à criação de novos produtos, sejam personalizados ou genéricos, chamados de projetos de software. Este produto atende todas as características de um projeto, por isso as fábricas de software tendem a utilizar práticas existentes no PMBoK, o qual segundo ÁVILA (2006), é um “guia do conjunto de conhecimentos em gerenciamento de projetos”.

O PMBoK foi desenvolvido por um instituto chamado Project Management Institute (PMI) que segundo o Project Management Institute do Amazonas (PMIAM) (2007) é a “principal associação mundial sem fins lucrativos em gerenciamento de projetos”. Em 1993, o PMI publicou na Internet um guia englobando todas as áreas do conhecimento que regem as regras do gerenciamento de projetos, esse guia ficou conhecido como PMBoK.

O PMBoK está divido em nove áreas do processo de gestão de projeto listadas em:

- Gerenciamento de integração do projeto;
- Gerenciamento do escopo do projeto;
- Gerenciamento de tempo do projeto;
- Gerenciamento de custos do projeto;
- Gerenciamento de qualidade do projeto;
- Gerenciamento de recursos humanos do projeto;
- Gerenciamento das comunicações do projeto;
- Gerenciamento de riscos do projeto e;
- Gerenciamento de aquisições do projeto.

Como o foco deste trabalho está na negociação entre desenvolvedor e cliente de software, faz-se necessário detalhar a área de processo gerenciamento de aquisições, a qual está dividida em duas fases principais que possuem seis subprocessos listados em: Planejar compras e aquisições; Planejar contratações; Solicitar respostas de desenvolvedores; Selecionar desenvolvedores; Administração de contrato e Encerramento de contrato. Os subprocessos podem ser visualizados na Figura 2.1.

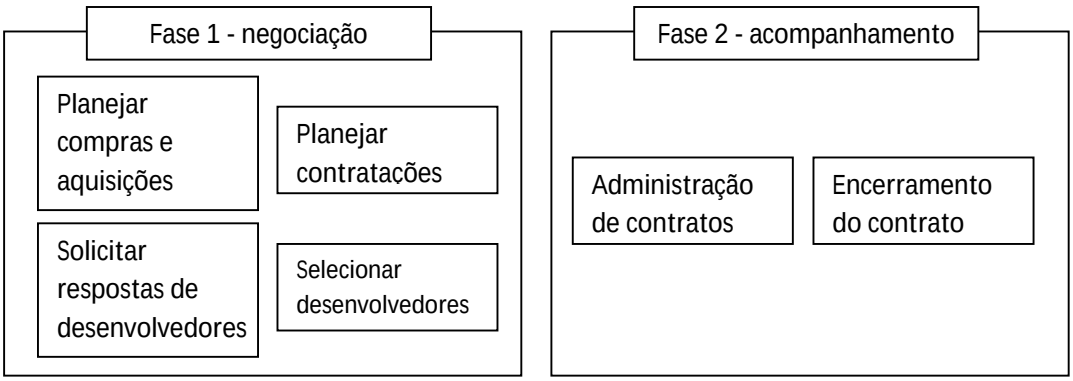

Figura 2.1: Fases do processo de gerenciamento de aquisições.

A partir de pesquisas em vários editais de concorrência pública, percebeu-se que estes possuem um elemento comum que torna compreensível os requisitos entre as partes, entenda-se: desenvolvedor e cliente. Conforme Lei Federal nº 8.666, de 21 de Junho de 1993, art. 40, os editais devem indicar obrigatoriamente alguns aspectos, entre eles, o “objeto da licitação de forma sucinta e clara”.

Nos editais e solicitações de requisitos dos produtos de software não é diferente. Estes também precisam de uma linguagem comum. Segundo VASCONCELOS (2005), as medidas de tamanho de software mais comuns são “número de linhas de código, número de pontos de função e número de pontos de caso de uso”. Antes de discorrer sobre as medidas comuns de software ou sobre seu relacionamento com as etapas do processo de negociação, será realizada uma reflexão sobre os conceitos de negociação.

Assim, LEWICKI e LITTERER (*apud* Ramires, 2004) definem negociação como um “processo social básico usado para resolver conflitos”. Para CARVALHAL (2004) negociação é um processo “em que duas ou mais partes, com interesses comuns e antagônicos se reúnem para confrontar e discutir propostas explícitas com o objetivo de alcançarem um acordo”. Portanto, nota-se que a negociação se inicia de forma divergente no intuito de terminar de forma convergente.

De acordo com a visão de KERSTEN (*apud* Ramires, 2004) o processo de decisão na negociação “envolve procurar um ponto de intersecção dentro de um conjunto de alternativas aceitáveis” com o objetivo de satisfazer todos os sujeitos envolvidos. Durante sucessivas negociações estas alternativas podem sofrer alterações, aumentando ou diminuindo seu propósito até ser encontrado o ponto de interseção.

Completando esta visão, RIELLI (2007) enfatiza que negociação é um “processo de alcançar objetivos por meio de um acordo nas situações em que existam interesses comuns, complementares e opostos”. CAMARGO (2007) ilustra o ciclo da negociação exemplificado na Figura 2.2.

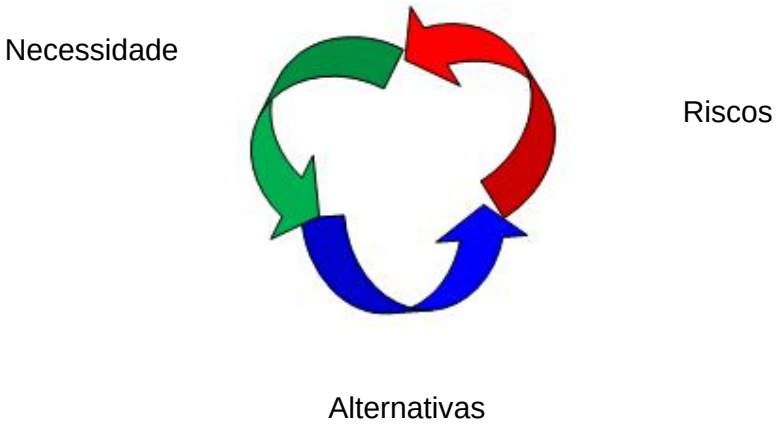

Figura 2.2: Ciclo da negociação.

O ciclo da negociação ilustrado na Figura 2.2, inicia-se com a identificação

das necessidades. Em seguida, entra a fase das alternativas, onde são comparadas as vantagens e as características de cada uma. Neste ponto, CAMARGO (2007) pondera que durante a fase das alternativas “não se está ainda no ponto de decidir, mas necessita-se de informações e argumentos que possibilitem a comparação e a formação de um juízo crítico sobre cada possibilidade”, fazendo uma análise crítica de cada alternativa.

Após o levantamento das alternativas, identificam-se os riscos envolvidos em cada uma. Sem a ponderação dos riscos, a negociação fica prejudicada e o seu ciclo não se completa. Para JUNQUEIRA (apud Picolo, 2004), o processo de negociação é “muito semelhante ao processo de tomada de decisão. Em certo sentido, pode-se dizer que a negociação não é nada mais do que uma tomada de decisão”.

Diante desses fatos, percebe-se que a negociação deve ser encarada como uma forma de beneficiar as partes envolvidas, pois não se negocia contra alguém, mas em companhia de alguém. Ao analisar as alternativas envolvidas, verifica-se a existência de pontos em comum. PICOLO (2004) defende que no fim a “negociação pode ser um acordo ou um impasse, porém, o resultado é considerado eficiente quando não há outra solução”.

Como já foi mostrada na Figura 2.1, a primeira fase do processo de aquisição é chamada de negociação, a qual possui quatro etapas distintas e complementares. A primeira etapa se chama “Planejar compras e aquisições” e define-se, segundo o PMBoK (2004) como: a “determinação do que comprar ou adquirir e de quando e como fazer isso”.

Após o planejamento das compras e aquisições, parte-se para o “Planejamento das contratações” que se define, na visão da tradução do PMBoK (2004) como a “documentação dos requisitos de produtos, serviços e resultados e identificação de possíveis desenvolvedores”.

O terceiro subprocesso da Fase 1 é o “Solicitar respostas de

desenvolvedores". A tradução do PMBoK (2004) mostra que este subprocesso refere-se sobre a "obtenção de informações, cotações, preços, ofertas ou propostas, conforme adequado". Por fim, o último subprocesso da Fase 1 é o "Selecionar desenvolvedores" que compreende a "análise de ofertas, escolha entre possíveis desenvolvedores e negociação de um contrato por escrito com cada desenvolvedor".

Na Fase 1, os processos trabalham em conjunto, cada um envolve o esforço de uma ou mais pessoas com base nas necessidades do projeto. Um exemplo de uso desses processos seria uma licitação, onde são executados os dois processos de planejamento e posteriormente ocorre uma pesquisa de mercado, buscando uma empresa que forneça o serviço requerido da forma mais viável. Uma vez selecionado o desenvolvedor, inicia-se a negociação.

O resultado da etapa "Selecionar desenvolvedores" serve de base para a realização da Fase 2, a qual refere-se ao processo de acompanhamento como na Figura 1. As informações de saída dessa etapa conforme o PMBoK (2004) são: Desenvolvedores selecionados; Contrato; Plano de gerenciamento de contrato; Disponibilidade de recursos; Plano de gerenciamento de requisições (atualização) e Mudanças solicitadas.

A Fase 2 do processo de aquisição pode ser dividida em: Administração de contrato e Encerramento do contrato. O manual do PMBoK (2004) aponta o primeiro subprocesso como o "gerenciamento do contrato e da relação entre o comprador e o desenvolvedor". Nesta etapa, realiza-se uma análise sobre o contrato atual e verifica-se o seu andamento.

Por fim, ocorre o Encerramento do contrato que se caracteriza por "terminar e liquidar cada contrato", ou seja, encerra os contratos vigentes que não possuam nenhuma pendência.

Depois de apresentar as fases do processo de aquisição, pretende-se apresentar algumas métricas de tamanho de software, enfatizando a métrica de

Pontos de Caso de Uso (PCU), a qual se mostra como protagonista deste trabalho, sem ofuscar o papel das demais medidas existentes.

2.2 MÉTRICAS

Em uma fábrica de software, o foco na negociação direciona-se ao sistema de informação, o qual segundo o Brazilian Function Point User Group (BFPUG) (2007) tem seu tamanho funcional “medido através de métricas”, entre elas, a métrica de Pontos de Caso de Uso (PCU). Assim como os engenheiros civis que produzem seus orçamentos baseados em metros quadrados.

Na visão de PRESSMAN (1995), “a medição é fundamental para qualquer atividade de engenharia e na Engenharia de Software não é diferente”. Além disso, destaca que a medição permite obter entendimento a partir de um mecanismo para avaliação objetiva.

Ao longo dos anos, muitas métricas foram criadas com o objetivo de fornecer ao gerente de projetos uma ferramenta que suporte a determinação do esforço para execução de um projeto. Entre as mais conhecidas estão os modelos de estimativas de linha de código (LOC), o modelo Constructive Cost Mode (COCOMO), a análise de pontos de função (PF) e, recentemente, a de Pontos de Caso de Uso (PCU). Um dos primeiros modelos a surgir foi o COCOMO idealizado por Boehm desde 1970. O COCOMO possui inúmeras versões, a mais recente intitulada COCOMO II utiliza linhas de código, pontos de função ou pontos de objetos para calcular suas estimativas.

Conforme HARMAN (2007), um problema enfrentado pelo COCOMO é “o uso das linhas de código como medida de tamanho”. Desta forma cita-se como exemplo, um código de 1.000 linhas feito em Java que na teoria representa muito mais funcionalidades do que um código feito em Assembler. Deste modo, a dificuldade em representar o tamanho funcional dos softwares através do número de linhas codificadas gerou o surgimento de novas métricas, entre elas, a mais utilizada hoje em dia, chamada de Pontos de Função (PF).

AGUIAR (1998) defende que a métrica de PF teve seus conceitos “introduzidos por Allan Albrecht, em uma conferência do GUIDE (Grupo de usuários IBM) em 1979. A métrica se caracteriza por medir as funcionalidades fornecidas por um software do ponto de vista de seu usuário”, ou seja, a métrica PF busca medir o que o software faz e não como ele foi construído.

FATTO (2007) enfatiza que o objetivo da métrica PF é “encontrar uma técnica de estimativa para esforço de desenvolvimento de software que fosse independente da linguagem de programação utilizada”. Atualmente, a técnica é mantida pelo International Function Point User Group (IFPUG) que possui a extensão brasileira denominada BFPUG. Além disso, FATTO (2007) adverte que a técnica é tão utilizada que sofre “constantes evoluções”. Atualmente, o IFPUG possui um manual que recebe atualizações, no entanto tais atualizações não visam mudanças radicais, apenas esclarecimentos em relação às definições e regras de contagem melhorando a medição. Apesar do sucesso, a técnica possui algumas desvantagens. Entre as principais desvantagens, destaca-se o treinamento especializado que deve ser ministrado para as pessoas que a usarão, por ser uma técnica extensa e complexa. Na falta de uma métrica mais simples e com o surgimento de novos paradigmas de linguagem de programação, outras técnicas foram surgindo. Entre esses paradigmas, a programação orientada a objetos ganha destaque e está, cada vez mais, inserida nas organizações de desenvolvimento de software no Brasil. Com isso foram surgindo novas métricas derivadas dela, entre elas, a métrica de Pontos de Caso de Uso (PCU).

2.3 MÉTRICA DE PONTOS DE CASO DE USO (PCU)

A métrica de Pontos de Caso de Uso (PCU) conforme FREIRE (2003) foi criada em 1993 por Gustav Karner da empresa Objectory AB. Esta métrica permite fazer estimativas no início do projeto com base no modelo de casos de uso. BELGAMO e FABBRI (*apud* Heimberg e Grahl, 2004) evidenciam que “a filosofia dos Pontos de Casos de Uso é baseada na definição da Análise de Pontos de Função (APF), na qual a funcionalidade vista pelo usuário é a base para a estimativa do tamanho do software”.

Segundo FREIRE (2003), “a técnica de análise de dimensão por Casos de Uso foi criada para permitir que seja possível estimar o tamanho do sistema ainda na fase de levantamento de Casos de Uso”, possibilitando uma visão mais rápida do tamanho do sistema. Um Quadro comparativo para verificar alguns dados das principais métricas de tamanho de software pode ser visualizado no Quadro 2.1.

Quadro 2.1: Comparação entre as métricas

	COCOMO	PF	PCU
Paradigma de Programação	Estruturada	Estruturada e Orientada a Objetos	Orientada a Objetos
Consultoria	Não existe	Existe	Não foi encontrado
Certificação	Não foi encontrado	CFPS	Não existe
ONG Mantenedora	Não foi encontrado	IFPUG	Não foi encontrado
Suporte no Brasil	Não foi encontrado	BFPUG	Não foi encontrado
Acervos de Dados (Google)	407.000 ocorrências	2.790.000 ocorrências	176 ocorrências

O Quadro 2.1 motivou o uso da PCU como métrica neste trabalho, pelo fato de mostrar que não existe muito material sobre a métrica, como no caso do acervo de dados, que obteve 407.000 ocorrências para COCOMO, 2.790.000 ocorrências para PF e apenas 176 ocorrências para PCU e no caso da ONG mantenedora onde a métrica PF possui a IFPUG e as outras métricas não possuem suporte. A seguir tem-se uma explicação sobre o funcionamento da métrica para, no Capítulo 4, mostrar como o protótipo funciona.

Conforme MEDEIROS (2004), o processo de contagem da métrica PCU consiste em 14 passos, porém segundo MONTEIRO e BELCHIOR (2006) podem

ser resumidos em 6 passos, os quais estão exibidos na Figura 2.3.

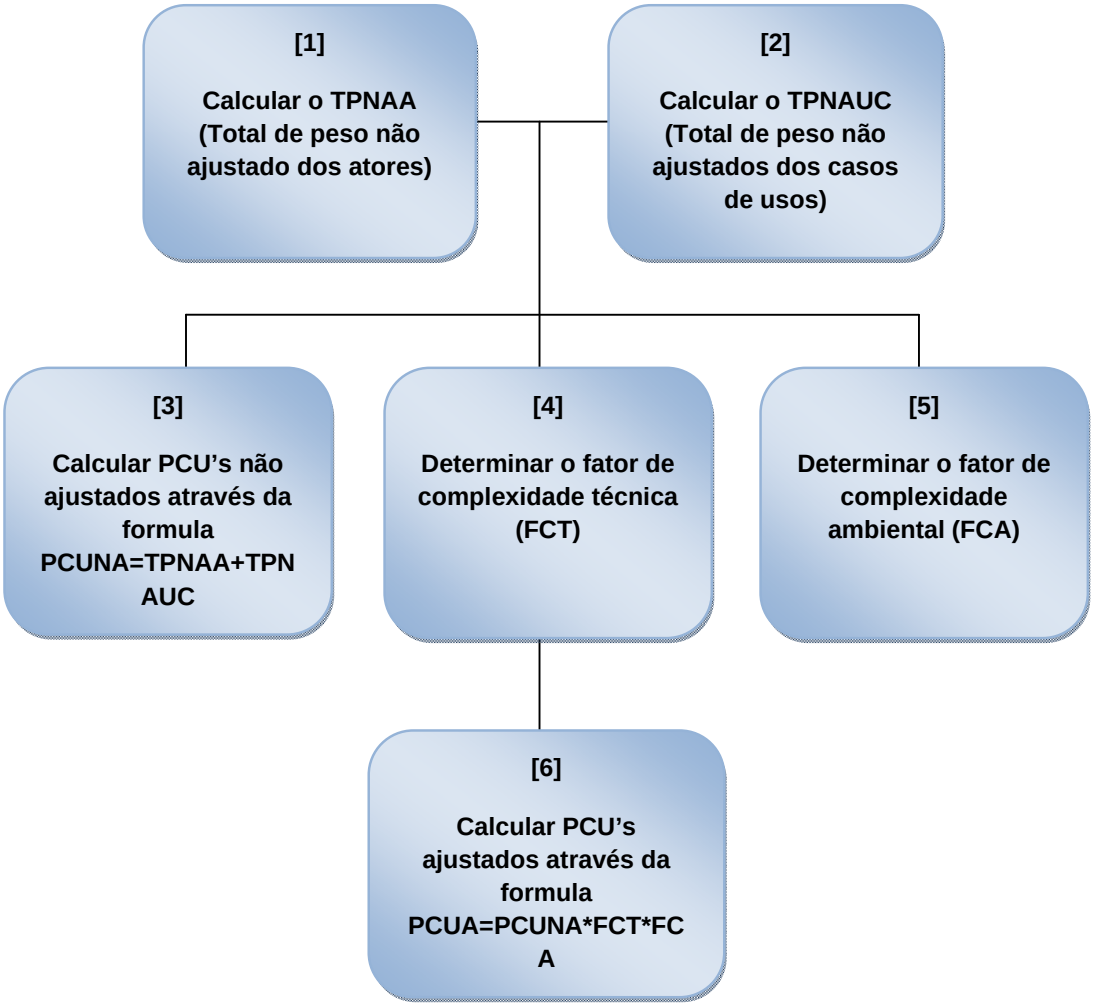

Figura 2.3: Passos da métrica PCU.

No passo 1, calcula-se o Total de Peso Não Ajustado dos Atores (TPNAA). Conforme MONTEIRO e BELCHIOR (2006) para se identificar os atores do sistema devem-se pensar primeiramente nos "usuários que o utilizarão, e também como esses atores podem ser categorizados".

Posteriormente, os atores devem ser determinados, relacionados e depois classificados de acordo com seu nível de complexidade (simples, intermediário ou complexo) atribuindo respectivamente os pesos 1, 2 ou 3, conforme o Quadro 2.2.

Quadro 2.2: Complexidade dos atores

Tipo de Ator	Descrição	Peso
Simples	Aplicação com API definida	1
Intermediário	Outro sistema interagindo através de um protocolo de comunicação, como TCP/IP ou FTP.	2
Complexo	Um usuário interagindo através de uma interface gráfica (stand-alone ou Web)	3

Com os atores determinados e classificados, faz-se necessário realizar o cálculo do Total de Pesos Não Ajustados dos Atores (TPNAA) somando os produtos da quantidade de atores pelo seu peso. Por exemplo, um sistema que possua um ator intermediário (peso 2) e outro complexo (peso 3) terá um TPNAA igual a 5.

Após o cálculo do TPNAA, parte-se para o passo 2, onde será calculado o Total de Pesos Não Ajustados dos Casos de Uso (TPNAUC). A melhor forma de identificar os casos de uso segundo MONTEIRO e BELCHIOR (2006), é “considerar o que cada ator exige do sistema”.

Após determinar os casos de uso, os mesmos devem ser contados e recebem seu grau de complexidade. Este grau de complexidade está relacionado a quantidade de transações realizadas pelo sistema. Feito isto calcula-se o TPNAUC somando os produtos da quantidade de casos de usos pelo respectivo

peso.

O peso relativo a complexidade, baseada no número de transações, é informado no Quadro 2.3.

Quadro 2.3: Tipos de caso de uso.

Tipos de Caso de Uso	Número de Transações	Peso
Simples	Até 3 transações	5
Intermediário	De 4 a 7 transações	10
Complexo	Acima de 7 transações	15

Com o TPNAA e o TPNAUC calculados, parte-se para o passo 3, onde ocorre o cálculo dos Pontos de Caso de Uso Não Ajustados (PCUNA) que são os Pontos de Caso de Uso de forma bruta, como na equação 2.1.

$$PCUNA = TPNAA+TPNAUC$$

Equação 2.1: Equação do PCUNA.

Para ajustar os Pontos de Caso de Uso alguns fatores técnicos devem ser determinados e posteriormente aplicados a uma fórmula. Esse procedimento é executado nos últimos 3 passos da técnica PCU e serão explicados a seguir. O passo 4 determina os Fatores de Complexidade Técnica (FCT), descritos no Quadro 2.4. Conforme FREIRE (2003), os FCT cobrem “uma série de requisitos funcionais do sistema”. Os FCTs variam em uma escala de 0 a 5, de acordo com o grau de dificuldade do software a ser construído. Estes requisitos precisam ser analisados cuidadosamente, pois é necessária a participação de todos os envolvidos no sistema para alcançar este valor, tendo em vista que os Fatores de Complexidade

Técnica (FCT) irá abstrair dados exatos, porém dados abstratos como, por exemplo, a eficiência do usuário final.

Quadro 2.4: Escala dos FCT.

Descrição	Peso
Sistemas Distribuídos	2
Desempenho da aplicação	1
Eficiência do usuário final (on-line)	1
Processamento interno complexo	1
Reusabilidade do código em outras aplicações	1
Facilidade de instalação	0,5
Usabilidade (facilidade operacional)	0,5
Portabilidade	2
Facilidade de manutenção	1
Concorrência	1
Características especiais de segurança	1
Acesso direto para terceiros	1
Facilidades especiais de treinamento	1

Freire completa que “o valor 0 indica nenhuma influencia, 3 indica influência moderada e 5 indica forte influência”. Após determinar o valor dos FCTs, deve-se multiplicar pelo respectivo peso ilustrado no Quadro 2.4, somar o total e aplicar a seguinte equação.

Fator de Complexidade Técnica (FCT) = 0.6 + (0.01*somatório do fator técnico)

Equação 2.2: Equação do FCT.

Com o FCT calculado, parte-se para o passo 5, onde deve-se determinar o Fator de Complexidade Ambiental (FCA). FREIRE (2003) advoga que os FCAs cobrem os “requisitos não-funcionais associados ao processo de desenvolvimento”. Estes FCAs, descritos no Quadro 2.5, são determinados através da escala de 0 a 5.

Quadro 2.5: Escala dos FCA.

Fator	Descrição	Peso
F1	Familiaridade com o processo de desenvolvimento de software	1,5
F2	Experiência na aplicação	0,5
F3	Experiência com OO, na linguagem e técnica de desenvolvimento.	1
F4	Capacidade do líder de análise	0,5
F5	Motivação	1
F6	Requisitos estáveis	2
F7	Trabalhadores com dedicação parcial	-1
F8	Dificuldade da linguagem de programação	-1

Após determinar o valor de cada fator, deve-se multiplicar pelo peso e somar o total dos valores. Em seguida, aplicar a seguinte equação.

Fator de Complexidade Ambiental (FCA) = 1,4 + (-0,03 * somatório do fator ambiental)

Equação 2.3: Equação do FCA.

O último passo da técnica de Pontos de Caso de Uso (PCU) é calcular os Pontos de Casos de Uso Ajustados (PCUA). Os resultados dos passos anteriores servem de entrada para execução da fórmula do passo 6. Esse cálculo é realizado com base na multiplicação dos PCU não ajustados (PCUNA) com os Fatores de Complexidade Técnica (FCT) e os Fatores de Complexidade Ambiental (FCA) através da equação do PCUA.

PCUA = PCUNA * Fator de Complexidade Técnica * Fator de Complexidade Ambiental

Equação 2.4: Equação do PCUA.

HEIMBERG e GRAHL (2007) explanam que Karner, o criador da estimativa, sugere a utilização de 20 pessoas-hora por unidade de PCU, para calcular estimativas de horas de programação. É importante notar que Karner não calcula horas para outras atividades como teste, análise, projeto entre outras. Após uma análise de Schneider e Winters a sugestão de Karner passou pelo seguinte refinamento:

- X = total de itens de F1 a F6 com pontuação abaixo de 3;
- Y = total de itens de F7 a F8 com pontuação acima de 3;
- Se X + Y <= 2, usar 20 como unidade de homens/hora;
- Se X + Y = 3 ou X + Y = 4, usar 28 como unidade de homens/hora e;
- Se X + Y >= 5, deve-se tentar modificar o projeto de forma a baixar o número, pois o risco de insucesso é relativamente alto.

Com os valores de X e Y determinados aplica-se a equação 2.5 para se obter a estimativa de horas de programação de um projeto de software.

Estimativa de horas = PCUA * pessoas hora por unidade de PCU

Equação 2.5: Estimativa de horas de programação.

CAPÍTULO 3

METODOLOGIA DA PESQUISA

Este capítulo visa mostrar ao leitor em quais categorias de pesquisa esse trabalho está inserido e posteriormente explana a metodologia ICONIX.

3.1 TIPOS DE PESQUISA

MARCONI e LAKATOS (2002) defendem a existência de dois grupos de pesquisa: pura e aplicada. A pesquisa pura é “aquela que procura o progresso científico, a ampliação de conhecimentos teóricos, sem a preocupação de utilizá-los na prática. É a pesquisa formal tendo em vista: generalizações, princípios, leis”. A pesquisa aplicada consiste na aplicação de seus resultados ou produtos na solução de problemas reais. Desta maneira acredita-se que esta pesquisa seja aplicada, pois o resultado dessa pesquisa será utilizado com dados de um problema real utilizado no trabalho de Heimberg & Grahl.

Conforme DEMO (*apud* Baffi, 2007), as pesquisas podem ser classificadas como teórica, metodológica, empírica e prática, partindo desse princípio essa pesquisa se classifica como empírica, pois DEMO (*apud* Baffi, 2007), afirma que tal pesquisa é dedicada ao tratamento da "face empírica e factual da realidade; produz e analisa dados, procedendo sempre pela via do controle empírico e factual". Este trabalho também é considerado uma pesquisa exploratória porque envolve levantamento bibliográfico e qualitativo, pois, os dados obtidos não podem ser quantificáveis. Para CASTILLO (2003), a pesquisa qualitativa “ajuda a identificar questões e entender porque elas são importantes”.

Quanto ao objeto, essa pesquisa se classifica como bibliográfica, devido a pesquisa ser baseada em materiais já publicados e também como pesquisa de laboratório, porque será executada em um ambiente preparado para o desenvolvimento do software. Laboratório se entende pelo local onde será reproduzido o ambiente natural do fato estudado e segundo MARCONI e

LAKATOS (2002), “se restringem a determinadas manipulações”. Após esclarecimentos sobre o tipo de pesquisa, faz-se necessário apresentar a metodologia ICONIX, pois, ela será utilizada no desenvolvimento dos artefatos que irão auxiliar na programação do software. O ICONIX será utilizado por ser uma metodologia robusta e baseada em casos de uso.

3.2 ICONIX

O ICONIX é um processo de desenvolvimento de software. Este modelo não gera tanta documentação, como o RUP (Rational Unified Process, ou Processo Unificado Racional, é um processo proprietário de Engenharia de software criado pela Rational Software Corporation). O ICONIX é um processo simples, como o XP (eXtreme Programming ou programação extrema ou simplesmente XP, é uma metodologia ágil para equipes pequenas e médias e que irão desenvolver software com requisitos vagos e em constante mudança), porém apesar da simplicidade é eficiente na Análise e Projeto.

O ICONIX faz uso da linguagem de modelagem UML (OMG®, 2001) e possui uma característica exclusiva chamada "Rastreabilidade dos Requisitos" (Traceability of Requirements). Este modelo permite verificar em todas as fases se os requisitos estão sendo atendidos, por isso sua abordagem é flexível e aberta, isto é, se for necessário usar outro recurso da UML (OMG®, 2001) para complementar os recursos usados nas fases do ICONIX, não há problema algum.

Principais fases do ICONIX:

- Modelo de Domínio;
- Modelo de Caso de Uso;
- Análise Robusta;
- Diagrama de Sequência e;
- Diagrama de Classe.

ICONIX pode ser considerada uma metodologia pura, prática e simples, mas também poderosa e com um componente de análise e representação dos problemas sólido e eficaz, por isso, a metodologia ICONIX é caracterizada como um Processo de Desenvolvimento de Software desenvolvido pela ICONIX Software Engineering.

MAIA (2005) afirma que o ICONIX é uma metodologia de desenvolvimento de software que ajuda a "planejar, projetar e avaliar o software de forma mais simples". SILVA e BORBINHA (2001), ratificam essa ideia quando afirma que o ICONIX "é uma metodologia prática, intermediária entre a complexidade da RUP, que gera muita documentação e a simplicidade da XP". Por outro lado, o ICONIX possui uma característica exclusiva chamada Traceability of Requirements, que conforme MAIA (2005) "permite checar em todas as fases, se os requisitos estão sendo atendidos". Esta característica é chamada por SILVA e BORBINHA (2001), de rastreabilidade, a qual representa "a capacidade de seguir a relação entre os diferentes artefatos produzidos". BONA (apud Borges, 2007) enfatiza que o ICONIX se "baseia em responder algumas questões sobre o software que será desenvolvido. Para isto, utiliza técnicas da UML (OMG®, 2001) para auxiliar na obtenção da melhor resposta". As questões e técnicas estão listadas no Quadro 3.6.

Quadro 3.6: Questões e técnicas sobre o software a ser desenvolvido

Modelo	Questões e Técnicas	Diagrama a ser utilizado	Fase
Dinâmico	Quem são os usuários do sistema, e o que eles estão tentando fazer?	Casos de Uso	Análise
Estático	Q que são, no "mundo real", os objetos e as associações entre eles?	Classe	Projeto
Dinâmico	Que objetos são necessários para cada caso de uso?	Análise Robusta	Análise
Dinâmico	Como os objetos estão colaborando e interagindo dentro de cada caso de uso?	Sequência	Projeto
Estático	Como realmente será construído o sistema em um nível prático?	Modelo de Domínio	Análise

As respostas das perguntas apresentadas no Quadro 3.6, serviram de apoio para o início das fases do ICONIX que, conforme MAIA (2005), são compostas pela construção de “cinco artefatos interdependentes e rastreáveis”, listados em: Modelo de Domínio; Modelo de Caso de Uso; Análise Robusta; Diagrama de Sequência; Diagrama de Classe. Além disso, MAIA (2005) preconiza que “o ICONIX é dividido em dois grandes setores, Modelo Estático e Modelo Dinâmico, que podem ser desenvolvidos em modo paralelo e de forma recursiva”. Os modelos são exemplificados na Figura 3.4.

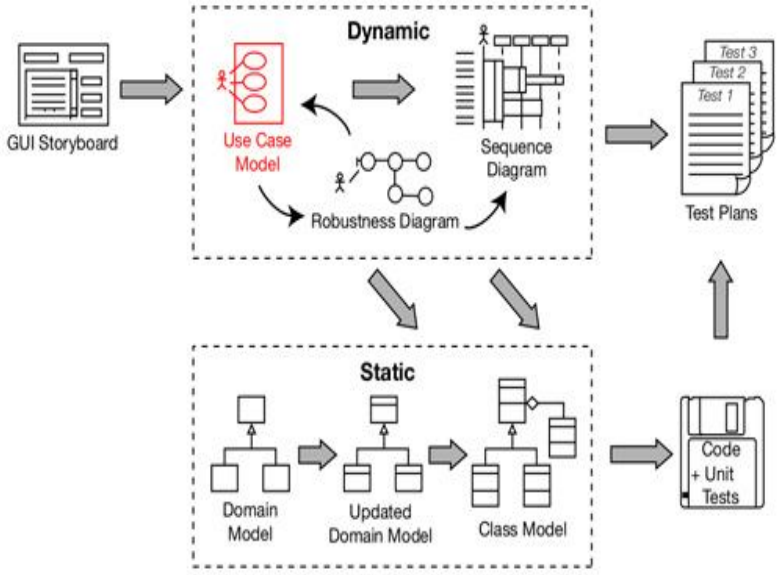

Figura 3.4: Modelo do ICONIX.

O Modelo Dinâmico apresentado na parte superior da Figura 3.4 possui como entrada um protótipo. Como o ICONIX não possui um modelo padrão para servir de entrada, será utilizado o padrão elaborado por Wazlawick (2004).

No seu livro, Wazlawick (2004) elabora uma visão geral do sistema que se caracteriza como “um documento de texto em formato livre, no qual se deve escrever aquilo que conseguiu descobrir de relevante sobre o sistema após conversas com os clientes”. Um exemplo de visão geral de um sistema de vídeo-locadora proposto no livro de WAZLAWICK (2004) encontra-se a seguir:

“É proposto o desenvolvimento de um sistema de vide locadora, que vai informatizar as funções de empréstimo, devolução e reserva de fitas. O objetivo do sistema é agilizar o processo de empréstimo e garantir maior segurança, ao mesmo tempo possibilitar um melhor controle das informações por parte da gerência. Deverão ser gerados relatórios de empréstimos por cliente, empréstimos por fita e empréstimos no mês. O sistema deverá calcular automaticamente o valor dos pagamentos a serem efetuados em cada empréstimo, inclusive multas e descontos devidos. A cada devolução de fitas corresponderá um pagamento, não sendo possível trabalhar com sistema de créditos. A impossibilidade de efetuar um pagamento deve deixar o cliente suspenso, ou seja, impossibilitado de tomar emprestadas novas fitas até saldar a dívida”.

Com base nestes dados se obtém o Modelo Dinâmico apresentado na parte superior da Figura 9. O Modelo de Domínio mostra o usuário interagindo com o sistema através de ações onde o sistema apresenta alguma resposta ao usuário em tempo de execução. O Modelo Dinâmico é constituído pelos Diagramas de Caso de Uso, Sequência e Análise Robusta. Na parte inferior da Figura 9, encontra-se o Modelo Estático. Este modelo é formado pelos Diagramas de Domínio ou Modelo de Domínio e Diagramas de Classe que modelam o funcionamento do sistema sem nenhuma interação com o usuário. Após o detalhamento dos modelos, faz-se necessária uma apresentação aos Diagramas.

O primeiro diagrama que o ICONIX possui é o Diagrama de Domínio que, segundo BONA (*apud* Borges, 2007), refere-se à “tarefa de se descobrir objetos que representam coisas e conceitos do mundo real”. Esta tarefa é considerada indispensável no auxílio da fase de projeto partindo dos casos de uso. Através do Diagrama de Domínio são descobertos todos os objetos do mundo real de um determinado problema. Um exemplo de Diagrama de Domínio encontra-se na Figura 3.5.

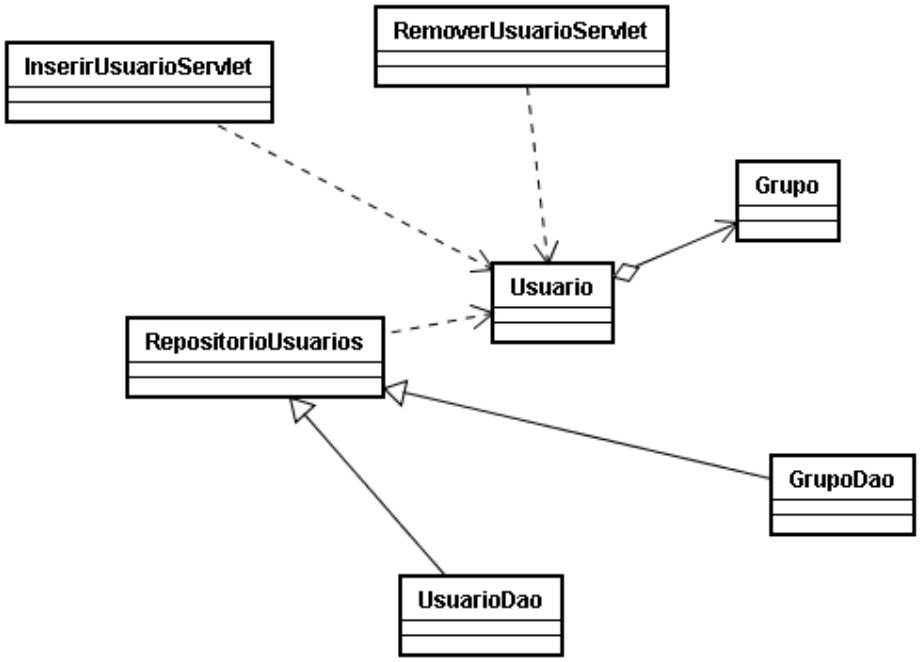

Figura 3.5: Diagrama de Domínio.

A Figura 3.5 é composta por classes e relacionamentos de um sistema. Desta forma, COSTA (2007) afirma que classes são “descrições de conjuntos de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica”. No Diagrama de Domínio da Figura 3.5, Grupo é um exemplo de classe. Relacionamentos são ligações entre as classes e são divididos em dependências, generalizações ou associações. Segundo o Laboratório de Engenharia de Software da PUCRio (LES) (2007), dependência é “um relacionamento semântico entre dois itens, nos quais a alteração de um pode afetar a semântica do outro”.

Generalizações são definidas pelo LES (2007) como um “Relacionamento de especialização/generalização, onde objetos ‘filhos’ compartilham a estrutura e o comportamento dos objetos ‘pais’”.

De acordo com MAIA (2005), para construir um Diagrama de Domínio, os seguintes passos devem ser realizados:

1. Encontrar as classes que representam as abstrações do Domínio do

problema. Estas classes servirão como base para a construção do sistema;

2. Destacar os substantivos e frases que possuem substantivos. Pois estes possivelmente se tornarão objetos e atributos. Verbos e frases com verbos se tornarão associações. Palavras como “seu”, “nosso” e “seus” que indicam substantivos, devem ser atributos no lugar de objetos;

3. Eliminar classes ambíguas, incorretas e itens desnecessários;

4. Fazer generalização, se necessário. Aplicar técnicas de composição e agregação, caso existam;

5. Revisar os Diagramas construídos, analisando principalmente as associações e generalizações.

O segundo Diagrama do ICONIX é o Diagrama de Casos de Uso, que BORGES (2005) aponta que são “especificações do comportamento do sistema a partir da descrição das funcionalidades do sistema”, ou seja, ele mostra como o usuário usará o software. Na visão de BOOCH (2000), Caso de Uso “é uma descrição de um conjunto de sequências de ações, resultantes da interação do sistema com um ator”. Um exemplo de Casos de Uso encontra-se na Figura 3.6.

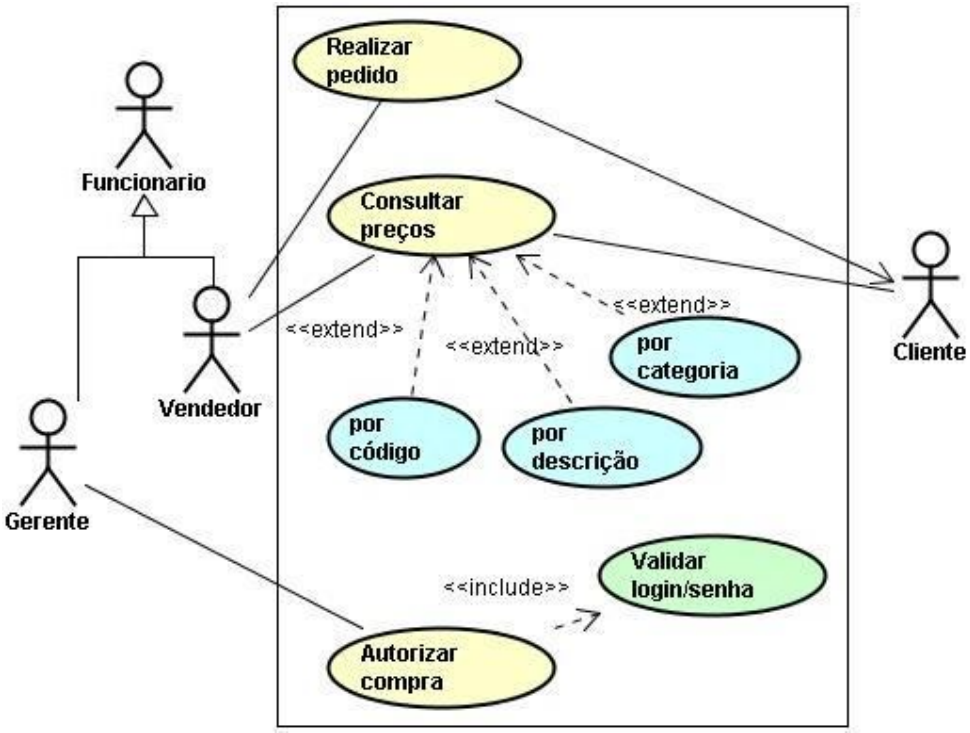

Figura 3.6: Diagrama de Casos de Uso

A Figura 3.6 é composta pelos atores e os casos de uso de um sistema clínico. Conforme WAZLAWICK (2004) atores “interagem com o sistema por meio do caso de uso”. MAIA (2005) mostra que para se construir um modelo de Caso de Uso, é importante executar os seguintes passos:

1. Identificar o maior número de casos de uso possível e descrevê-los cada um detalhadamente, com clareza e sem ambiguidades.
2. Refinar o texto, verificando se as orações estão bem descritas. O formato do texto deve seguir a sequência substantivo-verbo-substantivo, para facilitar a identificação dos atores e objetos.
3. Verificar se os casos de uso atendem aos requisitos funcionais desejados no sistema.

Após a definição dos Diagramas de Domínio e Caso de Uso, inicia-se a criação do Diagrama de Análise Robusta que de acordo com BONA (apud Borges, 2007), “tem por finalidade, conectar a parte de análise com a parte de projeto, assegurando que a descrição dos casos de uso esteja correta”. Este diagrama serve de ponte entre a parte de análise e a parte de projeto. BONA (*apud* Borges, 2007) evidencia que a análise de robustez se “concentra em construir um modelo analisando as narrativas de texto dos casos de uso, identificando um conjunto de objetos que participarão de cada caso de uso”. Estes objetos podem ser classificados em três tipos: fronteira, controle e entidade. Um exemplo dos objetos encontra-se na Figura 3.7.

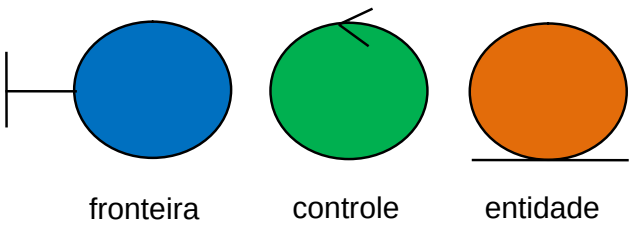

Figura 3.7: Tipos de objeto.

A Análise Robusta possui certos passos que segundo BONA (*apud* Borges, 2007) são citados a seguir:

- Verificar razoabilidade: verifica se os casos de uso foram escritos de

forma correta, e se estão de acordo com os requisitos do sistema;

- Verificar a perfeição: válida se todas as referências de execução do sistema estão descritas nos fluxos dos casos de uso;
- Identificar objetos: permite identificar novos objetos, que não foram vistos no Modelo de Domínio e;
- Projeto preliminar: os Diagramas da Análise Robusta são menos complexos e mais fáceis de ler do que os Diagramas de Sequência.

A Análise Robusta possui um conjunto de regras para uso dos objetos que devem ser seguidas, listadas abaixo:

1. Atores podem se comunicar somente com objetos interface;
2. Objetos interface podem se comunicar somente com atores e controladores;
3. Objetos entidade podem se comunicar somente com objetos controladores;
4. Objetos controladores podem se comunicar somente com objetos entidade, interface e também com outros controladores, mas não com atores.

Um exemplo de Diagrama de Análise Robusta encontra-se na Figura 3.8.

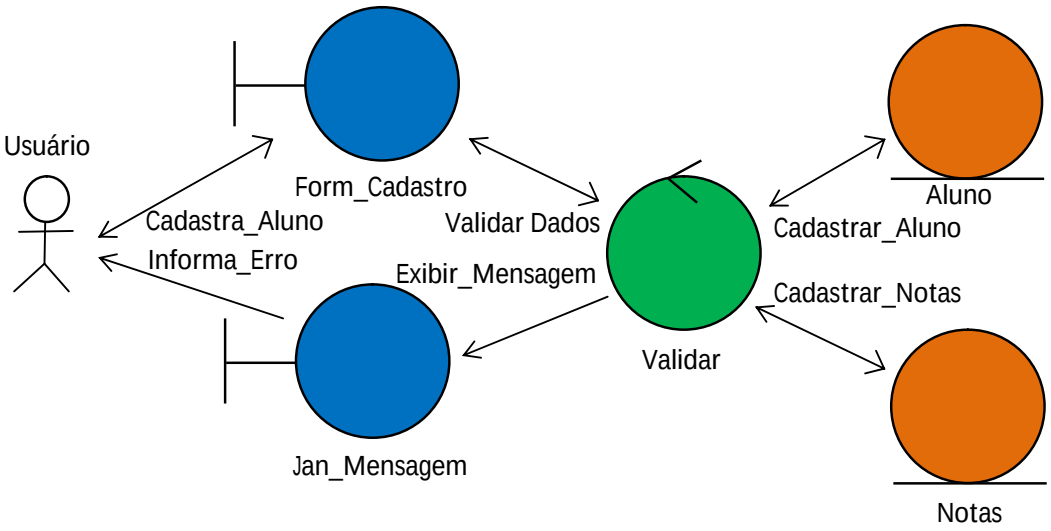

Figura 3.8: Diagrama de Análise Robusta.

O penúltimo Diagrama do ICONIX é o Diagrama de Sequência. O Diagrama de Sequência é um diagrama representando a sequência de processos (mais especificamente, de mensagens passadas entre objetos) num programa de computador. Como um projeto pode ter uma grande quantidade de métodos em classes diferentes, pode ser difícil determinar a sequência global do comportamento. O diagrama de sequência representa essa informação de uma forma simples e lógica.

Um diagrama de seqüência descreve a maneira como os grupos de objetos colaboram em algum comportamento ao longo do tempo. Ele registra o comportamento de um único caso de uso e exibe os objetos e as mensagens passadas entre esses objetos no caso de uso.

Em síntese, o **Diagrama de Sequência** é usado para representar interações entre objetos de um cenário, realizadas através de operações ou métodos *(procedimentos ou funções)*. Este diagrama é construído a partir do Diagrama de Casos de Usos. Primeiro, se define qual o papel do sistema (Use Cases), depois, é definido como o software realizará seu papel (Sequência de operações).

O **diagrama de sequência** dá ênfase a ordenação temporal em que as mensagens são trocadas entre os objetos de um sistema. Entende-se por mensagens os serviços solicitados de um objeto a outro, e as respostas desenvolvidas para as solicitações.

WAZLAWICK (2004) defende que o Diagrama de Sequência é “útil para representar a sequência dos eventos de sistema em um cenário de um caso de uso”. Durante a construção do Diagrama de Sequência, algumas metas devem ser alcançadas:

1. Alocar comportamentos entre os objetos interface, controlador e entidade, identificados anteriormente na Análise Robusta.
2. Mostrar as interações detalhadas que ocorrem na linha de tempo entre os objetos associados com o texto de caso de uso. Os objetos interagem entre si através de mensagens. Cada mensagem estimula algum objeto a

realizar alguma ação. Para cada comportamento de um caso de uso, devemos identificar as mensagens necessárias e os métodos.

3. Povoar o Diagrama de Classes, distribuindo as operações entre os objetos, já que nessa altura, já foram identificadas as maiorias dos atributos.
4. Após concluir o modelo de interação, tem-se bastante informação, portanto pode-se dispor do comportamento detalhado do esquema de sequência entre objetos no contexto de seus respectivos casos de uso. Finaliza-se então, procurando e alocando apropriadamente atributos e operações.

Dentro do ICONIX, BORGES (2005), enfatiza que o Diagrama de Sequência "representa o artefato principal do produto" de projeto. Um exemplo de Diagrama de sequência é apresentado na Figura 3.9.

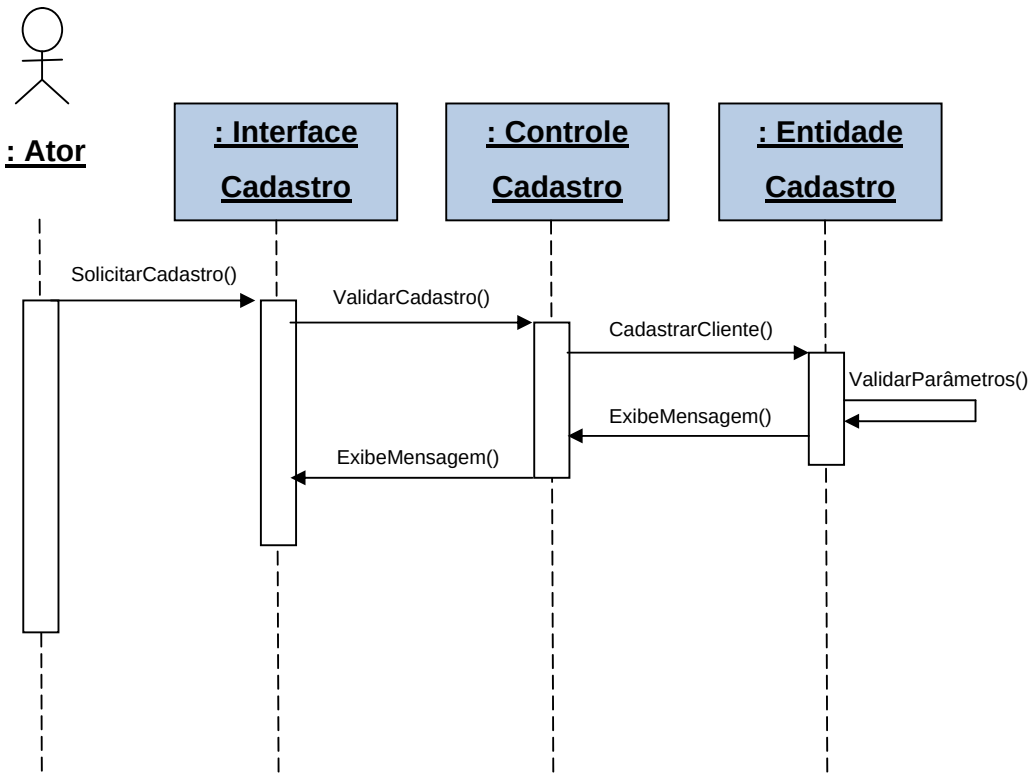

Figura 3.9: Diagrama de Sequência.

Segundo MAIA (2005), o Diagrama de sequência é composto por quatro componentes conforme Figura 3.9:

1. O texto que descreve os fluxos de ação para os casos de uso.
2. Os objetos (interface, controlador e entidade) que interagem entre si.
3. Mensagens.
4. Métodos ou operações.

A seguir serão apresentados quatro passos para se construir um Diagrama de sequência corretamente:

1. Copiar o texto de especificação do caso de uso no lado direito do Diagrama. O texto serve como uma lembrança continua do que é necessário fazer.
2. Adicionar os objetos entidade do Diagrama Robusto. Cada um destes objetos é um exemplo de classe que deve estar no Diagrama de Classe.
3. Adicionar os objetos interface do Diagrama de Análise Robusta.
4. Alocar os métodos em suas respectivas classes adequadamente.

O último Diagrama é chamado de Diagrama de Classes. Em programação, um **diagrama de classes** é uma representação da estrutura e relações das classes que servem de modelo para objetos.

O Diagrama de Classe é uma modelagem muito útil para o sistema, pois ele define todas as classes que o sistema necessita e é a base para a construção dos diagramas de comunicação e estados.

WAZLAWICK (2004) afirma que “se baseia em primeiro lugar no modelo conceitual adicionado de algumas informações que não era possível obter na fase de análise”, ou seja, o Diagrama de Classes irá se basear no Modelo de Domínio. Ele será composto pelo Modelo de Domínio acrescido de outras informações que

devem ser colhidas durante a construção dos outros artefatos, conforme Figura 3.10.

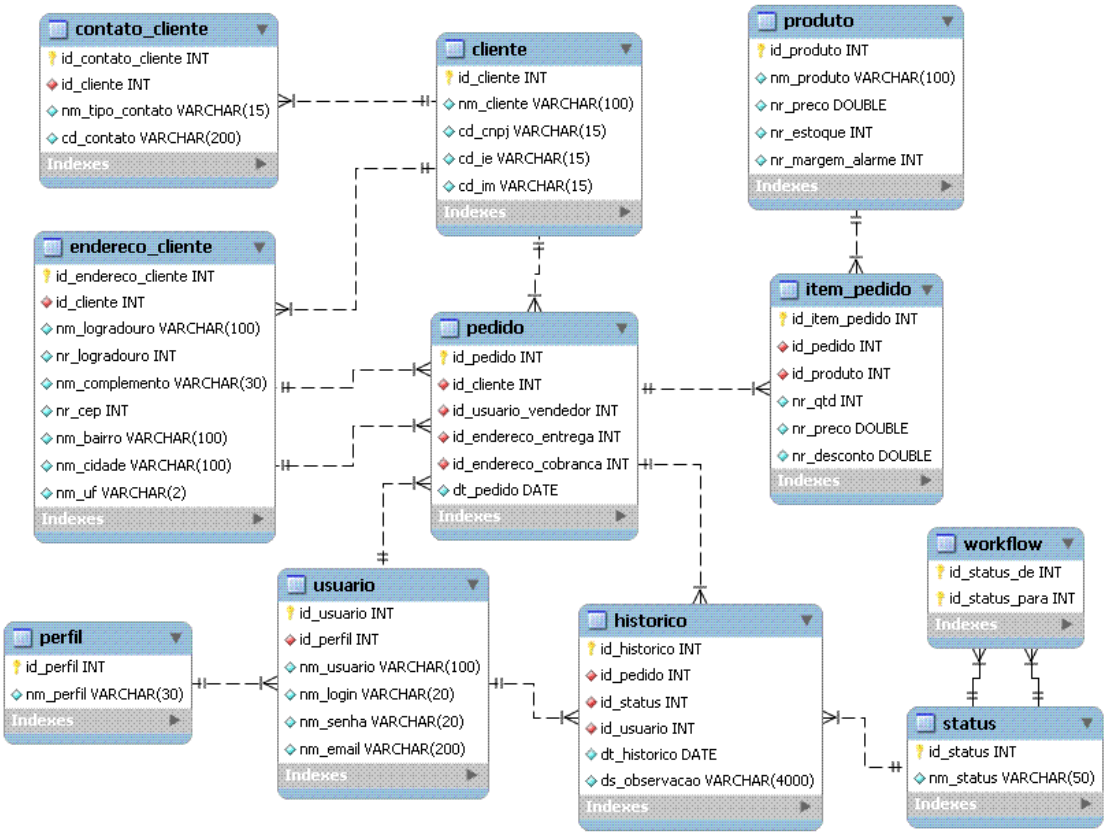

Figura 3.10: Diagrama de Classe.

A Figura 3.10 mostra um Diagrama de Classes constituído por classes, atributos, métodos e relacionamentos. O Diagrama de Classes é um Modelo de Domínio mais refinado. Esta é uma das vantagens do ICONIX, pois, um artefato serve de entrada para o outro, causando dependência e fazendo que no fim da análise, todos os artefatos estejam modelados da maneira correta.

CAPÍTULO 4

4.1 PROTOTIPAÇÃO

O sistema foi simulado na linguagem PHP e foram utilizadas as ferramentas PHP Editor, DBDesign, PHPMyadmin, JUDE e banco de dados MySQL. A escolha destas ferramentas motivou-se pelo fato de serem livres e com vastas documentações já publicadas.

4.2 TESTES

OLIVEIRA (2003) define teste como a “atividade de executar um software com o objetivo de revelar falhas”. Atualmente, existem várias formas de se testar um software. Esta seção conceitua e mostra os testes que foram realizados no sistema.

Analisando o processo através de um ponto de vista procedimental, uma série de testes foi implementada sequencialmente, focando cada módulo de forma individual, a fim de garantir uma completa e máxima detecção de erros. Cada módulo foi testado em diferentes compiladores e interpretadores. Em seguida, os módulos, foram montados para serem simulados em linguagens interpretadas.

Após a integração foi realizado um teste de validação como garantia final de que o modelo proposto atenda as exigências funcionais, comportamentais e de desempenho.

O modelo também foi combinado com outros elementos do sistema, como hardware e sistemas operacionais, verificando se todos se combinam adequadamente e se a função e/ou desempenho global do sistema é conseguida.

Durante a verificação de desempenho de hardware foi possível analisar que a arquitetura necessária, para a implementação do sistema, exige processadores de núcleo único com apenas o cache interno, memória ativa com capacidade de no mínimo 512 MB e unidades de memórias secundárias com aproximadamente 100 MB de espaço livre. O modelo foi testado nos ambientes operacionais: Microsoft

Windows 2003 Server, Microsoft Windows Vista Ultimate, Microsoft 7 Ultimate e no Linux Ubuntu. Em todos os ambientes o modelo funcionou com total compatibilidade de hardware e software.

As técnicas mais conhecidas de testes de software são a Caixa-Branca e Caixa-Preta. A primeira técnica, conforme MELO (2007), ocorre quando o desenvolvedor “tem acesso ao código fonte da aplicação”. Na técnica Caixa-Preta o desenvolvedor “não possui acesso algum ao código fonte do programa”, ou seja, na técnica Caixa-Preta são testados os eventos disparados pelos usuários. O primeiro teste realizado chama-se Teste Funcional. VIJAYKUMAR (2007) explana que o Teste Funcional é visto como teste “Caixa-Preta”, sem acesso ao código. Enfatiza ainda que o mesmo possui uma “preocupação com funcionalidade e não com detalhes de programação” e apresenta “ponto de vista de um usuário”. Para realizar esse tipo de teste foram utilizados dados dos Projetos dos trabalhos de Heimberg e Grahl (2007). Estes dados foram simulados no sistema com o intuito de comparar o resultado final.

Tabela 4.1: Dados do trabalho de Heimberg e Grahl.

Projetos	Atores	Casos de Uso	FCT	FCA	PCUNA	PCUA	Horas Estimadas
Projeto 1	4	5	1,00	0,81	54	43,74	437,40
Projeto 2	6	8	1,02	0,81	76	61,50	615,60
Projeto 3	7	4	1,03	0,81	57	46,17	461,70

CAPÍTULO 5

5.1 RESULTADOS ALCANÇADOS

Para construir um ponto de início no desenvolvimento do Modelo MAN (Modelo de Apoio à Negociação) foi elaborada a descrição do sistema, baseando-se na visão geral do sistema de vídeo-locadora proposta por WAZLAWICK (2004). A descrição é apresentada no Quadro 5.7.

Baseando-se na descrição apresentada no Quadro 4.7 foram definidos os Diagramas da ferramenta utilizando a metodologia ICONIX. O primeiro Diagrama elaborado foi o Diagrama de Domínio ou Modelo de Domínio, o qual está apresentado na Figura 5.11.

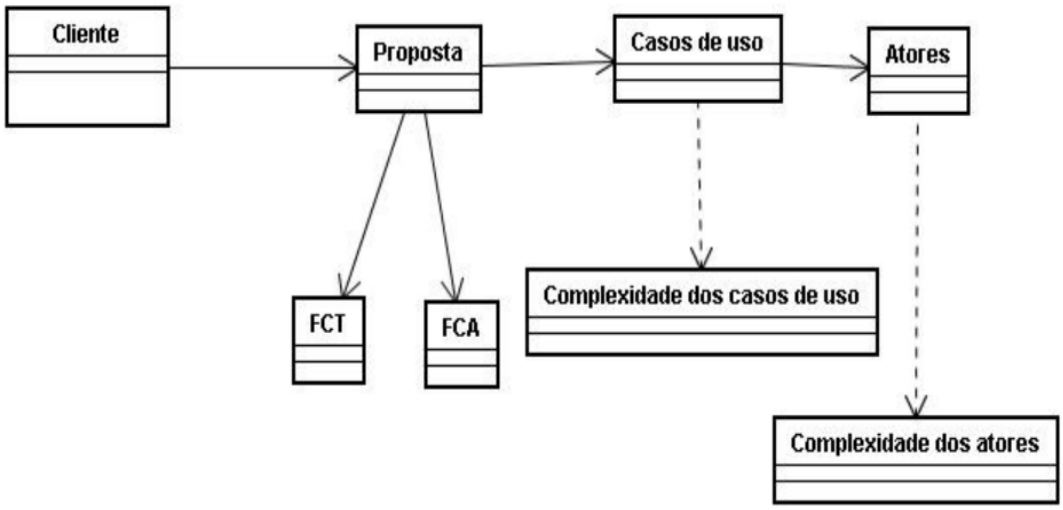

Figura 5.11: Modelo de Domínio do protótipo.

O Modelo de Domínio é composto por 8 classes e 7 relacionamentos. Entre os 7 relacionamentos 5 são associações enquanto dois são dependências. Entre os relacionamentos de dependência encontram-se os da classe Complexidade dos casos de uso com a classe Casos de uso e Complexidade dos atores com a classe Atores, por isso ambos possuem uma linha tracejada. O próximo Diagrama construído foi o Diagrama de Casos de Uso apresentado na Figura 5.12.

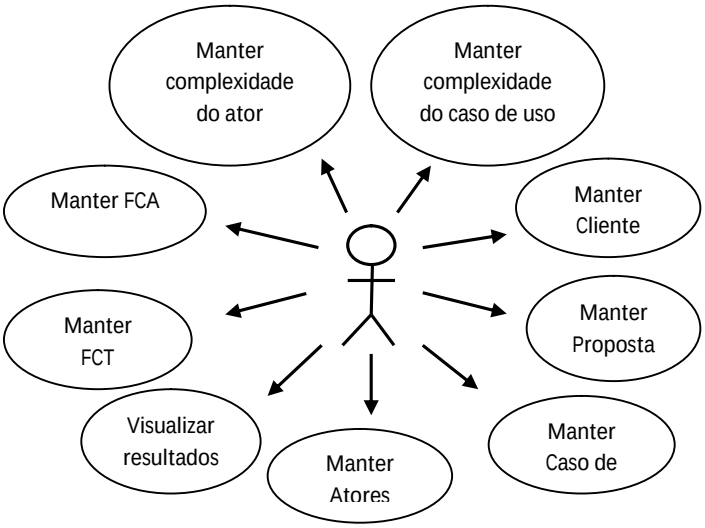

Figura 5.12: Diagrama de Casos de Uso do protótipo.

A utilização de um padrão nos Casos de Uso Manter diminuiu a quantidade de linhas na descrição dos mesmos, assim como a quantidade de documentos e aumentou a produtividade do programador.

Os Diagramas de Análise Robusta são formados por objetos entidade, interface e controlador, conforme o Quadro 5.7.

Quadro 5.7: Objetos do Diagrama de Análise Robusta.

Tipo de objeto	Nome do objeto
Entidade	Proposta e navegador.
Interface	Form de cadastro e telas de sucesso e erro.
Controlador	Gerenciador de cadastros.

Confirmou-se que o Diagrama possui dois objetos entidade, três objetos interface e um objeto controlador. Após o Diagrama de Análise Robusta foi construído o Diagrama de Sequência exemplificado na Figura 5.13.

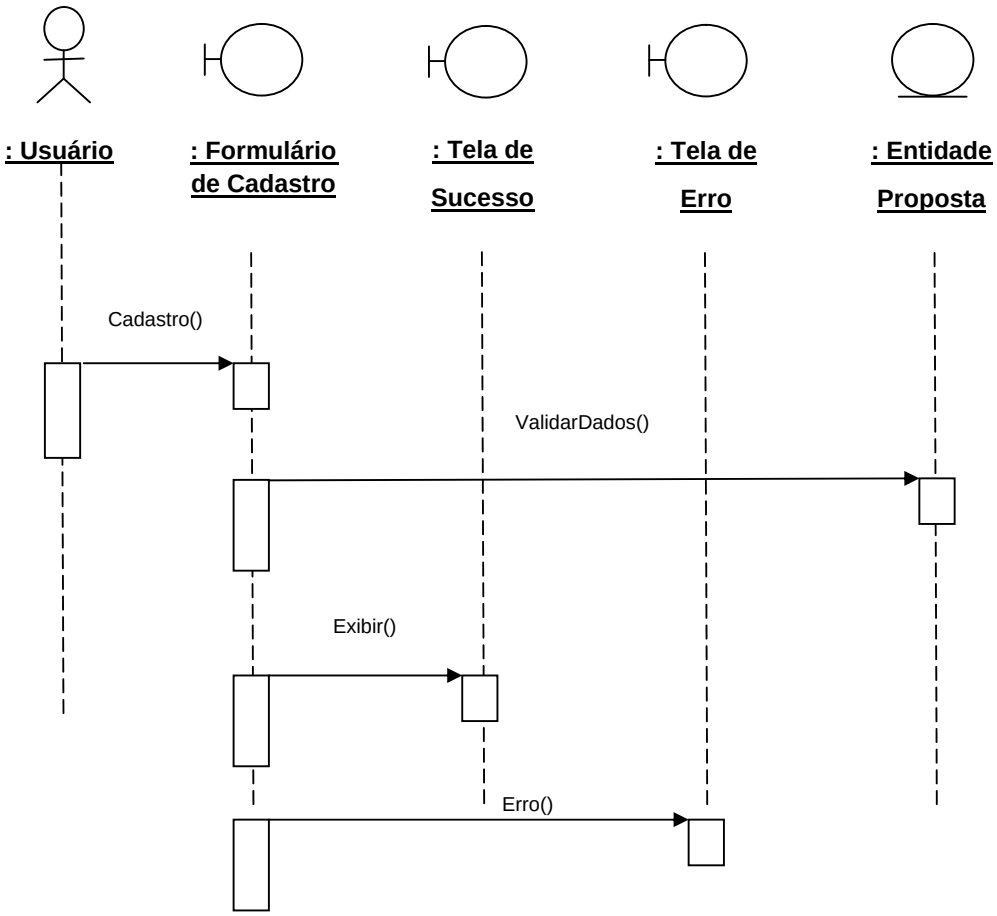

Figura 5.13: Diagrama de Sequência.

Assim como o Diagrama de Análise Robusta, o Diagrama de Sequência também é construído para cada caso de uso, o exemplo acima é o de cadastro de propostas, como foi dito anteriormente esse Diagrama possui mais detalhes que o Diagrama de Análise Robusta.

Esse Diagrama serviu como entrada para a fase de projeto. O Diagrama da

Figura 5.13 possui o ator usuário, que será o responsável pelo manuseio do sistema, as interfaces de cadastro, erro e sucesso que serão responsáveis por informar onde o ator está situado e a entidade do banco de dados que receberá as informações. Os outros Diagramas de Sequência encontram-se no apêndice. A seguir será apresentado o Diagrama de Classes final:

Diagrama de Classes final é um refinamento do Diagrama de Domínio. É composto pelos mesmos elementos do Diagrama de Domínio acrescido de atributos e métodos. Geralmente o Diagrama de Classes final apresenta mais classes que o Diagrama de Domínio, porém como o sistema é apenas um protótipo o mesmo número de classes foi identificado. Com isso o número de classes e relacionamentos foi o mesmo.

Todos os métodos desse Diagrama retornam boolean, isso ocorre pelo fato de todos os métodos serem de cadastro retornando apenas true ou false caso o dado seja inserido ou não com sucesso.

5.2 CASO REAL ESTUDADO

A metodologia apresentada neste trabalho foi utilizada no desenvolvimento de um projeto utilizado em uma empresa de segurança eletrônica do PIM (Polo Industrial de Manaus). O sistema consiste em um software de apoio aos sistemas de CFTV (Circuito fechado de Televisão).

5.2.1 ESPECIFICAÇÃO DO SISTEMA

Um sistema de CFTV consiste em uma série de equipamentos eletroeletrônicos, interligados entre si através de um protocolo padrão de comunicação, tendo como objetivo monitorar áreas através da captura de imagens. As imagens capturadas são enviadas a um computador responsável pelo armazenamento e gerenciamento dos dados.

Quando ocorre a necessidade de busca por uma determinada imagem, ou seja, algum evento ocorrido em uma data, hora e lugar específico, torna-se necessário a

solicitação destas imagens ao departamento responsável pelo gerenciamento das mesmas. Isto demanda tempo e uma série de processos burocráticos.

O problema pode estender-se, pois dependendo do sistema de CFTV do cliente é necessário parar o sistema de vigilância para realizar a busca por um determinado evento, causando falha na segurança e irregularidades nos procedimentos operacionais da empresa.

A partir desta situação torna-se necessário a implantação de um software que interaja com o Sistema de CFTV e tente minimizar ou solucionar tal problema.

5.2.2 SOLUÇÃO PROPOSTA

Como os processos de gravação ocorrem no computador a solução proposta foi transformar este computador em um servidor de dados, utilizando o protocolo TCP/IP, padrão da Internet, e desenvolver um software que compartilhe os dados desde servidor com os clientes na rede.

Desta forma o servidor terá como funções o recebimento e o armazenamento dos dados e os computadores clientes poderão acessá-lo remotamente para realizar busca pelos eventos sem a necessidade de parar os serviços do servidor e não sendo mais necessário interromper os serviços das pessoas envolvidas na área de segurança. O modelo chama-se SAE (Sistema de Acessos a Eventos).

5.3 APLICAÇÃO DA METODOLOGIA DE NEGOCIAÇÃO NO CASO REAL ESTUDADO

Utilizando as técnicas de PCU, mostradas no decorrer deste trabalho, foram analisados os dados que serão utilizados no modelo simulado SAE. O quadro 5.8 exibe as escalas dos pesos dos fatores de complexidades técnicas, os pesos adotados e sua justificativa.

Quadro 5.8: Escala dos FCT utilizados no modelo.

Descrição	Peso	Peso Adotado	Justificativa
Sistemas Distribuídos	2	2	Sistema voltado para o ambiente Web
Desempenho da aplicação	1	1	Alto desempenho para acesso aos dados
Eficiência do usuário final (on-line)	1	0	Não é necessário usuário experiente
Processamento interno complexo	1	0	Sistema de busca e exibição dos dados, não há a necessidade do processamento digital de imagens
Reusabilidade do código em outras aplicações	1	0	Arquitetura específica para este modelo
Facilidade de instalação	0,5	0,5	Acesso direto via servidor de dados
Usabilidade (facilidade operacional)	0,5	0,5	Interface gráfica e intuitiva
Portabilidade	2	2	Totalmente portável para qualquer sistema operacional
Facilidade de manutenção	1	1	Manutenção remota
Concorrência	1	0	Sem concorrência
Características especiais de segurança	1	1	Protocolos web de segurança
Acesso direto para terceiros	1	0	Software será registrado
Facilidades especiais de treinamento	1	1	Padrão de uso da Internet

A complexidade ambiental retrata as características do ambiente de desenvolvimento e dos desenvolvedores envolvidos no projeto. O quadro 5.9 exibe as escalas dos pesos dos fatores de complexidades ambientais, os pesos adotados e sua justificativa.

Quadro 5.9: Escala dos FCA utilizados no modelo.

Fator	Descrição	Peso	Peso Adotado	Justificativa
F1	Familiaridade com o processo de desenvolvimento de software	1,5	1,5	Desenvolvedores com experiência na área de desenvolvimento
F2	Experiência na aplicação	0,5	0,5	Desenvolvedores com experiência na área de desenvolvimento
F3	Experiência com OO, na linguagem e técnica de desenvolvimento.	1	1	Desenvolvedores com experiência na área de desenvolvimento
F4	Capacidade do líder de análise	0,5	0,5	Desenvolvedores com experiência na área de desenvolvimento
F5	Motivação	1	1	Equipe motivada
F6	Requisitos estáveis	2	2	Requisitos atendidos
F7	Trabalhadores com dedicação parcial	-1	0	Equipe com dedicação total
F8	Dificuldade da linguagem de programação	-1	0	Desenvolvedores com experiência na área de desenvolvimento

A seguir são mostrados os dados compilados para o cálculo da estimativa de horas.

Tabela 5.2: Dados do modelo SAE.

Projetos	Atores	Casos de Uso	FCT	FCA	PCUNA	PCUA	Horas Estimadas
SAE	3	3	0,69	1,21	18	15,03	300,6

Como o projeto baseia-se no acesso aos dados através da conexão com um servidor de dados, ou seja, o acesso é via Internet ou via Intranet, o peso dos atores é

3 (três), conforme a Tabela 5.2, pois está em um nível complexo conforme o Quadro 2.2.

O modelo trata 3 (três) casos de uso, conforme a Tabela 5.2:

1. Tela de login;
2. Tela para busca de eventos;
3. Tela de Resultados.

O Fator de Complexidade Técnica (FCT) é calculado conforme Equação 2.2 obtendo-se o valor 0,69, conforme a Tabela 5.2.

O Fator de Complexidade Ambiental (FCA) é calculado conforme Equação 2.3 obtendo-se o valor 1,21 (1,205), conforme a Tabela 5.2.

Os Pontos de Caso de Uso Não Ajustados (PCUNA) é calculado conforme Equação 2.1, onde o TPNAA = 3 e o TPNAUC = 15, obtendo-se o valor 18, conforme a Tabela 5.2.

Os Pontos de Caso de Uso Ajustados (PCUA) é calculado conforme Equação 2.4 obtendo-se o valor 15,03 (15,0282), conforme a Tabela 5.2.

A estimativa de horas é calculada conforme Equação 2.5 obtendo-se o valor 300,06, conforme a Tabela 5.2. Lembrando que o valor utilizado por pessoas hora por unidade de CPU é 20 (vinte), conforme HEIMBERG e GRAHL (2007).

Além do cálculo das estimativas de horas os requisitos do sistema, baseados no acesso, consulta, alteração e exclusão no banco de dados foram testados no modelo e são mostrados no Quadro 5.10.

Quadro 5.10: Requisitos testados.

Dados de entrada	Ação executada	Resultado esperado	Resultado obtido
Dados do FCT	Inserção no banco de dados	Ok	Ok
Dados do FCA	Inserção no banco de dados	Ok	Ok
Dados sobre a Complexidade de Atores	Inserção no banco de dados	Ok	Ok
Dados sobre a Complexidade dos casos de uso	Inserção no banco de dados	Ok	Ok
Dados do cliente	Inserção no banco de dados	Ok	Ok
Nome, data, descrição, justificativa, cliente, status, FCTs e FCAs da proposta	Inserção no banco de dados	Ok	Ok
Nome dos Casos de Uso, Atores e suas respectivas complexidades	Inserção no banco de dados	Ok	Ok
Campo data no cadastro de propostas	Aceitar apenas números e adicionar barras	Ok	Ok
Campo CNPJ do cliente	Aceitar apenas números e adicionar a máscara	Ok	Ok
Campo estimativa de horas da tela resultados finais	Receber o valor da multiplicação	Ok	Ok
Campo quantidade de dias da tela resultados finais	Receber o valor da divisão	Ok	Ok

5.4 ANÁLISE DOS RESULTADOS

O valor das horas estimadas, correspondente a 300.6 horas, pode ser interpretado da seguinte forma: 300.6 horas equivalem aproximadamente a 37.6 dias de trabalho. Dividindo 37.6 por 5 (cinco dias na semana – de segunda a sexta) obtém-se 7.52, que corresponde a sete semanas e meia, aproximadamente dois meses, que foi o tempo utilizado pela equipe para o desenvolvimento do sistema, confirmando que as estimativas de horas determinada pela métrica PCU está de acordo com o modelo tratado no mundo real.

Em função do tamanho e das características do projetos, que foram calculados nos Fatores de Complexidades Técnicas (FCT) e nos Fatores de Complexidades Ambientais (FCA), pode-se afirmar que as estimativas de horas estão

dentro do tempo de desenvolvimento utilizado pelas fábricas de software, ou seja, está dentro do tempo adotado inclusive por outras métricas.

Com base nos dados inseridos foi possível obter rapidamente alguns resultados através da metodologia do PCU. Os limites da confiança estão disponíveis para todos os parâmetros e resultados calculados.

O modelo permite a análise de modos de falha competitivos e de análise de sistemas, pois tem-se uma visão do tempo de desenvolvimento baseados em dados reais.

Como o modelo utiliza métodos paramétricos para analisar os eventos é possível modelar o número de ocorrências de um a cada possível solicitação de mudança.

Como a estimativa de horas é definida é possível identificar falhas nos equipamentos, baseando-se em sua performance, ou seja, se o tempo extrapolar a estimativa será possível analisar se a falha foi do ambiental de desenvolvimento ou dos equipamentos em uso.

O modelo permite a análise de dados não paramétricos, pois através de técnicas de orientação a objeto, como a herança, é possível utilizar dados herdados de outros modelos sem haja a necessidade de se saber os parâmetros utilizados no modelo original (reaproveitamento de códigos).

O modelo utiliza pouca quantidade de análises, baseando-se em dados, para obter um resultado confiável.

CAPÍTULO 6

6.1 CONCLUSÃO

O objetivo desse trabalho foi demonstrar a necessidade do uso das métricas na negociação entre cliente e desenvolvedor, tornando-a mais eficiente. Para tanto foi aplicada à técnica de Pontos de Caso de Uso (PCU).

Ao longo do trabalho foi confirmado que através da utilização da técnica PCU, e consequentemente dos Diagramas de Casos de Uso, é possível calcular automaticamente mudanças nas estimativas de tamanho do sistema a cada pequena alteração de requisitos, apenas refazendo alguns cálculos.

A técnica PCU, por ser uma técnica formal e ter mostrado resultados satisfatórios neste trabalho, se apresenta como alternativa para empresas que não possuem métricas definidas ou para empresas que não vem obtendo resultados satisfatórios com as métricas que utilizam hoje em dia.

Na confecção dos Diagramas do sistema foi utilizada a metodologia ICONIX, ela foi escolhida, pois, como a métrica PCU, ela também é baseada nos Diagramas de Caso de Uso, facilitando a construção dos artefatos. Uma vantagem da metodologia ICONIX é a facilidade de monitorar todas as suas fases para verificar se todos os seus requisitos estão sendo atendidos, isso ocorre através de uma característica exclusiva chamada Traceability of Requirements.

Durante a confecção dos diagramas, o ICONIX mostrou-se uma metodologia robusta e contínua, pois, cada artefato construído serviu como entrada para o próximo, com isso a metodologia correspondeu a todas as expectativas esperadas.

No cenário da negociação de projetos de software, a simulação demonstrou um exemplo de cálculo de estimativas de horas de programação. Essa característica pode agilizar uma futura negociação, pois os valores serão encontrados de uma forma mais rápida proporcionando uma ideia do tamanho da

proposta e consequentemente o tempo viável para seu desenvolvimento.

Todos os testes realizados no sistema obtiveram resultados positivos, inclusive a comparação com trabalhos anteriores.

6.2 PROPOSTAS PARA TRABALHOS FUTUROS

Para trabalhos futuros apresentam-se algumas sugestões.

- Atualizar o modelo para que possa informar outras estimativas, tais como risco e qualidade.
- Refinar a métrica PCU apresentando um padrão de escrita para os Diagramas de Caso de Uso no intuito de eliminar as diferenças na confecção dos mesmos pelos analistas;
- Realizar um estudo de caso aplicando a técnica PCU e posteriormente a técnica de Pontos de Função (PF) para comparar os resultados e;
- Desenvolvimento do modelo em Fuzzy.

REFERÊNCIAS

ADADE F., A. Análise de Sistemas Dinâmicos. S. José dos Campos-SP, CTA-ITA-IEMP, 1992.

AGUIAR, M. O que é um Ponto de Função. Disponível em: http://www.bfpug.com.br/Artigos/Dekkers-PontosDeFuncaoEMedidas.htm. Acesso em 12 mai. 2010.

ALVES, F.A.R. ICONIX Disponível em: http://fredslz.googlepages.com/pds_6pp.pdf. Acesso em 20 mai. 2010.

ÁVILA, M. d' PMBoK e Gerenciamento de Projetos http://www.mhavila.com.br/tópicos/gestao/PMBoK.html. Acesso em 20 fev. 2010.

BAFFI, T.A.M. MODALIDADES DE PESQUISA: UM ESTUDO INTRODUTÓRIO. Disponível em: http://www.pedagogiaemfoco.pro.br/met02a.htm. Acesso em 10 jan. 2010.

BEACHLE , N.H. & HARRISON, H.L. Introduction to Dynamic Systems Analysis. New York, Harper & Row, 1978.

BORGES, P.C.C. Sistema de controle de eventos utilizando metodologia ICONIX. Disponível em: http://www.unipe.br/graduacao/computacao/projetos/tcc20052/SISTEMA%20DE%20CONTROLE%20DE%20EVENTOS%20UTILIZANDO%20METODOLOGIA%20ICONIX.pdf. Acesso em 17 mai. 2010.

BOTTURA, C.P. Análise Linear de Sistemas. Rio de Janeiro, Guanabara Dois, 1982.

CAMARGO, R.J.F. A NEGOCIAÇÃO EFICAZ. Disponível em: http://www.iep.org.br/lit/neg.html. Acesso em 25 jan. 2010.

CANON, R.H. Dynamics of Physical Systems. New York, McGraw-Hill, 1967.

CARVALHAL, E. Negociação. Rio de Janeiro: Vision, 2004.

CASTILLO, R.A.F. de. Aprendendo sobre pesquisas. Disponível em: http://www.ead.unicamp.br/trabalho_pesquisa/Pesq_quali.htm. Acesso em 12 mai. 2010.

CASTOR, M.E. de Fábricas de Software: Passado, Presente e Futuro. Disponível em: http://www.unibratec.com.br/revistacientifica/diretorio/edicao1/artigo_fabricasw_tecnologus_ final.pdf. Acesso em 25 jan. 2010.

CLOSE, C.M. & FREDERICK, D.K. Modeling and Analysis of Dynamic Systems. Boston, Houghton Mifflin Co., 1978.

COSTA, X.A.H. Classes. Disponível em: http://www.dcc.ufla.br/~heitor/Artigos/ Artigo_009.html. Acesso em 12 mai. 2010.

DOEBELIN, E.O. System Modelling and Response. New York, Wiley, 1980. Acesso em 03 mar. 2010.

DORNY, C.N. Understanding Dynamic Systems: Approaches to Modeling, Analysis, and Design. NJ, Prentice-Hall, 1993.

FATTO. Consultoria e Sistemas. Disponível em: http://www.fattocs.com.br/faq.asp. Acesso em 25 jan. 2010. Acesso em 20 fev. 2010.

FREIRE, H. Calculando Estimativas: o Método de Pontos de Caso de Uso. Disponível em: http://www.cnnt.com.br/files/usecasepoints.pdf. Acesso em 21 abr. 2010.

HARMAN, M. How long is this going to take? Disponível em: http://www.dcs.kcl.ac.uk/staff/mark/exe11.html. Acesso em 25 jan. 2010.

HEIMBERG, V; GRAHL, E.A. Estudo de Caso de Aplicação da Métrica de Pontos de Casos de Uso numa Empresa de Software Disponível em: http://www.inf.furb.br/seminco/2005/artigos/130-vf.pdf. Acesso em 15 mai. 2010.

JUNIOR, A.P.S. Investigação concernente ao uso da videoconferência na negociação empresarial: um estudo de caso. Disponível em: http://64.233.169.104/search?q=cache:LCXmcK51RTcJ:www.ead.fea.usp.br/Semea

d/6 semea d/PGT/005PGT%2520-%2520Investiga%25E7ao%2520Concernente.doc Acesso em 21 abr. 2010.

LAKATOS, E. M; MARCONI, M. de A. Técnicas de Pesquisa. 5.ed. São Paulo:Atlas,2002. LES.UML (OMG®, 2001). Disponível em: http://web.teccomm.les.inf.puc-rio.br/index.php/UML (OMG®, 2001) . Acesso em 20 fev. 2010.

MAIA, J. A. Construindo Softwares com Qualidade e Rapidez Usando ICONIX. Disponível em: http://www.guj.com.br/content/articles/patterns/iconix_guj.pdf. Acesso em 23 mar. 2010.

MELO, W. “Qualidade” + “Testes de Softwares”=“Qualidade de Software”. Disponível em: http://www.ogerente.com.br/qual/dt/qualidade-dt-ws-testes_software.htm. Acesso em 28 jul. 2010.

MERTINS, C.F. Desenvolvimento e gestão de requisitos de software. Disponível em: http://www.inf.unisinos.br/alunos/arquivos/TC_Cristiane_Mertins.pdf. Acesso em 15 fev. 2010.

MONTEIRO, T. C; BELCHIOR, A. D; TUCP: Uma Extensão da Técnica UCP. Disponível em: http://www.inf.ufes.br/~sbqs2006/?q=node/8. Acesso em 19 jan. 2010.

MSW. Fábrica de Software. Disponível em: http://www.mswconsult.com.br/ fabrica.html. Acesso em 10 jan. 2010.

OGATA, K. - Engenharia de Controle Moderno. São Paulo, Prentice-Hall, 1983.

OGATA, K. System Dynamics. New Jersey, Prentice-Hall, 1978.

OGERENTE. A Importância do Ganha-Ganha na Negociação. Disponível em: http://ogerente.com/congestionado/2006/11/22/a-importancia-do-ganha-ganha-na-negociacao/. Acesso em 22 jan. 2010.

OLIVEIRA, F. Capacitação em Técnicas de Teste de Software. Disponível em: http://www.pucrs.br/inf/eventos2003/workshop/arquivo/MiniCurso_Teste.pdf. Acesso em 07 jul. 2010.

PALM III, N.J. Modeling, Analysis and Control of Dynamic Systems. New York, Wiley, 1983.

PAULA, M.V. Um sistema de suporte a negociação cooperativa para destinação de excessos. Disponível em: http://cronos.cos.ufrj.br/publicacoes/reltec/es62203.pdf. Acesso em 19 jan. 2010.

PHILLIPS, C.L. & PARR, J.M. Signals, Systems, and Transforms. NJ, Prentice-Hall, 1995.

PICOLO, A.A. Projeto de Software para Negociações. Disponível em: http://nourau.eadstrong.com.br/document/get.php/280/Picolo_AngeloA_tcci.pdf Acesso em 21 abr. 2010.

PMIAM. Project Management Institute. Disponível em: http://www.pmiam.org/paginas/instituto.html. Acesso em 15 jan. 2010.

PRESSMAN, R. S. Engenharia de Software. 2.ed. São Paulo:Makron Books,1995.

RAMIRES, V.C.J.J. Negociação de requisitos no processo de desenvolvimento de software. Disponível em: www.di.fc.ul.pt/sobre/documentos/tech-reports/04-19.pdf. Acesso em 19 jan. 2010.

RIELLI, E.M. A ARTE DE NEGOCIAR. Disponível em: http://www.bioenergetica. com.br/ novo/artigo.asp?cod_artigo=44. Acesso em 15 jan. 2010.

SHEARER, J.L et alii - Introduction to System Dynamics. Massachusetts, Addison-Wesley, 1967.

SILVA J. O. da; SALVIANO F.C.; JINO M. Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo. Disponível em: http://www. amplaconsultoria.com.br/ARTIGOS/SBQS2004.pdf. Acesso em 07 jul. 2010.

SILVA, A; BORBINHA, J. Análise e Concepção de Sistemas de Informação. Disponível em: https://dspace.ist.utl.pt/bitstream/2295/74372/1/11-acsi-metodologia-iconixjlb.ppt. Acesso em 21 abr. 2010.

VASCONCELOS, A. Introdução a Métricas de Software. Disponível em: www.cin.ufpe.br/~if720/slides/introducao-a-metricas-de-software.ppt. Acesso em 05 jun. 2010.

VIJAYKUMAR, N. L. Introdução aos testes de Software. Disponível em: http://www.lac.inpe.br/~vijay/download/ELAC2007/Vijay_Testes_De_Software_E_Avaliaca o_De_Desempenho_PARTE_II.pdf. Acesso em 07 jul. 2010.

WAZLAWICK, S. R. Análise e projeto de sistemas de informação orientados a objetos. Rio de Janeiro: Campus, 2004.

WELLSTEAD, P.E. Introduction to Physical System Modelling. London, Academic Press, 1979.

ZIEMER, R.E. et alii. Signals and Systems. New York, McMillan Co., 1983.