UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA DE CÁLCULO DE ESTIMATIVAS DE SOFTWARE

ALEXANDRE WENDERLICH

BLUMENAU
2008

2008/1-01

ALEXANDRE WENDERLICH

FERRAMENTA DE CÁLCULO DE ESTIMATIVAS DE SOFTWARE

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Paulo Roberto Dias, Orientador

BLUMENAU
2008

2008/1-01

FERRAMENTA DE CÁLCULO DE ESTIMATIVAS DE SOFTWARE

Por

ALEXANDRE WENDERLICH

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Paulo Roberto Dias, Orientador, FURB
Membro:	Prof. Everaldo Artur Grahl, Mestre – FURB
Membro:	Prof. Oscar Dalfovo, Doutor – FURB

Blumenau, 10 de julho de 2008

Dedico este trabalho a todos os familiares, amigos e professores, especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

A Deus, por todas as graças concedidas.

À minha família, por todo o carinho e compreensão.

Aos meus amigos, pela força, empurrões e cobranças.

Ao meu orientador, Paulo Roberto Dias, por ter acreditado na conclusão deste trabalho.

Ao professor Everaldo Artur Grahl pelas sugestões.

Bom senso é ver as coisas como elas são e fazê-las como devem ser feitas.

Josh Billings

RESUMO

Este trabalho apresenta o desenvolvimento de uma ferramenta para calcular estimativas de projetos de software utilizando as métricas de *Function Point Analysis* (FPA), *Use Case Points* (UCP) e *Constructive Cost Model* (COCOMO). Além desta ferramenta auxiliar na produção de estimativas, permite controlar os projetos através de consultas e relatórios. Informações complementares podem ser adicionadas conforme a necessidade sem necessidade de revisar tudo que já foi cadastrado. A ferramenta também permite recalibragem dos parâmetros das métricas em caso de necessidade ou revisão técnica das mesmas.

Palavras chaves: Métrica. Estimativa de software. FPA. UCP. COCOMO.

ABSTRACT

This paper presents the development of a tool to calculate estimates for projects of software using the metrics of Function Point Analysis (FPA), Use Case Points (UCP) and Constructive Cost Model (COCOMO). In addition to this tool help in the production of estimates, to control the project through consultations and reports. Additional information may be added as needed without the need to review everything that has already been registered. It also allows calibration of the parameters of metrics in case of need or technical review of them.

Key-words: Metric. Measurement. Estimate. FPA. UCP. COCOMO.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS

AE – Aumento no Esforço

CGS – Características Gerais do Sistema

COCOMO – *Constructive Cost Model*

EE – Estimativa de Esforço

FA – Fator de Ajuste

FCA – Fator de Complexidade Ambiental

FCT – Fator de Complexidade Técnica

FPA – *Function Point Analysis*

FOP – estimativa de *Object Point*

IFPUG – *International Function Point User Group*

JCL – *Job Control Language*

KLOC – milhares de LOC

LOC – *Lines of Code*

MFP – Multiplicador Final do Produto

MER – Modelo de Entidade Relacionamento

NI – Nível de Influência

OO – Orientado a Objeto

PDF - *Portable Document Format*

PFA – Pontos de Função Ajustados

PFNA – Pontos por Função Não Ajustados

PMBOK – *Project Management Body of Knowledge*

PROD – Produtividade

TAN – Total de Atores não Ajustado

TEM – Total de Entidades não ajustado

TFA – Total de Fatores Ambientais

TFT – Total de Fatores Técnicos

TTN – Total de Transações Não Ajustado

UC – *Use Cases*

UCN – UCP Não ajustados

UCP – *Use Case Points*

UML – *Unified Modelling Language*

SUMÁRIO

1 INTRODUÇÃO

Um dos maiores problemas enfrentados no desenvolvimento de softwares é conseguir estimar os projetos de forma adequada. Quanto mais exige-se precisão, maior o nível de complexidade para se obtê-la. E não há como gerenciar algo que não se consegue medir.

> Com a globalização da economia e maior competitividade no mercado, as empresas tornam-se mais dependentes dos seus sistemas de informação. Construir esses sistemas em tempo hábil para serem úteis aos negócios e com qualidade adequada é o desafio que as organizações que desenvolvem software estão enfrentando. (TAVARES; CARVALHO; CASTRO, 2002, p. 1).

O termo estimativa é usado quando existe um conjunto mínimo de variáveis ou dados sobre um problema e aplica-se um modelo visando a prever seu estado em pontos representativos de sua evolução. Para que obtenha-se uma estimativa, assume-se que o modelo ou não tem condições de determinar precisamente o resultado desejado ou não possui variáveis ou dados suficientes para fazê-lo (MAGELA, 2006, p. 297).

Quando feita adequadamente, a medição de produtos e processos pode fornecer uma base efetiva para a iniciação e gerência de atividades de melhoria de processos (FLORAC; PARK; CARLETON, 1997, p. 21)

> No ano de 1979, em busca de métricas eficientes, A. J. Albrecht (IBM) considerou a utilização dos aspectos externos visíveis de um software para gerar uma nova métrica, conhecida como Pontos de Função, e que foi um dos primeiros métodos a medir e prever o desenvolvimento de software com alguma precisão. (TAVARES; CARVALHO; CASTRO, 2002, p. 2).

A métrica de *Function Point Analysis* (FPA), mede o tamanho do software pela quantificação de sua funcionalidade externa, baseada no projeto lógico ou a partir do modelo de dados. Abrange a funcionalidade específica requerida pelo usuário (TAVARES; CARVALHO; CASTRO, 2002, p. 3).

Essa métrica permite medir o tempo e o custo de um projeto de software e estimar quais os impactos que o projeto pode sofrer com a adição de novos requisitos ou recursos, permitindo melhor gerenciamento do projeto pela organização, obtendo-se assim maior exatidão no processo.

Agora, apesar da FPA ser a métrica de tamanho mais utilizada no mercado, muitos autores criticam a sua adequação à estimativa de projetos atuais, como os projetos Orientados a Objetos (OO), pois a FPA foi criada em 1979, com base nos conceitos das técnicas de análise e projetos estruturados, diferentes dos conceitos baseados na tecnologia OO (ANDRADE, 2004, p. 14).

Com o aumento do uso da tecnologia OO para o desenvolvimento de projetos de software, a estimativa passou a ser realizada também através de *Use Case Points* (UCP), uma métrica proposta em 1993 tendo como base FPA e usada para a estimativa de projetos de software orientados a objetos (ANDRADE, 2004, p. 14).

UCP explora o modelo e a descrição dos casos de uso, substitui algumas características técnicas propostas pela FPA, cria fatores ambientais e propõe uma estimativa. Só pode ser utilizado por empresas que adotem os casos de uso como forma de expressão dos requisitos (PEREIRA et al, 2007, p. 4).

Hoje, uma das métricas que conta com significativo volume de pesquisa é a do modelo *Constructive Cost Model* (COCOMO), editada em 1981 como resultado de estudos do Prof. Dr. Barry Boehm sobre 63 projetos de grande porte. É um método de estimativa voltado à produção. Esta métrica já está na sua segunda fase de desenvolvimento, conhecida como COCOMO II, reflexo do estado de amadurecimento das tecnologias e da engenharia de software, e está condizente com as tecnologias dos anos 90 e com pensamento focalizado nas próximas décadas (TRINDADE; PESSOA; SPINOLA, 1999, p. 2-3).

Então, surgiu a necessidade de elaborar uma ferramenta capaz de auxiliar as organizações no cálculo de estimativas de seus projetos de software, com *interface* web, que permita aos envolvidos no projeto fazer uso de FPA, UCP e COCOMO. Com base nos dados cadastrados por projeto na ferramenta pelos envolvidos, são produzidas análises através de consultas e relatórios.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver uma ferramenta de cálculo de estimativas de projetos de software, utilizando-se das métricas de FPA, UCP e COCOMO.

Os objetivos específicos do trabalho são:

a) permitir que os envolvidos no projeto, consigam acompanhar os projetos em andamento, e alimentar a ferramenta com novas informações quando necessário;
b) controlar o acesso à ferramenta de acordo com o perfil do usuário, disponibilizando apenas as opções sobre os quais o usuário possui direitos de acesso;
c) permitir a calibração das métricas caso seja necessário devido à publicação de

novos índices ou interpretação do usuário;

d) fornecer relatórios e consultas que auxiliem os desenvolvedores a estimarem seus projetos.

1.2 ESTRUTURA DO TRABALHO

No primeiro capítulo aborda-se a introdução do trabalho, os objetivos a serem alcançados no seu desenvolvimento e sua estrutura.

No segundo capítulo aborda-se a fundamentação teórica do presente trabalho. Descreve-se de forma breve a importância das estimativas de software e o papel das métricas na apuração destas estimativas. Apresenta-se uma visão geral sobre as métricas de FPA, UCP e COCOMO citando um breve histórico, as variáveis envolvidas, parâmetros e fórmulas, além de algumas considerações finais sobre cada uma. Também é apresentado uma breve descrição da ferramenta GeneXus usada na implementação deste trabalho. Por fim são apresentados trabalhos correlatos.

O terceiro capítulo contempla o desenvolvimento do trabalho, tais como, a especificação ilustrando o Modelo de Entidade Relacionamento – MER. Contém a implementação do sistema desenvolvido, descrevendo técnicas e ferramentas utilizadas.

Finalizando, o quarto capítulo descreve as considerações finais, incluindo também as sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

No presente capítulo são descritos alguns aspectos teóricos relacionados à ferramenta desenvolvida.

2.1 ESTIMATIVAS

Segundo Sommerville (2003, p. 437), estimativas podem ser necessárias para estabelecer um orçamento para o projeto ou para definir um preço do software para o cliente. Para Pressman (2002, p. 117), software é o elemento virtualmente mais caro de todos os sistemas baseados em computador. Para sistemas complexos, feitos sob medida, um erro de estimativa grande pode fazer a diferença entre o lucro e o prejuízo. Excesso de custo pode ser desastroso para o desenvolvedor.

Em grande parte dos casos, as estimativas são feitas com base na experiência passada. No caso de se ter um projeto relativamente similar a um projeto já realizado, não fica difícil estimar questões como esforço, cronograma e custo, uma vez que estes serão muito próximos daqueles relativos ao projeto anterior (PRESSMAN, 2002, p. 118).

A partir da manutenção de uma base de dados históricos estimados e realizados dos projetos, a organização pode avaliar a adequação do processo de desenvolvimento e extrair indicadores de produtividade e qualidade cada vez mais próximos de sua realidade e, portanto, cada vez mais confiáveis. Com isto as estimativas dos futuros projetos de software podem ser realizadas com segurança o mais cedo possível durante o ciclo de vida de desenvolvimento, ocasionando decisões mais rápidas e com um menor custo para a organização (VAZQUEZ; SIMÕES; ALBERT, 2007, p. 156).

Isto pode funcionar relativamente bem, se o presente projeto for bastante semelhante a esforços anteriores e as outras influências do projeto (exemplos: o cliente, os prazos) forem equivalentes. Infelizmente a experiência anterior nem sempre é um bom indicador de resultados futuros (PRESSMAN, 2002, p. 118).

Devido a particularidades inerentes a cada projeto de software, determinar com exatidão seu custo final e a data de sua conclusão é uma tarefa que só pode ser realizada quando ele está finalizado. Antes disso, tudo o que pode ser feito, até os cálculos mais

rigorosos, são estimativas. Seja qual for o método empregado, o que muda é a precisão e a acurácia dos resultados obtidos quando comparados aos dados reais (VAZQUEZ; SIMÕES; ALBERT, 2007, p. 156).

As estimativas de custo e esforço de software jamais serão uma ciência exata. Muitas variáveis, humanas, técnicas, ambientais e políticas podem afetar o custo final do software e do esforço aplicado para desenvolvê-lo. Entretanto, as estimativas de projetos de software podem ser transformadas de mágicas em uma série de passos sistemáticos que oferecem estimativas com riscos aceitáveis para os interessados (REZENDE, 2005, p. 296).

Segundo Sommerville (2003, p. 443), cada técnica de estimativa tem seus próprios pontos positivos e negativos. No caso de grandes projetos, é necessário utilizar várias técnicas de estimativa de custos e comparar seus resultados. Se elas predizem custos radicalmente diferentes, isso sugere que não se tem informações suficientes para fazer a estimativa. É necessário obter mais informações e repetir o processo de estimativa, até que as estimativas sejam convergentes.

Para Pressman (2002, p. 118), ferramentas automatizadas de estimativa implementam uma ou mais técnicas que, quando combinados com uma *interface* gráfica do usuário, fornecem uma opção atraente para a estimativa.

2.2 MÉTRICAS

Métrica é o método de determinar, quantitativamente, a extensão em que o projeto, o processo e o produto de software têm certos atributos. Isto inclui a fórmula para determinar o valor da métrica como também sua forma de apresentação e as diretrizes de utilização e interpretação dos resultados obtidos no contexto do ambiente de desenvolvimento de software (FERNANDES, 1995, p. 81).

Para Pressman (2002, p. 524) métricas de software fornecem uma maneira de avaliar a qualidade de atributos internos do produto, habilitando assim o engenheiro de software a avaliar a qualidade antes do produto a ser construído. As métricas fornecem a visão aprofundada necessária para criar modelos efetivos de análise e projeto, código sólido, e testes rigorosos.

Para Peters e Pedrycz (2001, p. 430), a métrica de software faz com que os engenheiros de software possam medir e prever os processos de software, os recursos

necessários a um projeto e os artefatos relevantes ao esforço do desenvolvimento.

Para ser útil no contexto do mundo real, uma métrica de software precisa ser simples e calculável, persuasiva, consistente e objetiva. Deve ser independente de linguagem de programação e fornecer efetiva realimentação para o engenheiro de software (PRESSMAN, 2002, p. 524).

Um processo de medição pode ser caracterizado por cinco atividades (PRESSMAN, 2002, p. 507):

a) formatação: a derivação de medidas e métricas de software que são adequadas para a representação do software que está sendo considerado;
b) coleta: mecanismo usado para acumular os dados necessários para originar as métricas formuladas;
c) análise: cálculo de métricas e aplicação das ferramentas matemáticas;
d) interpretação: avaliação dos resultados das métricas num esforço para ganhar profundidade na qualidade da representação;
e) realimentação: recomendações derivadas da interpretação das métricas técnicas transmitidas à equipe de software.

2.3 FPA

FPA é um método padrão para a medição do desenvolvimento de software, visando estabelecer uma medida do tamanho do software em pontos de função, com base na funcionalidade a ser implementada, sob o ponto de vista dos requisitos dos usuários. Seus objetivos são medir as funcionalidades do sistema requisitadas e recebidas pelo usuário e, medir os projetos de desenvolvimento e manutenção de software, sem se preocupar com a tecnologia que será utilizada na implementação (REZENDE, 2005, p. 296).

De acordo com a técnica de FPA, uma aplicação de software, vista sob a ótica do usuário, é um conjunto de funções ou atividades do negócio que o beneficiam na realização de suas tarefas (TAVARES; CARVALHO; CASTRO, 2002, p. 4-5).

A FPA foi desenvolvida em meados da década de 70 e apresentada por Allan Albrecht em 1979. Desde então, vem evoluindo, principalmente após a criação do grupo internacional de usuários de pontos por função (*International Function Point User Group* - IFPUG) em 1986. Em 2002, FPA passou ao patamar de padrão internacional, através da norma ISO/IEC

20926. A técnica de contagem dos elementos através de sua importância no sistema foi obtida através da análise de centenas de programas.

Por se tratar de uma medida de funcionalidade, pode ser feita no início do projeto, sem que se tenha uma linha sequer de código implementada, sendo dessa forma, muito útil nas estimativas de novos projetos ou alterações em sistemas. Apesar de não necessitar de código implementado, a modelagem precisa estar feita, para se ter acesso às funcionalidades do sistema.

FPA foi uma das primeiras métricas a medir o tamanho de um sistema com alguma precisão. Isto é possível, pois sua utilização é independente da linguagem de programação que será utilizada, pois como descrito anteriormente, mede a aplicação pelas funções que ela desempenhará (ANDRADE, 2004, p. 22).

Segundo Trindade (1999, p. 64) a capacidade de utilização da FPA para medição do tamanho de sistemas mesmo antes do seu desenvolvimento é uma das vantagens na utilização desta métrica. A característica de mensurar pela visão do usuário permite que o exercício de medida se estabeleça sobre caixas-pretas (modelo conceitual do sistema), sem exigir envolvimento com níveis de detalhe comportamental não naturais na fase de análise e modelagem.

FPA é aplicada fazendo a contagem de pontos de função considerando-se cinco elementos básicos de um sistema modelado. Antes de verificar cada um dos elementos, serão tratados os tipos de dados que são utilizados para a apuração dos pontos por função, constituindo as tabelas de complexidade de cada um dos cinco elementos, conforme definido pelo IFPUG.

Nas tabelas de apuração da complexidade, existem três níveis que podem ser atribuídos. Estes níveis classificam a complexidade entre baixa, média e alta, onde baixa significa que a implementação da funcionalidade é simples e alta, onde a sua implementação é mais complexa.

2.3.1 Tipos de Dados

A seguir serão tratados os tipos de dados que são utilizados para a apuração dos pontos por função, conforme definido pelo IFPUG.

2.3.1.1 Tipo de Elemento de Dado (TED)

Representa um campo único, não recursivo e que seja reconhecido pelos usuários. Como exemplos de TED, tem-se: campos de entrada de dados, botões do sistema, campos em relatórios, etc. TED é utilizado na apuração de complexidade de todos os cinco elementos da contagem de FPA.

2.3.1.2 Tipo de Elemento de Registro (TER)

É um conjunto de dados reconhecíveis pelos usuários e que identifiquem uma informação completa do sistema. Pode-se citar como exemplo, uma tupla de uma tabela do sistema reconhecida pelo usuário. TER é utilizado na apuração de Arquivos Lógicos Internos e Arquivos de *Interfaces* Externas, que são tratados a seguir.

2.3.1.3 Tipo de Arquivo Referenciado (TAR)

São conjuntos de registros do sistema, tanto arquivos manipulados pelo sistema, quanto arquivos somente utilizados pelo mesmo. Podem ser identificados como Arquivos Lógicos Internos ou como Arquivos de *Interfaces* Externas.

2.3.2 Elementos utilizados para a apuração de FPA

A seguir uma breve descrição dos cinco elementos básicos de um sistema modelado.

2.3.2.1 Arquivo Lógico Interno (ALI)

Arquivos lógicos internos, são os arquivos utilizados e controlados pela própria aplicação que está sendo mensurada. Estes arquivos devem estar estruturados logicamente, ou

seja, devem estar apresentados de forma que os usuários percebam a sua função e a sua importância para o funcionamento do sistema.

Alguns exemplos de ALI:

a) arquivos-mestre do sistema (dados utilizados por mais de um módulo do sistema);
b) tabelas do sistema;
c) arquivos de configuração;
d) arquivos de *backup* (desde que especificados pelo usuário para atender necessidades do sistema);
e) arquivos de mensagens de erros;
f) arquivos de segurança da aplicação.

Dentre os arquivos que não devem ser considerados como ALI, podem ser citados:

a) arquivos temporários;
b) arquivos de trabalho;
c) arquivos de *backup* (não solicitados pelo usuário);
d) arquivos de configuração do sistema em função das tecnologias utilizada no seu desenvolvimento;
e) arquivos de índices de acesso a dados.

Para atribuir valores de complexidade aos ALI, do sistema, utiliza-se o Quadro 1 como referência.

Registros	Dados		
	1 a 19 TEDs)	20 a 50 TEDs	> 50 TEDs
1 TER	Baixa	Baixa	Média
2 a 5 TERs	Baixa	Média	Alta
> 5 TERs	Média	Alta	Alta

Quadro 1 – Complexidade atribuída aos ALI

2.3.2.2 Arquivo de *Interface* Externa (AIE)

Arquivos de *interfaces* externas**,** são os arquivos utilizados por mais de uma aplicação, ou seja, são arquivos de integração entre sistemas. Estes arquivos devem ser identificados pelo usuário como sendo um requerimento do sistema e devem ser somente utilizados pela aplicação que está sendo medida, ficando sua manutenção a cargo da aplicação externa a qual

o sistema se integrará.

Alguns exemplos de AIE:

a) arquivos-mestre de outras aplicações;
b) dados de outras aplicações utilizados pelo sistema;
c) arquivos de mensagens de erros ou de auxílio não modificados pela aplicação a ser implementada.

Da mesma forma dos ALI, existem arquivos que não devem ser classificados como AIE, conforme exemplos:

a) arquivos recebidos de aplicações externas, mas que sejam utilizados para alterar dados da própria aplicação, pois são considerados Entradas Externas;
b) arquivos acessados em outras aplicações, mas que sofrem alteração e manutenção por parte da aplicação mensurada;
c) arquivos trabalhados pela aplicação para serem utilizados em outros sistemas, pois se classificam como saídas externas.

Para atribuir a complexidade dos AIEs, são utilizados os mesmos valores de ALI (ver Quadro 1).

2.3.2.3 Entrada Externa (EE)

Entradas externas, são todas as entradas de dados no sistema, sejam elas feitas pelos usuários, por processamentos em lote (*batchs*) ou por aplicações externas e que modificam os ALI do sistema.

Alguns exemplos de EE:

a) entradas de dados feitas pelos usuários;
b) entradas de dados oriundas de processamento de *batchs*;
c) *interfaces* necessárias para as entradas de dados.

A quantidade de arquivos alterados pelas entradas e também as interfaces necessárias para possibilitar estas entradas, são os fatores aos quais serão auferidas as complexidades (Quadro 2).

Arquivos envolvidos	Dados		
	1 a 4 TED(s)	5 a 15 TEDs	> 15 TEDs
0 ou TAR	Baixa	Baixa	Média
2 TARs	Baixa	Média	Alta
> 2 TARs	Média	Alta	Alta

Quadro 2 – Complexidade atribuída aos IE

2.3.2.4 Saída Externa (SE)

Saídas externas, ao contrário das EE, são todas as saídas de dados do sistema, tanto para a utilização por parte dos usuários, quanto por parte de outras aplicações.

Alguns exemplos de SE:

a) relatórios do sistema;
b) mensagens de aviso;
c) arquivos de dados exportados para uso em outras aplicações.

Exemplos de arquivos que não são considerados SE:

a) telas de *help* do sistema;
b) relatórios criados pelos usuários do sistema.

Da mesma maneira das SEs, a quantidade de arquivos envolvidos na geração da saída são contados e recebem seu grau de complexidade, porém, os valores diferem dos utilizados na contagem de EE (Quadro 3).

2.3.2.5 Consulta Externa (CE)

Consultas externas, são resultantes da combinação de *inputs* e *outputs*, quando uma entrada gera uma saída imediata, sem a atualização dos ALI da aplicação. Para contabilizar CE, é necessário enquadrar tanto a entrada, quanto a saída gerada (Quadros 2 e 3). Após realizar a verificação da complexidade de ambos, serão contados os pontos por função e considerados somente os pontos do item de maior complexidade.

Arquivos envolvidos	Dados		
	1 a 5 TED(s)	6 a 19 TEDs	> 19 TEDs
0 ou 1 TAR	Baixa	Baixa	Média
2 a 3 TARs	Baixa	Média	Alta
> 3 TARs	Média	Alta	Alta

Quadro 3 – Complexidade atribuída aos SE

Alguns exemplos de CE:

a) *interfaces* de consulta dos dados da aplicação;
b) relatórios de geração imediata;
c) telas de geração de relatórios;
d) telas de *help* e acesso ao sistema.

Alguns exemplos de não enquadramento como CE:

a) mensagens de erro ou comunicação com os usuários;
b) telas de *help* acessadas através do sistema, mas que não fazem parte do mesmo, pois são acessados independente da aplicação;
c) controles de acesso ao sistema sem nenhum controle de segurança (telas de *login* do sistema sem a necessidade de uso de senhas);
d) *interfaces* que possibilitam a navegação pelos módulos da aplicação sem exibição de informações.

De acordo com as complexidades apuradas para cada elemento, são atribuídos os números de pontos de acordo com a complexidade (Quadro 4).

2.3.3 Pontos por Função Não Ajustados (PFNA)

Após apurar os atributos e lhes atribuir devidos pesos de acordo com suas complexidades, soma-se os pontos por função identificados e obtém-se assim, uma medida de complexidade do sistema que será criado. Denomina-se o resultado desta apuração como a contagem de PFNA.

Elemento	Complexidade	Pontos por função
ALI	Baixa	7
	Média	10
	Alta	15
AIE	Baixa	5
	Média	7
	Alta	10
EE	Baixa	3
	Média	4
	Alta	6
SE	Baixa	4
	Média	5
	Alta	7
CE	Baixa	3
	Média	4
	Alta	6

Quadro 4 – Pontos por função atribuídos de acordo com as complexidades

2.3.4 Características Gerais do Sistema (CGS)

Existem quatorze características para realizar o ajuste dos PFNA ao ambiente de desenvolvimento, acrescentando ou reduzindo pontos da contagem inicial obtida. Para cada um deles, é atribuído um peso de acordo com a sua influência no sistema. Esses pesos devem variar na escala de 0 a 5. Quando o fator tiver pouca ou nenhuma influência no sistema, atribui-se peso igual a 0 (zero) e 5 quando o fator tiver grande influência no sistema (VAZQUEZ; SIMÕES; ALBERT, 2007, p. 59, 117-130).

A seguir, seguem as quatorze CGS, sua descrição e os detalhes de quando se deve utilizar cada peso, de acordo com a sua interferência.

2.3.4.1 Comunicação de Dados

Descreve o nível em que a aplicação comunica-se diretamente com o processador. Os dados ou informações de controle utilizados pela aplicação são enviados ou recebidos através de recursos de comunicação. Protocolo é um conjunto de convenções que permitem a transferência ou intercâmbio de informações entre dois sistemas ou dispositivos. Todos os

links de comunicação necessitam de algum tipo de protocolo (Quadro 5).

Peso	Descrição
0	A aplicação é puramente *batch* ou uma estação de trabalho isolada.
1	A aplicação é *batch*, mas possui entrada de dados ou impressão remota.
2	A aplicação é *batch*, mas possui entrada de dados e impressão remota.
3	A aplicação possui coleta de dados *on-line*, *front-end* de teleprocessamento para um processamento *batch* ou sistema de consulta.
4	A aplicação é mais que um *front-end*, mas suporta apenas um tipo de protocolo de comunicação.
5	A aplicação é mais que um *front-end*, e suporta mais que um tipo de protocolo de comunicação.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 5 – Pesos para comunicação de dados

2.3.4.2 Processamento Distribuído

Descreve em que nível a aplicação transfere dados entre seus componentes (Quadro 6).

Peso	Descrição
0	A aplicação não participa na transferência de dados ou processamento de funções entre os componentes do sistema.
1	A aplicação prepara dados para processamento pelo usuário final em outro componente do sistema, como planilhas eletrônicas ou banco de dados.
2	Dados são preparados para transferência, então são processados em outro componente do sistema (não para processamento pelo usuário final).
3	Processamento distribuído e transferência de dados são feitos em linha e em apenas uma direção.
4	Processamento distribuído e transferência de dados são feitos em linha e em ambas as direções.
5	Os processamentos de funções são executados dinamicamente no componente mais apropriado do sistema.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 6 – Pesos para processamento distribuído

2.3.4.3 Performance

Descreve em que nível os requisitos estabelecidos pelo usuário, sobre tempo de resposta, influenciam no projeto, desenvolvimento, instalação e suporte da aplicação (Quadro 7).

Peso	Descrição
0	O usuário não estabeleceu nenhum requisito especial sobre performance.
1	Requisitos de performance e projeto foram estabelecidos e revisados, mas nenhuma ação em especial foi tomada.
2	Tempo de resposta ou taxa de transações são críticos durante as horas de pico. Não é necessário nenhum projeto especial para a utilização de CPU. O limite para o processamento é o dia seguinte.
3	Tempo de resposta ou taxa de transações são críticos durante todas as horas de trabalho. Não foi necessário nenhum projeto especial para a utilização de CPU. O limite de processamento é crítico.
4	Adicionalmente, requisitos especificados pelo usuário são exigentes o bastante para que tarefas de análise de performance sejam necessárias na fase de projeto.
5	Adicionalmente, ferramentas de análise de performance devem ser utilizadas nas fase de projeto, desenvolvimento e/ou implementação para que os requisitos de performance do usuário sejam atendidos.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 7 – Pesos para performace

2.3.4.4 Configuração Altamente Utilizada

Descreve em que nível restrições computacionais influenciam no desenvolvimento da aplicação. Por exemplo, o usuário deseja executar a aplicação em um equipamento já existente ou comprado e que será altamente utilizado (Quadro 8).

Peso	Descrição
0	Não existem restrições operacionais implícitas ou explícitas nos requisitos.
1	Existem restrições operacionais, mas são menos restritivas que uma aplicação típica. Não há esforço especial necessário ao atendimento destas restrições.
2	Existem restrições operacionais, mas são restrições típicas da aplicação. Há esforço especial necessário ao atendimento dessas restrições.
3	Existem requisitos específicos de processador para uma parte específica da aplicação.
4	Restrições operacionais explícitas necessitam de um processador dedicado ou utilização pesada do processador central.
5	Adicionalmente, existem limitações na aplicação nos componentes distribuídos do sistema.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 8 – Pesos para configuração altamente utilizada

2.3.4.5 Volume de Transações

Descreve em que nível o alto volume de transações influencia o projeto, desenvolvimento, instalação e suporte da aplicação (Quadro 9).

Peso	Descrição
0	Não é antecipado nenhum período de pico de transações.
1	São previstos períodos de pico de processamento (Ex.: Picos mensal, quinzenal, periódico, anual), mas o impacto no esforço do projeto é mínimo.
2	Volumes de transação regulares (Ex.: Picos semanais) são previstos. Há algum impacto no esforço do projeto.
3	Altos volumes de transação (Ex. Picos diários) são previstos, conseqüentemente com impacto significativo no esforço do projeto.
4	Altas taxas de transação definidos pelo usuário nos requisitos ou os níveis de serviço acordados são altos o bastante para requererem tarefas de análise de performance na fase de projeto.
5	Adicionalmente, existem requisitos de ferramentas de análise de performance nas fases de projeto, desenvolvimento e/ou instalação.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 9 – Pesos para volume de transações

2.3.4.6 Entrada de Dados On-line

Descreve em que nível são efetuadas entradas de dados na aplicação por meio de transações interativas (Quadro 10).

Peso	Descrição
0	Todas as transações são processadas em lote.
1	De 1% a 7% das transações são entradas de dados *on-line*.
2	De 8% a 15% das transações são entradas de dados *on-line*.
3	De 16% a 23% das transações são entradas de dados *on-line*.
4	De 24% a 30% das transações são entradas de dados *on-line*.
5	Mais de 30% das transações são entradas de dados *on-line*.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 10 – Pesos para entrada de dados o*n-line*

2.3.4.7 Eficiência do Usuário Final

As funções *on-line* fornecidas pela aplicação enfatizam um projeto para o aumento da eficiência do usuário final (Quadro 11). O projeto inclui: auxílio para navegação como, por exemplo:

a) teclas de função;
b) saltos, menus gerados dinamicamente;
c) menus;
d) ajuda *on-line* e documentação;
e) movimentação automática de cursor;

f) paginação;

g) impressão remota através de transações *on-line*;

h) teclas de função pré-definidas;

i) tarefas em lote submetidos de transações *on-line*;

j) *drop-down list box*;

k) uso intenso de vídeo reverso, brilho, cores e outros indicadores;

l) *interface* de *mouse*;

m) janelas *pop-up*;

n) utilização de número mínimo de telas para executar uma função do negócio;

o) suporte a dois idiomas (conte como 4 itens);

p) suporte a mais de dois idiomas (conte como 6 itens);

q) impressão de documentação.

Peso	Descrição
0	Nenhum dos itens ao lado.
1	De um a três dos itens ao lado.
2	De quatro a cinco dos itens ao lado.
3	Seis ou mais dos itens ao lado, mas não existem requisitos específicos do usuário associados a eficiência.
4	Seis ou mais dos itens ao lado, e requisitos explícitos sobre a eficiência para o usuário final são fortes o bastante para necessitarem de tarefas de projeto incluírem fatores humanos como minimizar o número de batidas no teclado, maximizar padrões de campo e uso de *templates*.
5	Seis ou mais dos itens ao lado e requisitos explícitos sobre a eficiência para o usuário final são fortes o bastante para necessitarem do uso de ferramentas e processos especiais para demonstrar que os objetivos foram alcançados.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 11 – Pesos para eficiência do usuário final

2.3.4.8 Atualização *On-Line*

Descreve em que nível os arquivos lógicos internos são atualizados de forma *on-line* (Quadro 12).

Peso	Descrição
0	Não há nenhuma atualização *on-line*.
1	Existe a atualização *on-line* de um a três arquivos de controle. Volume de atualização é pequeno e a recuperação é fácil.
2	Existe a atualização *on-line* de quatro ou mais arquivos de controle. Volume de atualização é pequeno e a recuperação é fácil.
3	A atualização da maioria dos arquivos internos é *on-line*.
4	Adicionalmente, a proteção contra a perda de dados é essencial e foi especialmente projetada e programada no sistema.
5	Adicionalmente, o alto volume de processamento torna necessária a análise do custo do processo de recuperação. São incluídos procedimentos altamente automatizados com um mínimo de intervenção do operador.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 12 – Pesos para atualização o*n-line*

2.3.4.9 Processamento Complexo

Descreve em que nível o processamento lógico ou matemático influencia o desenvolvimento da aplicação (Quadro 13).

Os seguintes componentes estão presentes:

a) controle sensível (por exemplo, processamento especial de auditoria) e/ou processamento específico de segurança da aplicação;
b) processamento lógico extensivo;
c) processamento matemático extensivo;
d) muito processamento de exceção resultando em transações incompletas que devem ser processadas novamente, por exemplo, transações incompletas em ATM em função de problemas de teleprocessamento, falta de dados ou problemas de edição;
e) processamento complexo para manipular múltiplas possibilidades de entrada e saída, como por exemplo, multimídia, ou independência de dispositivo.

Peso	Descrição
0	Nenhum dos itens ao lado.
1	Qualquer um dos itens ao lado.
2	Quaisquer dois itens ao lado.
3	Quaisquer três itens ao lado.
4	Quaisquer quatro itens ao lado.
5	Todos os cinco itens ao lado.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 13 – Pesos para processamento complexo

2.3.4.10 Reusabilidade

Descreve em que nível a aplicação e seu código foram especificamente projetadas, desenvolvidas, e suportadas para serem utilizadas em outras aplicações (Quadro 14).

Peso	Descrição
0	Não há código reutilizável.
1	Código reutilizável é utilizado na aplicação.
2	Menos de dez por cento do código-fonte da aplicação foi construído levando em consideração o uso em mais de uma aplicação.
3	Menos de dez por cento do código-fonte da aplicação foi construído levando em consideração o uso em mais de uma aplicação.
4	A aplicação foi especificamente empacotada e/ou documentada para fácil reutilização, ela é customizada pelo usuário ao nível de código.
5	A aplicação foi especificamente empacotada e/ou documentada para fácil reutilização, ela é customizada pelo usuário através de manutenção de parâmetros.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 14 – Pesos para reusabilidade

2.3.4.11 Facilidade de Instalação

Um plano e/ou ferramentas de conversão e instalação foram fornecidos e testados durante a fase de teste do sistema (Quadro 15).

Peso	Descrição
0	O usuário não definiu considerações especiais, assim como não é requerido nenhum setup para a instalação.
1	O usuário não definiu considerações especiais, mas é necessário setup para a instalação.
2	Requisitos de instalação e conversão foram definidos pelo usuário, e guias de conversão e instalação foram fornecidos e testados. Não é considerado importante o impacto da conversão.
3	Requisitos de instalação e conversão foram definidos pelo usuário, e guias de conversão e instalação foram fornecidos e testados. É considerado importante o impacto da conversão.
4	Além do item 2 acima, ferramentas de instalação e conversão automáticas foram fornecidas e testadas.
5	Além do item 3 acima, ferramentas de instalação e conversão automáticas foram fornecidas e testadas.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 15 – Pesos para facilidade de instalação

2.3.4.12 Facilidade de Operação

Descreve em que nível a aplicação atende a alguns aspectos operacionais como: inicialização, segurança e recuperação. A aplicação minimiza a necessidade de atividades manuais, como montagem de fitas, manipulação de papel e intervenção manual pelo operador (Quadro 16).

2.3.4.13 Múltiplos Locais

Descreve em que nível a aplicação foi especificamente projetada, desenvolvida e suportada para diferentes ambientes de hardware e software (Quadro 17).

Peso	Descrição
0	Não foram estabelecidas pelo usuário outra consideração que não os procedimentos de segurança normais.
1-4	Um, alguns ou todos os seguintes itens são válidos para a aplicação. Selecione todos aqueles que sejam válidos. Cada item tem um valor de um ponto, a exceção de onde seja citado o contrário: - procedimentos de inicialização, salva e recuperação foram fornecidos, mas é necessária a intervenção do operador; - procedimentos de inicialização, salva e recuperação foram fornecidos, e não é necessária a intervenção do operador (conte como dois itens); - a aplicação minimiza a necessidade de montagem de fitas; - a aplicação minimiza a necessidade de manipulação de papel.
5	Aplicação projetada para operação não assistida. Isto é, não é necessária nenhuma intervenção do operador para operar o sistema, que não seja a inicialização e término da aplicação. A recuperação automática de erros é uma característica da aplicação.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 16 – Pesos para facilidade de operação

Peso	Descrição
0	Os requisitos do usuário não consideram a necessidade de mais de um usuário/local de instalação.
1	Necessidade de múltiplos locais foi considerada no projeto, e a aplicação foi projetada para operar apenas nos mesmos ambientes de hardware e software.
2	Necessidade de múltiplos locais foi considerada no projeto, e a aplicação foi projetada para operar apenas ambientes de hardware e software similares.
3	Necessidade de múltiplos locais foi considerada no projeto, e a aplicação foi projetada para operar ambientes diferentes de hardware e software.
4	Adicionalmente aos itens 1 e 2, plano de suporte e documentação são fornecidos e testados para suportar a aplicação em múltiplos locais.
5	Adicionalmente ao item 3, plano de suporte e documentação são fornecidos e testados para suportar a aplicação em múltiplos locais.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).

Quadro 17 – Pesos para múltiplos locais

2.3.4.14 Facilidade de Mudanças

Descreve em que nível a aplicação foi especificamente desenvolvida para facilitar a mudança de sua lógica de processamento ou estrutura de dados (Quadro 18). As seguintes características podem ser válidas para a aplicação:

a) são fornecidos mecanismos de consulta e reporte flexível, que permitem a manipulação de pedidos simples; por exemplo, lógica de e/ou aplicada a apenas um arquivo lógico (conte como um item);
b) são fornecidos mecanismos de consulta e reporte flexível, que permitem a

manipulação de pedidos de média complexidade; por exemplo, lógica de e/ou aplicada a mais de um arquivo lógico (conte como dois itens);

c) são fornecidos mecanismos de consulta e reporte flexível, que permitem a manipulação de pedidos complexos; por exemplo, lógica de e/ou combinadas em um ou mais arquivos lógicos (conte como três itens);
d) dados de controle do negócio são mantidos pelo usuário através de processos interativos, mas as alterações só tem efeito no próximo dia útil;
e) dados de controle do negócio são mantidos pelo usuário através de processos interativos, e as alterações tem efeito imediato (conte como dois itens).

Peso	Descrição
0	Nenhum dos itens ao lado.
1	Qualquer um dos itens ao lado.
2	Quaisquer dois itens ao lado.
3	Quaisquer três itens ao lado.
4	Quaisquer quatro itens ao lado.
5	Todos os cinco itens ao lado.

Fonte: adaptado de Fattos Consultoria e Sistemas (2008, p. 2).
Quadro 18 – Pesos para facilidade de mudanças

2.3.5 Pontos de Função Ajustados (PFA)

Com a soma dos pesos atribuídos a cada uma das CGS, obtém-se um Nível de Influência (NI), que será utilizado no ajuste dos PFNA. Para tal, gera-se um Fator de Ajuste (FA) (Quadro 19).

FA = 0,65 + (0,01 * NI)

Quadro 19 – Fórmula do FA

O fator de ajuste, quando aplicado ao PFNA, pode produzir uma variação de 35% para mais ou para menos, resultando na contagem final dos PFA (VAZQUEZ; SIMÕES; ALBERT, 2007, p. 118) (Quadro 20).

PFA = PFNA * FA

Quadro 20 – Fórmula do PFA

Após aplicar o FA aos PFNA, descobre-se o PFA e encerra-se a apuração do número de pontos por função do sistema que será implementado.

2.3.6 Considerações finais de FPA

Quando a técnica de FPA foi apresentada por Allan Albrecht, em 1979, não haviam as CGS para determinar o FA. O FA era determinado de forma totalmente subjetiva, o qual poderia produzir uma variação de 25% para mais ou para menos nos PFNA. Na revisão técnica, em 1984, foram introduzidas as quatorze CGS e o fator de ajuste foi alterado para produzir a variação de 35% para mais ou para menos (VAZQUEZ; SIMÕES; ALBERT, 2007, p. 119).

Algumas das vantagens da utilização da FPA são (VAZQUEZ; SIMÕES; ALBERT, 2007, p. 51):

a) é uma ferramenta que possibilita determinar o tamanho de um pacote adquirido através da contagem de todas as funções incluídas;
b) provê auxílio aos usuários na determinação dos benefícios de um pacote para sua organização, através da contagem das funções que especificamente correspondem aos seus requisitos;
c) suporta a análise de produtividade e qualidade, seja diretamente, ou em conjunto com outras métricas como esforço, defeitos e custo;
d) apóia o gerenciamento de escopo de projetos;
e) complementa o gerenciamento dos requisitos ao auxiliar na verificação da solidez e complementação dos requisitos especificados;
f) é um meio de estimar custo e recursos para o desenvolvimento e manutenção de software;
g) é uma ferramenta para fundamentar a negociação de contratos;
h) é um fator de normalização para comparação de software ou para a comparação da produtividade na utilização de diferentes técnicas.

FPA oferece uma medida padronizada e normalizada do tamanho funcional dos requisitos lógicos dos usuários e, juntamente com outras medidas, podem ilustrar vários aspectos do processo de desenvolvimento de software, de modo a ensejar melhorias. (DEKKERS, 1998, p. 1-8).

FPA provê um mecanismo para acompanhar e monitorar o aumento do escopo de um projeto. As contagens de pontos de função ao fim do levantamento de requisitos do software até a sua implementação podem ser comparadas (LONGSTREET, 2000).

Apesar de ser a métrica mais utilizada, a FPA é muito criticada pela sua aplicação em

projetos orientados a objetos (cada vez mais utilizados atualmente), pois foi criada na década de 70 e baseada em técnicas de análise e de projeto da época de sua criação, que são diferentes das atuais (ANDRADE, 2004, p. 30).

Segundo ANJOS (2003, p. 46) as limitações e desvantagens da FPA são as seguintes:

a) para ter uma boa utilização é necessária uma base histórica;
b) é necessário ter uma boa visão (profundidade do sistema para poder estimar com menos insegurança);
c) não é muito boa para medir esforço de manutenção (correção de problemas);
d) utilização de pesos para definir a classificação das funções;
e) há variações das métricas, é preciso saber qual a versão da métrica, quando se vai medir tamanho do software.

2.4 UCP

UCP é um método de estimativa de projeto de software orientado a objetos proposto em 1993 por Karner, com base na FPA e no modelo de casos de uso. O UCP também estima o tamanho da função do software de acordo com a visão do usuário final (ANDRADE, 2004, p. 32).

UCP somente pode ser aplicada em projetos de software cuja especificação tenha sido expressa em casos de uso. Nela conta-se os atores e os casos de uso e identifica-se sua complexidade, para então proceder com o cálculo inicial, determinação de complexidades e então chegar ao resultado final (HAZAN; FUKS; LUCENA, 2005, p. 11-12).

Os *Use Cases* (UC) são técnicas baseadas em cenários para a obtenção de requisitos (SOMMERVILLE, 2003, p. 113). Ainda conforme este autor, na atualidade, os UC figuram como um dos modelos primordiais na notação da linguagem de modelagem unificada (*Unified Modelling Language* - *UML*) para a descrição dos modelos de sistemas OO.

Considerando a importância dos UC no desenvolvimento de softwares OO, existe a possibilidade de se medir custos e prazos, baseando-se na análise dos UCP. Segundo Andrade (2004, p. 32), esta técnica foi criada em 1993 por Gustav Karner, baseando-se em FPA, *Mark II* e nos modelos de UC. Conforme esta mesma autora, Karner analisou os modelos e descrições de UC, fez algumas modificações nas regras propostas pela FPA e criou com isso, uma proposta de estimativa de produtividade baseada em UC.

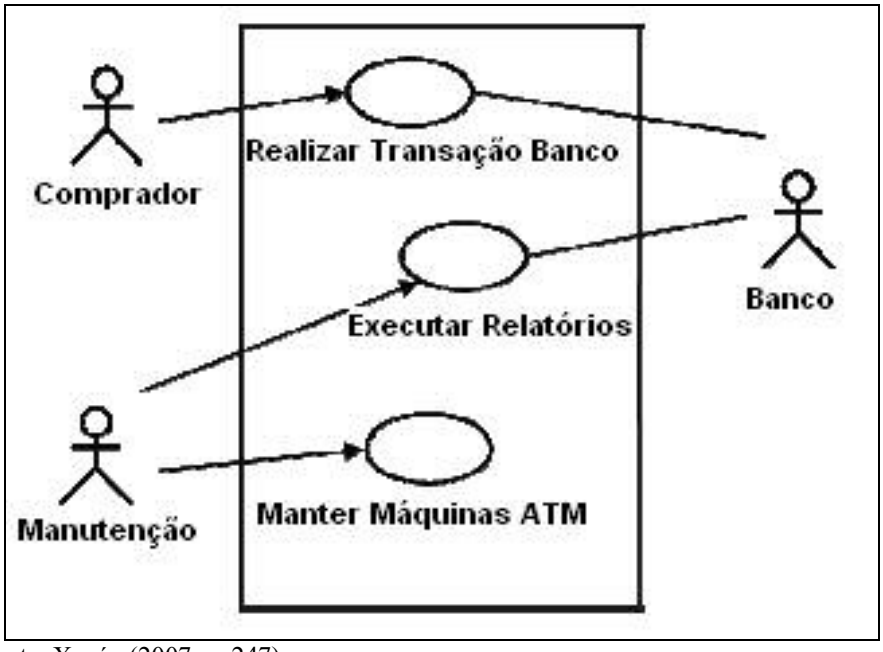
Como esta técnica surgiu baseando-se fortemente em FPA, elementos como PFNA e a utilização de fatores de ajuste estão presentes em UCP. (ROCHA, 2005, p. 27).

Assim como em FPA, UCP baseia-se na funcionalidade do sistema pela visão dos usuários. Karner propôs uma produtividade de 20 homens/hora para implementar um UCP. Este número é contestado por alguns autores, pois segundo eles, este valor está diretamente ligado aos fatores de ajuste que são utilizados na contagem. Como exemplos de autores que discutem este tema, podem ser citados: Schneider, Winters e Sparks (apud ANDRADE, 2004, p. 38).

A principal diferença entre FPA e UCP está relacionada ao momento da coleta de informações para a estimativa de projeto. A FPA possui resultados melhores à medida em que se tem mais informação da análise e do projeto de sistemas (tabelas, campos, associações, etc). Já UCP tem como proposta ser utilizado logo no início do ciclo de desenvolvimento, na fase de definição dos requisitos, com base no modelo de casos de uso. O modelo de UC é uma técnica largamente utilizada na indústria para capturar e descrever os requisitos de um software, consistindo de diagramas e descrições de UC (ANDRADE, 2004, p. 14).

Segundo Andrade (2004, p. 33), esta métrica estimula a utilização de medições de software, pois é um método simples, fácil e rápido de usar.

Na Figura 1, apresenta-se um exemplo de diagrama de UC.



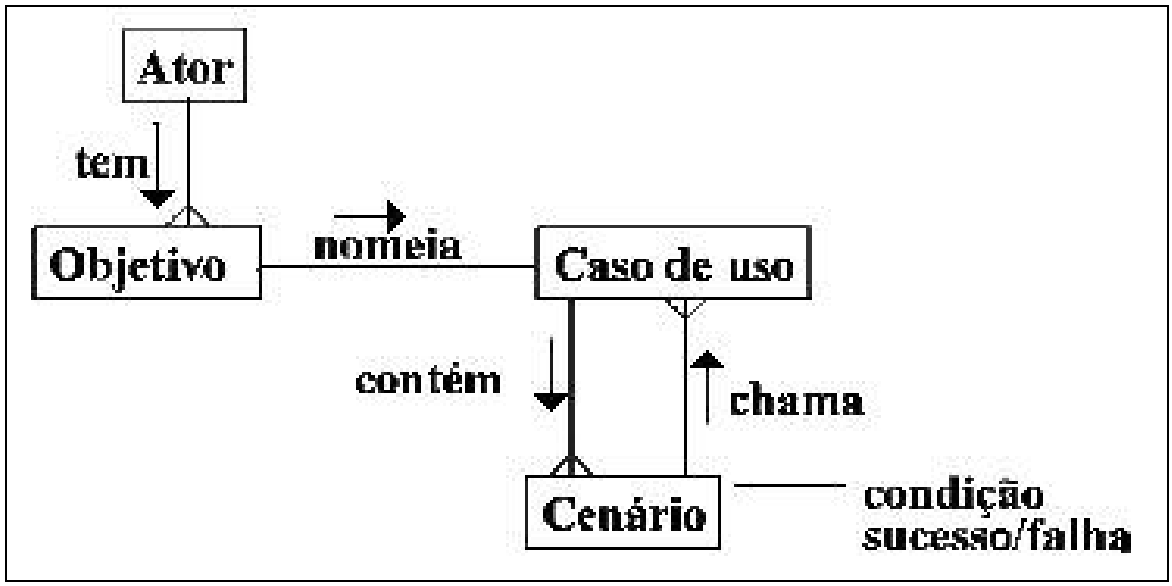
Fonte: Xexéo (2007, p. 247).

Figura 1 – Exemplo de diagrama de UC

2.4.1 Elementos

UC são criados utilizando alguns elementos para descrever as funcionalidades do sistema que serão representadas.

Na Figura 2 apresenta-se um diagrama com os principais elementos de UC.



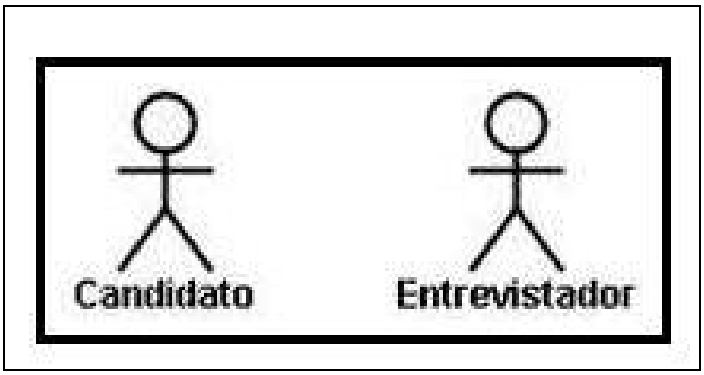
Fonte: Xexéo (2007, p. 243).

Figura 2 – Diagrama com principais elementos de UC

A seguir, uma breve descrição de alguns destes elementos.

2.4.1.1 Atores

Segundo Xexéo (2007, p. 240), são entidades externas ao sistema e que se comportam de forma própria, interagindo com o mesmo. Conforme este autor, exemplos de atores podem ser: pessoas, outros sistemas, organizações, etc. A representação de um ator num diagrama de UC é exibida na Figura 3.



Fonte: Xexéo (2007, p. 240).
Figura 3 – Exemplo de representação de atores em um UC

2.4.1.2 Objetivos

Como o próprio nome diz, são os objetivos, tanto dos atores, quanto do próprio UC. Podem existir vários objetivos num UC, como por exemplo: enviar currículo, marcar entrevista, realizar seleção, contatar pretendentes a vagas, dentre outros, baseando-se nos atores da Figura 1.

2.4.1.3 Cenários

Descrevem as situações do UC, definindo as atitudes que podem ser tomadas de acordo com o resultado de determinadas ações ou condições. Conforme Xexéo (2007, p. 241) existem cenários principais, alternativos, de falhas, dentre outros.

2.4.2 Passos para medir UCP

Para medir UCP, existem seis passos a serem seguidos, conforme Andrade (2004, p. 33-36), Rocha (2005, p. 27-30) e Vieira (2007, p. 41-43), os quais são descritos a seguir.

2.4.2.1 Verificar atores envolvidos e atribuir os devidos pesos

Consiste em verificar os atores do(s) caso(s) de uso analisados e auferir pesos de acordo com a complexidade dos mesmos, que podem variar entre simples, média e complexos. Multiplicando-se os atores pelos pesos, obtém-se o Total de Atores Não Ajustado (TAN) (Quadro 21).

Complexidade do ator	Descrição	Peso
Simples	Representa outro sistema acessado através de *interface* definida.	1
Médio	Representa outro sistema acessado através de algum protocolo de comunicação ou através de interação humana.	2
Complexo	Um usuário que interage através de uma *interface* gráfica, tanto numa aplicação *desktop*, quanto numa aplicação web.	3

Fonte: adaptado de Andrade (2004, p. 33) e Rocha (2005, p. 27).

Quadro 21 – Pesos dos atores de UC

2.4.2.2 Contagem e atribuição dos pesos dos casos de uso

De acordo com o número de transações de cada UC, atribui-se os pesos de acordo com a complexidade, apurando-se o Total de Transações Não Ajustado (TTN). Segundo Rocha (2005, p. 28), transação pode ser definida como uma série de processos que devem ocorrer conjuntamente, ou então, serem cancelados em sua totalidade caso um dos processos falhar. Baseado nesta definição atribui-se os pesos de acordo com a quantidade de transações dos casos de uso (Quadro 22).

Complexidade do UC	Descrição	Peso
Simples	1 a 3 transações incluindo os passos alternativos e deve ser realizado com menos de 5 classes de análise.	5
Médio	4 a 7 transações incluindo os passos alternativos e deve ser realizado com 5 a 10 classes de análise.	10
Complexo	> 7 transações incluindo os passos alternativos e deve ser realizado com pelo menos de 10 classes de análise.	15

Fonte: adaptado de Andrade (2004, p. 34).

Quadro 22 – Pesos dos UC em função das transações

A medição de complexidade dos UC pode ser feita também em função das entidades que participam do mesmo (ROCHA, 2005, p. 32). Este autor define entidade como sendo

cada conceito de negócio ou objeto do mundo real que se encontre no UC que está sendo medido. Um conceito pode ser uma idéia, uma coisa, ou um objeto, como por exemplo: dinheiro, saque, banco e saldo. Desta forma, são apresentadas as complexidades e pesos em função do número de entidades envolvidas. Ao realizar a contagem, cabe ao executor da mesma decidir se analisará a complexidade em função das transações ou das entidades. Através desta atribuição, apura-se o Total de Entidades Não Ajustado (TEN) (Quadro 23).

Complexidade do UC	Descrição	Peso
Simples	1 a 5 entidades	1
Médio	6 a 10 entidades	2
Complexo	> 10 entidades	3

Fonte: adaptado de Rocha (2005, p. 29).

Quadro 23 – Pesos dos UC em função das entidades envolvidas

Vale lembrar que os pesos atribuídos podem variar de acordo com a interpretação do autor.

2.4.2.3 Apurar UCP não ajustados

A apuração dos UCP Não Ajustados (UCN) se dá através da soma dos pesos atribuídos de acordo com a complexidade dos atores envolvidos nos UC e de acordo com os pesos atribuídos às entidades ou transações dos mesmos. Nos Quadros 24 e 25 apresentam-se as fórmulas de apuração de acordo com o critério adotado.

UCN = TAN + TTN

Quadro 24 – Fórmula de UCN usando TTN

UCN = TAN + TEN

Quadro 25 – Fórmula de UCN usando TEN

Como citado anteriormente, a simples contagem de UCN não pode ser utilizada, pois os critérios ambientais e técnicos não são levados em conta. Os itens a seguir destacam estes critérios, seus pesos e fórmulas de cálculo, que levarão à apuração de UCP. Cada um dos critérios deve ser avaliado e receber um peso que varia de 0 a 5. O peso 0 (zero) é atribuído quando o fator tem pouca ou nenhuma influência no sistema e o peso 5 é atribuído aos fatores de grande influência no sistema.

2.4.2.4 Determinação do fator de complexidade técnica

O Quadro 27 explicita os fatores de complexidade técnica e seus respectivos pesos.

Estes fatores consideram as dificuldades técnicas de implementação do sistema. Para cada fator deve ser atribuído o peso de 0 a 5 e multiplicar este peso atribuído pelo peso do fator de acordo com a tabela. Após realizar esta operação com todos os fatores, somam-se os resultados e obtém-se o Total de Fatores Técnicos (TFT). Com a posse deste total, executa-se a seguinte fórmula, que define o Fator de Complexidade Técnica (FCT) (Quadro 26).

FCT = 0.6 + (0.01 * TFT)

Quadro 26 – Fórmula de FCT

Fator	Peso
Sistemas Distribuídos	2.0
Desempenho da aplicação	1.0
Eficiência do usuário final (on-line)	1.0
Processamento interno complexo	1.0
Reusabilidade do código em outras aplicações	1.0
Facilidade de instalação	0.5
Usabilidade (facilidade operacional)	0.5
Portabilidade	2.0
Facilidade de manutenção	1.0
Concorrência	1.0
Características especiais de segurança	1.0
Acesso direto para terceiros	1.0
Facilidades especiais de treinamento	1.0

Fonte: Karner, apud Andrade (2004, p. 35).

Quadro 27 – Fatores de complexidade técnica e seus pesos

2.4.2.5 Determinação do fator de complexidade ambiental

Os fatores ambientais são relacionados à experiência da equipe que desenvolverá o sistema. Esta experiência se refere à habilidade com a linguagem de programação, entrosamento da equipe, disponibilidade dos funcionários, dentre outros aspectos. Os fatores ambientais e seus pesos são listados no Quadro 29. A apuração ocorre da mesma forma dos aspectos técnicos, atribuindo-se notas de 0 a 5, multiplicando pelo peso do atributo na tabela e somando todos os resultados dos fatores. Através da soma de todos os fatores, obtém-se o Total de Fatores Ambientais (TFA). Aplicando a seguinte fórmula, é calculado o Fator de Complexidade Ambiental (FCA) apresentado no Quadro 28.

FCA = 1.4 + (-0.03 * TFA)

Quadro 28 – Fórmula de FCA

Fator	Peso
Familiaridade com o Processo de Desenvolvimento de Software	1.5
Experiência na Aplicação	0.5
Experiência com OO, na Linguagem e na Técnica de Desenvolvimento	1.0
Capacidade do Líder de Projeto	0.5
Motivação	1.0
Requisitos estáveis	2.0
Trabalhadores com dedicação parcial	-1.0
Dificuldade da Linguagem de Programação	-1.0

Fonte: Karner, apud Andrade (2004, p. 36).

Quadro 29 – Fatores de complexidade ambiental e seus pesos

2.4.2.6 Cálculo dos UCP ajustados

O cálculo final, que define os UCP ajustados, é feito multiplicando-se os UCN pelos fatores de ajuste técnico e ambiental (Quadro 30).

UCP = UCN * FCT * FCA

Quadro 30 – Fórmula de UCP

2.4.3 Considerações finais de UCP

Finalizando esta métrica, cabe ressaltar a análise de Andrade (2004, p. 40), sobre a utilização de UCP. Através do estudo de alguns autores, esta autora concluiu que UCP não é tão amplamente utilizada, pois:

a) trata-se de uma métrica relativamente nova (se comparada com FPA, por exemplo);
b) ainda não alcançou nível de padronização;
c) não está incorporada em ferramentas disponíveis de medição de software;
d) ainda não possuem bases históricas de produtividade.

Ainda baseando-se nos estudos de Andrade (2004, p. 40-41), pode-se afirmar que os utilizadores de UCP têm obtido sucesso, tanto em experiências de medição, quanto em aplicações em sistemas reais e que as medições feitas com esta técnica vem alcançado valores próximos aos dados reais da implementação dos sistemas onde têm sido aplicadas.

2.5 COCOMO

O método de modelo de COCOMO, foi inicialmente proposto por Boehm em 1981 (MAGELA, 2006, p. 307).

COCOMO propõe-se a medir esforço, prazo, tamanho da equipe e custo envolvidos no desenvolvimento de software, tendo como premissa a dimensão do mesmo, fornecida em número de instruções-fonte. Este autor salienta que, com relação à variável custo, neste modelo não há uma forma direta de dimensioná-lo. Porém, tendo conhecimento de prazo e equipe de trabalho, torna-se possível estimar um valor (TRINDADE; PESSOA; SPINOLA, 1999, p. 3).

2.5.1 COCOMO 81

COCOMO 81 estima o esforço em pessoas-meses de trabalho (considerando 152 h/mês) e toma como principal fator de esforço o LOC, expressas em milhares de instruções-fonte disponibilizadas (KLOC). (PETERS; PEDRYCZ, 2001, p. 487) e (SOMMERVILLE, 2003, p. 447). Estas instruções-fonte incluem todas as instruções de programação e instruções de controle de trabalho de linguagem (*Job Control Language* - JCL). Não estão incluídos os comentários e os softwares utilitários não alterados. Ainda conforme estes autores, o COCOMO apóia-se em duas suposições: primeiramente, ligada ao modelo em cascata clássica de desenvolvimento de software e como segunda suposição, assume bons padrões de gerenciamento, sem tempo ocioso.

De acordo com Sommerville (2003, p. 446) este modelo sugere que o desenvolvimento do software se dê em cascata (surgindo ‘a partir do zero’), porém, aconteceram mudanças desde esta proposição. Softwares são implementados utilizando componentes reutilizáveis, são interligados por linguagens de roteiro, utilizam prototipação, desenvolvimento incremental e, em muitos casos, subsistemas de prateleira para a composição do software.

Para Peters e Pedrycz (2001, p. 487), o processo geral de elaboração de modelos envolve três tipos de sistemas:

a) embutido: caracterizada por restrições rigorosas e um ambiente mutável e pouco conhecido. Projetos desta natureza normalmente apresentam restrições temporais. Exemplos: sistemas de software em tempo real;

b) orgânico: abrange sistemas pequenos, no tocante ao tamanho do projeto e da equipe envolvida no mesmo. Exemplos: sistemas comerciais simples, de processamento de dados ou pequenas bibliotecas de software;
c) geminado: são uma mistura entre sistemas dos tipos embutido e orgânico. Exemplos: sistemas operacionais, de gerência de bancos de dados e de controle de estoque.

De forma geral, COCOMO obedece à fórmula de Estimativa de Esforço (EE) (Quadro 31) (MAGELA, 2006, p. 306).

$EE = a * TP^b * FA$

Fonte: adaptado de Magela (2006, p. 306).

Quadro 31 – Fórmula de estimativa de esforço

A seguir temos os componentes da fórmula de EE:

a) TP: representa o tamanho do projeto em KLOC;
b) a: representa um valor dependente da própria empresa e do tipo de software que está sendo desenvolvido;
c) b: reflete o tamanho do projeto.
d) FA: fator de ajuste.

Desnecessário dizer que não possuímos o tamanho KLOC do projeto em seu início. Portanto, embora sua aplicação seja simples e fácil, seu resultado é completamente imprevisível (MAGELA, 2006, p. 307).

Segundo Magela (2006, p. 307-309) e Fernandes (1995, p.138), existem 3 modelos neste método, que serão apresentados a seguir.

2.5.1.1 Modelo Básico

Versão aplicável a grande maioria de projetos de software de porte pequeno ou médio desenvolvidos internamente, apesar da limitação de não considerar fatores de restrição de hardware, de qualificação e experiência do pessoal de desenvolvimento, bem como emprego de técnicas modernas.

Neste modelo os valores de ‘a’, ‘b’ e FA são fixos (Quadro 32).

Tipo	FA	a	b
Embutido	1	3,6	1,20
Orgânico	1	2,4	1,05
Geminado	1	3,0	1,12

Fonte: Magela (2006, p. 307).

Quadro 32 – Valores modelo básico do COCOMO 81

Salienta-se que estes formatos e seus respectivos parâmetros resultam de ajustes de dados obtidos em caráter experimental, advindos de projetos de software anteriores que foram analisados na criação do modelo. Uma limitação imposta por esta metodologia diz respeito ao fato de que certos parâmetros podem não se adequar a projetos diferentes daqueles utilizados como base de criação do modelo (PETERS; PEDRYCZ, 2001, p. 488).

2.5.1.2 Modelo Intermediário

Adiciona fatores ao método básico como restrições de hardware, qualificação e experiência do pessoal. Está associado a um conjunto de direcionadores de custo, que incluem avaliações subjetivas do produto, do pessoal envolvido no desenvolvimento, das características e recursos do projeto, o que demonstra amplo detalhamento e capacidade mais acurada de estimativas.

Neste modelo os valores de ‘a’ sofreram modificações (Quadro 32).

Tipo	a	b
Embutido	2,8	1,20
Orgânico	3,2	1,05
Geminado	3,0	1,12

Fonte: Magela (2006, p. 307).

Quadro 33 – Valores modelo intermediário do COCOMO 81

O FA é obtido via multiplicação de um conjunto de quinze fatores e seus determinados pesos (Quadro 35). Cada fator possui uma variação de pesos de acordo com a sua influência no sistema (Quadro 34).

2.5.1.3 Modelo Detalhado

Incorpora ambas as versões. Apresenta técnicas para se estimar tanto a nível de módulo, subsistema e sistema individualizado por fase do projeto.

VL	Very Low	muito baixa
LO	Low	baixa
NM	Nominal	padrão
HI	High	alta
VH	Very High	muito alta
XH	Xtreme High	extra alta

Quadro 34 – Graus de influência no sistema

		Pesos					
Atributo	Descrição	VL	LO	NM	HI	VH	XH
RELY	Confiabilidade Necessária	0,75	0,88	1,00	1,15	1,40	-
DATA	Bytes de dados por IFD	-	0,94	1,00	1,08	1,16	-
CPLX	Complexidade	0,70	0,85	1,00	1,15	1,30	1,65
TIME	Tempo de Execução	-	-	1,00	1,11	1,30	1,66
STOR	Restrições de Memória	-	-	1,00	1,06	1,21	1,56
VIRT	Volatilidade da Máquina Virtual	-	0,87	1,00	1,15	1,30	-
TURN	Tempo de Resposta do Desenvolvimento	-	0,87	1,00	1,07	1,15	-
ACAP	Capacidades de Análise	1,46	1,19	1,00	0,86	0,71	-
AEXP	Experiência em Aplicações	1,29	1,13	1,00	0,91	0,82	-
PCAP	Capacidades de Programação	1,42	1,17	1,00	0,86	0,70	-
LEXP	Experiência em Linguagens	1,14	1,07	1,00	0,95	-	-
VEXP	Experiência em Máquina Virtual	1,21	1,10	1,00	0,90	-	-
MODP	Modernas Práticas de Desenvolvimento	1,24	1,10	1,00	0,91	0,82	-
TOOL	Ferramentas de *Software*	1,24	1,10	1,00	0,91	0,83	-
SCED	Efeitos do Cronograma	1,23	1,08	1,00	1,04	1,10	-

Fonte: adaptado de Sommerville (2003, p. 451) e Magela (2006, p. 308).

Quadro 35 – Parâmetros para cálculo do FA do modelo intermediário do COCOMO 81

2.5.2 COCOMO II

Já o modelo COCOMO II representa uma atualização do modelo COCOMO 81 e foi novamente apresentado por Boehm em 1995 (MAGELA, 2006, p. 309).

COCOMO II contempla as mudanças de abordagem no desenvolvimento de software anteriormente citados, como a prototipação e o desenvolvimento pela agregação de componentes. Este modelo mensura esforços, prazos e custos de maneira condizente com as tecnologias modernas, onde todas as fórmulas e parâmetros de análise resultam de regressões

matemáticas aplicadas sobre dados históricos coletados de projetos já desenvolvidos e conclusos, mesmo procedimento do COCOMO 81, porém com regressões não lineares (SOMMERVILLE, 2003, p. 447).

Com relação ao nível de modelo associado com as atividades no processo de software, aqueles que podem ser identificados no COCOMO II, segundo Sommerville (2003, p. 447) serão descritos a seguir.

2.5.2.1 Nível Inicial de Prototipação

Estimativas de tamanho são realizadas tomando como base pontos de objeto, sendo que o esforço requerido é estimado por uma fórmula simples de tamanho/produtividade (Quadro 36), onde é dividido o tamanho do software medido pela estimativa de *Object Point* (FOP) pela Produtividade (PROD).

EE = FOP / PROD

Fonte: Magela (2006, p. 309).

Quadro 36 – Fórmula de EE para o nível inicial de prototipação

O valor de PROD é fornecido pela média aritmética dos 2 fatores (Quadro 37).

Fator de Produtividade	VL	LO	NM	HI	VH
Capacitação e experiência dos desenvolvedores	4	7	13	25	50
Capacitação e maturidade da ferramenta CASE	4	7	13	25	50

Fonte: adaptado de Magela (2006, p. 309).

Quadro 37 – Fator de produtividade

2.5.2.2 Nível Inicial de Projeto

Refere-se à conclusão dos requisitos do sistema com a possibilidade de algum projeto inicial. Neste nível as estimativas são baseadas em FP convertidas para o LOC.

Neste nível de modelo retornamos à fórmula de EE demonstrada no COCOMO 81 (Quadro 31), porém a variável ‘a’ foi definida e estipulada para o valor de 2.5 e a variável TP é a medida em FP.

Após sua determinação, é utilizada uma tabela de conversão visando a transformar o FP em KLOC (Quadro 38).

Linguagem	LOC/FP
Ada	71
Assembly	320
C	128
C++	29
Pascal	91

Fonte: adaptado de Magela (2006, p. 310).

Quadro 38 – Valores típicos de conversão LOC/FP

A variável ‘b’ é obtida através de fórmula (Quadro 39) e reflete o Aumento no Esforço (AE) necessário de acordo com o tamanho do projeto (Quadro 40).

$b = 1{,}01 + 0{,}01 * \Sigma(AE)$

Fonte: Magela (2006, p. 310).

Quadro 39 – Fórmula para ‘b’

AE	VL	LO	NM	HI	VH
Experiência	4,0	3,2	2,4	1,6	0,8
Flexibilidade	6,0	4,8	3,6	2,4	1,2
Risco	4,2	3,4	2,5	1,8	0,8
Coesão	5,0	4,0	3,0	2,0	1,0
Processo	4,5	3,6	2,7	1,8	0,9

Fonte: Magela (2006, p. 310).

Quadro 40 – Parâmetros de AE

O FA será calculado de forma análoga ao modelo COCOMO 81, contudo utilizando apenas 7 atributos (Quadro 41).

Atributo	Descrição
RCPX	Disponibilidade e complexidade do software
RUSE	Reuso
PDIF	Dificuldade da plataforma
PERS	Capacitação profissional
PREX	Experiência profissional
SCED	Cronograma de desenvolvimento
FCIL	Facilidade de suporte

Fonte: adaptado de Magela (2006, p. 310).

Quadro 41 – Atributos para cálculo do FA do nível inicial do projeto

2.5.2.3 Nível Pós-arquitetura

Feita a arquitetura do sistema, pode-se realizar uma estimativa razoavelmente exata do tamanho do software. A estimativa desse nível faz uso de um conjunto mais amplo de multiplicadores, refletindo a capacidade pessoal, as características de produto e de projeto.

As variáveis ‘a’ e ‘b’ devem ser definidas de forma semelhante ao modelo anterior.

Já o FA possui agora 17 atributos (Quadro 42).

Atributo	Descrição	Pesos					
		VL	LO	NM	HI	VH	XH
RELY	Confiabilidade Necessária	0,75	0,88	1,00	1,15	1,39	-
DATA	Bytes de dados por IFD	-	0,93	1,00	1,09	1,19	-
CPLX	Complexidade	0,70	0,88	1,00	1,15	1,30	1,66
RUSE	Reuso	-	0,91	1,00	1,14	1,29	1,49
DOCU	Documentação	-	0,95	1,00	1,16	1,13	-
TIME	Tempo de Execução	-	-	1,00	1,11	1,31	1,67
STOR	Restrições de Memória	-	-	1,00	1,06	1,21	1,57
PVOL	Volatilidade da Plataforma de Desenvolvimento	-	0,87	1,00	1,15	1,30	-
ACAP	Capacidades de Análise	1,50	1,22	1,00	0,83	0,67	-
PCAP	Capacidades de Programação	1,37	1,16	1,00	0,87	0,74	-
PCON	Continuidade de Profissionais	1,24	1,10	1,00	0,92	0,84	-
AEXP	Experiência em Aplicações	1,22	1,10	1,00	0,89	0,81	-
PEXP	Experiência Programadores	1,14	1,07	1,00	0,95	-	-
LTEX	Experiência na Linguagem	1,22	1,10	1,00	0,91	0,84	-
TOOL	Ferramentas de Software	1,24	1,12	1,00	0,86	0,72	-
SITE	Disposição dos Ambientes de Desenvolvimento e a Qualidade de Comunicação entre eles	1,25	1,10	1,00	0,92	0,84	0,78
SCED	Efeitos do Cronograma	1,29	1,10	1,00	1,00	1,00	-

Fonte: adaptado de Magela (2006, p. 310).

Quadro 42 – Parâmetros para cálculo do FA do nível pós-arquitetura

2.5.3 Considerações finais de COCOMO

O modelo COCOMO é o mais concreto e documentado em termos de estimativas de esforço. Esta métrica baseia-se em uma análise de Boehm e fornece fórmulas que possibilitam determinar o cronograma do tempo de desenvolvimento, esforço geral envolvido no mesmo, assim como interrupção do esforço por fase/atividade e ainda o esforço de manutenção (PETERS; PEDRYCZ, 2001, p. 487).

Com relação às especificidades do COCOMO II, coloca que os resultados obtidos com o mesmo apontam para possibilidade de produção de estimativas com significativas reduções de esforços e que, quanto maior for o aprimoramento dos direcionadores de custo, maior será

a autonomia em esforços necessários. Ele postula que o ideal é que cada equipe de desenvolvimento em uma organização desenvolva seus próprios direcionadores de custo para melhoria da produtividade (TRINDADE; PESSOA; SPINOLA, 1999, p. 4).

Uma das grandes novidades do COCOMO II reside no fato deste modelo possibilitar que todos os fatores sejam alterados, quer por uma calibragem publicada pela equipe que desenvolve as pesquisas referentes ao modelo, quer por adequações da própria empresa que desenvolve softwares fazendo uso desta métrica, com base em suas experiências (TRINDADE; PESSOA; SPINOLA, 1999, p. 4).

Como ponto negativo, pode-se considerar que ainda há muito a desenvolver nessa métrica. Aspectos denotados como salto evolutivo, ou como grande vantagem do novo método, estão obscuros, sem uma definição clara de como serão conquistados (TRINDADE; PESSOA; SPINOLA, 1999, p. 15).

2.6 GENEXUS

GeneXus é um software de desenvolvimento de sistemas que permite uma visão integrada, ou seja, conhecer os processos associados aos dados. Os dados são modelados de uma visão da realidade, capturada pelo analista diretamente do usuário, produzindo um modelo em 3ª forma normal (GONDA; JODAL, p. 9.).

Como base teórica, o GeneXus está inspirado em Warnier e Orr, para o processo e estrutura dos dados, que constituem os objetos fundamentais e são muito compatíveis com a visão do usuário. A análise de dados e projeto da Base de Dados são baseados nas publicações de Bachman, Codd, Maier e Ullman, o que permite o desenvolvimento totalmente automático da Base de Dados. A especificação e geração dos programas foi baseada em Warnier e Orr para a estruturação de programas e algumas idéias sobre coesão foram obtidas de Yourdon e Constantine, Gane e de Marco (GONDA; JODAL, p. 15).

.Com isto, o GeneXus automatiza o processo de desenvolvimento de sistemas, permitindo desenhar, prototipar, gerar e manter automaticamente a base de dados e os programas.

Todo o desenvolvimento é feito em plataforma PC, gerando código fonte para rede de micros, AS/400, Windows e Unix.

GeneXus projeta e cria automaticamente uma base de dados na terceira forma normal,

como é proposto na teoria de banco de dados relacionais, partindo de definições de simples visões dos usuários.

Proporciona um ambiente radicalmente novo para desenvolvimento de aplicações em grande escala. Começando com a normalização e desenho de toda a base de dados até a geração automática dos programas, análises de impacto e a subseqüente regeração de programas.

GeneXus foi desenvolvido com a filosofia de usar o computador para automatizar tanto quanto possível o processo de desenvolvimento e implementação de sistema. Como resultado, os procedimentos GeneXus são inteiramente otimizados para explorar o que o computador faz de melhor - lidar com grandes quantidades de dados e relacionamentos complexos, liberando o analista para fazer aquilo que só ele consegue: localizar os problemas críticos associados com o modelo do negócio, comunicação com o usuário e o desenvolvimento de algoritmos complexos.

Na maioria das metodologias tradicionais, as modelagens de base de dados e normalização são descritos e automatizadas com simples processos. Mas isso só funciona quando há um número pequeno de entidades envolvidas. A normalização pode se tornar um assunto extremamente complicado para a maioria dos analistas quando o número de entidades se torna tão grande que é impossível concentrá-las na mesma tela ou em um mesmo diagrama.

Entretanto, com GeneXus é possível desenvolver bases de dados sem erros e completamente normalizadas, mesmo em aplicações muito grandes, por causa de sua característica de desenho automático da base de dados.

A habilidade do GeneXus para adaptar-se facilmente às mudanças, mesmo grandes mudanças, permite ao analista criar qualquer sistema incrementalmente. Pela primeira vez aplicações não precisam ser completamente definidas antes de começar. Usando GeneXus pode-se desenvolver uma aplicação em estágios, adicionando facilmente novos módulos ou elementos de dados conforme requisitados, crescendo aos poucos em complexidade e sofisticação.

A seguir serão demonstrados alguns tópicos que visam demonstrar o ciclo de desenvolvimento de software usando a ferramenta GeneXus (LISBOA, p. 16-164).

2.6.1 Relação com engenharia de software

GeneXus é um produto para desenhar e gerar 100% da aplicação. Utilizando uma

única base de conhecimento, GeneXus produz automaticamente a base de dados, os programas e a documentação da aplicação que está sendo desenvolvida. Esta base de conhecimento armazena os componentes da aplicação capturados das visões de cada usuário compreendendo regras, fórmulas, estruturas de dados, lay-out de telas, relatórios, etc. A partir destas informações e utilizando grande capacidade de inferência, GeneXus cria base de dados normalizada e a reorganiza sempre que necessário.

GeneXus simplifica o processo de desenvolvimento e manutenção da aplicação, que é construída e testada de forma interativa num PC. Conforme as mudanças são feitas, GeneXus as identifica e gera novamente os programas de aplicação e utilitários de reorganização da base de dados.

Quanto a documentação, toda a informação provida pelo analista ou inferida pelo GeneXus é armazenada em um relatório ativo que constitui um completo material de consulta on-line, permanentemente atualizado e sempre disponível.

2.6.2 Ciclo de desenvolvimento e manutenção

O Desenvolvimento e manutenção acontecem em quatro ambientes distintos, os quais serão descritos a seguir.

2.6.2.1 Desenho da aplicação (*Application Designer*)

Este componente permite ao analista coletar as visões do usuário e descrever as informações requeridas para a aplicação. Estas informações são definidas com base nas regras de negócio que são fáceis para o compreendimento do usuário, conforme a Figura 4.

Figura 4 – Especificações da transação no GeneXus

2.6.2.2 Prototipagem (*Prototype Manager*)

Um protótipo GeneXus é uma aplicação pronta, diferenciando-se da aplicação de produção apenas quanto a plataforma de execução. O protótipo permite que a aplicação seja totalmente testada antes de passar à produção. A aplicação pode ser construída e testada em etapas. A partir do teste realizado no protótipo, pode-se encontrar necessidades que não haviam sido previstas, ou falhas na interpretação da necessidade do usuário. Conforme as aplicações são implementadas, GeneXus avalia o impacto na base de dados (Figura 5) e nos programas afetados (Figura 6) fornecendo um relatório que é o resultado da análise de impacto por ele conduzida.

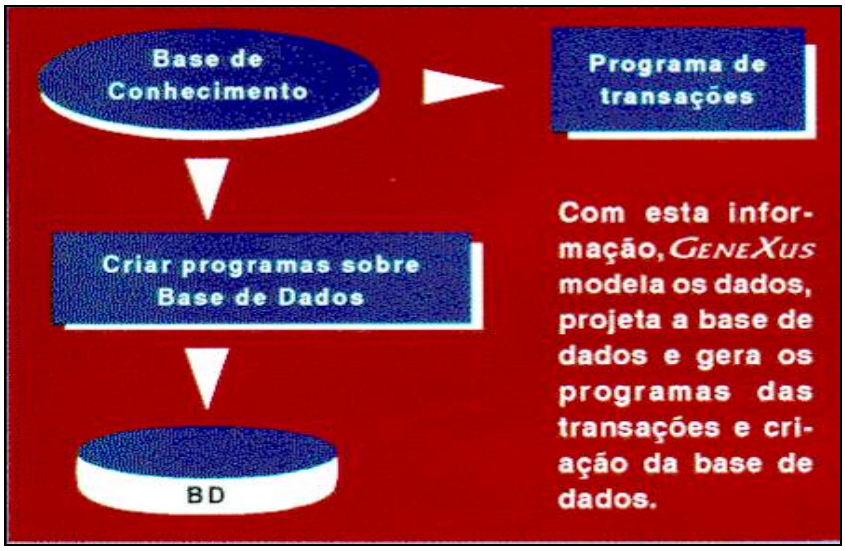


Figura 5 – Impacto na base de dados no GeneXus

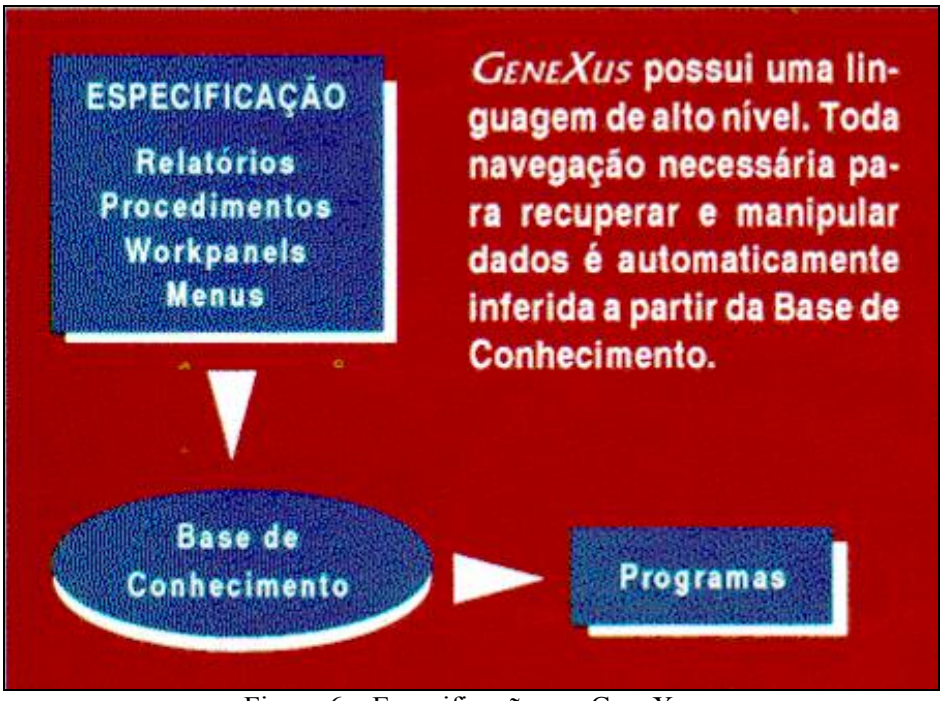


Figura 6 – Especificações no GeneXus

2.6.2.3 Ambiente de produção (*Production Manager*)

A aplicação de produção é automaticamente gerada a partir do protótipo final. Não requer nenhuma programação adicional. Um código isento de erros será referido para o computador destino onde será compilado, tanto para criação e manutenção das bases de dados (Figura 7), quanto para os programas (Figura 8).

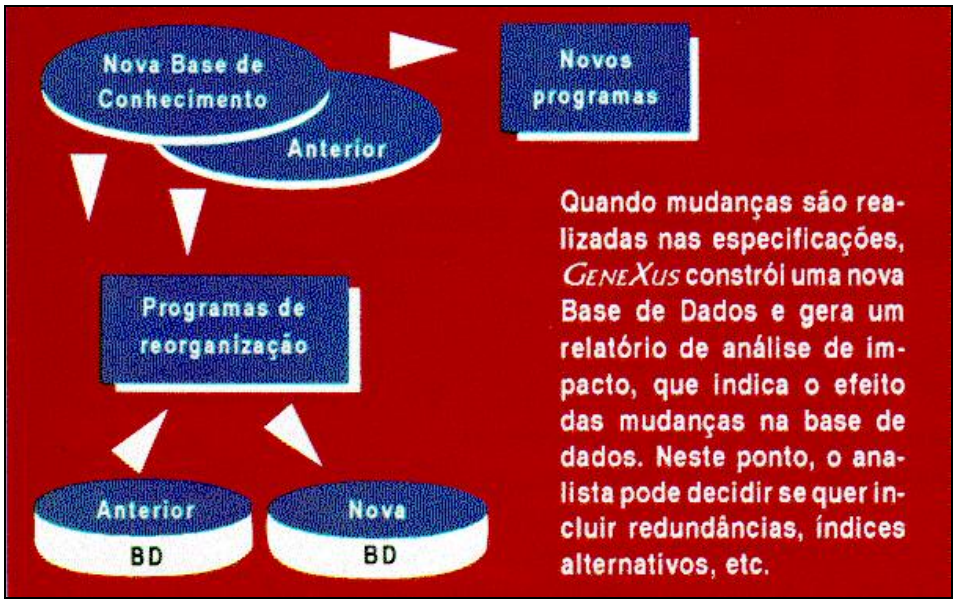


Figura 7 – Impacto na base de dados de produção no GeneXus

Figura 8 – Especificações de produção no GeneXus

2.6.2.4 Consolidação e distribuição do conhecimento (*Knowledge Manager*)

Um mesmo sistema pode ser desenvolvido por diversas pessoas. O módulo *Knowledge Manager* permite integrar e consolidar num modelo corporativo as partes desenvolvidas e testadas isoladamente. Da mesma forma, sistemas aplicativos completos ou partes destes, podem ser consolidados num sistema corporativo da organização. Vários módulos de uma

mesma aplicação podem ser desenvolvidos independentemente ao mesmo tempo. Analistas ou grupos podem desenvolver e prototipar seus próprios objetos específicos. Estes módulos individuais podem ser consolidados para criar uma única aplicação integrada, amplamente útil quando muitos analistas estão trabalhando no mesmo módulo ou sub-sistema.

2.7 TRABALHOS CORRELATOS

Chaves (2003, p. 1-46), apresenta um roteiro para aplicação de métricas e qualidade em sub-contratação de software. O autor visa analisar teoricamente os recursos e métodos que possam contribuir na avaliação de métrica e de análise de qualidade de softwares, baseando-se em FPA.

Já Pereira Junior (2003, p. 1-83) apresenta protótipo de uma ferramenta para gerência de custos em projetos de software baseada no modelo *Project Management Body of Knowledge* (PMBOK). O autor desenvolveu um estudo da aplicação dos conceitos de gerenciamento de projetos de software definidos pelo PMBOK — no âmbito da gerência de custos de projetos, aplicando também a FPA como métrica.

Também Gielow (2003, p. 1-71) apresenta uma ferramenta de suporte ao cálculo de estimativas, porém usando a métrica de UCP, a partir da leitura de diagramas gerados pela ferramenta CASE Rational Rose.

Valcanaia (1998, p. 1-126) apresenta um estudo sobre métricas de software, tendo como foco principal FPA, contemplando a especificação e implementação de um protótipo para contabilização e cálculo de pontos de função em sistemas desenvolvidos no banco de dados Access 97.

Por fim Berlanda (2005, p. 1-83) apresenta o desenvolvimento de um aplicativo para analisar e auxiliar na avaliação do custo para desenvolvimento de software utilizando FPA.

Diferentemente dos trabalhos anteriores que focam apenas nos aspectos teóricos ou na criação de ferramentas baseadas apenas em uma métrica, neste trabalho desenvolveu-se uma ferramenta que reúne ao mesmo tempo as métricas de FPA, UCP e COCOMO.

3 DESENVOLVIMENTO

Este capítulo descreve os principais requisitos, a especificação e a implementação da ferramenta proposta. Por fim, são apresentados os resultados e discussão.

Com base nos objetivos propostos por este trabalho, desenvolveu-se uma Ferramenta de Cálculo de Estimativas de Projetos de Software, denominada de SPE – Sofware Project Estimator.

3.1 REQUISITOS PRINCIPAIS DA FERRAMENTA

Os principais requisitos da ferramenta desenvolvida incluem:

a) permitir a calibração da ferramenta de acordo com os parâmetros publicados mais recentes para cada uma das métricas implementadas na ferramenta (FPA, UCP e COCOMO) (Requisito Funcional – RF);
b) permitir o cadastro de clientes que solicitaram projetos (RF);
c) permitir o cadastro e controle de mais de um projeto (RF);
d) identificar a métrica escolhida dentre as implementadas na ferramenta (FPA, UCP, ou COCOMO) a cada cadastro de projeto (RF);
e) verificar e disponibilizar somente as opções apropriadas à métrica selecionada para o projeto (RF);
f) permitir o cadastro dos dados do projeto pertinentes à métrica escolhida (RF);
g) analisar as entradas com base nas regras de cálculo da métrica escolhida (RF);
h) processar o cálculo de estimativas do projeto, de acordo com as regras e parâmetros calibrados para a métrica escolhida (RF);
i) fornecer relatórios em formato PDF com o cálculo das estimativas, variando de acordo com a métrica escolhida, permitindo a visualização, impressão e salvamento do arquivo (RF);
j) fornecer consultas em tela para as informações associadas a cada cadastro de forma a facilitar o acesso do usuário às informações e parâmetros (RF);
k) possuir *interface* web (Requisito Não Funcional - RNF);
l) fazer o controle de acesso à ferramenta, com senhas, permitindo acesso ao que for

liberado para o perfil do usuário informado (RNF);

m) ser implementado, utilizando-se do software GeneXus (RNF);

n) ser implementado, utilizando-se do banco de dados Oracle 9i (RNF);

o) ser implementado, utilizando-se do software Framework 2.0 (RNF);

p) ser compatível com o navegador de internet Microsoft Internet Explorer 6.0 e seus similares compatíveis (RNF).

3.2 ESPECIFICAÇÃO

Para a especificação do SPE foi usado o software Power Designer da Sybase, versão 12.0. O modelo escolhido foi o Modelo de Entidade Relacionamento (MER) (Figura 10).

Este modelo facilita muito para o desenvolvedor a criação da aplicação no software GeneXus. O próprio GeneXus fornece um diagrama com uma visão das tabelas (Figura 9).

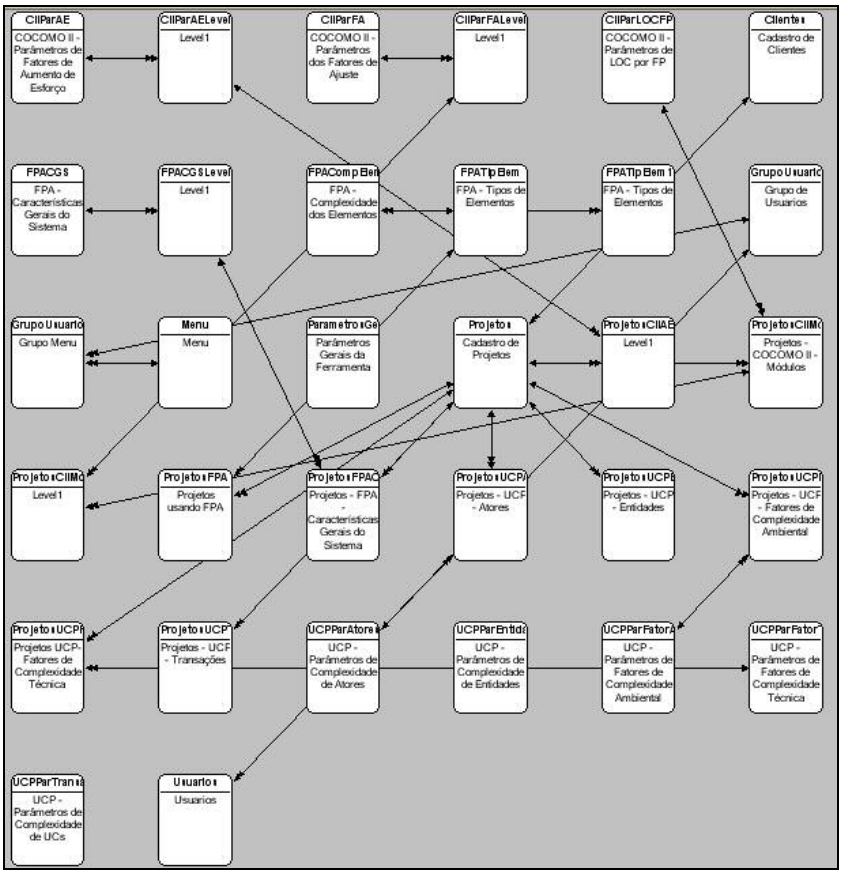


Figura 9 – Diagrama de tabelas no GeneXus

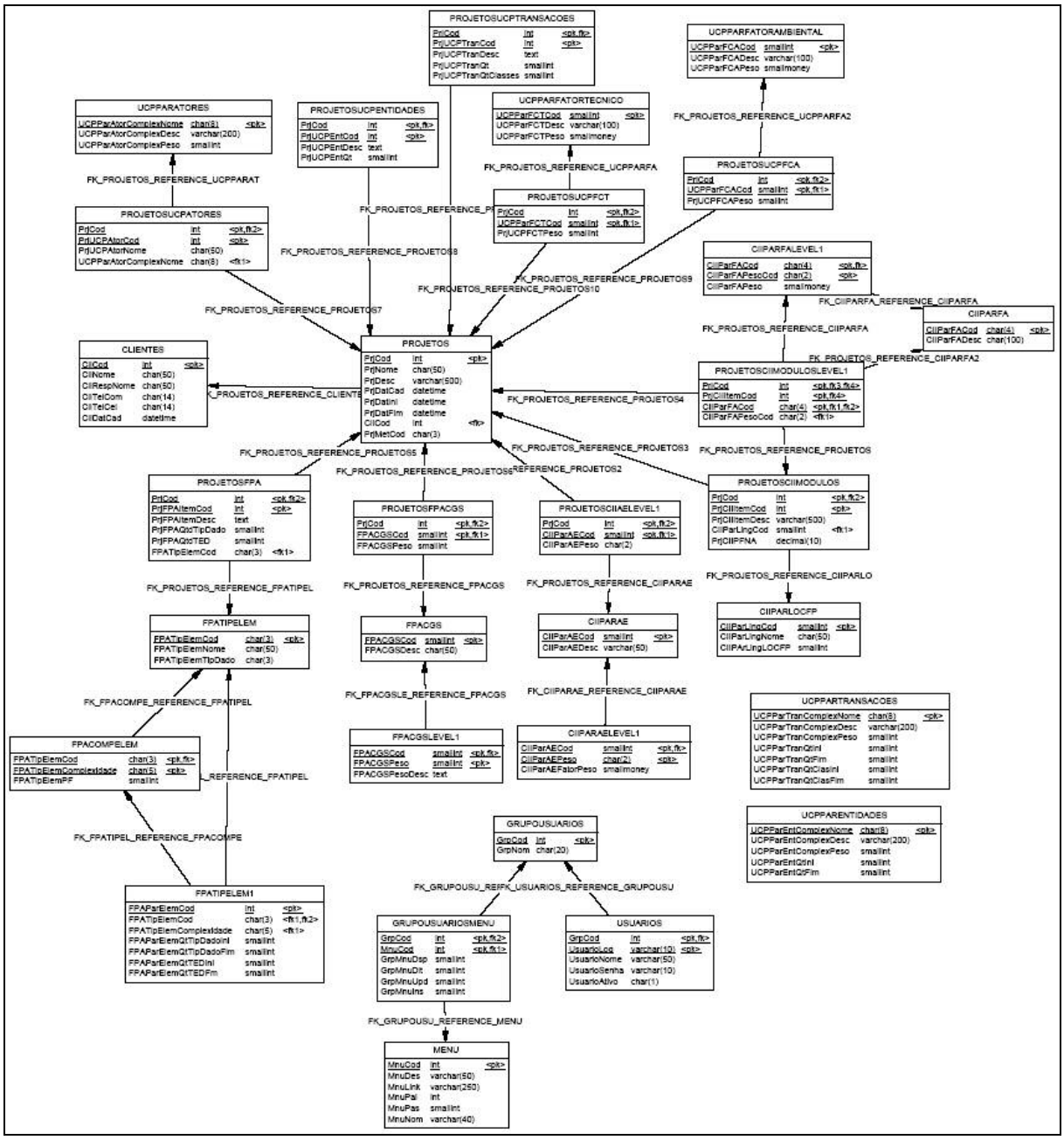


Figura 10 – MER do SPE

Como a visualização em uma única folha ficou prejudicada pela quantidade de tabelas presentes na ferramenta desenvolvida, a seguir são apresentadas as visões parciais do MER.

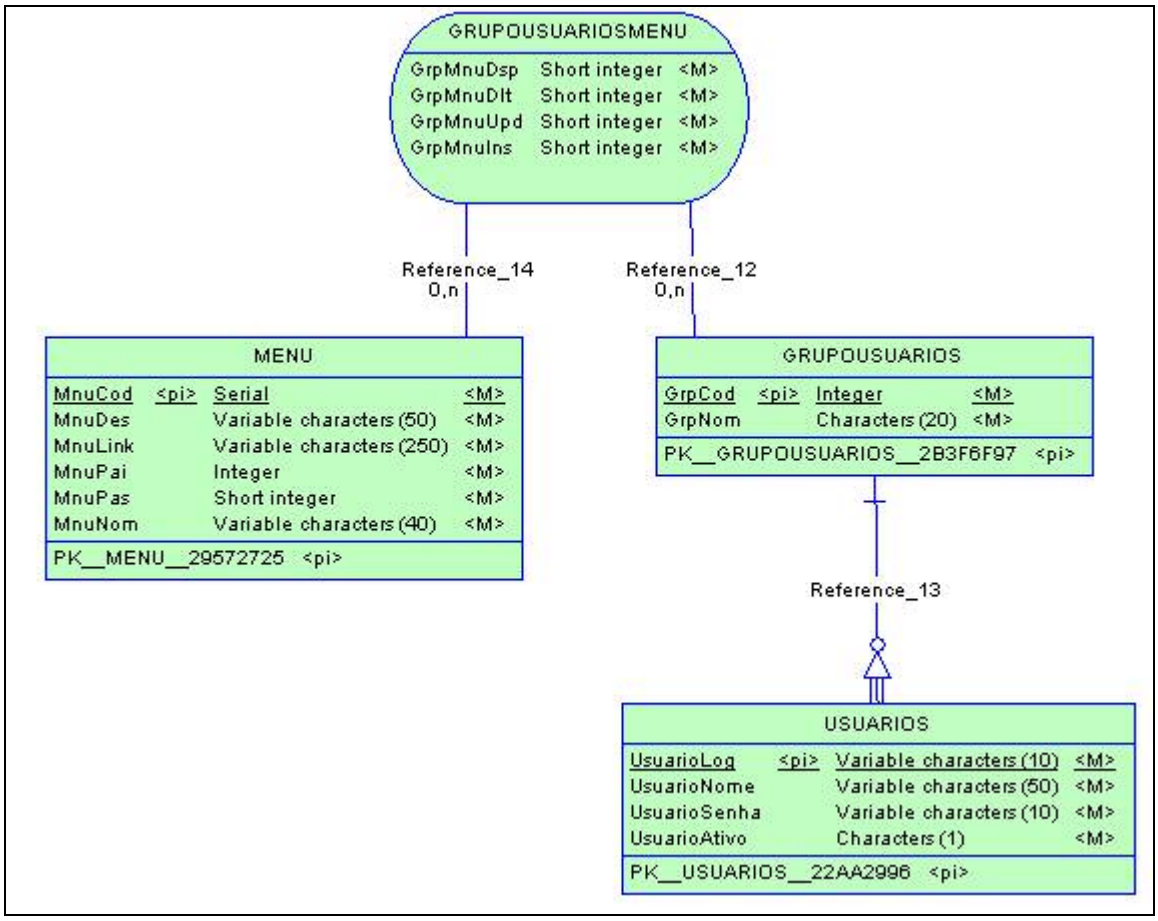


Figura 11 – Entidades referentes controle de acesso ao SPE

Na Figura 11 verifica-se a presença de entidades que visam permitir o cadastro de usuários, grupos de usuários e menus, e uma que faz a ligação para determinar quais grupos de usuários podem acessar determinadas opções do menu da ferramenta.

Isto permite que o usuário principal da ferramenta determine quem pode, por exemplo, acessar as opções de calibração ou opções de cadastro de projeto.

Na Figura 12 verifica-se a presença da entidade de clientes e projetos, e todas as entidades relacionadas a projetos que usam como métrica a FPA.

Na Figura 13 verifica-se a presença da entidade de clientes e projetos, e todas as entidades relacionadas a projetos que usam como métrica o UCP.

Na Figura 14 verifica-se a presença da entidade de clientes e projetos, e todas as entidades relacionadas a projetos que usam como métrica o COCOMO.

Observa-se nas Figuras 12, 13 e 14 a presença das entidades que contém as informações de calibração de cada uma das métricas implementadas. Também observa-se que todo o cálculo é gerado a partir do relacionamento entre as informações do projeto e as informações contidas nas entidades de calibração da ferramenta.

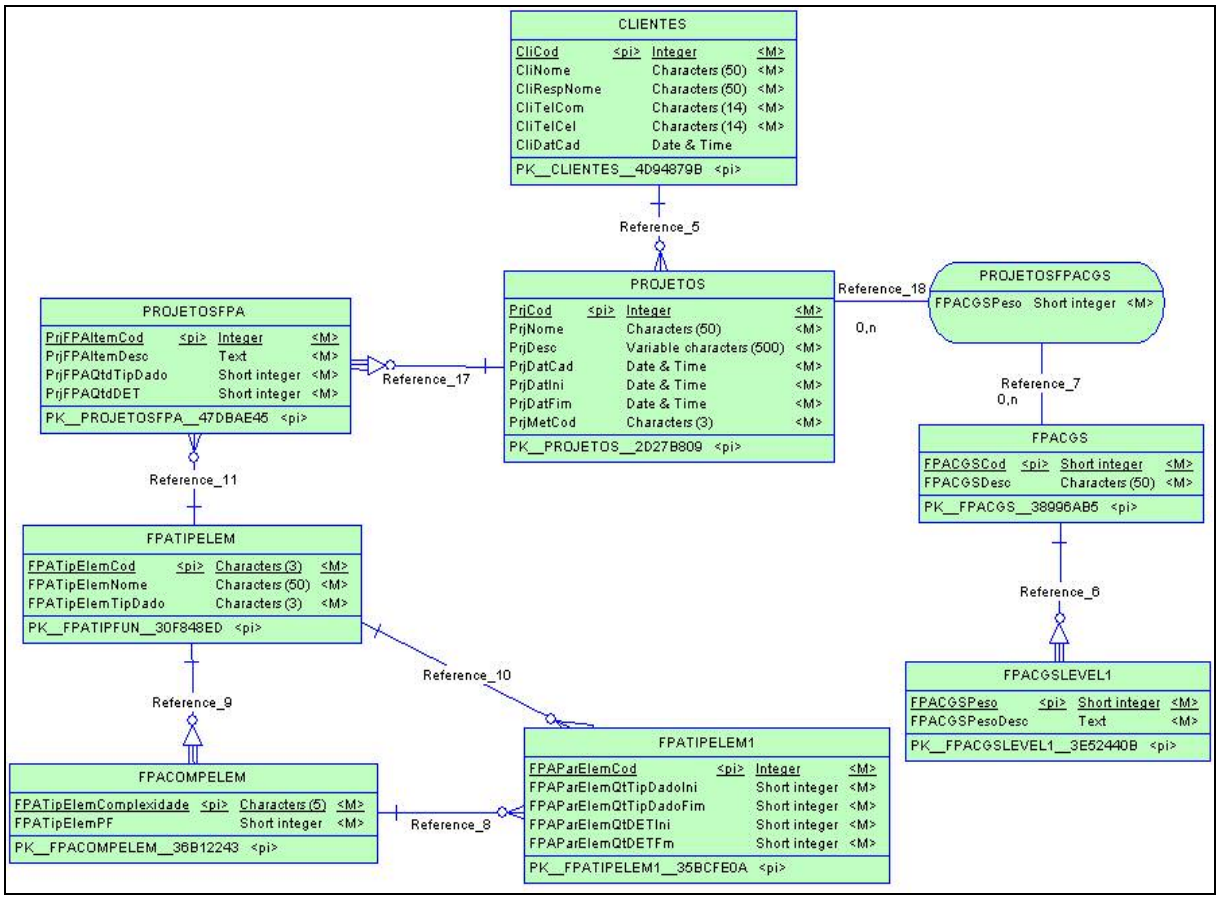


Figura 12 – Entidades referentes a projetos usando FPA

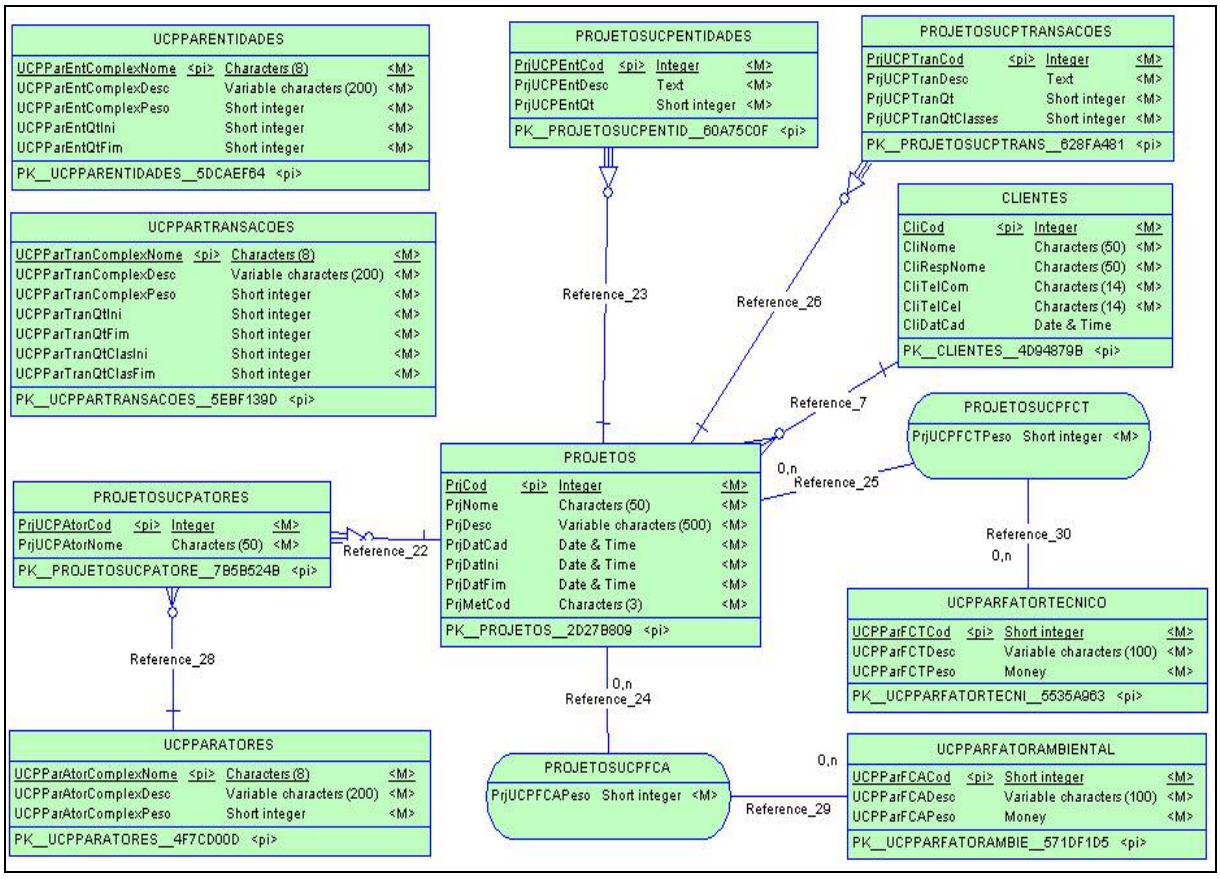


Figura 13 – Entidades referentes a projetos usando UCP

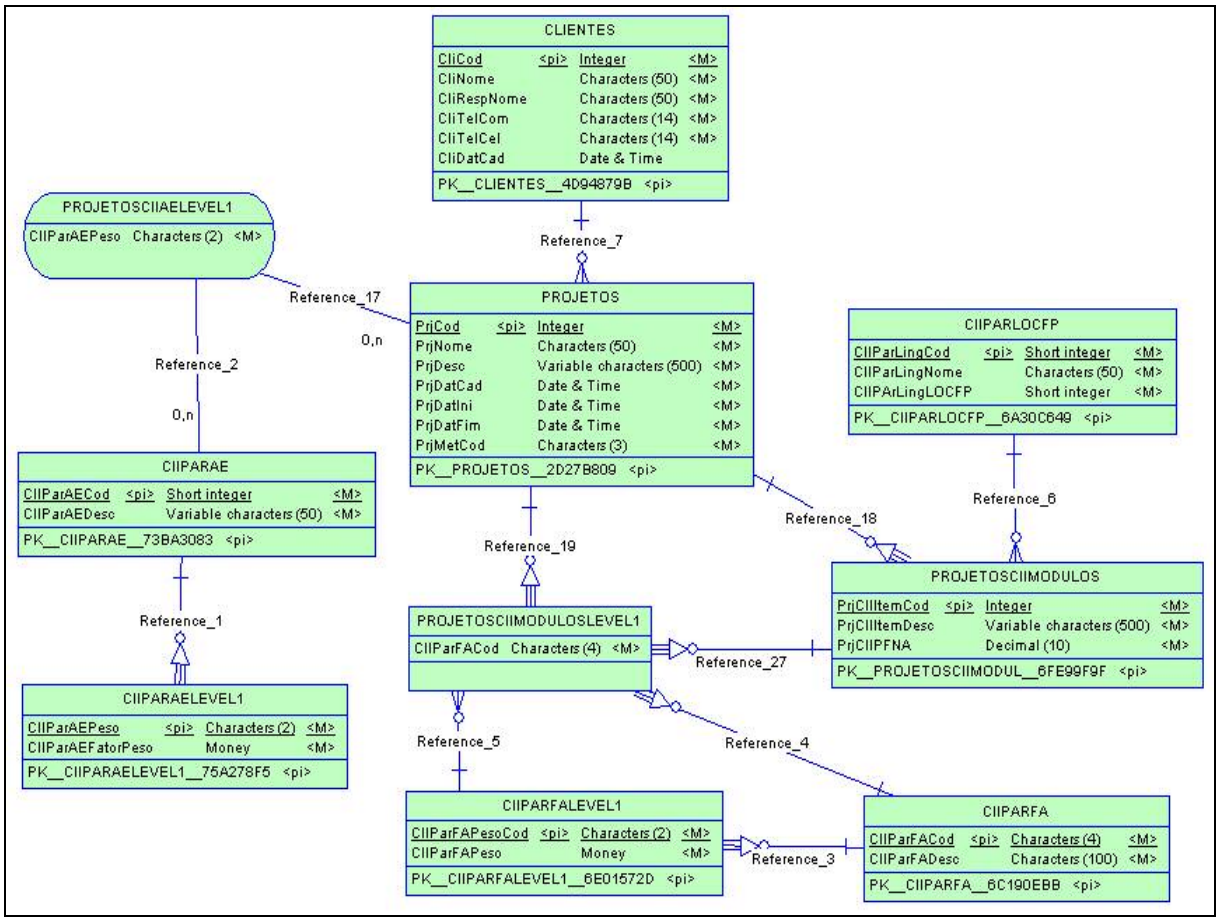


Figura 14 – Entidades referentes a projetos usando COCOMO

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

A principal ferramenta utilizada no desenvolvimento do SPE foi o software GeneXus versão 9, na sua atualização de número 4. Na fase inicial do desenvolvimento da ferramenta o mesmo mostrou-se um grande aliado na normalização e geração das tabelas do SPE baseando-se no desenho especificado e posteriormente cadastrado no GeneXus. Na confecção do protótipo novamente ele mostrou-se muito eficiente na geração dos códigos em .Net de

acordo com as programações (Figura 15) e especificações realizadas. Cada incremento na criação do SPE podia ser facilmente gerado e testado sem maiores complicações, garantindo agilidade ao processo de desenvolvimento.

```
header
    print Cabecalho
end
for each
    where PrjCod = &PrjCod
    print CabProjeto
    print CabElemento
    for each
        defined by PrjFPAItemCod
        for each
            if PrjFPAQtdDET >= FPAParElemQtDETIni and PrjFPAQtdDET <= FPAParElemQtDETFm and
               PrjFPAQtdTipDado >= FPAParElemQtTipDadoIni and PrjFPAQtdTipDado <= FPAParElemQtTipDadoFim
               &PFNA += FPATipElemPF
               Print Elemento
            endif
        endfor
    endfor
    //
    // Soma de NI (Nível de Influência) das CGS (Características Gerais do Sistema)
    print CabCGS
    for each
        &NI += FPACGSPeso
        print CGS
    endfor
endfor
//
// FA (Fator de Ajuste)
&FA = 0.65 + (0.01 * &NI)
//
// PFA (Pontos de Função Ajustados)
&PFA = &PFNA * &FA
print Resultado
```

Figura 15 – Exemplo de programação do SPE no GeneXus: relatório de projeto em FPA

3.3.2 Operacionalidade da implementação

Ao executar-se o SPE, depara-se com o menu principal da ferramenta, conforme a Figura 16, o qual possui as seguintes opções:

a) Controle de Acesso: reúne as opções de controle de acesso à ferramenta, como cadastro de usuários e acesso aos menus;

b) Calibração da Ferramenta: reúne as opções de calibração da ferramenta, permitindo informar os parâmetros de cada uma das métricas implementadas na ferramenta (FPA, UCP e COCOMO);

c) Clientes: cadastro de clientes dos projetos;

d) Projetos: cadastro de projetos.

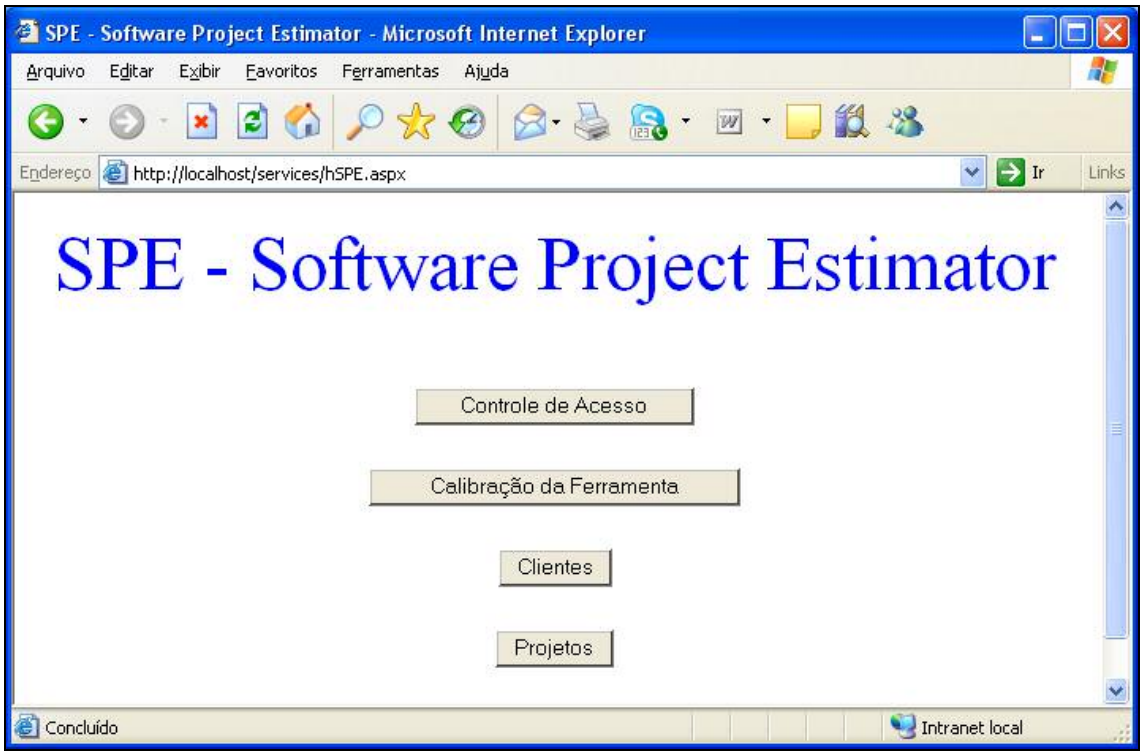


Figura 16 – Menu principal do SPE

No figura 17, observa-se o menu de Controle de Acesso, o qual possui as seguintes opções:

a) Menus Disponíveis: apresenta os menus disponíveis na ferramenta;

b) Controle de Acesso dos Usuários: permite o cadastro de usuários, e juntamente permite definir quais os menus liberados ao mesmo;

c) Menu Principal: volta ao menu principal da ferramenta.

Um dos primeiros passos para usar o SPE é calibrar a ferramenta com os parâmetros das métricas de FPA, UCP e COCOMO II. Isto em princípio pode parecer trabalhoso, mas é necessário apenas uma vez e garante que a ferramenta esteja atualizada de acordo com os valores mais atuais de cada métrica.

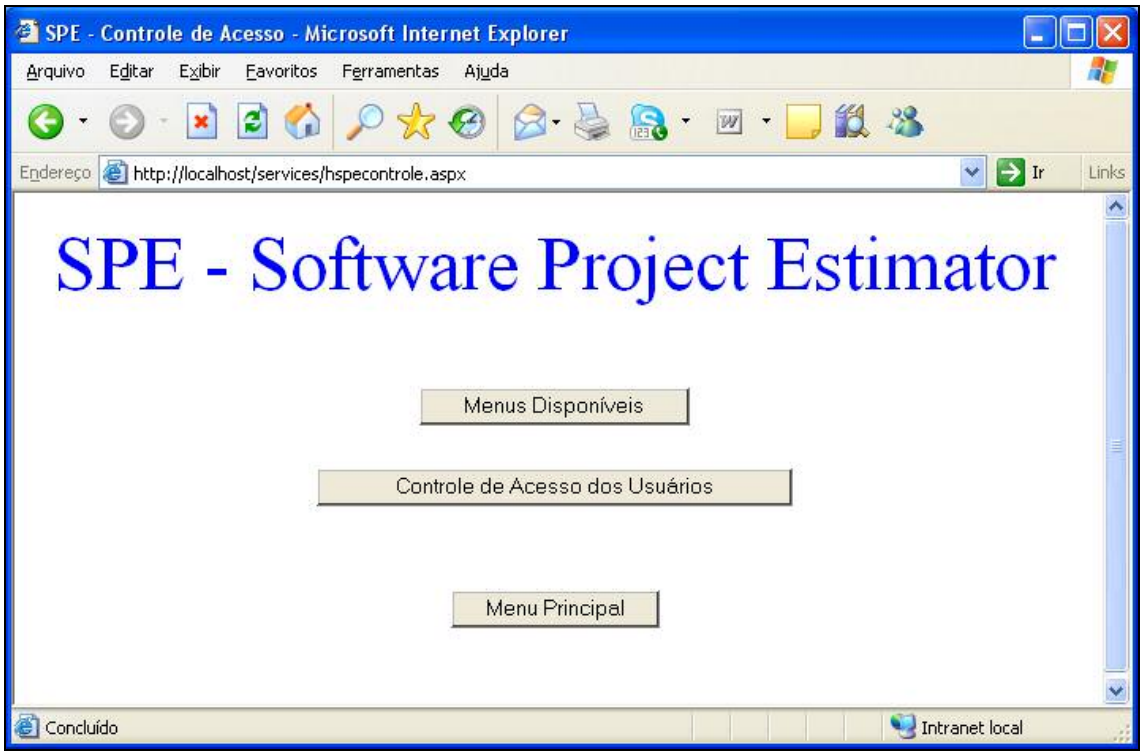


Figura 17 – Menu de controle de acesso do SPE

Na Figura 18 visualiza-se o menu de Calibração da Ferramenta. Nele cadastram-se todos os parâmetros pertinentes a cada uma das métricas implementadas:

a) FPA: parâmetros de elementos, elementos, e características gerais de sistema;
b) UCP: complexidade de atores, complexidade de transações, complexidade de entidades, fatores de complexidade técnica e fatores de complexidade ambiental;
c) COCOMO II: parâmetros de aumento de esforço, parâmetros de LOC por FP e parâmetros dos fatores de ajuste.

Na Figura 19, visualiza-se um exemplo do cadastro de parâmetros de elementos de FPA, informando os tipos de dados e os pontos de função contados em vista do grau de complexidade. Estes parâmetros que estão no exemplo obedecem aos valores de calibração presentes nas últimas atualizações das normas de cálculo da métrica.

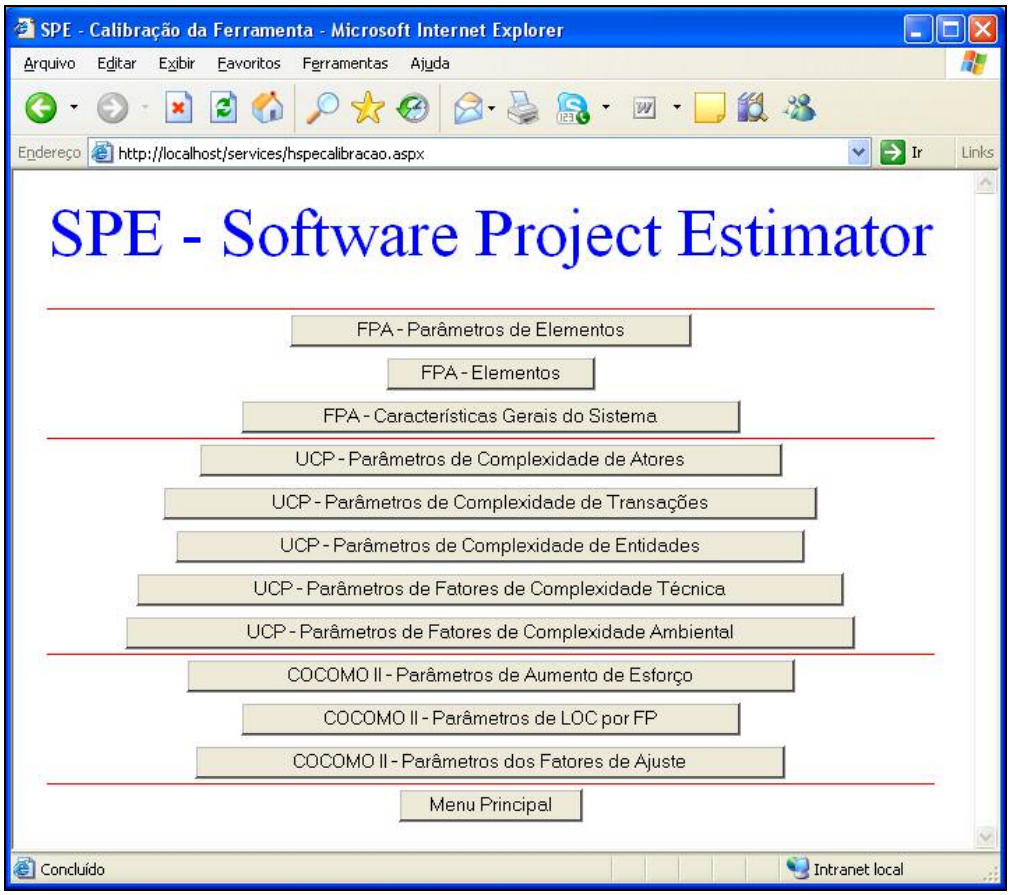


Figura 18 – Menu de calibração do SPE

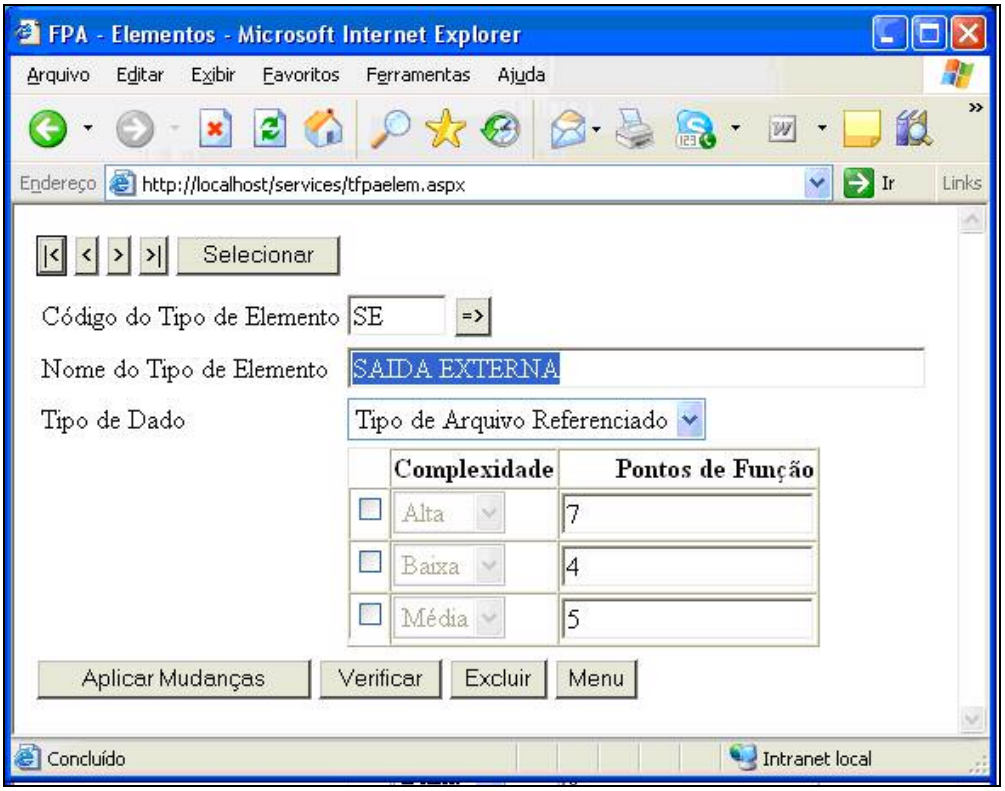


Figura 19 – Exemplo de parametrização de elementos da FPA

A seguir, a função dos botões presentes na Figura 19 e que serão terão o mesmo

significado nas demais telas da ferramenta:

a)  : carrega o primeiro registro do cadastro;

b)  : carrega o registro anterior ao corrente;

c)  : carrega o registro posterior ao corrente;

d)  : carrega o último registro do cadastro;

e)  : carrega o registro (no caso de o usuário ter informado a chave do cadastro);

f)  : abre uma tela de consulta e seleção para o cadastro;

g)  : salva as informações;

h)  : carrega novamente o registro corrente;

i)  : exclui o registro corrente;

j)  : volta ao menu anterior.

Na Figura 20, visualiza-se um exemplo do cadastro de parametrização de complexidade de elementos da FPA. Nele calibra-se a ferramenta de forma a permitir que ela identifique o grau de complexidade dos dados do projeto.

Na Figura 21, visualiza-se um exemplo de cadastro de característica geral do sistema da FPA. Nele calibra-se os pesos atribuídos a cada uma das CGS descrevendo-se a que casos se aplica.

Na Figura 22, visualiza-se um exemplo de cadastro de complexidade de atores do UCP. Nele calibra-se o peso do ator de acordo com sua complexidade.

Na Figura 23, visualiza-se um exemplo de cadastro de complexidade de transações de UCP. Nele calibra-se o peso da transação de acordo com a complexidade, quantidade de transações e quantidade de classes.

Na Figura 24, visualiza-se um exemplo de cadastro de complexidade de entidades de UCP. Nele calibra-se o peso da entidade de acordo com a complexidade e quantidade de entidades.

Na Figura 25, visualiza-se um exemplo de cadastro de fator de complexidade técnica de UCP. Nele calibra-se o peso de cada fator de complexidade técnica.

Na Figura 26, visualiza-se um exemplo de cadastro de fator de complexidade ambiental de UCP. Nele calibra-se o peso de cada fator de complexidade ambiental.

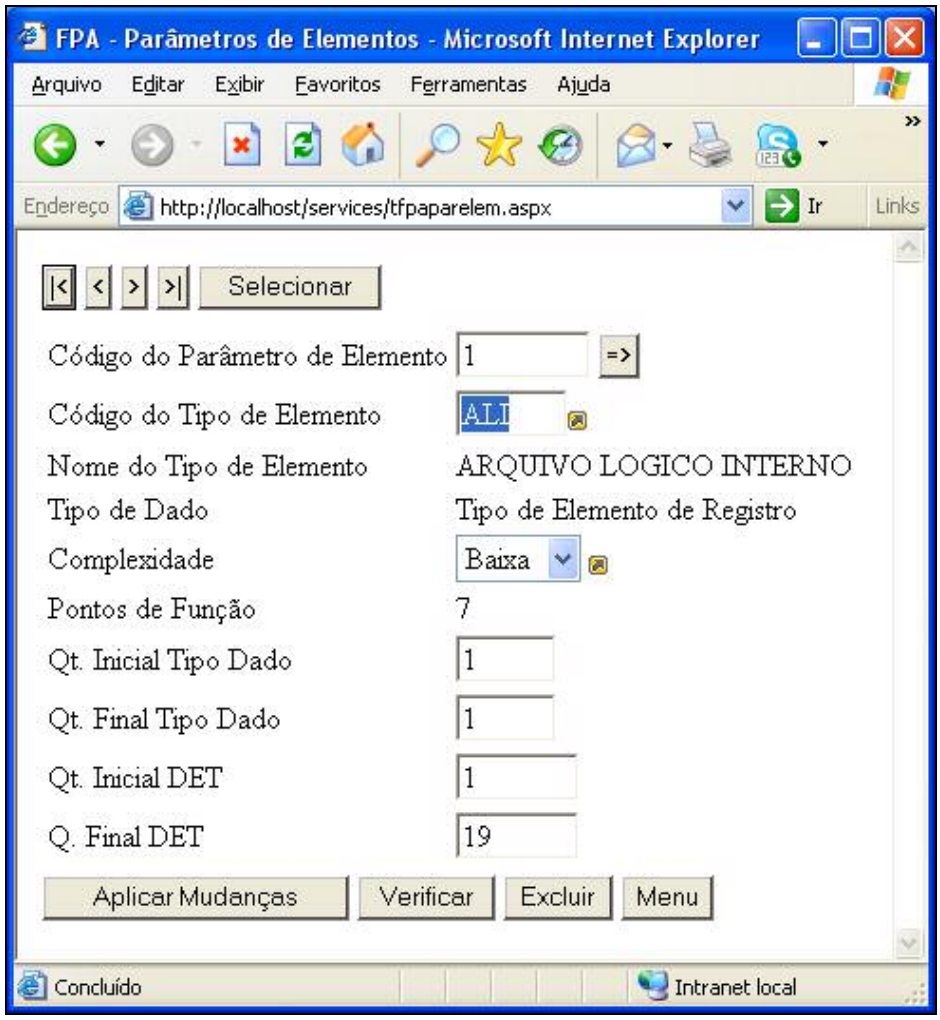


Figura 20 – Exemplo de parametrização de complexidade de elementos da FPA

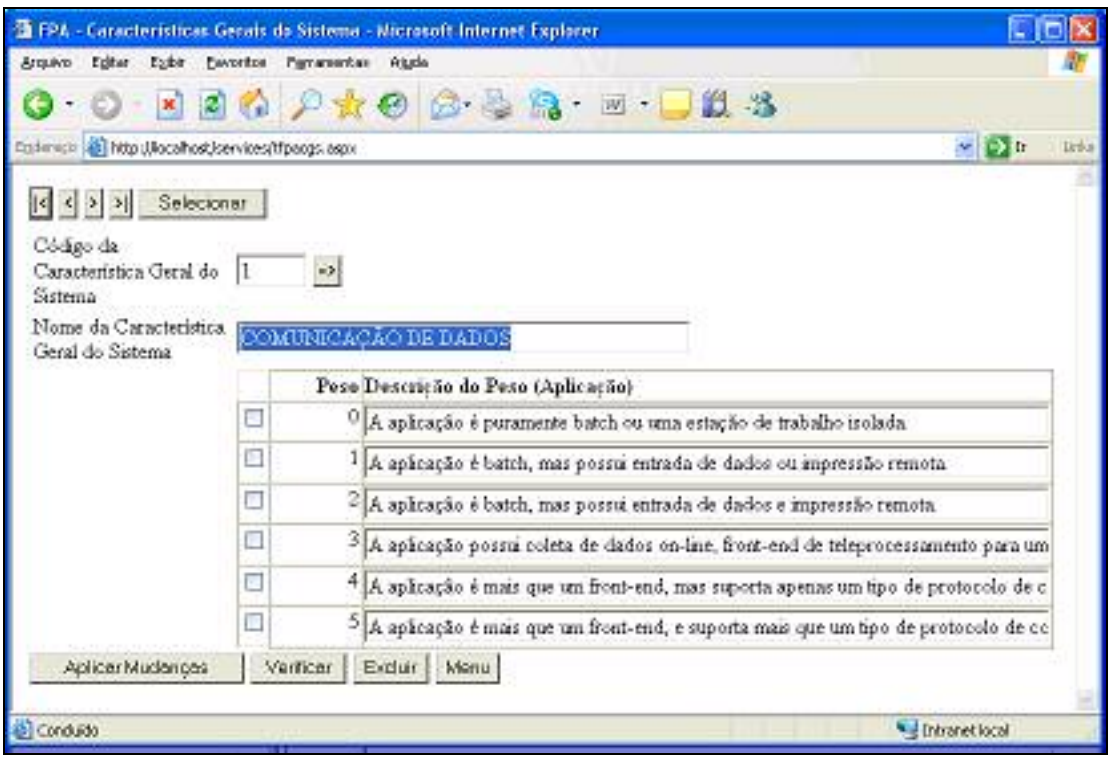


Figura 21 – Exemplo de parametrização de CGS da FPA

Figura 22 – Exemplo de parametrização de complexidade de atores do UCP

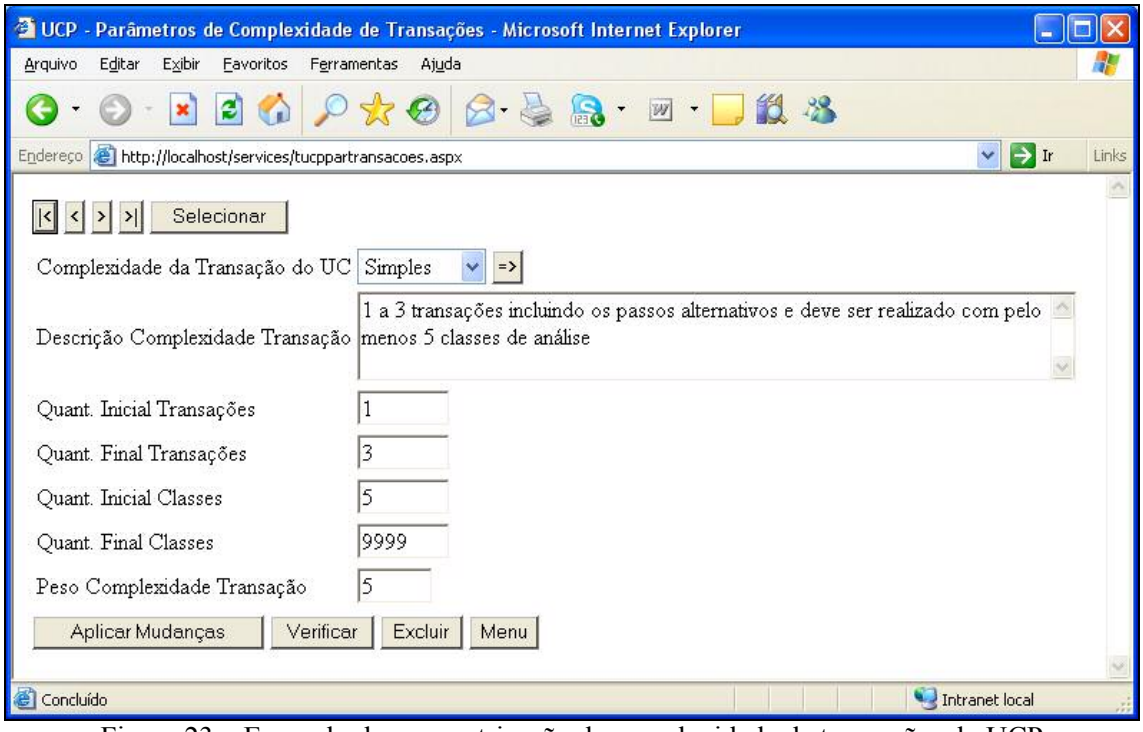


Figura 23 – Exemplo de parametrização de complexidade de transações do UCP

Na Figura 27, visualiza-se um exemplo de cadastro de fatores de aumento de esforço do COCOMO II. Nele calibra-se qual o peso do fator de esforço de acordo com sua complexidade.

Na Figura 28, visualiza-se um exemplo de cadastro de LOC por FP do COCOMO II. Nele calibra-se qual a quantidade de LOC por FP para cada linguagem de programação.

Na Figura 29, visualiza-se um exemplo de cadastro de fatores de ajuste do COCOMO II. Nele calibra-se qual o peso do fator de ajuste de acordo com sua complexidade.

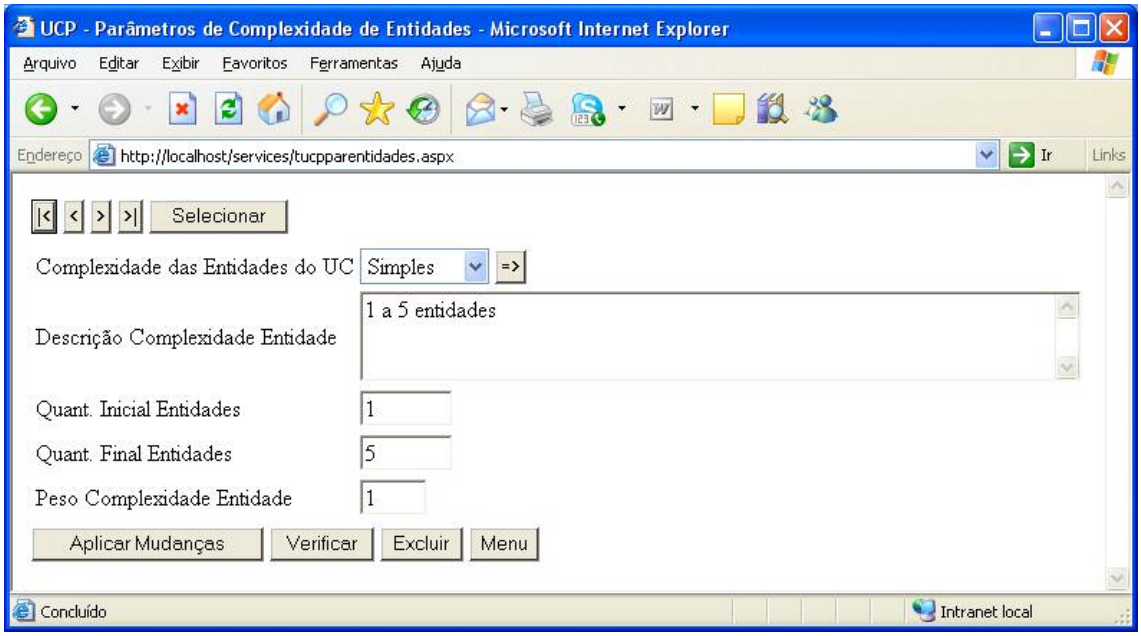


Figura 24 – Exemplo de parametrização de complexidade de entidades do UCP

Figura 25 – Exemplo de parametrização de FCT do UCP

Figura 26 – Exemplo de parametrização de FCA do UCP

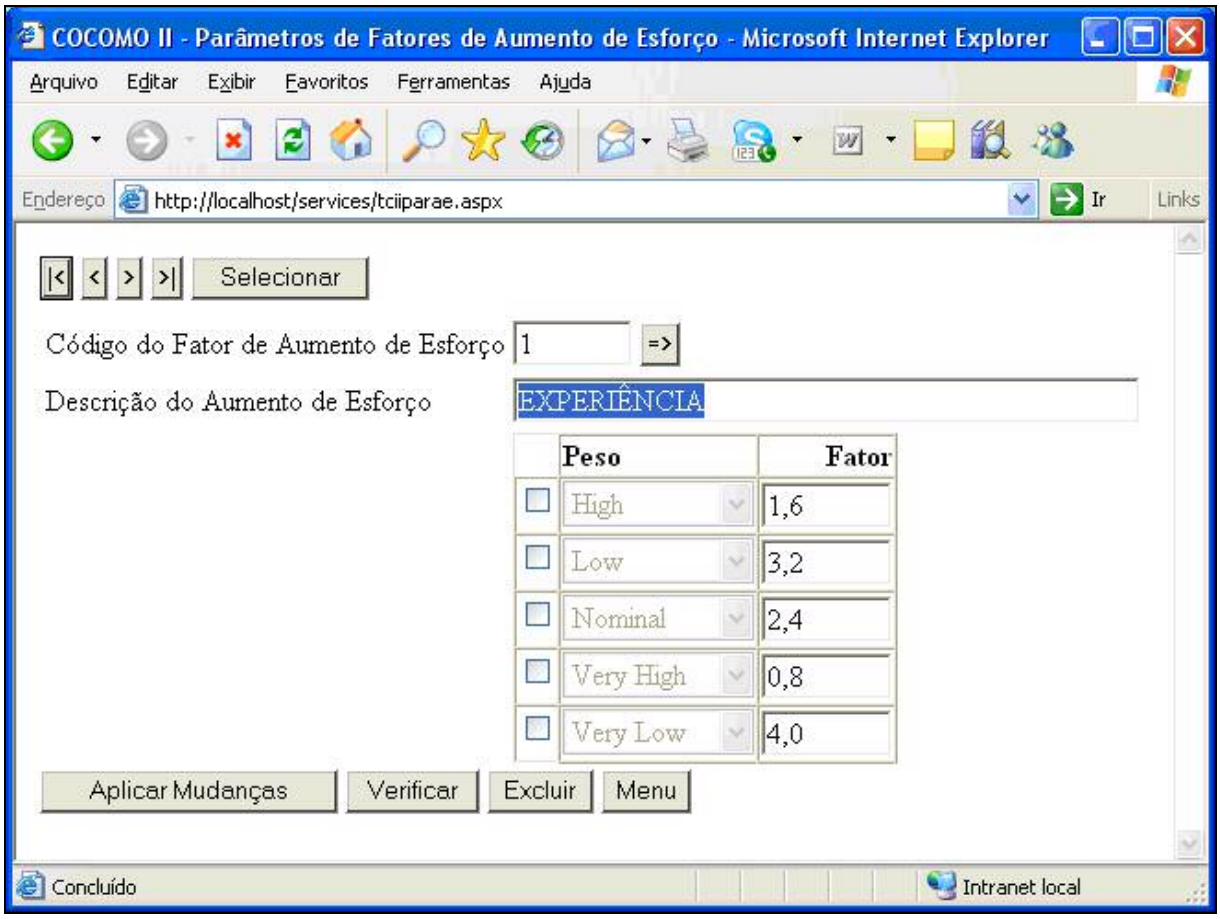


Figura 27 – Exemplo de parametrização de FCA do UCP

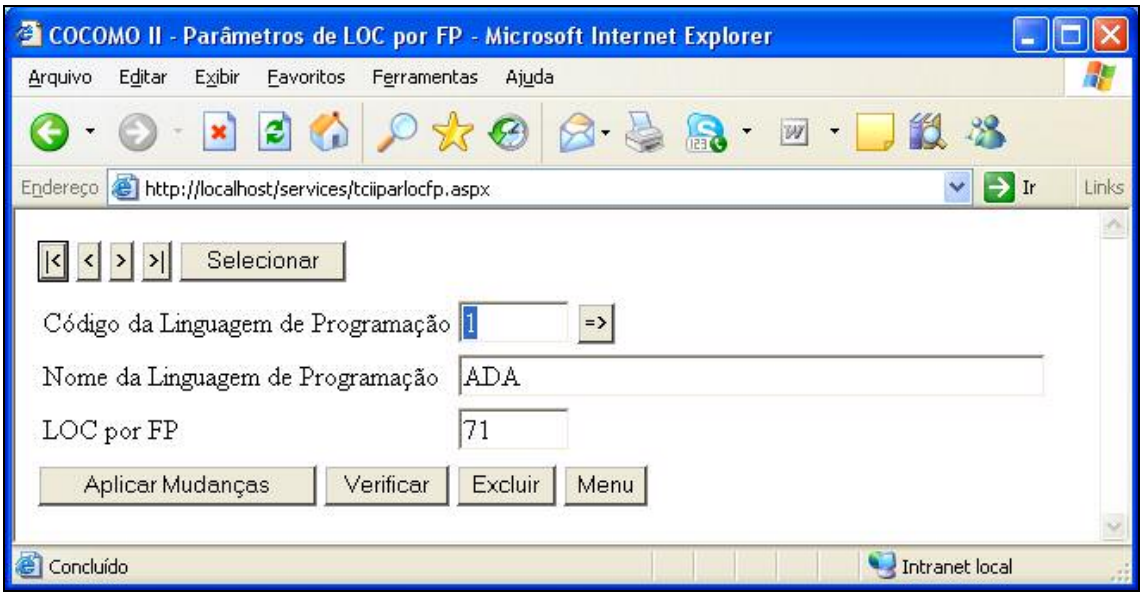


Figura 28 – Exemplo de parametrização de Linguagens do COCOMO II

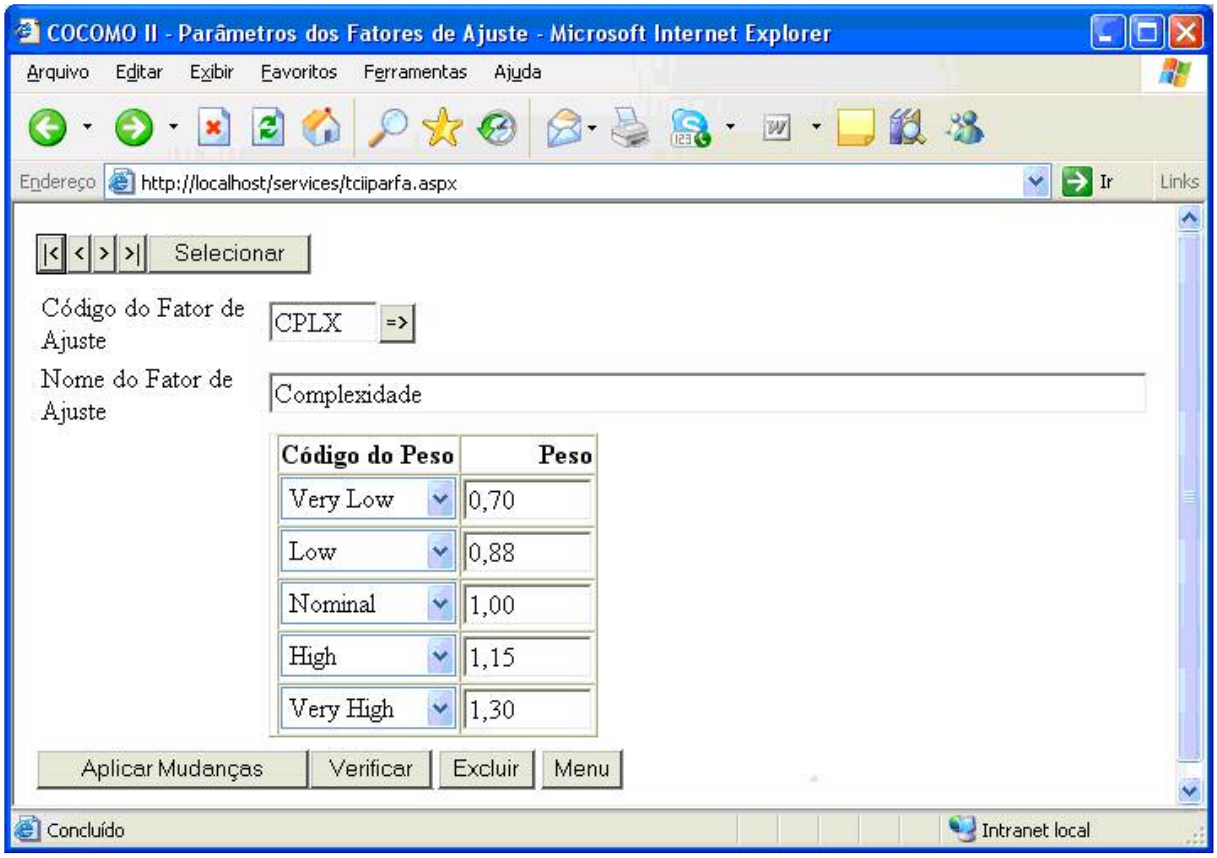


Figura 29 – Exemplo de parametrização de FA do COCOMO II

Após a parametrização de todas as métricas, o SPE está apto a receber o cadastro dos clientes do usuário (Figura 30), bem como seus projetos (Figura 31).

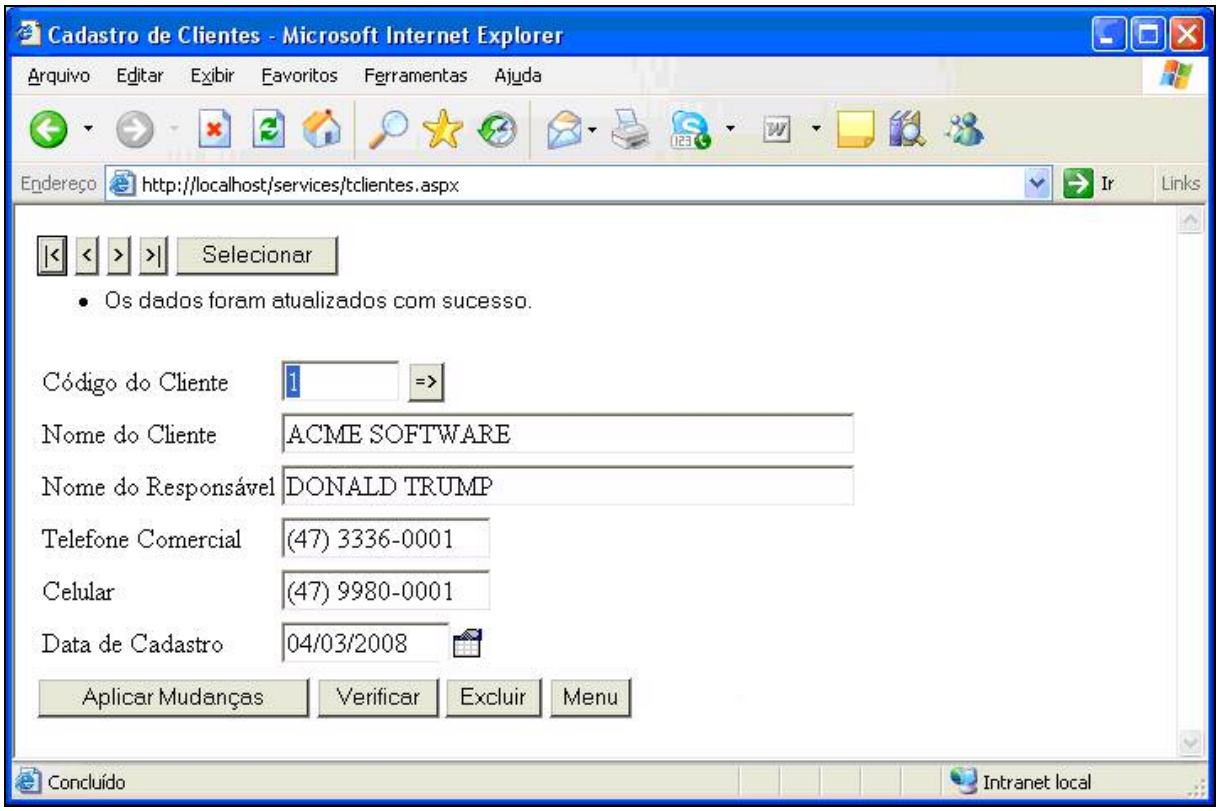


Figura 30 – Exemplo de cadastro de cliente

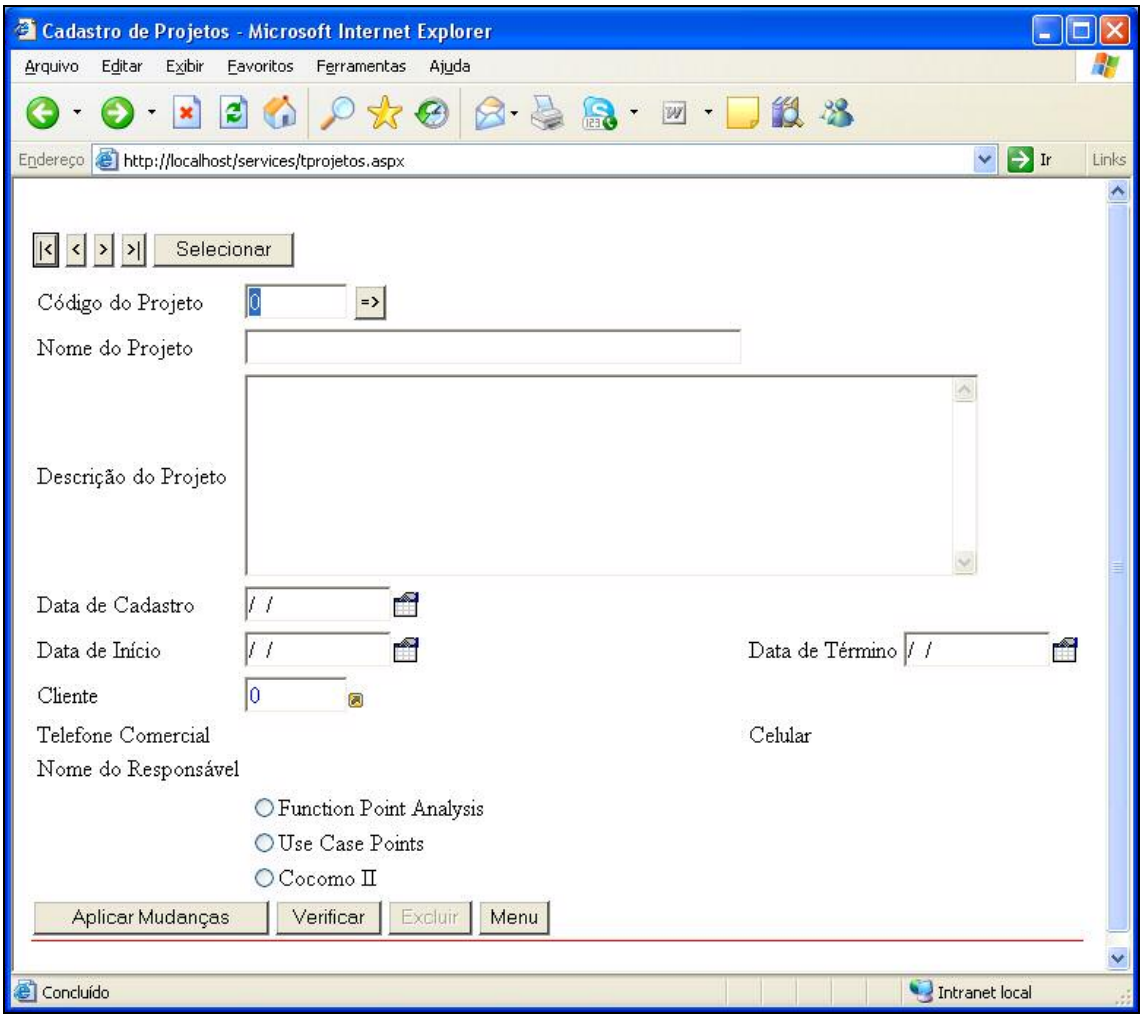


Figura 31 – Exemplo de cadastro de projeto

Observa-se na Figura 31, que é no momento do cadastro do projeto que o usuário escolhe qual a métrica que será usada no cálculo da estimativa.

Cadastrado o projeto, basta dar entrada com os itens que o compõem através dos botões que são ativados abaixo da linha vermelha no cadastro de projetos (Figura 31), observando que os mesmos variam de acordo com a métrica escolhida no momento do cadastro.

Botões disponíveis para projetos usando a métrica de FPA:

a) Itens do Projeto: permite acesso ao cadastro de itens correspondentes ao projeto (Figura 32);

b) Características Gerais: permite acesso ao cadastro de características gerais do projeto (Figura 33);

c) Consulta / Relatório: permite acesso à consulta do andamento do projeto, que também funciona como relatório que pode ser salvo (formato PDF) ou

impresso (Figura 41).

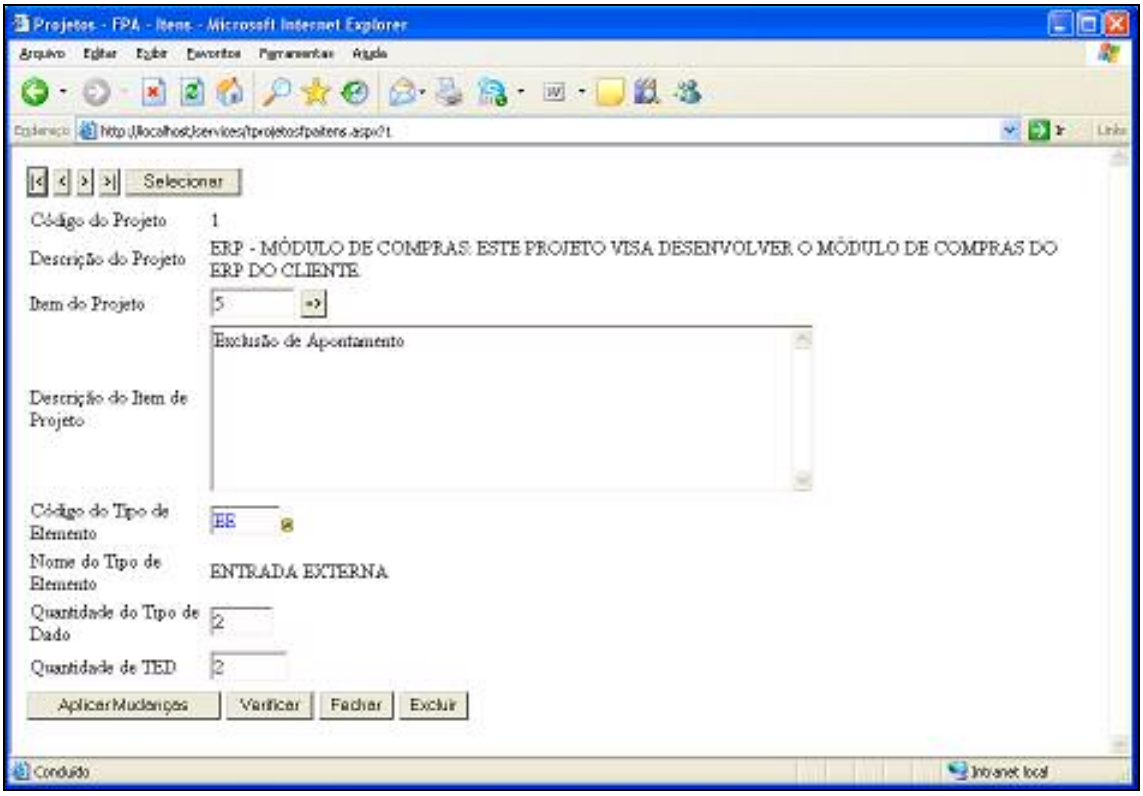


Figura 32 – Exemplo de cadastro de item de um projeto usando FPA

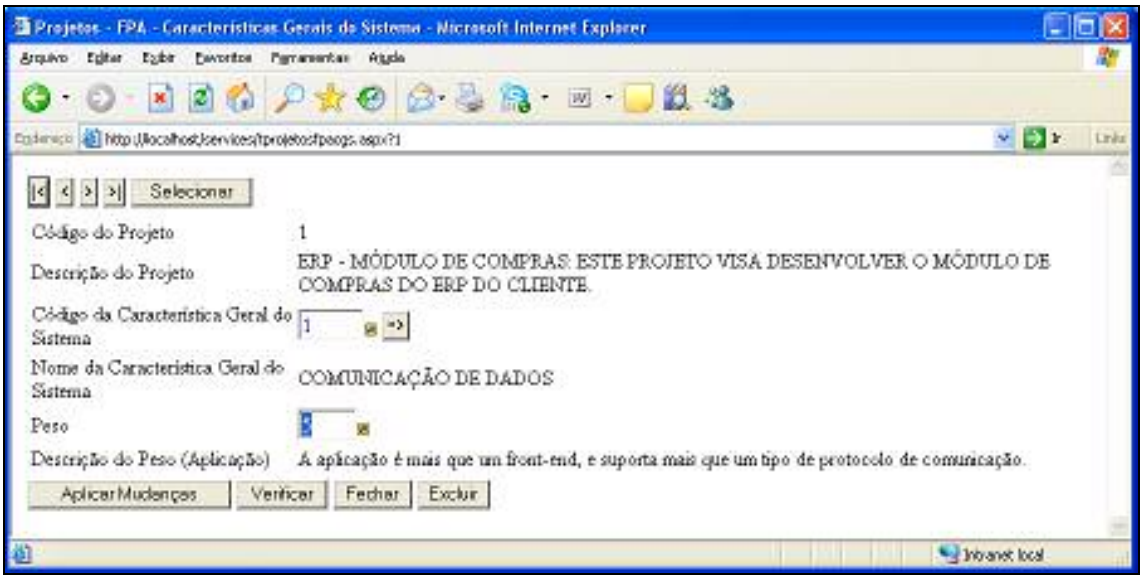


Figura 33 – Exemplo de cadastro de CGS de um projeto usando FPA

Nas Figuras 32 e 33 observa-se a presença do botão Fechar. Sua função é fechar o cadastro do componente do projeto e voltar para o cadastro de projetos.

Botões disponíveis para projetos usando a métrica de UCP:

d) Atores: permite acesso ao cadastro de atores presentes no projeto (Figura 34);

e) Transações: permite acesso ao cadastro de transações presentes no projeto

(Figura 35);

f) Entidades: permite acesso ao cadastro de entidades presentes no projeto (Figura 36);

g) Fatores Técnicos: permite acesso ao cadastro de fatores técnicos do projeto (Figura 37);

h) Fatores Ambientais: permite acesso ao cadastro de fatores ambientais do projeto (Figura 38);

i) Consulta / Relatório: permite acesso à consulta do andamento do projeto, que também funciona como relatório que pode ser salvo (formato PDF) ou impresso (Figura 42).

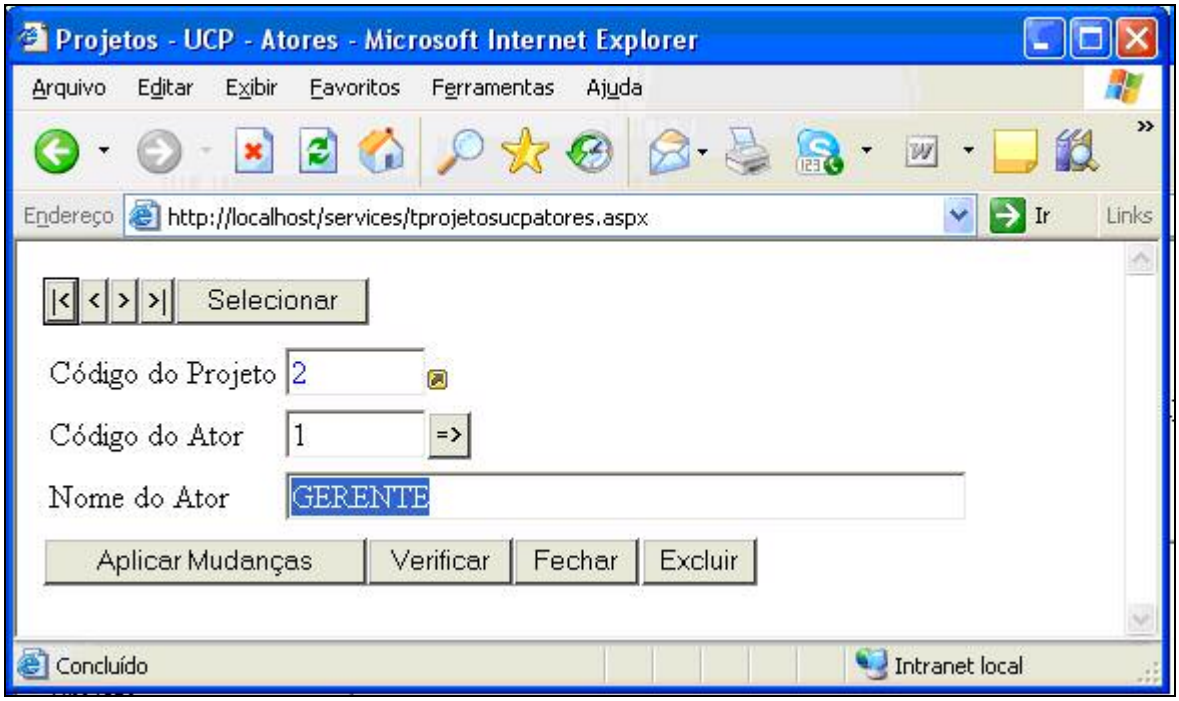


Figura 34 – Exemplo de cadastro de ator de um projeto usando UCP

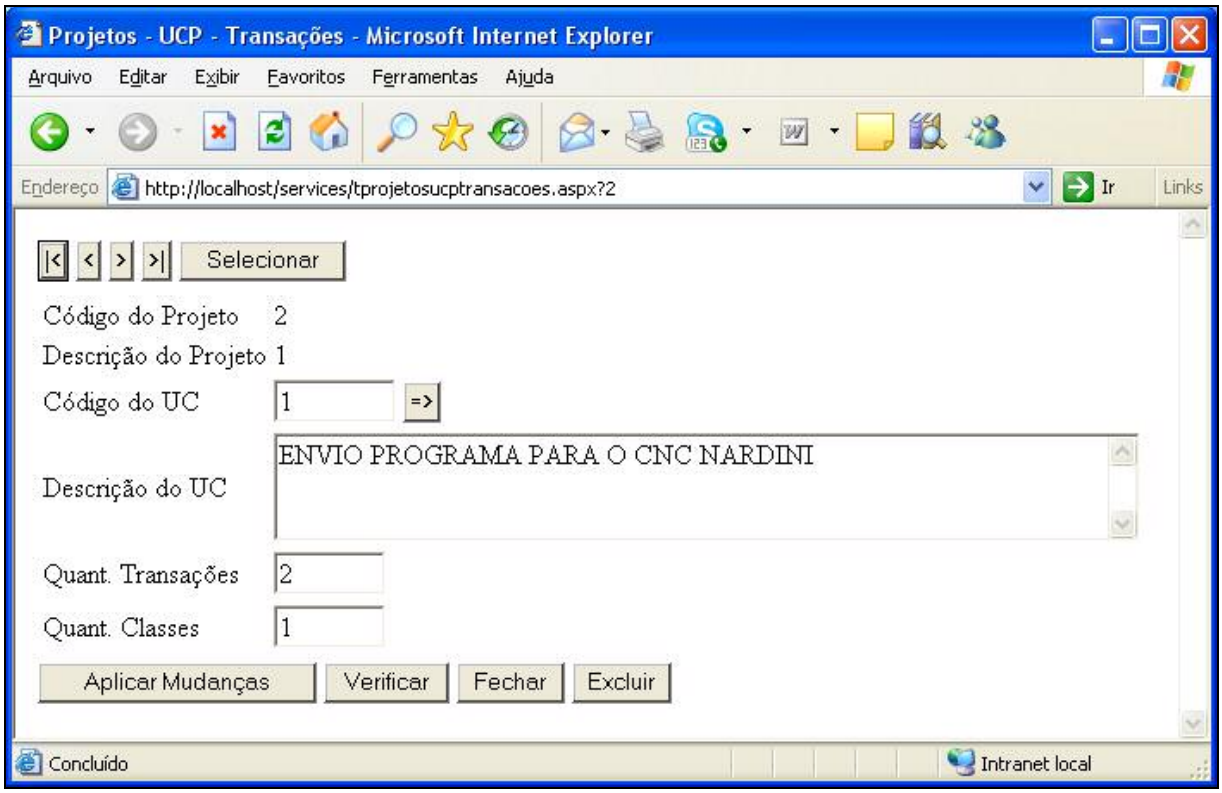


Figura 35 – Exemplo cadastro de UC em função das transações em projeto usando UCP

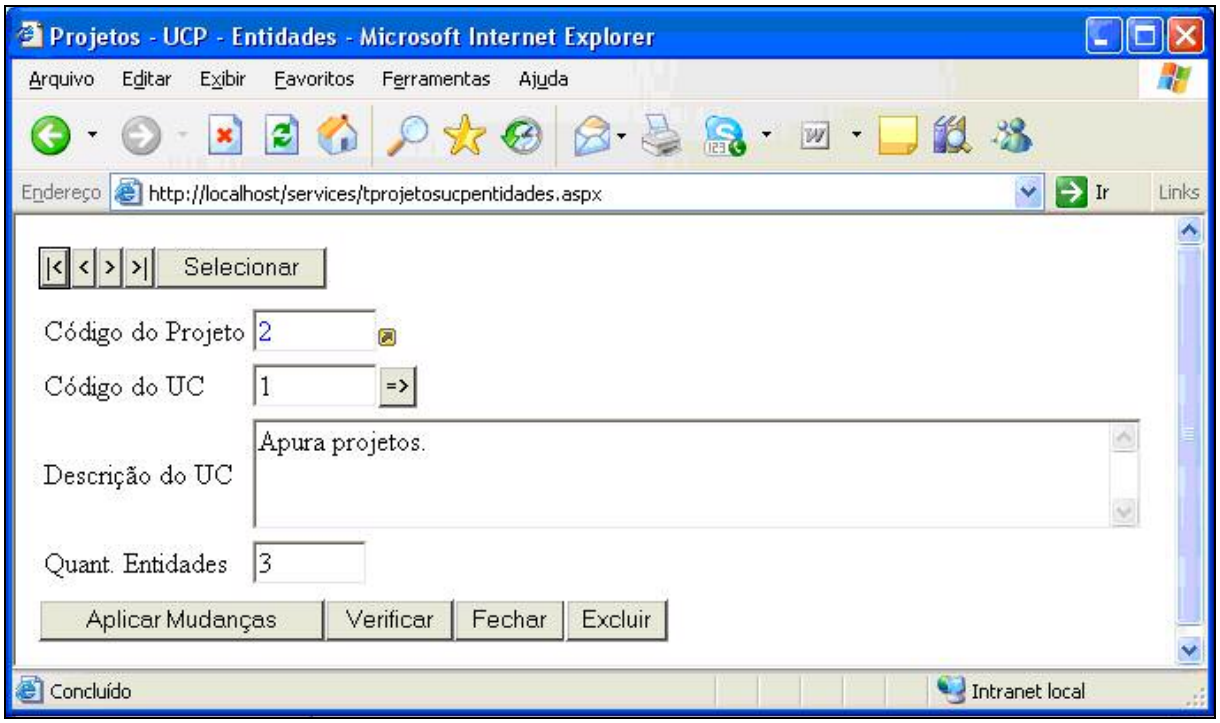


Figura 36 – Exemplo cadastro de UC em função das entidades em projeto usando UCP

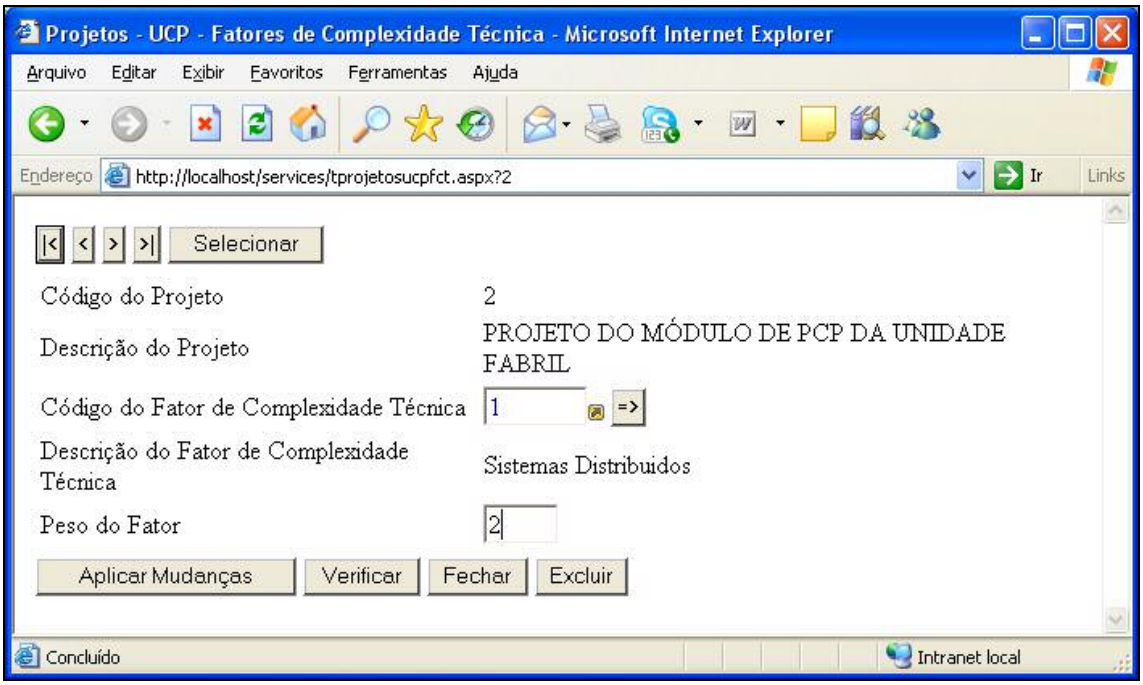


Figura 37 – Exemplo de cadastro de FCT em um projeto usando UCP

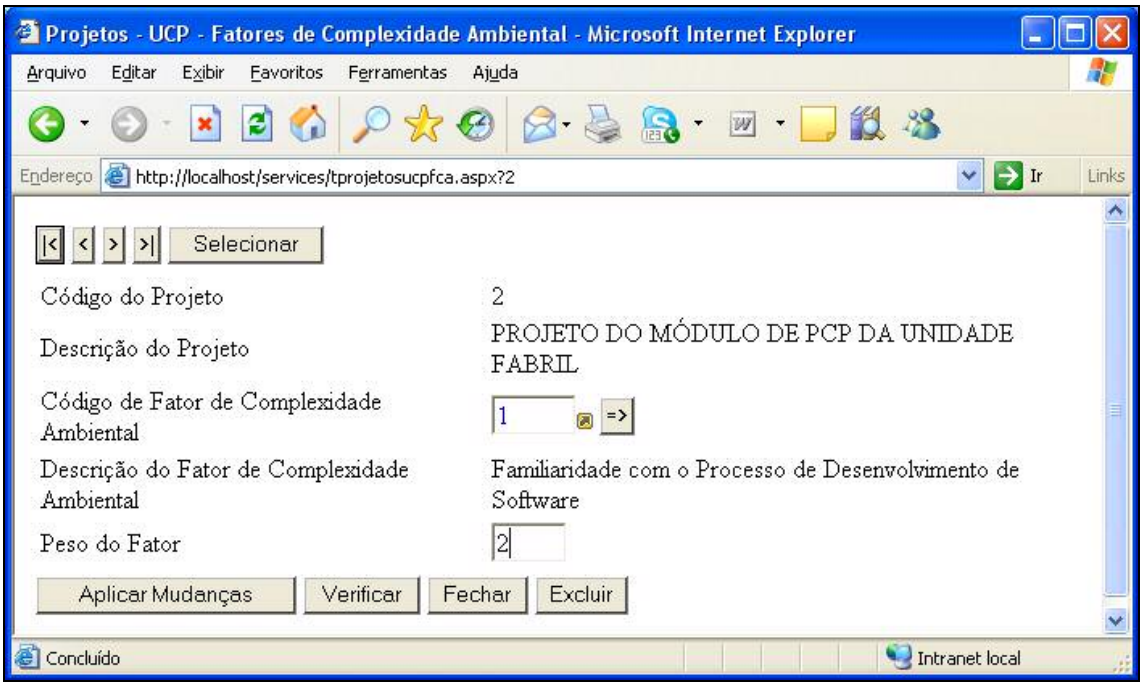


Figura 38 – Exemplo de cadastro de FCA em um projeto usando UCP

Botões disponíveis para projetos usando a métrica de COCOMO II:

a) Fatores Esforço: permite acesso ao cadastro de fatores de aumento de esforço no projeto (Figura 39);

b) Módulos: permite acesso ao cadastro dos módulos que compõem o projeto (Figura 40);

c) Consulta / Relatório: permite acesso à consulta do andamento do projeto,

que também funciona como relatório que pode ser salvo (formato PDF) ou impresso (Figura 43).

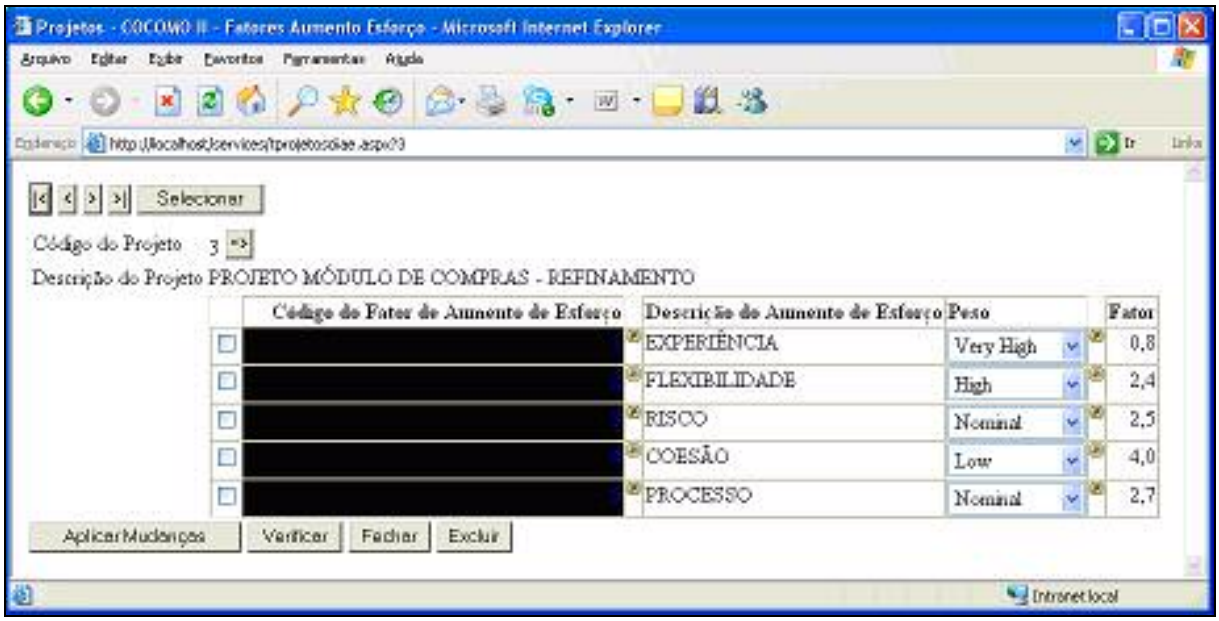


Figura 39 – Exemplo de cadastro de AE em um projeto usando COCOMO II

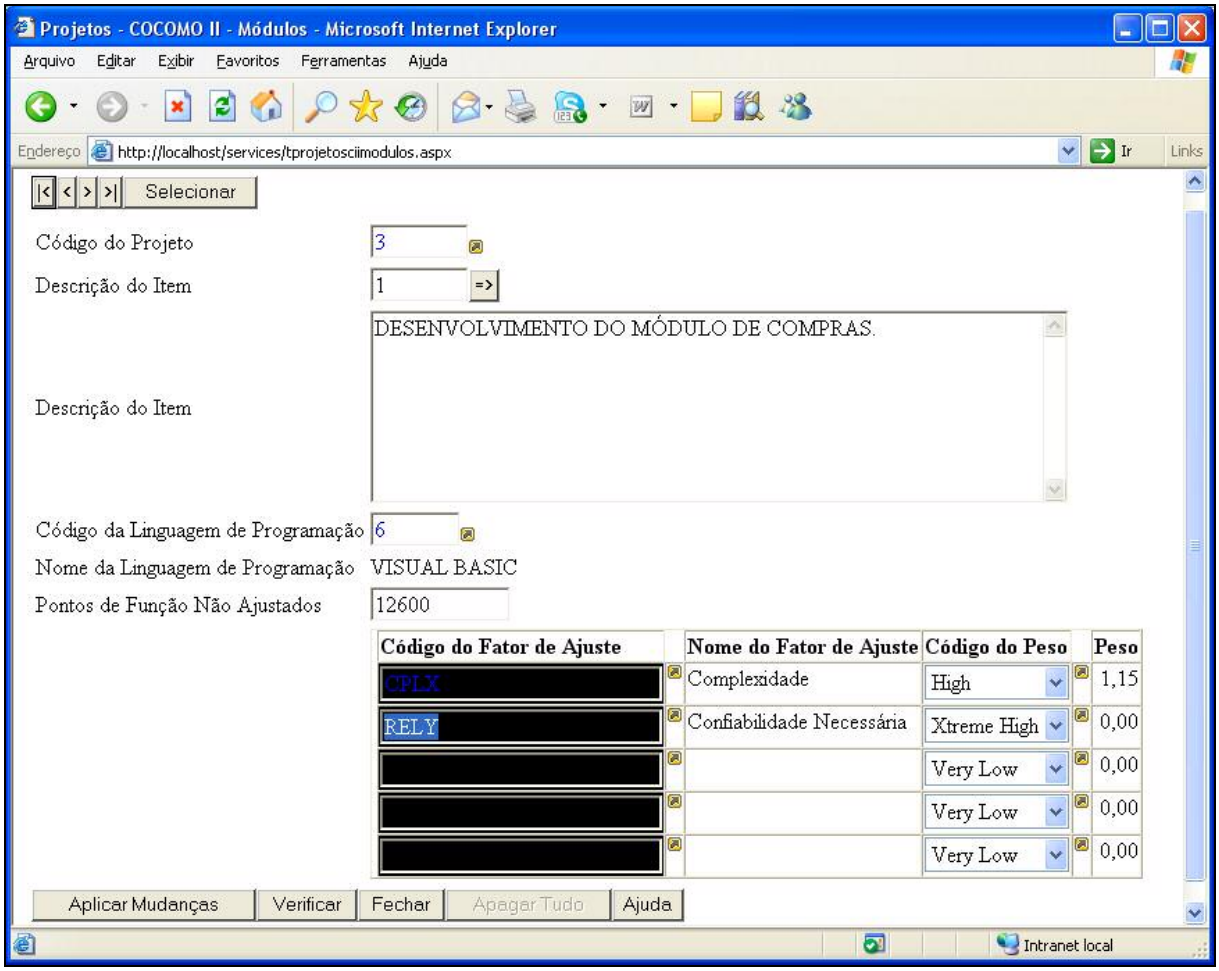


Figura 40 – Exemplo de cadastro de item em um projeto usando COCOMO II

Ao finalizar o cadastro de todas as informações pertinentes ao projeto, pode-se gerar o relatório de projeto que então produz o cálculo da estimativa, seja baseado na métrica de FPA (Figura 27), UCP (Figura 28) ou COCOMO II (Figura 29). Isto também pode ser feito a qualquer momento que o usuário do SPE desejar, não necessitando de novas parametrizações ou cuidados especiais. O relatório é gerado em tela em formato PDF e pode ser impresso ou salvo em arquivo conforme necessidade do usuário.

SPE Software Project Estimator

Relatório de Projeto

Data: 22/06/08
Hora: 23:43:24
Página: 1

Métrica: Function Point Analysis

Projeto: 1 - ESTE PROJETO VISA DESENVOLVER O MÓDULO DE COMPRAS DO ERP DO CLIENTE.
Cliente: 1 - ACME SOFTWARE

Item	Descrição	Tipo Elemento	Qt. Tipo Dado	Qt.DET	Complex.	PF
1	Login	EQ	4	1	MEDIA	5
2	Registro de Ponto	EI	3	1	MEDIA	4
3	Consulta Apontamento Diário	EQ	3	1	BAIXA	3
4	Apontamento com Justificativa	EI	5	3	MEDIA	4
5	Exclusão de Apontamento	EI	2	2	BAIXA	3

CGS	Descrição	Peso
1	COMUNICAÇÃO DE DADOS	5

PFNA - Pontos de Função Não Ajustados:	19,00
NI - Nível de Influência das CGS:	5,00
FA - Fator de Ajuste:	0,70
PFA - Pontos de Função Ajustados:	13,30

Figura 41 – Exemplo de relatório de projeto usando FPA

SPE Sotware Project Estimator

Relatório de Projeto

Métrica: Use Case Points

Data: 03/06/08
Hora: 00:29:49
Página: 1

Projeto: 2 - PROJETO EXEMPLO
Cliente: 1 - ACME SOFTWARE

Cód.	Nome do Ator	Complex.
1	GERENTE	COMPLEXO

UC T.	Descrição	Transações	Classes	Complex.
1	Apura projetos.	2	1	SIMPLES

UC E.	Descrição	Entidades	Complex.

FCT	Descrição	Peso
1	Sistemas Distribuidos	2

FCA	Descrição	Peso
1	Familiaridade com o Processo de Desenvolvimento de Soft	4

TAN - Total de Atores Não Ajustado:	3,00
TTN - Total de Transações Não Ajustado:	5,00
TEN - Total de Entidades Não Ajustado:	0,00
UCN - UCPs Não Ajustados:	8,00
TFT - Total de Fatores Técnicos:	4,00
FCT - Fator de Complexidade Técnica:	0,64
TFA - Total de Fatores Ambientais:	6,00
FCA - Fator de Complexidade Ambiental:	1,22
UCP - UCPs Ajustados:	6,25

Figura 42 – Exemplo de relatório de projeto usando UCP

SPE Sotware Project Estimator

Relatório de Projeto

Métrica: COCOMO II

Data: 31/05/08
Hora: 14:13:49
Página: 1

Projeto: 3 - TESTE COCOMO
Cliente: 1 - ACME SOFTWARE

Item	Descrição	Linguagem	PFNA
1	MÓDULO DE COMPRAS	VISUAL BASIC	12600

FA	Descrição	Influência	Peso
CPLX	Complexidade	MI	1,15
RELY	Confiabilidade Necessária	VH	1,39

Figura 43 – Exemplo de relatório de projeto usando COCOMO II

3.4 RESULTADOS E DISCUSSÃO

Com o desenvolvimento deste trabalho, através da automatização de rotinas de cálculo de métricas, proporcionou-se muito mais agilidade e segurança no decorrer do processo de apuração de estimativas de projetos de software. As operações são realizadas facilmente e em um curto espaço de tempo. Já é possível se obter um controle sobre os projetos uma vez que

as informações são armazenadas de maneira padronizada no sistema, sendo possível consultá-las rapidamente e gerar de relatórios permitindo o acompanhamento das operações efetuadas.

A ferramenta SPE foi desenvolvida a partir das definições das métricas de FPA, UCP e COCOMO, e das necessidades iniciais presentes em projetos de software. Nada impede que mais adiante sejam incrementadas novas melhorias e demais funcionalidades para atender futuras necessidades.

Com relação aos trabalhos correlatos citados na seção 2.6 do trabalho, apresentam da mesma forma que este, o desenvolvimento de aplicações voltadas para a apuração de estimativas baseadas em métricas. No entanto, a ferramenta desenvolvida neste trabalho de conclusão de curso torna flexível a escolha pelo usuário por qual métrica melhor atende às suas necessidades de projeto. Também permite a calibração da mesma caso hajam mudanças nos índices publicados para cada métrica conforme sua evolução.

Por fim, a *interface* web facilita a publicação e acesso à ferramenta e segue as tendências atuais presentes no desenvolvimento de soluções de software.

4 CONCLUSÕES

Para mim foi gratificante desenvolver a ferramenta SPE. Ela vai de encontro à solução de um grande problema que existe na área de desenvolvimento: a geração de estimativas confiáveis em projetos de software. Tanto que a partir do seu desenvolvimento já comecei a usá-la em minhas atividades profissionais.

A ferramenta SPE permite o cadastro e controle de mais de um projeto ao mesmo tempo, sendo que cada projeto pode optar pelas métricas de FPA, UCP, ou COCOMO II. Isto torna a escolha para o usuário mais flexível e os resultados mais abrangentes, pois podemos contemplar vários tipos de projetos de software na mesma ferramenta.

Cada projeto permite dar a entrada dos dados pertinentes à métrica escolhida para o projeto e a ferramenta produz analises das entradas com base nas regras de cálculo da métrica escolhida. Em qualquer momento, pode-se processar o cálculo de estimativas do projeto.

A ferramenta apresentará os resultados de acordo com a métrica escolhida em forma de relatórios em formato PDF visualizados em tela que podem servir para consulta e impressão.

No geral não houveram grandes problemas no desenvolvimento da ferramenta. A maior dificuldade foi na fase de levantamento da fundamentação teórica, mais especificamente quando o assunto é COCOMO. Apesar desta métrica já estar em sua segunda versão, existe pouca material disponível sobre a mesma, tanto que a maioria dos autores faz no máximo breves comentários e que em sua maioria são sobre a versão 81.

O SPE possui *interface* web e funciona a partir do navegador de internet. Os testes foram feitos usando o navegador Microsoft Internet Explorer versão 6.0. Ao acionar o link para o SPE, o mesmo permite realizar controle de acesso com uso de senhas, permitindo acesso apenas aos usuários cadastrados.

Para implementação do SPE foi usada a ferramenta GeneXus versão 9, na sua atualização de número 4. A linguagem adotada na geração dos fontes foi o Microsoft .Net com Framework versão 2.0.

O banco de dados escolhido para armazenamento das tabelas da ferramenta foi o Oracle 9i, devido à sua popularidade no mercado de software. Também foram realizados testes com o Microsoft SQL Server 2000 com pleno sucesso.

Conforme citado anteriormente, estudos mostram que o gerenciamento e a execução de projetos podem obter maior sucesso se auxiliados pela utilização de métricas que permitam mensurar e consequentemente gerar estimativas de prazo, custos e recursos.

Para minimizar distorções é importante que a medida do produto do trabalho seja padronizada e uniforme para tarefas iguais ou similares e é preferível que o esforço seja medido em termos de dedicação exclusiva ao trabalho em questão.

FPA, UCP e COCOMO possuem como objetivo básico comum o auxilio no cálculo de estimativas de desenvolvimento de software, provendo um fator de parâmetro em projetos de software.

O problema é que os cálculos em si são complexos e levam em conta uma série de parâmetros e variáveis. Calcular isto, mesmo com o auxílio de planilhas, torna-se arriscado. Se dados não forem considerados ou alimentados corretamente, os resultados não expressarão a realidade com exatidão. Também deve-se lembrar que quanto maior o projeto, maior a quantidade de dados a serem inseridos nos cálculos.

Pelas vantagens obtidas com a adoção das métricas de FPA, UCP e COCOMO no cálculo de estimativas dos projetos de software, muito melhor é ter uma ferramenta web que utilize estas métricas como parâmetro no gerenciamento de projetos de software e padronize o trabalho dos envolvidos.

4.1 EXTENSÕES

Algumas sugestões para trabalhos futuros incluem:

a) melhorar a *interface* da ferramenta, tornando-a mais amigável através do uso de cores, fontes diferenciados e imagens nos botões e telas. Isto torna a ferramenta mais atrativa do ponto de vista do usuário;
b) evoluir a versão do GeneXus e fazer a geração de gráficos dos projetos para as métricas implementadas, apresentando os impactos de cada componente. Com isto fornece-se uma forma menos técnica e de fácil compreensão para os leigos nas métricas e que necessitam de compreender as estimativas para acompanhamento e tomadas de decisão sobre os projetos;
c) criar históricos de projetos buscando comparar o previsto com o realizado. Produzir novas consultas e relatórios que apontem estas diferenças. A contagem ao final do levantamento de requisitos e/ou projeto pode ser comparada com pontos de função realmente entregues pelo desenvolvedor do software. Se o índice tiver aumentado, pode-se assumir que o projeto tornou-se mais bem definido, ou

realmente cresceu em tamanho. A quantidade crescida é um indicador da qualidade do levantamento dos requisitos e/ou de sua comunicação à equipe do projeto. O aumento do tamanho dos projetos deve ser acompanhado em todos os casos. Se a taxa de crescimento dos projetos diminuir ao longo do tempo, é natural assumir que houve melhoria na comunicação com o usuário.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDRADE, Edméia L. P. **Pontos de casos de uso e pontos de função na gestão de estimativa de tamanho de projetos de software orientados a objetos**. 2004. 143 f. Dissertação (Mestrado em Gestão do Conhecimento e Tecnologia da Informação) – Curso de Pós-graduação em Gestão do Conhecimento e Tecnologia da Informação, Universidade Católica de Brasília, Brasília.

ANJOS, Lúcio A. M. **Uma introdução à análise de pontos de função**. Recife, 2003. Disponível em: <http://www.cin.ufpe.br/~crsj/pos/disciplinas/TAES3/Pontos-de-funcao lucio6>. Acesso em: 11 nov. 2007.

BERLANDA, Ana P. **Sistema de cálculo de custo de desenvolvimento de software utilizando FPA**. 2005. 83 f. Trabalho de Conclusão de Curso (Sistemas de Informação - Bacharelado) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CHAVES, Frederico F**. Roteiro para aplicação de métricas e qualidade em subcontratação de software**. 2003. 46 f. Monografia (Curso de Especialização em Informática Pública da Prodabel – Empresa de Informática e Informação do Município de Belo Horizonte) – Centro de Capacitação, Pontífice Universidade Católica de Minas Gerais, Belo Horizonte.

DEKKERS, Carol A. **Desmistificando pontos de função: e**ntendendo a terminologia. [S.l.], 1998. Disponível em: <http://www.bfpug.com.br/Artigos/Desmistificando%20Pontos%20de%20Fun%C3%A7%C3%A3o.pdf>. Acesso em: 20 set. 2007.

FATTO CONSULTORIA E SISTEMAS. **Análise de pontos de função**. [S.l.], 2008. Disponível em: <http://www.fattocs.com.br/download/cartaoAPFCompleto.pdf>. Acesso em: 24 maio 2008.

FERNANDES, Aguinaldo A. **Gerência de software através de métricas**: garantindo a qualidade do projeto, processo e produto. São Paulo: Atlas, 1995.

FLORAC, William A.; PARK, Robert E.; CARLETON, Anita D. **Practical software measurement:** measuring for process management and improvement. [S.l.], 1997. Disponível em: <http://www.sei.cmu.edu/pub/documents/97.reports/pdf/97hb003.pdf>. Acesso em: 20 set. 2007.

GALVÃO, Ana M. **Pontos de função como ferramenta de gerenciamento de projetos**. [S.l.], 1999. Disponível em: <www.bfpug.com.br/Artigos/Palestra_introdutoria_FPA.ppt>. Acesso em: 20 set. 2007.

GIELOW, Sandra C. **Ferramenta de suporte ao cálculo dos use case points**. 2003. 71 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

HAZAN, Claudia; FUKS, Hugo; LUCENA, Carlos J. P. **Avaliação do tamanho funcional de ferramentas de e-learning**. 2005. 28 f. Monografia (Bacharelado em Ciência da Computação) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

IFPUG - International Function Point Users Group. [S.l.], 2008. Disponível em: <http://www.ifpug.org/>. Acesso em: 10 jun. 2008.

ETS: listening, learning, leading. [S.l.], 2004. Disponível em: <http://www.toefl.org>. Acesso em: 15 maio 2004.

GONDA, Breogán; JODAL, Nicolás. **Knowloede-based development**: philosophy and theorical foundation of GeneXus. Mondevideo, 2007. Disponível em: <http://www.genexus.com/portal/hgxpp001.aspx?2,32,583,O,E,0,MNU;E;130;1;MNU;,>. Acesso em: 20 jun. 2008.

LONGSTREET, David. **A utilidade dos pontos de função**. [S.l.], 2000. Disponível em: <http://www.bfpug.com.br/Artigos/MuitosUsos.htm>. Acesso em: 20 set. 2007

LISBOA, Daniel M. **GeneXus**: desarrollo basado em el conocimiento. Montevideo. Grupo Mago, 2006.

MAGELA, Rogério. **Engenharia de software aplicada:** fundamentos. Rio de Janeiro: Alta Books, 2006.

PEREIRA, Roberto et al. Estimativas de software: o estudo de uma aplicação prática utilizando a técnica de use case points. In: ESCOLA REGIONAL DE INFORMÁTICA – PARANÁ, 14, 2007, Guarapuava. **Anais eletrônicos...** Guarapuava: UEM, 2007. p.1-12. Disponível em: <http://www.sbc.org.br/bibliotecadigital/download.php?paper=686>. Acesso em: 11 nov. 2007.

PEREIRA JUNIOR, Lindolfo. **Protótipo de uma ferramenta para gerência de custos em projetos de software baseada no modelo PMBOK**. 2003. 83 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

PETERS, James F.; PEDRYCZ, Witold. **Engenharia de software:** teoria e prática. Rio de Janeiro: Campus, 2001.

PRESSMAN, Roger S.. **Engenharia de software**. 5. ed. Rio de Janeiro: McGraw-Hill, 2002.

REZENDE, Denis A. **Engenharia de software e sistemas de informação.** 3. ed. Rio de Janeiro: Brasport livros e multimídia Ltda, 2005.

ROCHA, Cláudio M. **Explorando o relacionamento entre métricas baseadas em caso de uso e o número de casos de teste.** Curitiba: 2005. 81 f. Dissertação (Mestrado em Informática) – Setor de Ciências Exatas da Universidade Federal do Paraná, UFPR, 2005. Disponível em: <http://dspace.c3sl.ufpr.br/dspace/bitstream/1884/2580/1/dissfim.pdf>. Acesso em: 25 maio 2008.

SOMMERVILLE, Ian. **Engenharia de software**. 6. ed. São Paulo: Addison Weskey, 2003.

TAVARES, Helena C.A.B.; CARVALHO, Ana E.; CASTRO, Jaelson F.B. Medição de pontos de função a partir da especificação de requisitos. In: WORKSHOP EM ENGENHARIA DE REQUISITOS, 5., 2002, Valencia, Espanha. **Anais eletrônicos**... Valencia, Espanha: 2002. p.1-21. Disponível em: < http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER02/tavares.pdf>. Acesso em: 20 set. 2007.

TRINDADE, André L. P.; PESSOA, Marcelo S. P.; SPINOLA, Mauro M. COCOMO II: uma compilação de informações sobre a nova métrica. In: CONGRESSO INTERNACIONAL DE ENGENHARIA INFORMÁTICA DA UNIVERSIDADE DE BUENOS AIRES, 5, 1999, Buenos Aires, Argentina. **Anais eletrônicos...** Buenos Aires, Argentina: USP, 1999. p. 1-17. Disponível em: <http://sites.ffclrp.usp.br/ccp/(SEM%205)/ES1/Cocomo/cocomo2.pdf>.

TRINDADE, André L. P. **Métricas para orçamento e planejamento da produção de software**. São Paulo: 1999. 156 f. Dissertação (Mestrado em Engenharia de Produção) – Escola Politécnica da Universidade de São Paulo, São Paulo.

VALCANAIA, Tibério C. **Protótipo de ferramenta de cálculo de FPA sobre Microsoft Access 97**. 1998. 126 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

VAZQUEZ, Carlos E.; SIMÕES Guilherme S.; ALBERT, Renato M. **Análise de pontos de função**: medição, estimativas e gerenciamento de projetos de software. 7. ed. São Paulo: Érica, 2007.

VIEIRA, Caetano Y. **UC-Measurer**: sistema para cálculo de prazos e custos na implementação de softwares baseado na apuração de pontos por caso de uso. 2007. 113 f. Trabalho de Conclusão de Curso (Curso de Ciência da Computação) – Instituto de Ciências Exatas e Tecnológicas, Centro Universitário Feevale, Novo Hamburgo.

XEXÉO, Geraldo. **Modelagem de sistemas de informação** – da análise de requisitos ao modelo de *interface*. Edição Ago/2007. San Francisco: Creative Commons, 2007. Disponível em: < http://wiki.xexeo.org/tiki-download_file.php?fileId=163>. Acesso em: 25 maio 2008.