UNIVERSIDADE NOVE DE JULHO – UNINOVE

ANDREA DAS NEVES AMORIM
ELIZA SANTOS GOTARDI

ESTUDO SOBRE TÉCNICAS DE DETERMINAÇÃO DE TAMANHO DE SOFTWARE: UMA ANÁLISE COMPARATIVA ENTRE PONTOS POR FUNÇÃO E PONTOS POR CASO DE USO.

SÃO PAULO
2012

ANDREA DAS NEVES AMORIM
ELIZA SANTOS GOTARDI

ESTUDO SOBRE TÉCNICAS DE DETERMINAÇÃO DE TAMANHO DE SOFTWARE: UMA ANÁLISE COMPARATIVA ENTRE PONTOS POR FUNÇÃO E PONTOS POR CASO DE USO.

Trabalho de Conclusão de Curso apresentado à Universidade Nove de Julho para obtenção do grau de Bacharel em Ciência da Computação.

Linha de Pesquisa: Engenharia de Software

Orientador: Prof. Gabriel Lara Baptista

SÃO PAULO
2012

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AIE	Arquivos de Interface Externa.
ALI	Arquivos Lógicos internos.
APF	*Function Point Analysis* (Análise de Ponto por função).
C#	*C Sharp* (Linguagem de programação orientada a objeto)
CE	Consulta Externa.
CFP	Número de Pontos de Função não Ajustados das Funções de Conversão.
CFPS	*Certified Function Point Specialist* (Especialistas em Pontos por Função).
CMM	*Capability Maturity Model* (Modelo de Capacidade e Maturidade)
CMMI	*Capability Maturity Model for Development* (Modelo de Capacidade e Maturidade Integrada).
COCOMO II	*Constructive Cost Model* (Modelo construtivo para custo).
CT	Contagem Total (Pontos por Função Brutos).
DFP	Número de Pontos de Função do Projeto em desenvolvimento.
EE	Entrada Externa.
EF	*Environmental Factor* (Fator Ambiental).
FA	Fator de Ajuste.
FCT	Fatores de Complexidade Técnica.
FI	Fator de Ajuste de Complexidade.
FIOO	Fator de Ajuste para Orientação a Objeto.
PFOO	*Object Oriented Function Points* (Pontos por Função Orientados á Objeto).
IEC	*International Electrotechnical Commission (*Comissão Electrotécnica Internacional*).*

IFPUG	*International Function Point Users Group* (Grupo internacional de usuários de pontos de função).
ISO	*International Standardization Organization* (Organização Internacional para Padronização).
LOC	*Lines of Code* (Linhas de código).
PCU	*Use Case Points* (Pontos por Casos de Uso).
PF	*Function Point* (Ponto por função).
PFT	Pontos de Fatores Técnicos.
PSM	*Practical Software and Systems Measurement* (Medição prática de Softwares e sistemas).
PTA	Pontos de Atores.
PTCU	Totais de caso de uso.
RUP	*Rational Unified Process* (Processo Unificado Racional)
SE	Saída Externa.
SFT	Solução do calculo de Fatores Técnicos.
UAW	*Unadjusted Actor Weight* (Somatória dos Atores).
UFP	Número de Pontos de Função não Ajustados disponíveis depois da instalação.
UUCW	*Unadjusted Use Case Weight* (Pesos de caso de uso não ajustados).
VFA	Valor do Fator de Ajuste.

SUMÁRIO

DEDICATÓRIA

Dedicamos este trabalho, aos nossos pais e familiares que são a razão da nossa existência e que tanto nos apoiaram nesta difícil tarefa.

AGRADECIMENTOS

A nossas companheiras de Iniciação Científica Caroline Oliveira e Juliana Alves que compartilharam pesquisas e estudos. Ao nosso orientador Gabriel que esteve ao nosso lado nos orientando e procurando da melhor forma nos proporcionar crescimento na área acadêmica e a todos os companheiros de trabalho que contribuíram para que este trabalho fosse concluído.

EPÍGRAFE

"Falta de tempo é desculpa daqueles que perdem tempo por falta de métodos."

Albert Einstein

RESUMO

A utilização das métricas pontos por função e pontos por caso de uso ajudam a identificar e quantificar as funcionalidades requeridas para um projeto de *Software*, além de auxiliar na redução de falhas em projetos por questão de prazos não cumpridos, funcionalidades abaixo do esperado e custos elevados. No entanto, por existirem diferenças e limitações em cada uma das técnicas, e por haver dificuldade na escolha de qual desses pontos deve-se utilizar em determinados projetos, o presente trabalho apresenta um estudo comparativo entre essas duas técnicas de determinação de tamanho. Além disso, para auxiliar o gerente de projetos na escolha entre a melhor técnica foi produzida uma ferramenta para análise de cenário que será baseada nas informações de cada projeto.

Palavras-chave: Métricas de Software; Estimativas de Tamanho de Software; Análise de Pontos por Função; Análise de Pontos de Caso de Uso.

ABSTRACT

The use of Function Point and Use Case Point helps to identify and quantify the features required in a software project. Besides, these techniques help reduce project failure in deadlines, expected functionalities and costs. However, there are differences, limitations and difficulties to choose the technique to be used in certain projects. For this reason, this paper presents a comparative study between these two approaches. Furthermore, the project developed a tool for assisting the project manager to choose between the best techniques, according to scenario analysis, based on project information.

Keywords: Software Metrics; Estimation of Software Size, Function Point Analysis, Use Case Points Analysis.

1 INTRODUÇÃO

A Engenharia de *Software* é a disciplina que lida com a produção e entrega de *Software* dentro de um determinado prazo e orçamento, com o objetivo de atender as necessidades dos usuários (SOMMERVILLE, 2007). Dentro da Engenharia de *Software*, uma das dificuldades está relacionada com a definição dos custos, que muitas vezes causam impactos na produtividade e qualidade de *Software* (PRESSMAN, 1995).

Segundo Bartié (2002), um dos principais fatores que afetam a atribuição de preço de *Software* são incertezas de estimativas de custo. Sommerville (2007), afirma que não existe uma maneira simples de fazer uma estimativa precisa de esforço necessário para desenvolver um *Software.* A estimativa inicial deve ser baseada em uma definição de requisitos de usuário de mais alto nível. Ele ainda afirma que existe uma grande dificuldade em avaliar a precisão das diferentes abordagens para as técnicas de estimativas de custo.

Sommerville (2007) também afirma que a comparação de produtividade por meio de linguagem de programação pode dar impressões enganosas da produtividade do programador. Por tal razão quanto mais expressiva for a linguagem, menor será a produtividade aparente. Um exemplo disso é a métrica de linhas de código (LOC), que se aplica apenas ao processo de programação. Mas quando se calcula o tempo de desenvolvimento, é necessário que todas as atividades de desenvolvimento sejam consideradas (KOSCIANSKI, 2002). Portanto, se uma linguagem requer mais linhas de código do que outra para implementar a mesma funcionalidade, a estimativa de produtividade será anormal (SOMMERVILLE, 2007).

Contudo, para evitar as anomalias mostradas anteriormente Sommerville (2007) sugere o uso de alguma medida de funcionalidade do código, uma vez que a funcionalidade é independente da linguagem de implantação.

McDonnell (1994) descreve e compara brevemente várias medidas baseadas em funções. A mais conhecida dessas métricas é a contagem de pontos de função que foi proposta por Albrecht (1979) e refinada por Albrecht e Gaffney (1983).

Outra medida que vem sendo explorada é conhecida como pontos por caso de uso. A métrica de pontos por caso de uso foi desenvolvida por Gustav Karner (1993), com adaptação específica dos pontos por função.

Baseados nas afirmações mencionadas, esse trabalho contém um estudo comparativo sobre as técnicas de determinação de tamanho de pontos por função e pontos por caso de uso. Espera-se que a ferramenta produzida de análise de cenário auxilie na escolha entre a melhor técnica baseado em informações de projetos.

1.1 DELIMITAÇÃO DO TEMA

O objeto de estudo desse trabalho aborda duas técnicas para estimativa de tamanho de *Software*, realizando uma análise comparativa entre elas. As técnicas em questão são pontos por função e pontos por caso de uso.

1.2 PROBLEMA

Segundo Bartié (2002), existem diversos fatores que afetam um processo de *Software.* Em modo geral, o problema está na dificuldade em associar estimativa de esforço e tempo com atividades do projeto. Segundo Pressman (1995) as técnicas para realizar estimativas são mal desenvolvidas, e refletem uma pressuposição não expressa que é bastante inverídica.

A grande dificuldade em estabelecer um tipo de métrica para cada projeto acontece frequentemente, muitos gerentes aplicam diferentes técnicas de estimativas usando uma como prova cruzada para a outra, mas muitas das vezes essa não é a melhor maneira para adquirir uma estimativa de esforço e tempo de forma quantitativa (PRESSMAN, 1995; KOSCIANSKI, 2007; SOMMERVILLE, 2007).

Segundo dados de estudos americanos mencionado por Bartié (2002), demonstram que mais de 30% dos projetos são cancelados antes de serem finalizados e que mais de 70% dos projetos falham nas entregas das funcionalidades

esperadas. Além dos custos extrapolados os prazos excedem em mais de 200% dos cronogramas originais e com isso os custos chegam a mais de 180% os valores originalmente previstos.

Dados de uma pesquisa realizada pelo grupo Standish (2009), apenas 32% dos projetos de *Software* são finalizados com sucesso, os demais encerram com falha por problemas de custo, prazo e/ou funcionalidades.

Dados recentes de uma pesquisa realizada por Baptista (2011) demonstram que ainda existe uma grande dificuldade para avaliar o desempenho dos projetos de *Software*, já que 55% afirmam que tal avaliação não faz parte da avaliação dos resultados financeiros da empresa. Vale destacar que 87% das empresas respondentes estão localizadas no estado de São Paulo.

Por tais razões, a presente pesquisa responde quais são as diferenças e limitações existentes nas técnicas de estimativa de ponto por função e ponto por caso de uso. A pesquisa também responde se existem situações em que uma das técnicas deve ser escolhida por conta de características do projeto de *Software*, tornando-se assim mais vantajosa do que a outra.

1.3 HIPÓTESE(S)

Supõe-se que o estudo sobre técnicas de determinação de tamanho de *Software*, através de uma análise comparativa entre pontos por função e pontos por caso de uso, auxiliará na determinação de uma ferramenta de análise de cenário que buscará responder a que tipo de projeto uma das métricas é melhor do que a outra.

1.4 OBJETIVOS

1.4.1 Objetivos gerais

Tem-se como objetivo geral da pesquisa comparar as técnicas de medição de tamanho de pontos por função e pontos por caso de uso, no que diz respeito a que tipo de projeto uma é melhor do que a outra.

1.4.2 Objetivos específicos

Conhecer as principais características da PF e PCU. Analisar se há relação entre as métricas PF e PCU no desenvolvimento do projeto de *Software* e levantar pontos comparativos entre as técnicas.

Criar uma ferramenta de análise de cenário que suporte o gerente de projetos na tomada de decisão entre uma das técnicas para assim monitorar as decisões a serem tomadas durante o controle e acompanhamento do projeto.

Sendo assim, orientar os gerentes sobre uma melhor forma de estimar, revisar e recalcular o tamanho do projeto a partir das métricas PF e PCU à medida que novos artefatos estejam disponíveis durante o processo de desenvolvimento do projeto de *Software,* provendo subsídios para melhorar as decisões gerenciais.

1.5 JUSTIFICATIVA

A competição entre as organizações de desenvolvimento de *Software* vem aumentando com o crescimento do mercado tecnológico (BARTIÉ, 2002). Em consequência desta realidade, as organizações têm-se preocupado cada vez mais com a melhoria da qualidade de seus produtos de *Software* e com os custos efetivos para cumprimento de cronogramas em seus projetos de desenvolvimento (COELHO

et. al., 2007). Todas estas características dependem de um gerenciamento do processo de desenvolvimento de *Software* eficiente e eficaz (ANDRADE e OLIVEIRA, 2004).

Andrade (2004) afirma que são necessárias estimativas de tamanho no início do projeto para discussão de contratos ou determinação da viabilidade do projeto em termos da análise de custos e benefícios.

Andrade (2004) ainda afirma que é necessário à obtenção de estimativas mais precisas no início do projeto, durante todo o ciclo de desenvolvimento do mesmo para que seja possível reduzir os problemas de gestão de altos custos, atrasos no cronograma, insatisfação do usuário e dificuldades de medição do progresso do projeto.

Com base nos dados levantados, a presente ferramenta foi desenvolvida com o intuito de auxiliar gerentes de projeto a estimar de forma eficiente e eficaz os custos e prazos de cada projeto utilizando as técnicas de pontos por função ou pontos por caso de uso, além de conscientizá-los da importância da aplicação de tais técnicas.

1.6 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado da seguinte forma: O primeiro capítulo apresentou o principal objetivo deste trabalho, um estudo sobre técnicas para medir *Software*: pontos por função e pontos por caso de uso e apresentou também as principais razões e objetivos para solucionar os problemas apresentados. No segundo capitulo, será apresentado o conhecimento mais detalhado de planejamento e medição de *Software* relacionado aos pontos por função e pontos por caso de uso. O terceiro capítulo apresenta a metodologia utilizada para o desenvolvimento da ferramenta que fará a comparação entre PF e PCU. No quarto capitulo será apresentado a modelagem da ferramenta de apoio e seu respectivo código fonte desenvolvido em C# além da apresentação da tabela de comparação entre as técnicas e tabela de decisão entre PF e PCU. Será apresentado também o conteúdo dos estudos feito em base de análise de cenário das técnicas de medição

de tamanho através da ferramenta comparativa onde permitirá ao gerente de projetos, tomar decisão sobre qual das técnicas é a melhor escolha para determinados projetos. Por fim, o quinto capítulo apresenta a conclusão da pesquisa, abordando questões relacionadas aos objetivos alcançados, os pontos de aprendizado, as dificuldades e as limitações encontradas no trabalho além de apresentar sugestões para trabalhos futuros.

2 REVISÃO DA LITERATURA

2.1 PLANEJAMENTO E MEDIÇÃO DE SOFTWARE

O processo de planejamento de *Software* inclui passos para medir o tamanho do *Software* e com isso obter as estimativas de recursos necessários para produzir uma programação, identificar e avaliar os riscos de *Software*, e também negociar compromissos (ABSTS et. al., 2000).

O propósito do planejamento de projeto de *Software* é estabelecer planos razoáveis baseados no desenvolvimento de estimativas realistas e assim estabelecer os compromissos necessários para desenvolver (CHRISSIS et. al., 1995).

Cimino (1987), da área de Engenharia Civil, afirma que quando se decide um processo construtivo para a execução de um planejamento, é necessário fazer uma análise de eventos comuns e especiais para cada tipo de obra. Assim sendo, recorre-se a cultura de conhecimento na execução dos serviços ou especialistas, para obter informações cadastradas que facilitam e agilizam todo processo de desenvolvimento de um planejamento. Isso não é diferente na área de Engenharia de *Software* (SOMMERVILLE, 2007).

A Engenharia de *Software* depende muito do calendário, de modelos e técnicas de estimativa que incluem o orçamento e analise de risco, para a precisão da estimativa global e análise adicional importante das sensibilidades de custo e cronograma de decisões de projeto de *Software* para o escopo, recursos humanos, ferramentas, reutilização, etc. (ABSTS et. al., 2000).

Na Engenharia de *Software* o plano é documentado e mantido como uma ferramenta necessária para gerenciar o projeto de *Software*. Sendo assim, o objetivo é gerenciar requisitos e estabelecer um entendimento comum entre o cliente e o projeto de *Software*. Este acordo com o cliente é a base de tudo para planejar e gerenciar o projeto de *Software* para então estabelecer as restrições e metas que definem e limitam cada projeto de *Software* (CHRISSIS et. al., 1995).

Contudo, DeMarco (1991), afirma que os engenheiros de *Software* são reconhecidamente maus estimadores. Ele ainda afirma que isso ocorre por muitas vezes os engenheiros de *Software* não se especializam em estimativas e com isso não fazem as previsões adequadas para contrabalança o efeito de distorções e então não sabem exatamente como lidar com os problemas políticos que dificultam o processo de estimativa.

Baptista (2011) ainda afirma que a Engenharia de *Software*, vem sofrendo ao longo dos anos com questões relacionadas a prazos, custos e funcionalidades, o que torna tais fatores desafiadores para a área.

DeMarco (1991), também afirma que a estimativa é a essência da dificuldade em controlar projetos de *Software* e por esta razão, quando as expectativas excedem qualquer possibilidade de resultado, os projetos são condenados.

Bartié (2002) considera que um dos principais fatores que afetam a atribuição de preço de *Software* são incertezas de estimativas de custo. Estudos apresentados por Bartié (2002) demonstram que a maior incidência dos erros se concentra nas fases iniciais do processo de desenvolvimento onde a maior parte das incidências de defeitos acontece na analise dos requisitos com 56% dos problemas, 27% na analise e modelagem, 7% na implementação e os demais 10% são outros.

Por esta razão, para controlar o desenvolvimento do *Software* dentro dos custos, prazos e níveis de qualidade desejados a engenharia de *Software* recomenda a implantação de atividades de estimativas de tamanho, esforço, prazo e custo, como formas de melhorar o planejamento e o acompanhamento de projetos de *Software* a partir de uma estimativa inicial (PRESSMAN, 1995).

Koscianski, (2007) sugere o uso de métricas em processos de *Software* para obter estimativas precisas com qualidade desde o inicio da produção até final, dessa forma, poderá ser identificado prematuramente os erros e com isso impedir que erros migrem para outras fases.

Para evitar que erros migrem de uma fase a outra Bartié (2002), sugere testar o projeto de *Software* nas fases iniciais, pois sairá mais barato do que nas fases mais tardias do processo de desenvolvimento, caso contrário os custos de correção irão se multiplicar.

A preocupação com a medição acontece não só na área de engenharia como também na área de gestão de projetos. Drucker (1993) afirma que não se pode medir o que não se pode gerenciar, o que demonstra a importância da utilização de um tipo de métrica para obter uma estimativa precisa de um projeto de *Software.* DeMarco (1991), também afirma que não se pode controlar o que não se pode medir.

Segundo Almeida e Paes (2005) é possível acompanhar o desenvolvimento de um projeto, e verificar se as estimativas iniciais são válidas no decorrer do projeto através das medições.

Sendo assim é essencial obter estimativas de custo e tempo para que o produto seja entregue dentro do prazo e do orçamento estimado e com isso garantir a qualidade do *Software* (BARTIÉ, 2002; KOSCIANSKI, 2007).

Entretanto, Sommerville (2007) afirma que o relacionamento entre o custo de projeto e o preço para o cliente não é normalmente tão simples de se obter, pois as atribuições de preço devem ser obtidas levando em conta considerações amplas sobre questões organizacionais, econômicas, políticas e de negócios. Por conta disso que a qualidade do *Software* também depende muito da produtividade do programador (PRESSMAN, 1995).

Fatores ambientais afetam o nível de experiência da equipe e a estabilidade do projeto. Quaisquer valores negativos significam que serão gastas mais horas treinando pessoal ou corrigindo problemas devido à instabilidade, e com isto menos tempo será dedicado ao projeto (GRAHL e HEIMBERG, 2005). Por tal razão é preciso estar atento a qualquer ajuste nos fatores ambientais, pois pode implicar em grandes mudanças no resultado final da estimativa. Mesmo meio ponto pode implicar em uma mudança muito significativa (SOMMERVILLE, 2007).

2.2 PONTOS POR FUNÇÃO

Segundo Aguiar (2009), a métrica de ponto de função é muito indicada por vários pesquisadores por ter uma grande capacidade de armazenar dados de diversas organizações.

Essa técnica tornou-se padrão internacional em 2002 através da norma ISO/IEC 20.926 (IFPUG, 2003). Com isso, a PF estabelece uma medida de tamanho, para se torna visíveis as atividades de produtividade da empresa (AGUIAR, 2003).

Sommerville (2007) afirma que para que as estimativas por pontos de função funcionem corretamente o gerente sênior deve tomar decisões estratégicas para a atribuição de preço do projeto.

Dados de uma pesquisa realizada em 2008 pelo IFPUG revelam que o Brasil contém o maior número de profissionais certificados pelo IFPUG com especialização em pontos de função no mundo (BFPUG, 2008).

2.3 PASSOS PARA UTILIZAR A TÉCNICA APF

O manual de práticas de contagem de pontos de função escrito por vários especialistas sugerem que antes de tudo os desenvolvedores devem traduzir a informação do usuário para informações em linguagem técnica a fim de prover uma solução, pois a visão do cliente representa uma descrição formal das necessidades do negócio do usuário (BROWN et. al., 2005).

Com base nesta descrição, Andrade e Oliveira (2005) afirmam que o processo de contagem da APF será bem estabelecido e deverá ser realizado em sete passos. Na fase de projeto, a contagem é revista com base na complexidade das funções e ao término do desenvolvimento do *Software,* é realizada uma contagem detalhada, obtida a partir do grau de complexidade das funções levantadas no processo funcional, no modelo de dados onde contém as descrições de telas e relatórios.

Os sete passos do processo de contagem da PF se resume da seguinte forma: Determinar o tipo de contagem, Identificar o escopo de contagem e a fronteira da aplicação, contar funções do tipo de dados, contar funções tipo transação, determinar valor do tipo fator de ajuste, determinar contagem de pontos de função não ajustados e por fim calcular o número de pontos de função ajustados (DRACH, 2005).

Os elementos da contagem para funções tipo dado são definidos basicamente pelas funções relacionadas aos dados utilizados pelo *Software* que englobam os arquivos lógicos internos (ALI), onde se calculam também os arquivos de interface externa (AIE) as funções transacionais que são as funções básicas que o *Software* deve conter como entrada externa (EE), saída externa (SE) e consulta externa (CE), (ANDRADE, 2005; DRACH, 2005; SIMÕES, 2004).

Segundo Machado (2008) depois de identificados e contados os tipos de dados o próximo passo será classificar a complexidade funcional para então efetuar o calculo dos pontos de função não ajustados. A tabela 1 representa os valores das complexidades funcionais para ALI e AIE.

Registros Lógicos	**Tipos de dados**		
	1 – 19	**20 – 50**	**>50**
1	Baixa	Baixa	Média
2-5	Baixa	Média	Alta
>5	Média	Alta	Alta

Tabela 1 – Complexidade funcional para ALI e AIE.

Fonte: Machado (2008).

Os arquivos lógicos internos (ALI) representam grupo de dados mantidos pelo sistema e relacionados logicamente e os arquivos de interface externos (AIE) representam grupo de dados referenciados pelo sistema de dados relacionados logicamente (DRACH, 2005; MACHADO, 2008; NESMA, 2009).

As entradas externas (EE) são a transação lógica que introduz dados recebidos de outra aplicação no sistema e que causam mudança nos ALI de alteração, exclusão ou inserção de dados. Isso ocorre, pois as entradas externas são um processo elementar que processam dados ou informação de controle que entram pela fronteira da aplicação e o objetivo principal será manter um ou mais arquivos lógicos internos e/ou alterar o comportamento do sistema (HAZAN, 2005; MACHADO, 2008). A tabela 2 é referente a classificação desses tipos de dados:

Arquivos Referenciados	Tipos de dados		
	1 – 4	5 - 15	>15
<2	Baixa	Baixa	Média
2	Baixa	Média	Alta
>2	Média	Alta	Alta

Tabela 2 – Complexidade funcional para EE.

Fonte: Machado (2008).

As saídas externas (SE) são responsáveis pelas transações lógicas em que dados são enviados, já as consulta externa (CE) são responsáveis pelas transações lógicas, ou seja, será um processo elementar que enviará dado ou informação de controle para fora da fronteira da aplicação. O objetivo principal será apresentar informação para o usuário através da recuperação de dados ou informação de controle de ALIs ou AIEs. Sendo assim o processamento lógico não há qualquer calculo matemático e não criará dados derivados. Com isso, nenhum arquivo lógico interno é mantido durante o processamento, nem o comportamento do sistema é alterado (FUKS e HAZAN, 2005; MACHADO, 2008, SERPRO, 2010). A tabela 3 é referente a classificação desses tipos de dados:

Arquivos Referenciados	Tipos de dados		
	<6	6 - 19	>19
<2	Baixa	Baixa	Média
2-3	Baixa	Média	Alta
>3	Média	Alta	Alta

Tabela 3 – Complexidade funcional para SE e CE.

Fonte: Machado (2008).

Segundo Andrade, (2005) os PF não ajustados são calculados com base no total de ALI, AIE, EE, SE e CE, e assim é determinado também o fator de ajuste com base nas 14 características gerais do *Software*, onde indicam o grau de dificuldade

para construir o *Software* e, finalmente, é aplicada uma fórmula para determinar os Pontos de Função (PF) ajustados. Segue a tabela 4 referente aos tipos de complexidade das funções não ajustadas:

Tipo de Função	**Complexidade dos Tipos de Função**		
	Baixa	**Média**	**Alta**
AIE	5 PF	7 PF	10 PF
ALI	7 PF	10 PF	15 PF
EE	3 PF	5 PF	7 PF
SE	4 PF	5 PF	7 PF
CE	3 PF	4 PF	6 PF

Tabela 4 – Calculo dos pontos de função não ajustados.

Fonte: Machado (2008).

Segundo Machado (2008), para calcular os pontos por função ajustados é necessário obter os valores de DFP que será o número de pontos de função do projeto em desenvolvimento e os UFP que será o número de pontos de função não ajustados das funções disponíveis depois da instalação. Por fim, obter os valores de CFP que será o número de pontos de função não ajustados das funções de conversão e VFA que será o valor do fator de ajuste. O calculo será como na seguinte formula: DFP = (UFP + CFP) X VFA. Sendo assim, o fator de ajuste será calculado baseado em 14 (catorze) características gerais do sistema, listadas na tabela 5, a seguir:

Características			
1	Comunicação de Dados	8	Atualização On-Line
2	Processamento Distribuído	9	Complexidade de Processamento
3	Performance	10	Reutilização
4	Configuração Altamente Utilizada	11	Facilidade de Instalação
5	Volume de Transações	12	Facilidade de Operação
6	Entrada de Dados On-Line	13	Múltiplos Locais
7	Eficiência do Usuário Final	14	Facilidade de Mudanças

Tabela 5 – Características gerais do sistema.

Fonte: Machado (2008).

Segundo Machado (2008), após determinar todos os níveis de influencia o fator de ajuste (FA) poderá ser representado os valores das 14 características do sistema com a seguinte formula: FA = 0,65 + ((n1 + n2 + n3 + +... + n14) x 0,01). Assim, será possível identificar no sistema os níveis de influencia sobre a aplicação, variando de 0 a 5, como demonstra na tabela 6, a seguir:

Valor	Grau de Influência
0	Nenhuma Influência
1	Influência Mínima
2	Influência Moderada
3	Influência Média
4	Influência Significativa
5	Grande Influência

Tabela 6 – Níveis de influência das características gerais do sistema.

Fonte: Machado (2008).

Simões (2004) afirma que os PF mensuram o tamanho do software independentemente de como será desenvolvido, além de serem diretamente compatíveis com as abordagens de casos de uso. Miranda (2002) ainda afirma que para a aplicação da Métrica PF em projetos orientados a objetos, é essencial adicionar um peso às características das classes, a qual produzirá uma medida de complexidade das mesmas, assim como se faz com os parâmetros de medidas que são aplicados nas medições de *Software* não orientados a objetos.

Miranda (2002) também afirma que o foco maior deve ser às características de reusabilidade de código produzido, assim como, considerar a utilização de componentes pré-implementados, quando da definição das bases para o novo fator de ajuste de complexidade (FI), conforme PFOO = CT x [0,65+ (0,01 x FIOO)], onde: PFOO = Pontos por Função orientados a objeto, CT = Contagem total (Pontos-por-FunçãoBrutos) e FIOO = Fator de Ajuste para orientação a objeto.

Para isso, Miranda (2002) sugere efetuar mensurações em uma grande quantidade de *Software*, para a obtenção dos dados históricos necessários ao embasamento das estimativas de próximo *Software* a serem desenvolvidos.

Com isso, será possível obter uma maior facilidade de efetuar esta coleta de dados, na medida em que desenvolve um *Software*, do que tentar obter de *Software* desenvolvidos anteriormente, sendo assim, será possível adquirir uma grande quantidade de tempo para uma implementação concretizada. (MIRANDA, 2002)

2.4 PRODUTIVIDADE POR PF DAS PRINCIPAIS LINGUAGENS

Souza (2009) sugere que primeiramente o responsável por medir o Software conheça a produtividade nas linguagens dominantes, pois a produtividade nada mais é do que quantas horas você precisa para produzir um PF.

Segundo Souza (2011) é importante ter em mente que cada organização deve conhecer a produtividade real da empresa, pois a produtividade em horas PF depende de diversos fatores e varia de empresa para empresa e o conhecimento no ambiente do cliente, senioridade, conhecimento do negócio e diversos outros fatores influenciam diretamente em ser mais ou menos produtivo.

Em um levantamento de dados de produtividade das linguagens mais utilizadas por Souza (2011) referente à produtividade por PF, foi possível perceber que a linguagem COBOL é a que mais sofre variação entre – 5,5h e + 12,75h, tendo um total de 11,5h por PF. A tabela 7 apresenta os dados levantados por Souza (2011):

Linguagem de Programação	Horas/PF	Variação
ASP	6h/PF	Com variação entre -2h e +6h
.Net (C#)	8h/PF	Com variação entre -3h e +6h
COBOL	11,5h/PF	Com variação entre -5,5h e +12,75h
Delphi	7,5h/PF	Com variação entre -1,5h e +2,5h
Java	10h/PF	Com variação entre -3h e +4,5h
Lotus Notes	4h/PF	Com variação entre -0,5h e +3h
Natural	9h/PF	Com variação entre -3h e +5h
PHP	5h/PF	Com variação entre -1h e +7h
SQL	6h/PF	Com variação entre -1,5h e +3h
VBA	8h/PF	Com variação entre -2,5h e +2h
Visual Basic	8h/PF	Com variação entre -2h e +3h

Tabela 7 – Linguagem e Variações em PF.

Fonte: Muller (2010).

2.5 VANTAGENS NA UTILIZAÇÃO DE PONTOS POR FUNÇÃO

Segundo Martins (2007), desde 1970 a métrica pontos por função predomina a indústria nacional e internacional desenvolvedoras de *Software* além de ser muito indicada por vários pesquisadores.

Um das principais vantagens da contagem de PF é a certificação CFPS promovido pelo IFPUG, pois com a certificação é possível armazenar dados de diversas organizações além de garantir a utilização correta do método APF por um especialista certificado (AGUIAR, 2009).

Hazan (2000) também afirma que atualmente a prática da APF tem demonstrado grande utilidade na garantia da completa especificação de requisitos, pois a contagem em PF usa uma descrição formal das necessidades do negócio na linguagem do usuário e apoia a uma revisão da contagem dos requisitos do sistema com o usuário.

Aguiar (2009) afirma que PF é uma excelente métrica para aferição de tamanho funcional dos requisitos elicitados. Segundo Souza (2010), PF é bem transparente e foca na visão do usuário e em eventos de mudanças de escopo, e com isso fica fácil de ambos entenderem e chegarem a um consenso.

Segundo Hazan (2005) a contagem de PF independe da forma como os requisitos do software foram expressos. Por tal razão Vazquez, (2008) afirma que a APF é conhecida como padrão para medir *Software* do ponto de vista do usuário através da quantificação da funcionalidade fornecida.

Andrade e Oliveira (2005) ainda afirmam que os PF são uma das métricas de estimativa que proporcionam resultados cada vez mais precisos à medida que novos artefatos da fase de análise e projeto são gerados além de permitirem uma contagem indicativa no início do desenvolvimento sem conhecer detalhes do modelo de dados.

Segundo Hazan (2003) a métrica PF também apoia a implantação dos modelos CMM e CMMI embora a principal finalidade da métrica pontos de função seja aferir o tamanho dos projetos de desenvolvimento e manutenção de *Software*. Hazan (2003) ainda afirma que os PF apoiam a implantação das áreas chaves de gerência de requisitos, planejamento e acompanhamento do projeto de *Software* e gerência de subcontratação de *Software* do nível 2 do modelo CMM.

Por tal razão a métrica PF não deixa de ser adequada para estimar sistemas orientados a objetos ou sistemas modelados por meio de casos de uso, além de ser capaz de apoiar implantações dos modelos CMM e CMMI, pois os PF medem o

tamanho funcional, independentemente da tecnologia que será utilizada (HAZAN, 2005; SIMÕES, 2004).

Também é possível perceber que a técnica de APF dimensiona uma aplicação na perspectiva do usuário, ao invés das características técnicas da linguagem utilizada e com isso dimensiona o *Software* e quantifica a funcionalidade proporcionada ao usuário, baseando principalmente no desenho lógico para então entrar de acordo com as medidas de comportamento externo de contagem, que são as entradas, as saídas, os requisitos, os arquivos internos e as interfaces (GUSTAFSON, 2003; HAUFE, 2001).

2.6 DESVANTAGENS NA UTILIZAÇÃO DE PONTOS POR FUNÇÃO

A métrica PF estabelece uma medida de tamanho, para se tornar visíveis as atividades de produtividade da empresa (AGUIAR, 2003). Mas para que as estimativas por pontos PF funcionem corretamente, segundo Sommerville (2007) o gerente sênior deverá tomar decisões estratégicas para a atribuição de preço do projeto.

Mesmo que seja garantida a utilização correta do método APF por um especialista com certificação CFPS, é importante destacar que a certificação deverá ser renovada a cada três anos, para garantir a atualização do especialista nas novas versões do manual ou até mesmo na versão em que ele foi certificado. (AGUIAR, 2009, HAZAN, 2005)

Martins (2007), também afirma que existem algumas desvantagens em usar a métrica pontos por função. A métrica pontos por função não é apoiada por nenhuma ferramenta que faça a contagem automaticamente, sendo que esta é uma operação manual e totalmente dependente da experiência do profissional que está fazendo a contagem, tornando a métrica totalmente subjetiva e dependente do ponto de vista de cada profissional. (MARTINS, 2007)

Vazquez (2008) também adverte que PF não são recomendados para estimativas pontuais de atividades ou projetos muito pequenos. Albert (2004) também afirma que a métrica PF não contém um preço único e por tal razão é

necessário avaliar o conjunto de atividades relativas à entrega das funcionalidades medidas em pontos de função, e o modelo de contrato ditará a remuneração de um PF e também os aspectos não funcionais que será desconsiderado na medição dos pontos de função. Haufe, (2001) também afirma que a contagem PF deve ser simples para minimizar o trabalho adicional do processo de mensuração.

Segundo Meirelles (2008) a métrica PF foi validada em alguns trabalhos ao comparar o LOC e PF como medidores de esforço de desenvolvimento e relacionando PF com a produtividade em um ambiente desenvolvimento, mas que em outros projetos, também foi possível perceber que PF por si só é insuficientemente precisa para a predição e esforço de desenvolvimento de *Software*.

Miranda (2002) também afirma que a métrica pontos por função apresenta algumas falhas no desenvolvimento orientado a objetos, pois existem atributos deste tipo de projeto que invalidam alguns PF. Belgamo (2004), afirma que a métrica pontos por função é baseada no paradigma procedimental, o qual separa dados de função, deixando PF pouco adequado para novos desenvolvimentos baseados no paradigma de orientação a objeto, o qual trabalha com dados e funcionalidades de forma combinada.

2.7 PONTOS POR CASO DE USO

Gustav Karner (1993) sugere o uso dos pontos de caso de uso, pois esses fornecem uma estimativa de *Software* logo na fase inicial, tendo como entrada os casos de uso. Ribu (2001) destaca que pontos de caso de uso são fáceis utilizar, pois seu processo de contagem é simples, suas estimativas podem ser rapidamente calculadas com uma planilha eletrônica, produz estimativas na fase de levantamento de requisitos e no processo de desenvolvimento de *Software*.

Monteiro (2005) afirma que as técnicas PCU fornecem uma estimativa de *Software* logo na fase inicial que possibilitam um acompanhamento mais detalhado de estimativas nas principais etapas do ciclo de vida do *Software* orientado a

objetos, mas que, no entanto, muitas empresas têm dificuldades de se obter resultados plenamente satisfatórios ao se utilizar a PCU.

Aguiar (2009) afirma que os pontos por caso de uso são estudados por vários pesquisadores no meio acadêmico e na indústria. Segundo Aguiar (2009), em um estudo realizado em 2001, por Bente Anda da Universidade de Oslo apresentou resultados da aplicação dos PCU na estimativa de esforço para alguns projetos e constatou-se que existem variações nos estilos dos casos de uso que pode impactar na quantidade dos PCU.

Isso acontece, pois na descrição dos casos de uso não há método formal para descrevê-los, o que torna esta tarefa um pouco difícil, principalmente para analistas pouco experientes (VIEIRA, 2007).

Por tal razão, Aguiar (2003) sugere a padronização nos estilos de casos de uso, para obter estimativas confiáveis de esforço, pois sem padrões universais para a construção de casos de uso, ficará difícil comparar projetos de diferentes organizações.

Entretanto, segundo Aguiar (2009) pesquisa feita em novembro de 2009 pela IBM Site *DeveloperWorks* relatou um forte crescimento entre 2002 a 2009, de empresas que utilizam PCU.

2.8 PASSOS PARA UTILIZAR A TÉCNICA PCU

Segundo Andrade e Oliveira (2005) o processo de contagem da métrica PCU consiste em seis passos. Nesse processo são contados os atores e os casos de uso e com base nessa contagem são calculados os PCU não ajustados. São determinadas as complexidades dos fatores técnicos, semelhante às características gerais da APF e ambiental que verifica as características da equipe e do ambiente em que será desenvolvido o *Software* e é aplicada uma fórmula para calcular os PCU ajustados.

Segundo Monteiro (2005), os seis passos do processo de contagem de PCU se resume da seguinte forma: Primeiro é contar os atores e atribuir o grau de complexidade, segundo é contar os casos de uso e atribuir o grau de complexidade,

terceiro passo é somar o total de atores com o total de casos de uso para obter o PCU não ajustado, o quarto é determinar a complexidade do fator técnico. Por fim determinar a complexidade do fator ambiental e calcular o PCU ajustado.

Segundo Muller (2010), para calcular o sistema e classificar os atores envolvidos em cada caso de uso, de forma a obter um somatório de pontos não ajustado é importante saber os pontos de atores (PTA) do sistema (*Unadjusted Actor Weight,* ou UAW) que será calculado pela soma dos produtos do número de atores de cada tipo pelo respectivo peso, como demonstrado na tabela 8.

Tipo de Ator	Peso	Descrição
Ator Simples	1	Outro sistema acessado através de uma API de programação
Ator Médio	2	Outro sistema interagindo através de um protocolo de comunicação, como TCP/IP ou FTP.
Ator Complexo	3	Um usuário interagindo através de uma interface gráfica (*Stand-alone* ou *Web*)

Tabela 8 – Classificar os atores envolvidos em cada caso de uso.

Fonte: Muller (2010).

O segundo passo será calcular os pontos dos casos de uso (PCU) (*Unadjusted Use Case Weight*, ou UUCW), o peso por transações e o peso por entidades. Para isso deverá ser calculado pelo UUCW, pelo peso ou pelas entidades sob os totais de caso de uso (PTCU) (MACHADO, 2008; MULLER, 2010), como demonstrado na figura1:

UUCP = UAW + UUCW e PTCU= PTA + PCU.

Figura 1 – Calculo de peso por transações e entidades.

Fonte: Machado (2008) e Muller (2010).

Segundo Muller (2010), para avaliar casos de uso é importante saber determinar a quantidade de Casos de Uso, verificar o número de transações de cada

um, inclusão/alteração/exclusão e incluir cenários alternativos, a saber, a classificação de cada caso de uso, da seguinte forma demonstrada na tabela 9:

Complexidade	**Transações**
Simples	Até 3 transações
Médio	De 4 a 7 transações
Complexos	Com 8 ou mais transações

Tabela 9 – Complexidades e Transações.

Fonte: Muller (2010).

Poderá ser avaliado também através do método alternativo que calcula a quantidade de classes para execução de casos de uso, (MULLER, 2010) como demonstrado na tabela 10:

Complexidade	**Classes**
Simples	Até de 4 classes de análise
Médio	De 5 a 10 classes de análise
Complexo	Mais de 10 classes de análise

Tabela 10 – Complexidades e Classes.

Fonte: Muller (2010).

Segundo Karner (1993), o cálculo dos pontos de casos de uso não ajustados será calculados pela somatória dos atores (UAW), onde será somado ao valor dos pesos de caso de uso não ajustados (UUCW), conforme a figura 2:

$$\mathbf{UUCP = \Sigma\ UAW + \Sigma\ UUCW.}$$

Figura 2 – Calculo de pontos por caso de uso não ajustados.

Fonte: Karner (1993).

Segundo Muller (2010), o próximo passo será calcular os pontos de fatores Técnicos (PFT). Para calcular os pontos de fatores técnicos o responsável por calcular deverá utilizar a tabela abaixo que destaca os 13 fatores de complexidade técnica (FCT). Cada FCT varia de 0 a 5, sendo que o valor 0 indica nenhuma influência, 3 indica influência moderada e 5 indica forte influência, conforme mostrado na tabela 11.

Fator	Requisito	Peso
T1	Sistema distribuído	2
T2	Tempo de Resposta	1
T3	Eficiência	1
T4	Processamento complexo	1
T5	Código reusável	1
T6	Facilidade de instalação	0.5
T7	Facilidade de uso	0.5
T8	Portabilidade	2
T9	Facilidade de mudança	1
T10	Concorrência	1
T11	Recursos de segurança	1
T12	Acessível por terceiros	1
T13	Requer treinamento especial	1

Tabela 11 – Fatores Técnicos.

Fonte: Muller (2010).

Segundo Muller (2010), considerando as seguintes formula: T1=1,T2=1,T3=1, T4=0,T5=3,T6=1,T7=1,T8=0, T9=1,T10=0,T11=0,T12=0,T13=0. Será obtido o seguinte resultado, como demonstrado na figura 3:

SFT=1*2+1*1+1*1+0*1+3*1+1*0.5+1*0.5+0*2+1*1+0*1+0*1+ 0*1+0*1=9

Figura 3 – Exemplo- Resultado do calculo de Fatores Técnicos.

Fonte: Muller (2010).

Para calcular os fatores ambientais, será preciso identificar os valores de um item que variam entre 0 e 5. Esses fatores medem a eficiência da equipe, pois o nível de influência indica o nível de disponibilidade de cada recurso no decorrer do projeto. Por tal razão, determinar um fator em nível de influência alta, será preciso atribuir a ele o valor 5 (MACHADO, 2008; MULLER, 2010). Os fatores ambientais previstos pela metodologia de PCU e seus pesos associados podem ser encontrados na tabela 12:

Fator	Requisito	Peso
E1	Familiaridade com RUP ou outro processo formal	1.5
E2	Experiência com a Aplicação em desenvolvimento	0.5
E3	Experiência em Orientação a Objetos	1
E4	Presença de analista experiente	0.5
E5	Motivação	1
E6	Requisitos estáveis	2
E7	Desenvolvedores em meio-expediente	-1
E8	Linguagem de programação difícil	-1

Tabela 12 – Fatores Ambientais.

Fonte: Muller (2010).

Muller (2010) afirma que um grau de influência mínimo (0) atribuído ao fator E3 indica uma equipe com total desconhecimento de orientação a objetos, enquanto que o grau máximo (5) indica a disponibilidade de uma equipe experiente neste paradigma de desenvolvimento. O fator ambiental (EF) é calculado pela seguinte fórmula: PFA = 1.4 + (-0.03 x SomaDosProdutos). Onde o valor de SomaDosProdutos = soma dos produtos entre o peso de cada fator (E1 a E8) e seu grau de influência atribuído.

Depois de calcular os fatores de ajuste que envolve fatores técnicos relacionados aos requisitos funcionais e os fatores de ambiente relacionados aos requisitos não funcionais, finalmente, podemos calcular o valor total do sistema em pontos por caso de uso ajustados, (MULLER, 2010) utilizando a fórmula apresentada no na figura 4:

UCP = PTCU x PFT x PFA.

Figura 4 – Calculo dos Fatores de Ajuste.

Fonte: Muller (2010).

2.9 VANTAGENS NA UTILIZAÇÃO DE PONTOS POR CASO DE USO

Com base em analise de modelos produzidos para esforço de estimação em sistemas de *Software* feita por diversos pesquisadores foi possível observar que a grande parte dos problemas que os desenvolvedores de *Software* e gerentes enfrentam começa nas fases iniciais de *Software,* onde o esforço de estimativa é calculado. Por tal razão, recomenda-se a utilização de casos de uso, pois os casos de uso ajudam na organização das informações coletadas e podem ser usados como uma fonte precisa para calcular as estimativas de esforço e tempo (LIVELY; SIMMONS, 2007).

Aguiar (2009) também afirma que a especificação de casos de uso é um excelente método para modelar requisitos. Por tal razão, Gustav Karner (1993) sugere o uso dos pontos de caso de uso, pois fornecem uma estimativa de *Software* logo na fase inicial, tendo como entrada os casos de uso.

Monteiro (2005) também afirma que as técnicas PCU fornecem uma estimativa de *Software* logo na fase inicial que possibilitam um acompanhamento mais detalhado de estimativas nas principais etapas do ciclo de vida do *Software* orientado a objetos.

Ribu (2001) destaca que os pontos de caso de uso são fáceis de utilizar, pois seu processo de contagem é simples, as estimativas podem ser rapidamente calculadas com uma planilha eletrônica, produzem estimativas na fase de levantamento de requisitos e em todo o processo de desenvolvimento de *Software*.

Aguiar (2009) afirma que os pontos por caso de uso são estudados por vários pesquisadores no meio acadêmico e na indústria. Uma pesquisa feita em 2009 pela IBM relatou um forte crescimento entre 2002 a 2009, de empresas que utilizam PCU (Aguiar, 2009).

A métrica PCU é uma solução simples para as estimativas de tamanho, especialmente para os que não conhecem as práticas de contagem de PF ou as consideram complexas para serem aplicadas (IFPUG, 2004).

Martins (2008) ainda afirma que os PCU usa um modelo como base para cálculo da métrica de tamanho, sendo assim fica viável a construção de ferramentas para a realização do cálculo automático, dessa forma o processo será mais barato, rápido e manutenível.

2.10 DESVANTAGENS NA UTILIZAÇÃO DE PONTOS POR CASO DE USO

Segundo Hazan (2005) os PCU somente podem ser aplicados em projetos de *Software* cuja especificação tenha sido expressa em casos de uso. Contudo, muitas empresas têm dificuldades de se obter resultados plenamente satisfatórios ao se utilizar a PCU, pois existem variações nos estilos dos casos de uso que pode impactar na quantidade dos PCU (AGUIAR, 2009).

Segundo Martins (2008), a maneira de especificação do caso de uso e o grau de detalhamento usando podem influenciar muito a contagem dos PCU bem como tornar inviável a automatização dos processos.

Hazan (2005) também afirma que os principais problemas do processo de estimativas acontecem quando se utiliza da métrica PCU, apesar dessa métrica ser apresentada para as equipes de desenvolvimento como uma métrica para estimativas de tamanho simples e eficaz.

Isso ocorre por não haver na descrição dos casos de uso um método formal para descrevê-los, o que torna esta tarefa um pouco difícil, principalmente para analistas pouco experientes (VIEIRA, 2007). Sendo assim, sem padrões universais para a construção de casos de uso, ficará difícil comparar projetos de diferentes organizações. (AGUIAR, 2003).

Martins (2007) também afirma que o problema é que um caso de uso pode ser especificado de diversas maneiras, a saber: Texto estruturado informal, texto estruturado formal, pseudocódigo, fluxograma, redes de Petri ou linguagens de programação e ainda podendo fazer uso de diversos graus de detalhamento.

Segundo Hazan (2005) o método PCU não é aderente à norma ISO/IEC 14143 que define um modelo para a medição funcional de *Software*, por tal razão a técnica do PCU não oferece certificação de profissionais, sendo assim não existe uma forma adequada para aplicá-la.

Partindo para o ponto de vista do usuário, Hazan (2005) lista os principais problemas que os desenvolvedores encontraram na utilização da métrica PCU. O primeiro problema encontrado é a falta de acurácia das estimativas de esforço, devido a ausência de base histórica. A segunda é dificuldade de estimar pequenos projetos de manutenção ou apurações especiais, onde não há modelagem de requisitos, onde depende do desenvolvimento de uma rotina para atualização de um arquivo e geração de um relatório. Outro problema identificado são as dificuldades de geração de dados históricos de produtividade, devido à subjetividade do método.

2.11 UTILIZAÇÃO DAS TÉCNICAS DE PF E PCU

Segundo Sommerville (2007), de alguma forma todos os modelos algorítmicos sofrem das mesmas dificuldades de estimativas de custo e prazo. Ele ainda afirma que é muito complicado estimar o tamanho do projeto no estágio inicial quando temos apenas uma especificação disponível.

Segundo Andrade e Oliveira (2005) a estimativa de tamanho é uma das métricas mais utilizadas na gestão de projetos de *Software*, porque a partir dessa dimensão é possível definir o esforço, o prazo e os custos necessários para o desenvolvimento do software.

Segundo Meirelles (2008) a métrica PF foi validada em alguns trabalhos ao comparar o LOC e PF como medidores de esforço de desenvolvimento e relacionando PF com a produtividade em um ambiente desenvolvimento, mas que em outros projetos, também foi possível perceber que PF por si só é insuficientemente precisa para a predição e esforço de desenvolvimento de *Software*.

Sommmerville (2007) sugere os pontos de função e pontos de objeto para calcular as estimativas por serem mais fáceis de produzir do que as estimativas de

tamanho de código, mas, contudo ele adverte que as estimativas por pontos de função e pontos de objeto são frequentemente, imprecisas.

Entretanto, Andrade (2004) propõe utilizar a PF e PCU de forma combinada para facilitar o gerente de projeto definir qual das técnicas é mais vantajosa para utilizar em determinados projetos.

Dados de uma pesquisa de campo realizada pelo diretor da empresa ti Métricas e do IFPUG Aguiar (2003) demonstram que em 2000, linhas de pesquisa como o PSM do exército norte-americano e o COCOMO II da Universidade do Sul da Califórnia passaram a considerar PF como alternativa e que no Brasil sua utilização vem crescendo num ritmo acelerado. A quantidade de profissionais certificados pelo IFPUG como especialistas em Pontos de Função (CFPS) revela um intenso crescimento entre 1999 a 2002.

Uma nova pesquisa realizada em 2008 pela IFPUG informa que, a grande maioria de CFPS estão localizados no Brasil. O Brasil teve o posto em primeiro lugar com total de 269 CFPS, a Coréia ficou em segundo lugar com o total de 264, a Índia obteve 176, a Itália 99 e os EUA em quinto lugar com 93 profissionais certificados pelo IFPUG com especialização em pontos de função (BFPUG, 2008).

Pesquisa realizada em 2002 pela empresa *Rational Website* obteve resultados baixos sobre o uso de pontos por caso uso. Foram encontradas aproximadamente 12.700 ocorrências de pontos de função, a mesma pesquisa retornou apenas 213 ocorrências de pontos de caso de uso (AGUIAR, 2009).

Segundo Aguiar (2009) essa pesquisa foi renovada em novembro de 2009 pela IBM Site DeveloperWorks onde relatou um forte crescimento entre 2002 a 2009, com aproximadamente 113.000 ocorrências de pontos de função em oposição a 11.400 de pontos por caso de uso.

2.12 ANALISE COMPARATIVA ENTRE AS TÉCNICAS PF E PCU

Segundo Andrade (2004), na aplicação de métricas para estimar tamanho, a métrica de pontos de função possui uma vantagem adicional que é a sua utilização como insumo para o COCOMO, modelo utilizado para a derivação das estimativas

de esforço, prazo e custo a partir das estimativas de tamanho. Assim, o PCU não traz nenhum benefício adicional sobre o PF. (ANDRADE, 2004)

Andrade (2005) ainda afirma que a precisão da estimativa com PF melhora à medida que se obtém mais informações da análise e do projeto do Software, já o PCU foi proposto para ser utilizado no início do ciclo de desenvolvimento, na fase de requisitos, mas que as duas métricas são adequadas em diferentes fases do processo de desenvolvimento de *Software*.

A métrica APF, conforme o manual do IFPUG versão 4.1.1, está padronizada sob a norma ISO/IEC 20926 (IFPUG, 2003). Por tal razão Souza (2012) recomenda mais o uso de PF que o uso de PCU. Segundo Andrade (2004), é importante destacar a maturidade da métrica PF na indústria de *Software* é bem maior em relação à métrica PCU.

Andrade e Oliveira (2005) também afirmam que os PCU permitem fazer estimativas no início do projeto com base no modelo de casos de uso, por ser mais completo que outros métodos de requisitos, mas que por outro lado, os PF são uma das métricas de estimativa que proporciona resultados cada vez mais precisos à medida que artefatos da fase de análise e projeto são gerados.

Contudo, a análise comparativa entre as duas técnicas feita por Machado (2008) possibilitou a verificação de que há uma diferença no número de pontos por função e no número de pontos por caso de uso, mas que apesar de haver diferenças entre as duas técnicas, isso não indica que PF seja melhor ou menos confiável que PCU, pois essa análise deve ser feita com um número maior de testes somente.

A Tabela 13 apresenta uma comparação entre as principais características de PCU e PF levantadas por Machado (2008). Pode-se perceber nesta tabela que a métrica pontos por caso de uso é nova no mercado e ainda não existe uma padronização para ela, sendo assim, não oferece treinamentos e certificação. Já a métrica pontos por função é padronizada internacionalmente desde 2002, oferece certificação e treinamento.

Análise de Pontos por Função (APF)	**Pontos de Caso de Uso (UCP)**
Método criado em 1979 e reconhecido como padrão internacional desde 2002 por meio da norma ISO/IEC 20926.	Método criado em 1993 e que não possui padronização do método.
E amparada por diversos grupos, entre eles o IFPUG *(International Function Point Users Group).*	Não ha um grupo ou organização responsável pela métrica.
Possui regras definidas e padronizadas para o processo de contagem de pontos de função.	Não ha padrões para a descrição dos casos de uso, ha duvidas na contagem de casos de uso incluídos e estendidos.
Aplicada na fase de analise do sistema.	Aplicada na fase de projeto, na especificação de requisitos.
Oferece treinamento e certificação.	Ainda não oferece certificação.
Muito material disponível na literatura.	Pouco material disponível.

Tabela 13 – Comparativo entre PF e PCU.

Fonte: Machado (2008).

3 METODOLOGIA

Para produzir a ferramenta de análise de cenário que auxiliará na escolha entre a melhor técnica, através de uma analise comparativa entre pontos por função e pontos por caso de uso, foi preciso coletar muitas informações para obter corretamente a comparação entre esses dois pontos.

Sendo assim, o primeiro passo foi criar uma tabela para analise das práticas com a finalidade de obter o maior número de resultados de comparação entre as técnicas de estimativa de tamanho de *Software* relacionadas aos pontos por função e pontos por caso de uso de forma combinada.

Para a realização desta comparação de ferramentas, foi feito um estudo bibliográfico para então obter um entendimento teórico sobre as definições de parâmetros das técnicas.

O segundo passo foi criar consultas na base de dados para buscas e comparações entre pontos de função e pontos por caso de uso, onde visou-se conhecer, analisar e interpretar as informações dos projetos de *Software* para encontrar a relação entre a aplicação das duas métricas. Sendo assim, a última etapa foi a criação do código fonte, desenvolvido na linguagem C#.

4 RESULTADOS

4.1 COMPARAÇÃO ENTRE AS TÉCNICAS PF E PCU

A Tabela 14 apresenta uma analise das práticas relacionadas aos PF e PCU levantadas na base em pesquisas por diversos pesquisadores que compararam as duas técnicas de medição de tamanho de *Software*. Pode-se perceber nesta tabela que a métrica PF é padronizada internacionalmente e oferece treinamento e certificação. Sendo assim, é a mais utilizada. A métrica PCU é nova no mercado e ainda não existe uma padronização para ela e por isso ainda não oferece treinamentos e certificação. Contudo, não se pode afirmar que uma é melhor que a outra, apenas pode-se dizer que para cada métrica precisa de uma análise diferente, mais detalhada, ou mais simples, mas com um número maior de testes para então aplicá-la.

Pontos de Função	Pontos por Caso de Uso
Baseada no paradigma procedimental, o qual separa dados de função (MARTINS, 2007).	Baseada no modelo de casos de uso (MARTINS, 2008).
Possui padronização e oferece treinamentos e certificação CFPS promovido pelo IFPUG (ANDRADE, 2009).	Não oferece treinamentos e certificação de profissionais e por isso não existe uma forma adequada para aplicá-la (MACHADO, 2008).
Predomina a indústria nacional e internacional desenvolvedoras de Software desde 1970 (AGUIAR, 2009).	Métrica relativamente nova e não muito utilizada (ANDRADE, 2009).
Garante a utilização correta do método Análise de Pontos de Função por um especialista certificado (BFPUG, 2008).	Simples para as estimativas de tamanho, especialmente para os que não conhecem as práticas de Contagem de PF (IFPUG, 2004).
Métrica totalmente subjetiva (MARTINS, 2007).	Métrica totalmente subjetiva (MARTINS, 2007).
Mais utilizado no final das fases de análise e projeto (HAZAN, 2005).	Mais utilizado na fase inicial do projeto (ANDRADE; OLIVEIRA, 2005).

Dependente do ponto de vista de cada profissional (MARTINS, 2007).	Dependente do ponto de vista de cada profissional (BELGAMO, 2004).
Está padronizada sob a norma ISO/IEC 20926 (BFPUG, 2008).	Não é aderente à norma ISO/IEC 14143 que define um modelo para a medição funcional de software (HAZAN, 2005).
Indicada como melhor forma de estimar, revisar e recalcular o tamanho do projeto à medida que novos artefatos estejam disponíveis durante o processo de desenvolvimento (HAUFE, 2001).	Proposto para ser utilizado no início do ciclo de desenvolvimento, na fase de requisitos (KARNER, 1993).
Pouco adequada para trabalhar com dados e funcionalidades de forma combinada (BELGAMO, 2004).	Adequada para trabalhar com dados e funcionalidades de forma combinada (BELGAMO, 2004).
Mede tamanho funcional, independentemente da tecnologia utilizada. Capaz de aplicar em sistemas cujos requisitos tenham sido modelados utilizando casos de uso (MACHADO, 2008).	Podem ser computados logo no início do desenvolvimento desde que adote um padrão de especificação de Caso de Uso (MARTINS, 2007)
Não é apoiada por nenhuma ferramenta que faça a contagem automaticamente (MARTINS, 2007).	Viável a construção de ferramentas para a realização do cálculo automático dessa métrica, dessa forma o processo fica mais barato, rápido e manutenível (MARTINS, 2008).
Uma excelente métrica para aferição de tamanho funcional dos requisitos elicitados (AGUIAR, 2009).	Mais completo que outros métodos de requisitos (ANDRADE; OLIVEIRA, 2005).
Tem aumentado o uso e a publicação de estudos na literatura (SOUZA, 2012).	Largamente discutida na literatura (ANDRADE; OLIVEIRA, 2005).
Revista com base na complexidade das funções e ao término do desenvolvimento do Software (MACHADO, 2008).	Usa um modelo como base para cálculo da métrica de tamanho (MARTINS, 2008).
Necessário conhecer a produtividade nas linguagens dominantes (Souza, 2009).	Necessário conhecer a produtividade nas linguagens dominantes (ANDRADE; OLIVEIRA, 2005).
Grande utilidade na garantia da completa especificação de requisitos (HAZAN, 2005).	Excelente método para modelar requisitos (AGUIAR, 2009).
Transparente e foca na visão do usuário (SOUZA, 2010).	Necessário conhecer e especificar Caso de Uso (HAZAN, 2005).

Tabela 14 – Analise comparativa entre as técnicas PF e PCU.

4.2 TABELA DE DECISÃO ENTRE PF E PCU

As características imputadas receberam valores atribuídos em variáveis e em cada IF será contado o valor recebido em favor da métrica PF e/ou em favor da métrica PCU. Depois da coleta de todas as características que será informada pelo gerente/responsável por medir o *Software* será feita a soma das características separada por PF e PCU. Por fim, a métrica que obtiver o maior número de resultado a favor será a métrica mais indicada no projeto apresentado. Como mostrada na tabela 15, a primeira coluna contém as possíveis características avaliadas no projeto e a segunda coluna representa a métrica mais recomendada.

Características do Projeto	**Recomendação**
Profissional certificado pela IFPUG	PF
Profissional não certificado pela IFPUG	PF/PCU
Projeto Simples/ Pequeno porte	PF/PCU
Projeto de médio porte	PF/PCU
Projeto de grande porte	PCU
Projeto complexo	PCU
Calculo Automático	PCU
Calculo Manual	PF
Requisitos modelados em caso de uso	PF/PCU
Requisitos descritos de forma textual	PF
Projeto em fase Inicial	PCU
Projeto em fase intermediaria	PF
Projeto em fase final	PF
Facilidade de Implantar, operar e Mudar.	PF/PCU

Reusabilidade	PF
Processamento Complexo	PF/PCU
Configuração de Equipamento	PF
Presença de analista experiente	PCU
Desenvolvedores em meio-expediente	PCU
Equipe motivada	PF/PCU
Familiaridade com RUP ou outro processo formal	PCU
Experiência com a aplicação em desenvolvimento	PCU
Experiência em Orientação a Objetos	PCU
Linguagem de programação difícil	PCU
Linguagem de programação simples	PF

Tabela 15 – Tabela de decisão entre as técnicas PF e PCU.

4.3 MODELAGEM DA FERRAMENTA DE APOIO

A ferramenta trabalhará da seguinte maneira, o usuário terá que imputar as informações sobre o projeto e através de um botão inicialmente o usuário deverá clicar na opção análise do projeto. A partir dai será inicializado uma comparação entre as técnicas de determinação de custo e esforço.

Essa comparação será feita automaticamente a partir da solicitação de comparação do usuário que foi feita através do botão. A busca na base de dados será através de consultas onde analisarão as características essenciais do projeto, sendo assim logo após alguns segundos a ferramenta apontará qual tipo de métrica é mais indicada para o tipo de projeto apresentado.

A Figura 5 apresenta o diagrama de casos de uso utilizado como base para construção da ferramenta. É possível notar que primeiramente o gerente de projetos deverá inserir as características do projeto para então processar os dados e obter a métrica mais indicada para o projeto apresentado. Com isso, é possível notar

também que o gerente de projeto deverá estar certo das características informadas, para que então, os dados imputados sejam avaliados da melhor forma possível.

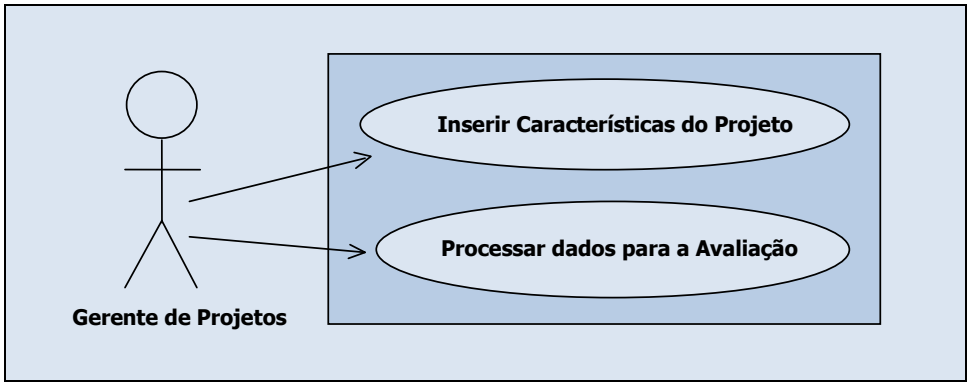

Figura 5 – Casos de uso da ferramenta de definição entre APF e PCU.

A Figura 6 apresenta o diagrama de classes gerado para construção da ferramenta. As informações existentes nas classes foram obtidas através da comparação entra as técnicas, mencionadas no capítulo 4.1 e 4.2.

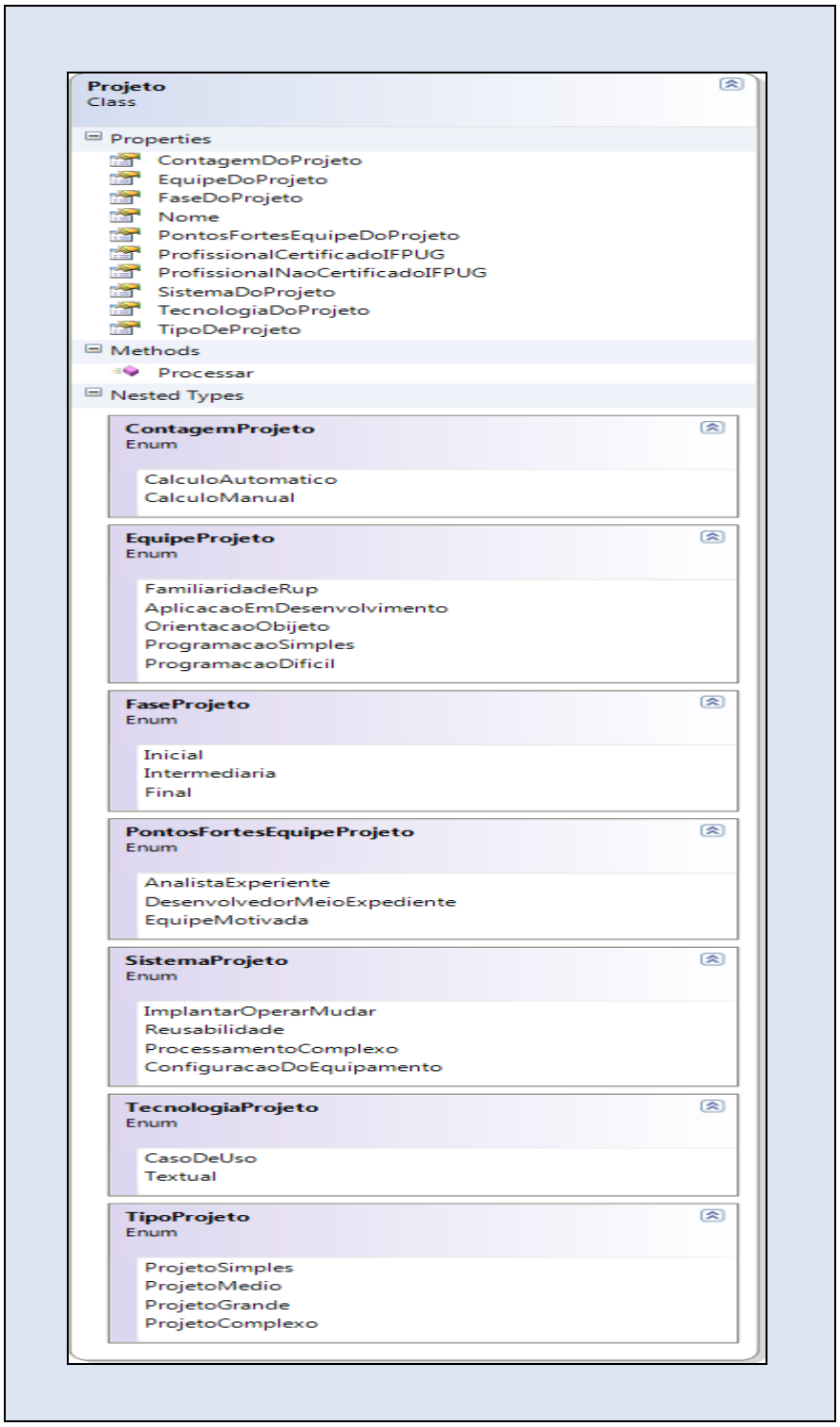

Figura 6 – Diagrama de Classe da ferramenta de definição entre APF e PCU.

A Figura 7 apresenta o fluxograma com a lógica de programação aplicada para tomar a decisão entre PF ou PCU. Este fluxo também foi obtido através da comparação entre as técnicas, mencionadas no capítulo 4.1.

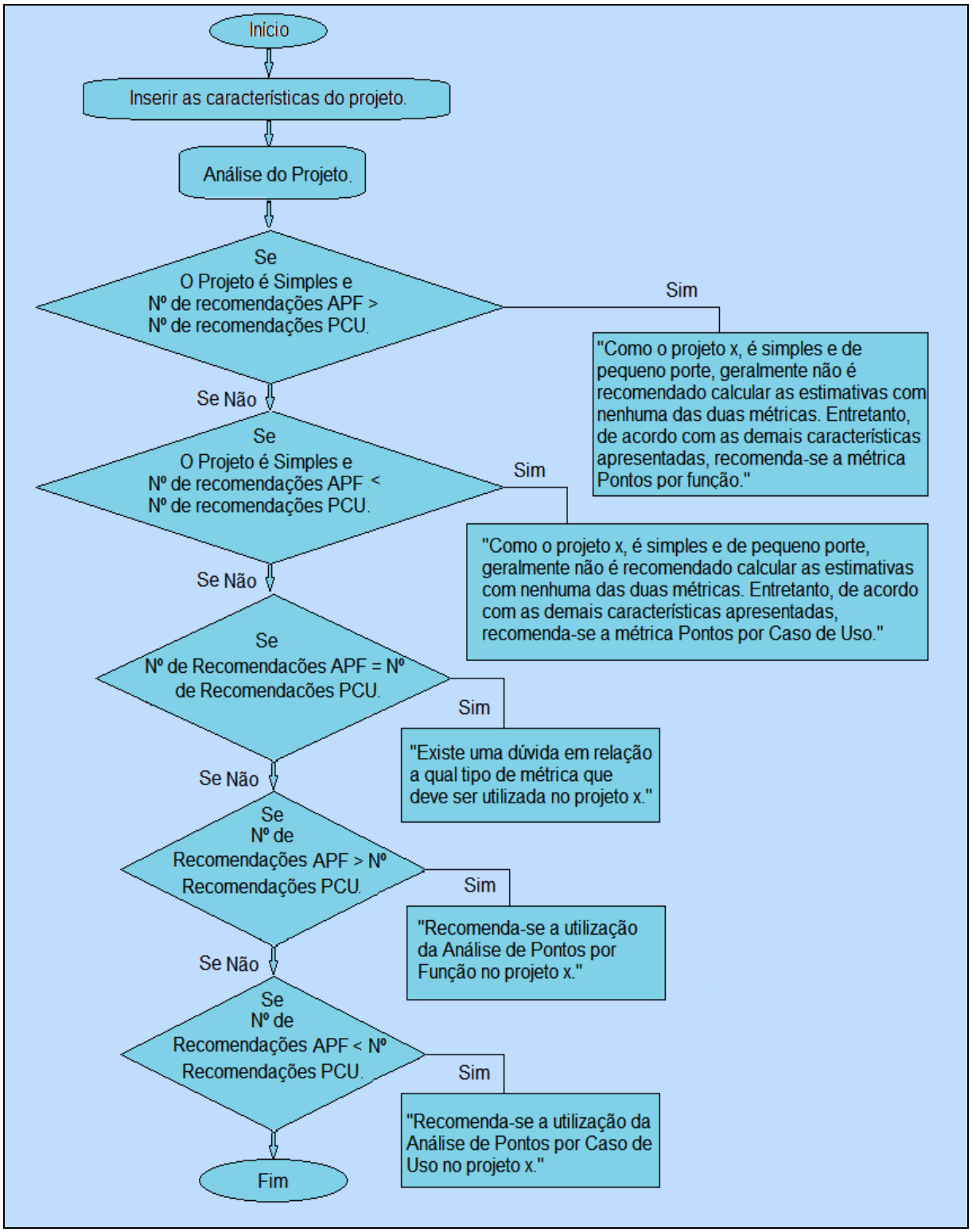

Figura 7 – Fluxograma da ferramenta de definição entre APF e PCU

4.4 CÓDIGO FONTE

A figura 8 demonstra a lógica desenvolvida para a análise do projeto na linguagem C#.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace ProjetoTeste
{
   class Projeto
   {

      public enum TipoProjeto
      {
         ProjetoSimples,
         ProjetoMedio,
         ProjetoGrande,
         ProjetoComplexo
      }

      public enum FaseProjeto
      {
         Inicial,
         Intermediaria,
         Final,
      }

      public enum TecnologiaProjeto
      {
         CasoDeUso,
         Textual,
      }

      public enum SistemaProjeto
      {
         ImplantarOperarMudar,
         Reusabilidade,
         ProcessamentoComplexo,
         ConfiguracaoDoEquipamento,
      }

      public enum PontosFortesEquipeProjeto
      {
         AnalistaExperiente,
         DesenvolvedorMeioExpediente,
         EquipeMotivada,
      }

      public enum ContagemProjeto
      {
```

```
        CalculoAutomatico,
        CalculoManual,
    }

    public enum EquipeProjeto
    {
        FamiliaridadeRup,
        AplicacaoEmDesenvolvimento,
        OrientacaoObijeto,
        ProgramacaoSimples,
        ProgramacaoDificil,
    }


    public string Nome { get; set; }

    public bool ProfissionalCertificadoIFPUG { get; set; }

    public bool ProfissionalNaoCertificadoIFPUG { get; set; }

    public TipoProjeto TipoDeProjeto { get; set; }

    public FaseProjeto FaseDoProjeto { get; set; }

    public TecnologiaProjeto TecnologiaDoProjeto { get; set; }

    public SistemaProjeto SistemaDoProjeto { get; set; }

    public PontosFortesEquipeProjeto PontosFortesEquipeDoProjeto { get; set; }

    public ContagemProjeto ContagemDoProjeto { get; set; }

    public EquipeProjeto EquipeDoProjeto { get; set; }


    public void Processar()
    {
        int numeroRecomendacoesAPF = 1;
        int numeroRecomendacoesPCU = 1;

        // lógica com base no fluxograma
        if (ProfissionalCertificadoIFPUG)
            numeroRecomendacoesAPF++;
        if (ProfissionalNaoCertificadoIFPUG)
            numeroRecomendacoesPCU++;

        // Tipo do Projeto
        switch (TipoDeProjeto)
        {
            case Projeto.TipoProjeto.ProjetoSimples:
                numeroRecomendacoesAPF++;
                numeroRecomendacoesPCU++;
                break;
            case Projeto.TipoProjeto.ProjetoMedio:
                numeroRecomendacoesAPF++;
                numeroRecomendacoesPCU++;
                break;
            case Projeto.TipoProjeto.ProjetoGrande:
                numeroRecomendacoesPCU++;
                break;
```

```
            case Projeto.TipoProjeto.ProjetoComplexo:
                numeroRecomendacoesPCU++;
                break;
        }

        //Fase do Projeto
        switch (FaseDoProjeto)
        {
            case Projeto.FaseProjeto.Inicial:
                numeroRecomendacoesPCU++;
                break;
            case Projeto.FaseProjeto.Intermediaria:
                numeroRecomendacoesAPF++;
                break;
            case Projeto.FaseProjeto.Final:
                numeroRecomendacoesAPF++;
                break;
        }

        //Tecnologia do Projeto
        switch (TecnologiaDoProjeto)
        {
            case Projeto.TecnologiaProjeto.CasoDeUso:
                numeroRecomendacoesPCU++;
                numeroRecomendacoesAPF++;
                break;
            case Projeto.TecnologiaProjeto.Textual:
                numeroRecomendacoesAPF++;
                break;
        }

        //Sistema
        switch (SistemaDoProjeto)
        {
            case Projeto.SistemaProjeto.ImplantarOperarMudar:
                numeroRecomendacoesPCU++;
                numeroRecomendacoesAPF++;
                break;
            case Projeto.SistemaProjeto.Reusabilidade:
                numeroRecomendacoesAPF++;
                break;
            case Projeto.SistemaProjeto.ProcessamentoComplexo:
                numeroRecomendacoesAPF++;
                numeroRecomendacoesPCU++;
                break;
            case Projeto.SistemaProjeto.ConfiguracaoDoEquipamento:
                numeroRecomendacoesAPF++;
                break;
        }

        //Pontos forte da Equipe
        switch (PontosFortesEquipeDoProjeto)
        {
            case Projeto.PontosFortesEquipeProjeto.AnalistaExperiente:
                numeroRecomendacoesPCU++;
                break;
            case Projeto.PontosFortesEquipeProjeto.DesenvolvedorMeioExpediente:
                numeroRecomendacoesPCU++;
                break;
            case Projeto.PontosFortesEquipeProjeto.EquipeMotivada:
```

```
                numeroRecomendacoesAPF++;
                numeroRecomendacoesPCU++;
                break;
            }

            //Contagem
            switch (ContagemDoProjeto)
            {
                case Projeto.ContagemProjeto.CalculoAutomatico:
                    numeroRecomendacoesPCU++;
                    break;
                case Projeto.ContagemProjeto.CalculoManual:
                    numeroRecomendacoesAPF++;
                    break;
            }

            //Nível de experiencia da Equipe
            switch (EquipeDoProjeto)
            {
                case Projeto.EquipeProjeto.FamiliaridadeRup:
                    numeroRecomendacoesPCU++;
                    break;
                case Projeto.EquipeProjeto.AplicacaoEmDesenvolvimento:
                    numeroRecomendacoesPCU++;
                    break;
                case Projeto.EquipeProjeto.OrientacaoObijeto:
                    numeroRecomendacoesPCU++;
                    break;
                case Projeto.EquipeProjeto.ProgramacaoSimples:
                    numeroRecomendacoesAPF++;
                    break;
                case Projeto.EquipeProjeto.ProgramacaoDificil:
                    numeroRecomendacoesPCU++;
                    break;
            }

            // Aqui o retorno para o usuário
            if ((TipoDeProjeto == TipoProjeto.ProjetoSimples) && (numeroRecomendacoesAPF > numeroRecomendacoesPCU))
                MessageBox.Show(String.Format("Como o projeto {0}, é simples e de pequeno porte, geralmente não é recomendado calcular as estimativas com nenhuma das duas métricas. Entretanto, de acordo com as demais características apresentadas, recomenda-se a métrica Pontos por função", Nome));
            else
            {
                if ((TipoDeProjeto == TipoProjeto.ProjetoSimples) && (numeroRecomendacoesAPF < numeroRecomendacoesPCU))
                    MessageBox.Show(String.Format("Como o projeto {0}, é simples e de pequeno porte, geralmente não é recomendado calcular as estimativas com nenhuma das duas métricas. Entretanto, de acordo com as demais características apresentadas, recomenda-se a métrica Pontos por Caso de Uso", Nome));
                else
                {
                    if (numeroRecomendacoesAPF == numeroRecomendacoesPCU)
                        MessageBox.Show(String.Format("Existe uma dúvida em relação a qual tipo de métrica que deve ser utilizada no projeto {0}.", Nome));
                    else
                    {
                        if (numeroRecomendacoesAPF > numeroRecomendacoesPCU)
                            MessageBox.Show(String.Format("Recomenda-se a utilização da Análise de
```

```
Pontos por Função no projeto {0}.", Nome));
                else
                {
                    if (numeroRecomendacoesAPF < numeroRecomendacoesPCU)
                        MessageBox.Show(String.Format("Recomenda-se a utilização da Análise de
Pontos por Caso de Uso no projeto {0}.", Nome));
                }
            }
        }
    }
  }
}
```

Figura 8 – Código fonte desenvolvido para a análise do projeto.

A figura 9 demonstra o código fonte do Formulário.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace ProjetoTeste
{
   public partial class frPrincipal : Form
   {
      public frPrincipal()
      {
         InitializeComponent();
      }

      private void btnAvaliar_Click(object sender, EventArgs e)
      {
         Projeto projeto = new Projeto();
         //Profissional
         projeto.ProfissionalCertificadoIFPUG = rbProfissionalIFPUG.Checked;
         projeto.ProfissionalNaoCertificadoIFPUG = rbProfissionalSemIFPUG.Checked;
         //Nome do Projeto
         projeto.Nome = txtNomeProjeto.Text;

         //Tipo do Projeto
         if (rbProjetoSimples.Checked)
            projeto.TipoDeProjeto = Projeto.TipoProjeto.ProjetoSimples;
         if (rbProjetoMedio.Checked)
            projeto.TipoDeProjeto = Projeto.TipoProjeto.ProjetoMedio;
         if (rbProjetoGrande.Checked)
            projeto.TipoDeProjeto = Projeto.TipoProjeto.ProjetoGrande;
         if (rbProjetoComplexo.Checked)
            projeto.TipoDeProjeto = Projeto.TipoProjeto.ProjetoComplexo;

         //Fase do Projeto
         if (rbInicio.Checked)
```

```
            projeto.FaseDoProjeto = Projeto.FaseProjeto.Inicial;

        if (rbInter.Checked)
            projeto.FaseDoProjeto = Projeto.FaseProjeto.Intermediaria;

        if (rbFinal.Checked)
            projeto.FaseDoProjeto = Projeto.FaseProjeto.Final;


        //Tecnologia utilizada
        if (rbCasoDeUso.Checked)
            projeto.TecnologiaDoProjeto = Projeto.TecnologiaProjeto.CasoDeUso;

        if (rbTextual.Checked)
            projeto.TecnologiaDoProjeto = Projeto.TecnologiaProjeto.Textual;


        //Sistema
        if (cbImplatarOperarMudar.Checked)
            projeto.SistemaDoProjeto = Projeto.SistemaProjeto.ImplantarOperarMudar;

        if (cbReusabilidade.Checked)
            projeto.SistemaDoProjeto = Projeto.SistemaProjeto.Reusabilidade;

        if (cbProcessamentoComplexo.Checked)
            projeto.SistemaDoProjeto = Projeto.SistemaProjeto.ProcessamentoComplexo;

        if (cbConfiguracaoDeEquipamento.Checked)
            projeto.SistemaDoProjeto = Projeto.SistemaProjeto.ConfiguracaoDoEquipamento;


        //Pontos forte da Equipe
        if (cbPresençaDeAnalistaExperiente.Checked)
            projeto.PontosFortesEquipeDoProjeto
=Projeto.PontosFortesEquipeProjeto.AnalistaExperiente;

        if (cbDesenvolvedoresMeioExpediente.Checked)
            projeto.PontosFortesEquipeDoProjeto
=Projeto.PontosFortesEquipeProjeto.DesenvolvedorMeioExpediente;

        if (cbEquipeMotivada.Checked)
            projeto.PontosFortesEquipeDoProjeto
=Projeto.PontosFortesEquipeProjeto.EquipeMotivada;


        //Contagem
        if (rbSim.Checked)
            projeto.ContagemDoProjeto = Projeto.ContagemProjeto.CalculoAutomatico;

        if (rbNao.Checked)
            projeto.ContagemDoProjeto = Projeto.ContagemProjeto.CalculoManual;


        //Nível de experiencia da Equipe
        if (cbFamiliaridadeRup.Checked)
            projeto.EquipeDoProjeto = Projeto.EquipeProjeto.FamiliaridadeRup;

        if (cbAplicacaoEmDesenvolvimento.Checked)
            projeto.EquipeDoProjeto = Projeto.EquipeProjeto.AplicacaoEmDesenvolvimento;
```

```
            if (cbExperienciaOrientacaoObjeto.Checked)
                projeto.EquipeDoProjeto = Projeto.EquipeProjeto.OrientacaoObijeto;

            if (cbProgramacaoSimples.Checked)
                projeto.EquipeDoProjeto = Projeto.EquipeProjeto.ProgramacaoSimples;

            if (cbProgramacaoDificil.Checked)
                projeto.EquipeDoProjeto = Projeto.EquipeProjeto.ProgramacaoDificil;

            // No final, faz o processamento
            projeto.Processar();

        }
    }
}
```

Figura 9 – Código fonte do Formulário.

4.5 INTERFACE GRÁFICA

O formulário principal contém 8 etapas para serem avaliadas, sendo elas: Profissional, Tipo/Tamanho, Fase do Projeto, Tecnologia, Sistema, Pontos fortes da equipe, Contagem e nível de experiência da equipe, como mostra a figura 10.

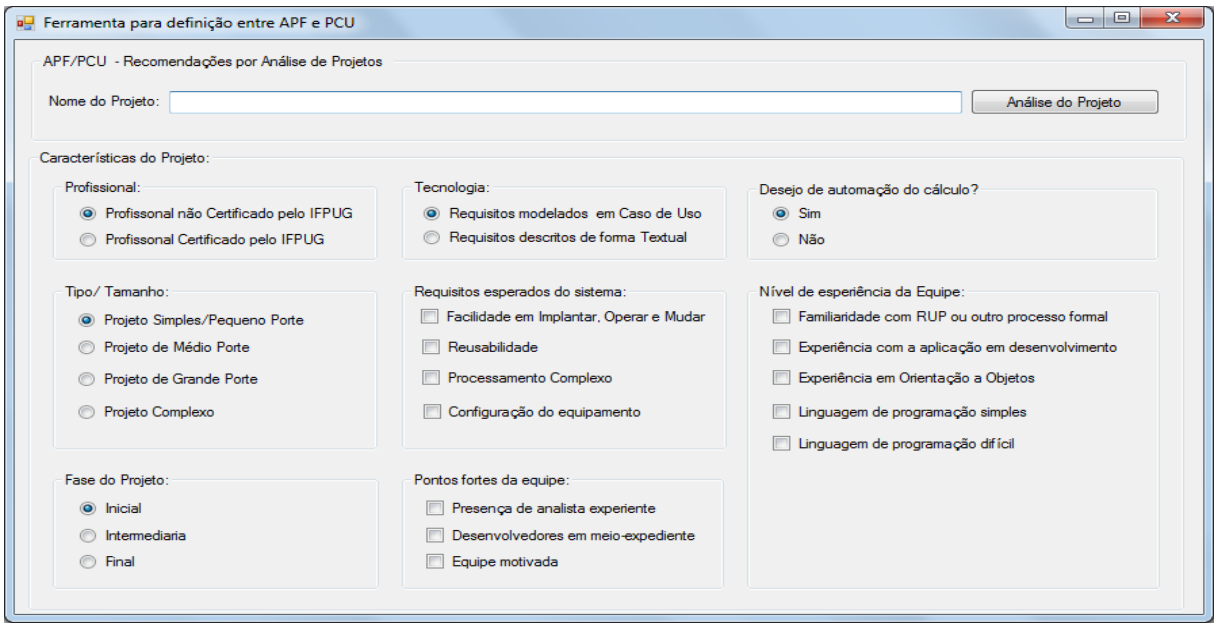

Figura 10 – Formulário principal da ferramenta de definição entre APF e PCU.

4.6 EXEMPLOS DE USO DO SOFTWARE

O primeiro passo será inserir o nome do projeto, o segundo será preencher o formulário com as características do projeto e profissional. Além das características do projeto e do profissional, o responsável deverá informar também pelo menos um ponto forte e um nível de experiência da equipe, como mostra a figura 11.

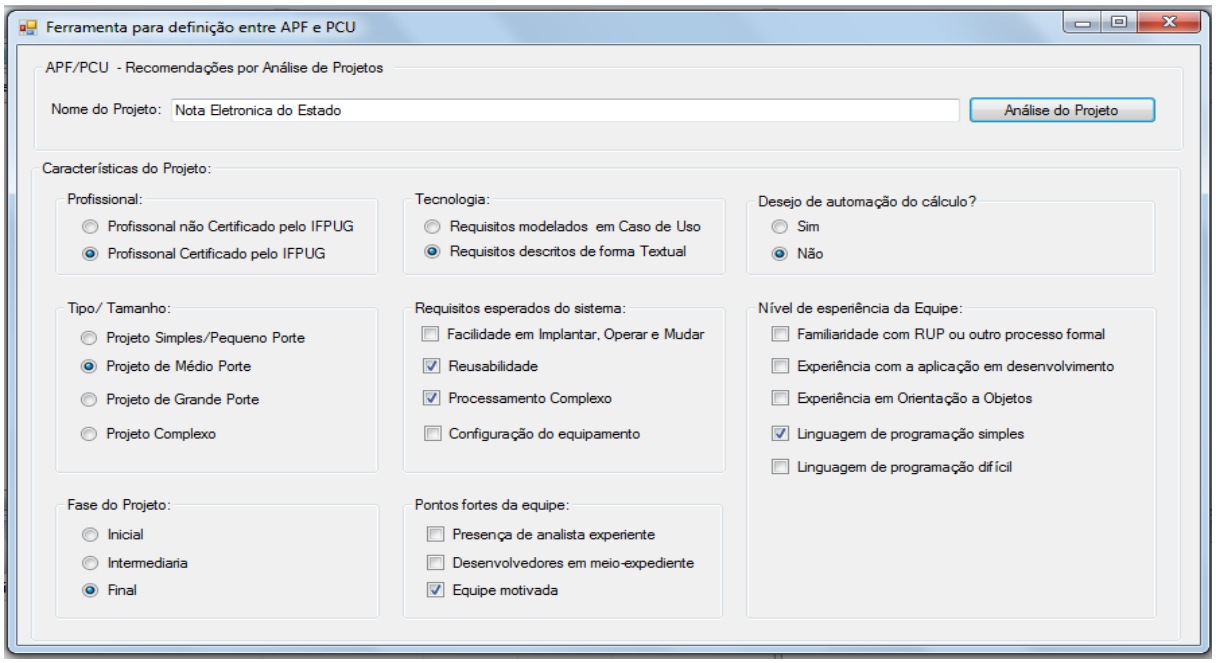

Figura 11 – Exemplo de uso da ferramenta de definição entre APF e PCU.

O terceiro passo será clicar no botão Análise de Projeto onde será feita a análise comparativa do projeto. Segundo as características apresentadas na figura 12, o projeto Nota Eletrônica do Estado deverá usar pontos por função.

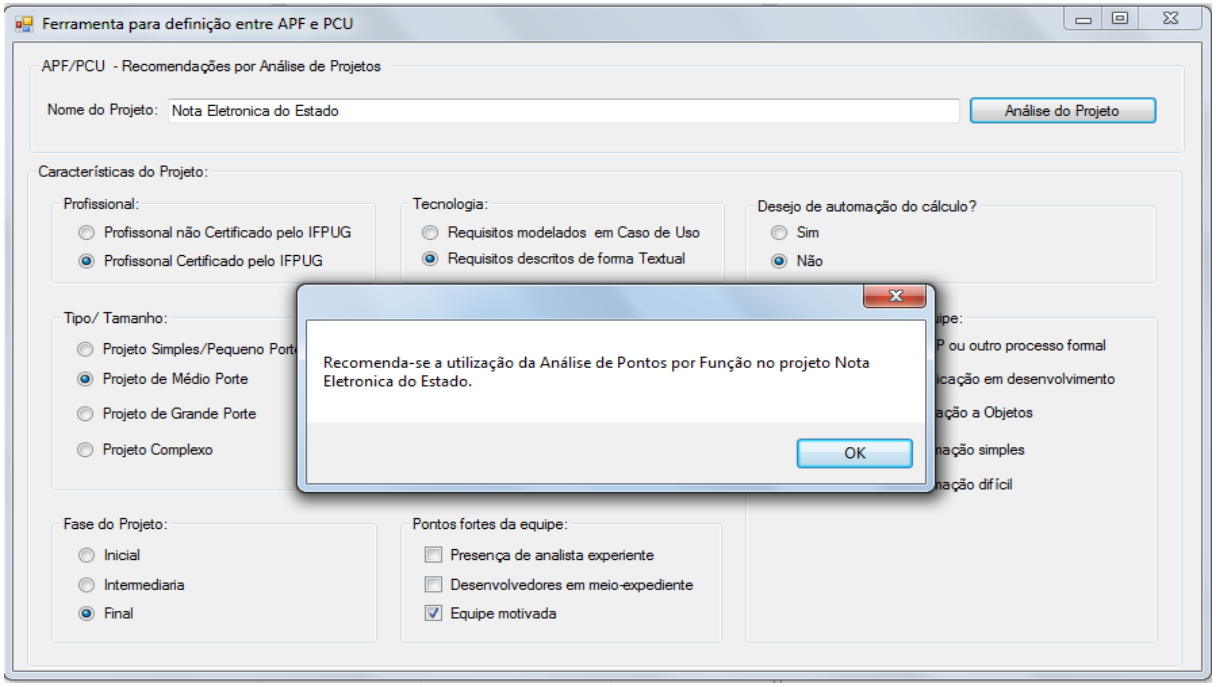

Figura 12 – Recomendação da métrica PF.

Segundo as características apresentadas na figura 13, o projeto Caixa Eletrônico deverá usar pontos por caso de uso.

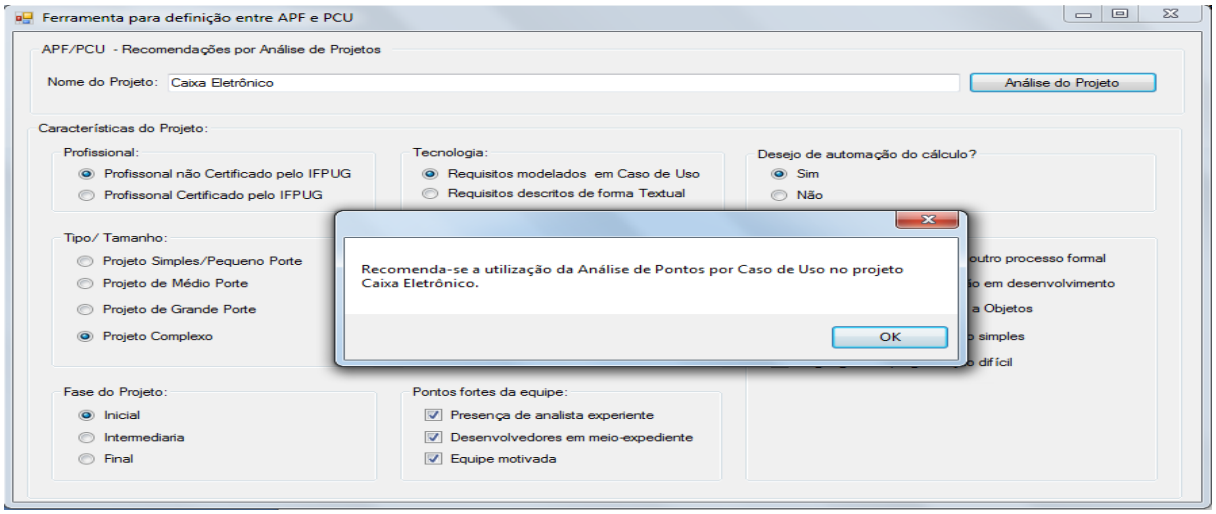

Figura 13 – Recomendação da métrica PCU.

Em alguns momentos ao imputar alguns dados, a ferramenta indicará que não poderá identificar qual tipo de métrica utilizar. Isso ocorrerá porque muitas vezes os

dados imputados estão divididos, pois alguns requisitos podem se igualar como mostrado na figura 14.

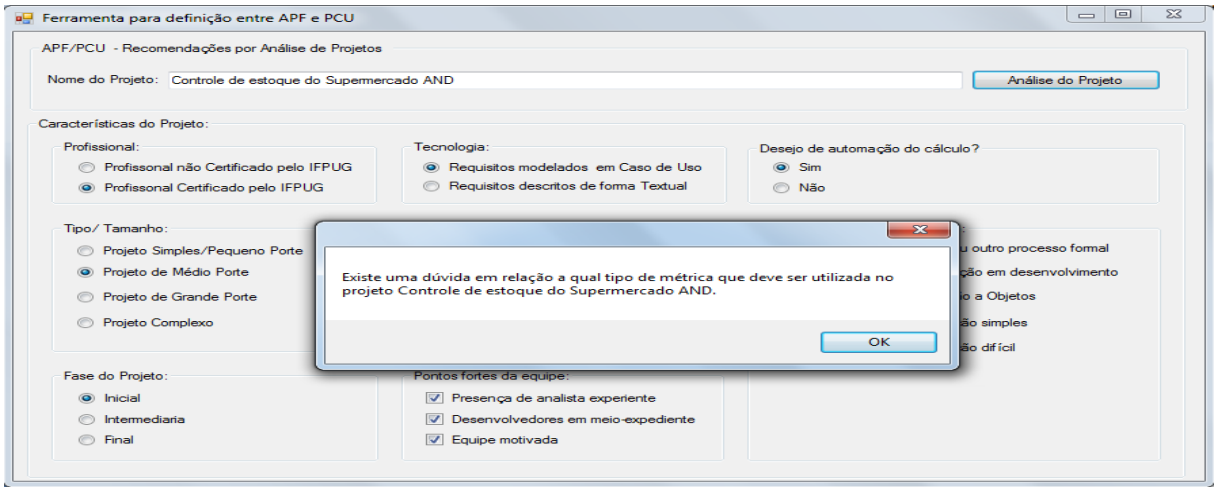

Figura 14 – Recomendação Indefinida.

Quando ocorrer a indecisão na analise comparativa o gerente de projetos poderá rever os requisitos do *Software*, pois geralmente quando isso acontece algo pode estar errado e então ao estimar o *Software* poderá ocorrer algum tipo de erro.

Ao selecionar o campo Projeto Simples/Pequeno Porte a ferramenta alertará que não existe um bom histórico em projetos simples/pequeno porte que utilizaram a PCU ou PF, como mostra a figura 15.

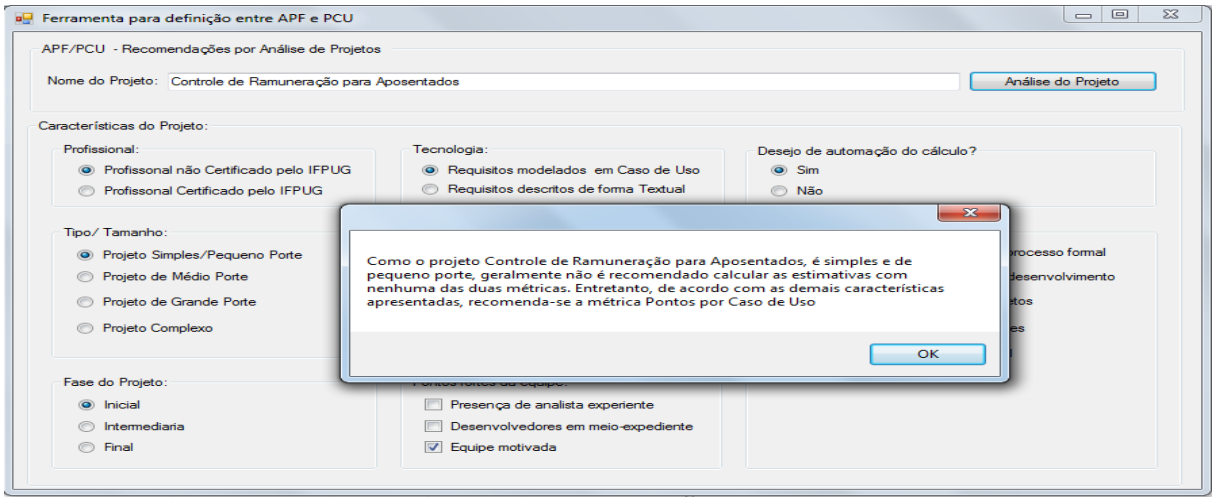

Figura 15 – Alerta e recomendação da Métrica PCU.

Analisando a figura 15 pode-se perceber que a ferramenta apresentará o tipo de métrica mais indicada, pois apesar de não ter um bom histórico serão avaliadas as demais características do projeto, a figura 15 recomendou os pontos por caso de uso e a figura 16 recomenda-se os pontos por função.

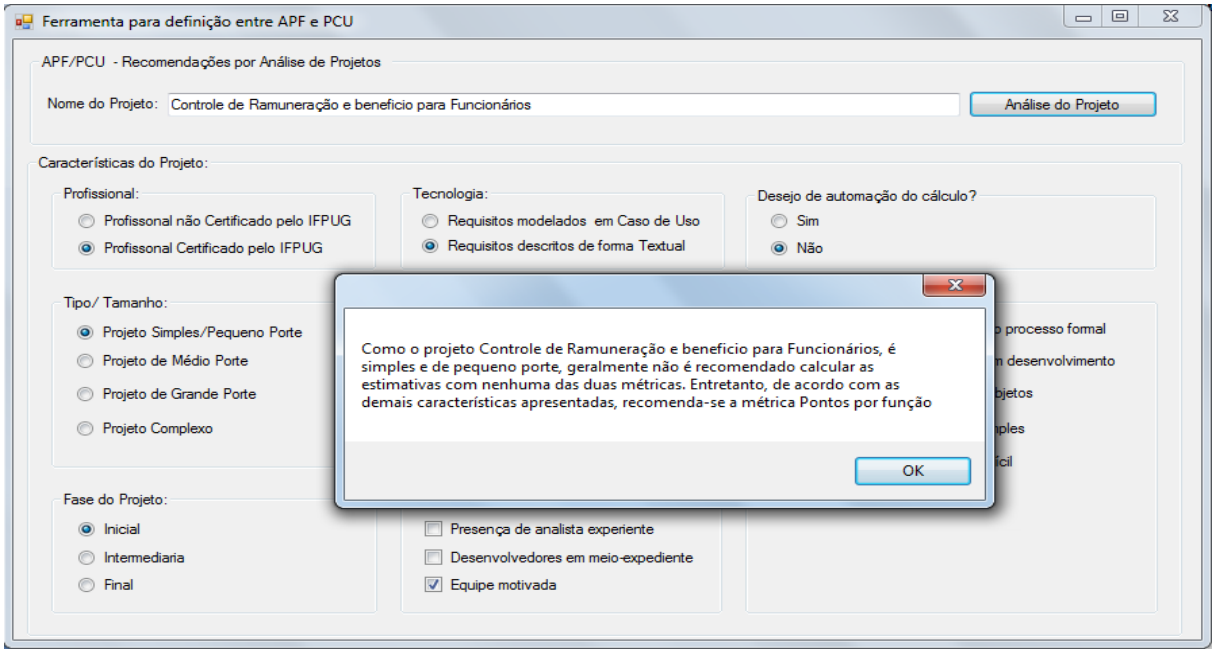

Figura 16 – Alerta e recomendação da Métrica PF.

5 CONCLUSÃO

Este trabalho teve como objetivo um estudo detalhado das métricas PCU e PF, no intuito de promover a melhoria da qualidade do processo de desenvolvimento de *Software*. É importante destacar que para se ganhar credibilidade e um produto estável, em conformidade com os requisitos do usuário, empresas desenvolvedoras de *Software* devem investir em métricas que calculam custo, prazo e o esforço de cada projeto.

O estudo bibliográfico realizado em comparação entre PF e PCU possibilitou a confirmação de que o uso dessas métricas para estimativas de *Software* auxiliam e promovem a qualidade dos projetos, mas que precisam ser pré-definidas antes de serem aplicadas.

A análise comparativa feita entre as métricas PF e PCU foi satisfatória, pois confirmou a existência de diferenças e limitações em cada uma das técnicas. Entretanto, não se pode afirmar que a métrica PF ou a PCU seja melhor ou menos confiável, apenas ambas precisam ser analisadas antes de serem aplicadas em determinados projetos de *Software*.

Os resultados obtidos com a ferramenta para análise de cenário também foram satisfatórios, pois agregam informação útil e auxiliam o gerente de projetos na escolha entre a melhor técnica baseando-se nas informações de cada projeto. Com isso a pesquisa responde que existem situações em que uma das técnicas em determinados projetos torna-se mais vantajosa do que a outra.

Porém, existem momentos em que nenhuma das duas métricas será recomendada plenamente, pois mesmo a APF sendo padronizada internacionalmente pela IFPUG, e a métrica PCU ser baseada na APF, cada uma contém suas peculiaridades.

Como recomendações para trabalhos futuros, sugere-se uma análise das demais métricas e tratamento de dados históricos de produtividade de cada uma delas, criando assim uma ampliação da ferramenta criada no presente trabalho, formando assim um manual eletrônico de recomendações de métricas de *Software*.

REFERÊNCIAS BIBLIOGRÁFICAS

ABSTS, C., BARRY, B., and CHULANI, S. *Software development cost estimation approaches - A survey*, 2000.

AGUIAR, M. *Function Points or Use Case Points?, 2009.*

AGUIAR, M. *Pontos de Função ou Pontos por Caso de Uso? Como estimar projetos Orientados a objetos*, 2003.

ALBERT, R. M., SIMÕES, G. S., VAZQUEZ, C. E. *Análise de Pontos de Função: Medição, Estimativas e Gerenciamento de Projetos de Software.* 10ª ed. São Paulo: Érica, 2008.

ALBERT, R. M., SIMÕES, G. S., VAZQUEZ, C. E. *Gestão de Contratos de Desenvolvimento de Software com a Análise de Pontos de Função*, 2004.

ALBRECHT, A.J. and GAFFNEY, J.R. *Software function, source lines of code and development effort prediction: a software science validation, IEEE Transactions on Software Engineering*, 1983.

ALBRECHT, A. J. *Measuring application development productivity, Proc. IBM Application Development Symposium,* 1979.

ANDRADE, L. P. E. OLIVEIRA M. K. *Aplicação de Pontos de Função e Pontos de Casos de Uso de Forma Combinada no Processo de Gestão de estimativa de tamanho de projetos de Software Orientado a Objeto*, 2004.

ANDRADE, L. P. E. OLIVEIRA M. K. *Uso combinado de analise de pontos de função e pontos por caso de uso na gestão de estimativa de tamanho de projetos de software orientado a objetos*, 2005.

BAPTISTA, G. L. *Um estudo sobre a utilização de sistemas de medição de desempenho em projetos de desenvolvimento de software: Perspectiva de gestores e empregados, 2011.*

BARTIÉ, A. *Garantia da Qualidade de Software*. 2ª ed. Rio de Janeiro: Elsevier, 2002.

BELGAMO, A. e FABBRI S. *Um Estudo sobre a Influência da Sistematização da Construção de Modelos de Casos de Uso na Contagem dos Pontos de Casos de Uso*, 2004.

BFPUG. *Teoria Explanatória para Estimativa Baseada em Casos de Uso no desenvolvimento orientado a objeto,* 2008*. Disponível em http://www.bfpug.com.br.*

BROWN, B. S., D´SOUZA, M. FISCHER, E. J. GARMUS, D., HOUSTON, K. T., MARTHALER, V., TIMP, A., VLIET E. V. *Manual de Práticas de Contagem de Pontos de Função*. Versão 4.2.1, 2005.

CHRISSIS, M. B., CURTIS, B. PAULK, M. C., WEBER, C. V. *The Capability Maturity Model: Guidelines for improving The Software Process.* 2ª ed. Rio de Janeiro: Addison – Wesley Longman, 1995.

CIMINO, R. *Planejar para Construir*. São Paulo: Editora Pini, 1987.

COELHO, F. A. M., SILVA, F. B., SOUZA, A. M. *Análise do custo e beneficio de técnicas estimativas de tamanho em diferentes cenários de aplicações de software*, 2007.

DEMARCO, T. *Controle de Projeto de Software*. 5ª ed. Rio de Janeiro: Editora Campos, 1991.

DRACH, M. D. *Aplicabilidade de Métricas por Pontos Por Função em Sistemas Baseados em Web*, 2005.

DRUCKER, P. F. *Administrando em tempos de grandes mudanças*. 5ª ed. São Paulo: Editora Afiliada, 1993.

FUKS, H., HAZAN C., LUCENA, C. J. P*. Avaliação do Tamanho Funcional de Ferramentas de E-learning*, 2005.

GRAHL, E. A., HEIMBERG, V. *Agentes e Métricas no Processo de Desenvolvimento de Software em Equipes Distribuídas*, 2005.

GUSTAFSON, D. A. *Teoria e problemas de Engenharia de Software.* Coleção Schaum. Porto Alegre: Bookman, 2003.

HAUFE, M. I. *Estimativa da Produtividade no desenvolvimento de Software*, 2001.

HAZAN, C. *Análise de Pontos por Função: Uma Abordagem Gerencial*, 2000.

HAZAN, C. *Análise de Pontos por Função: Uma ferramenta na implantação do Modelo CMM. Revista Tema, TemaTec, SERPRO,* 2003.

HAZAN, C., STAA A. V. *Análise e Melhoria de um Processo de Estimativas de Tamanho de Projetos de Software* *, 2005.

IFPUG. *Counting Practices Manual. Version 4.2*, 2004.

IFPUG. *Functional Sizing Method published as an International Standard. Version IFPUG 4.1*, 2003.

KARNER, G. *Resource Estimation for Objectory Projects. Objectory Systems*, 1993.

KOSCIANSKI, A.; SOARES, M. *Qualidade de Software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de Software*. 2ª ed. São Paulo: Novatec Editora, 2007.

LIVELY, W., KIM, S., SIMMONS, D. *An Effort Estimation by UML Points in the Early Stage of Software Development*, 2007.

MACHADO, J. B. *Métricas para qualidade de Software: um estudo de caso comparando análise de Pontos por Função e Pontos de Caso de Uso,* 2008.

MARTINS, M. D. C*. Geração de pontos de casos de uso no ambiente cocar*, 2007.

MCDONNELL, L. M. *Assessment Policy as Persuasion and Regulation*. American Journal of Education, 1994.

MEIRELLES, P. R. M. *Levantamento de Métricas de Avaliação de Projetos de Software Livre*, 2008.

MIRANDA, E. A. *Uma análise de métricas de Software orientadas à função e sua aplicação ao desenvolvimento orientado a objetos*, 2002.

MONTEIRO, T. C. *Pontos de Caso de Uso Técnicos (TUCP): Uma Extensão da UCP*, 2005.

MULLER, G. I. *Desenvolvimento de Software: Estimativa por Pontos de Caso de Uso,* 2010.

NESMA. *Function Point Analysis for Software Enhancement Guidelines.* Version 2.2.1, 2009.

PRESSMAN, R. S. *Engenharia de Software*. 3ª ed. São Paulo: Makron Books, 1995.

RIBU, K. *Estimating Object-Oriented Software Projects with Use Cases*, 2001.

SERPRO. Roteiro de Contagem de Pontos de Função, 2010.

SIMÕES, C. *Implantação de uma sistemática de métricas: A Difícil Arte de Estimar Tempo para implementação de Sistemas de informação*, 2004.

SOMMERVILLE, I. *Engenharia de Software.* 8ª ed. São Paulo: Pearson Addison-Wesley, 2007.

SOUZA, W. *CMMI APF – Análise de Pontos por Função ou UCP – Use Case Points?, 2012. Disponível em http://www.blogcmmi.com.br*

SOUZA, W. *CMMI - Como avaliar se meus fornecedores seguem CMMI/MPS.Br?, 2010. Disponível em http://www.blogcmmi.com.br*

SOUZA, W. *CMMI - Produtividade em horas por PF das principais linguagens, 2011. Disponível em http://www.blogcmmi.com.br*

SOUZA, W. *CMMI - Tutorial mini projeto parte 2 – Estimativa utilizando PF – Análise de pontos por função, 2009. Disponível em http://www.blogcmmi.com.br*

STANDISH GROUP INTERNATIONAL. *Chaos Summary 2009 Report*, 2009.

VIEIRA, L. E. *Uso do conceito de passos obrigatórios para aprimorar o processo de contagem do método "Pontos de Caso de Uso",* 2007.

FOLHA DE APROVAÇÃO DO TCC

ANDREA DAS NEVES AMORIM
ELIZA SANTOS GOTARDI

ESTUDO SOBRE TÉCNICAS DE DETERMINAÇÃO DE TAMANHO DE SOFTWARE: UMA ANÁLISE COMPARATIVA ENTRE PONTOS POR FUNÇÃO E PONTOS POR CASO DE USO.

Trabalho de Conclusão de Curso, apresentado a Universidade Nove de Julho como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, sob a orientação do Prof. Gabriel Lara Baptista.

Data: ____/____/____ _____

Assinatura do professor orientador

OBSERVAÇÕES: _____

_____

_____

_____

_____