UNIVERSIDADE FEDERAL DE MATO GROSSO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

MÉTRICAS PARA QUALIDADE DE SOFTWARE: UM ESTUDO DE CASO COMPARANDO ANÁLISE DE PONTOS POR FUNÇÃO E PONTOS DE CASO DE USO

JOICE BASÍLIO MACHADO

CUIABÁ – MT
2008

UNIVERSIDADE FEDERAL DE MATO GROSSO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

MÉTRICAS PARA QUALIDADE DE SOFTWARE: UM ESTUDO DE CASO COMPARANDO ANÁLISE DE PONTOS POR FUNÇÃO E PONTOS DE CASO DE USO

JOICE BASÍLIO MACHADO

Orientador: Prof. Dr. CRISTIANO MACIEL

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Mato Grosso, para obtenção do Título de Bacharel em Ciência da Computação.

CUIABÁ – MT
2008

UNIVERSIDADE FEDERAL DE MATO GROSSO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

CERTIFICADO DE APROVAÇÃO

Título: **Métricas para Qualidade de Software: um estudo de caso comparando Análise de Pontos por Função e Pontos de Caso de Uso**

Autora: Joice Basílio Machado

Aprovada em 10 / 12 / 2008

Prof. Dr. Cristiano Maciel
Instituto de Computação - UFMT
(Orientador)

Prof. Dra. Patricia Cristiane de Souza
Instituto de Computação - UFMT

Prof. M.Sc. Eunice Nunes
Instituto de Computação – UFMT

Prof. M.Sc. João Bosco Freitas
Tribunal Regional do Trabalho - TRT

DEDICATÓRIA

Dedico este trabalho, como expressão da minha gratidão, a Deus, Pai de Infinita Bondade, que me permitiu durante os quatro anos da graduação vivenciar tantas graças, adquirir conhecimento e amadurecer pessoal e profissionalmente. E também, *à Santa Maria Mãe de Deus*, que nesses meses esteve ao meu lado, segurando em minha mão.

Dedico de maneira muito especial aos meus pais e ao meu irmão, que me encorajaram em todos os momentos, e principalmente, que me suportaram e me compreenderam nesses últimos meses de trabalho monográfico.

E dedico ao meu querido, exigente e atencioso orientador, que me direcionou e me impulsionou a executar este trabalho arduamente.

AGRADECIMENTOS

Agradeço imensamente a Deus, por me cumular com suas Graças, durante toda a graduação e nesses últimos meses de forma especial, dando-me sabedoria, discernimento e força para a realização deste trabalho monográfico.

Quero agradecer também...

A minha família, meus pais e meu irmão, que se dispuseram, que me incentivaram e encorajaram em todos os momentos;

Ao meu orientador, Cristiano Maciel, que de maneira atenciosa me orientou, me direcionou e ampliou a minha visão quanto a pesquisa realizada, além de ser um professor-amigo e compreensivo.

Ao professor e instrutor do TRT, João Bosco Freitas, que me direcionou no início da pesquisa, indicando-me o tema deste trabalho, e que sempre me auxiliou na gratuidade. E também, a toda a equipe de TI do Tribunal, que me permitiu crescer profissionalmente durante o estágio.

Aos meus professores da graduação, em especial, Eunice Nunes, Patricia Cristiane e José Neves (Professor Zezinho), que mais do que professores foram amigos, pai e mãe em muitos momentos.

Aos meus amigos de turma, que me propiciaram muito aprendizado, alguns surpreendentes. Amigos que viveram tantos e tantos momentos de cansaço, de alegria, de desespero quase, mas que sempre encontravam uma forma de dar boas risadas, pois como aprendemos: "*Tudo dá certo no final*.".

Aos meus amigos-irmãos do Grupo de Oração Universitário Acadêmicos de Cristo e de todo o Ministério Universidades Renovadas, que estiveram ao meu lado em todos os momentos na graduação e me propiciaram alguns dos melhores momentos vividos na Universidade Federal de Mato Grosso.

A todos, que colaboraram direta ou indiretamente durante a minha graduação e principalmente nesses meses dedicados a monografia.

RESUMO

A utilização de métricas de software é reconhecida como uma das boas práticas dentro da engenharia de software. Por meio das métricas de estimativa de tamanho, é possível definir e conhecer o esforço, o custo e o prazo necessários para o desenvolvimento de um projeto. Isso proporciona maior qualidade ao processo de produção de software, pois a equipe segue um cronograma proposto em conformidade com os fatores reais de desenvolvimento, propiciando menos pressão e menos imprevistos para a equipe, qualidade ao produto final e satisfação do cliente. Neste trabalho foram estudadas as métricas de estimativa de tamanho de software Análise de Pontos por Função (APF) e Pontos de Caso de Uso (UCP). Esses métodos foram escolhidos por serem simples, eficientes e de fácil implantação nas equipes, podendo ser aplicados na fases iniciais do desenvolvimento de sistemas. De forma geral, a APF possui regras definidas e padronizadas para o processo de contagem de pontos de função, é reconhecida como padrão desde 2002 pela norma ISO/IEC 20926 e é aplicada na fase de projeto. Já a UCP é baseada nos casos de uso, feita na fase de especificação de requisitos e está ligada a projetos orientados a objetos. Neste trabalho faz-se a contagem dos pontos de função e dos Pontos de Caso de Uso no estudo de caso feito no Tribunal Regional do Trabalho (TRT) da 23ª Região de MT, e elabora-se uma análise comparativa entre as métricas abordadas. Com isso, pretende-se distinguir a relevância das métricas, discutir os resultados alcançados e a utilidade desses para o TRT e ressaltar a qualidade obtida no processo de produção de software.

Palavras-chave: métricas de software, qualidade de software, análise de pontos por função, pontos de caso de uso.

LISTA DE SIGLAS E ABREVIATURAS

ABNT	Associação Brasileira de Normas Técnicas
AIE	Arquivos de Interfaces Externos
ALI	Arquivos Lógicos Internos
APF	Análise por Ponto de Função
API	*Application Programming Interface*
BID	Banco Interamericano de Desenvolvimento
CE	Consulta Externa
CMM	*Capability Maturity Model*
CMMI	*Capability Maturity Model Integration*
COCOMO	*COnstructive COst MOdel*
COSMIC	*Common Software Measurement International Consortion*
DJE	Diário de Justiça Eletrônico
DTI	Departamento de Tecnologia de Informação
EE	Entradas Externas
EF	*Environmental Factor*
FA	Fator de Ajuste
FFP	*Full Function Points*
FINEP	Financiadora de Estudos e Projetos
GQS	Garantia da Qualidade de *Software*
IEC	*International Electrotechnical Commission*
IFPUG	*International Function Point Users' Group*
ISO	*International Standardization Organization*
LOC	*Lines Of Code*
MCT	Ministério da Ciência e Tecnologia

MPS.BR	Melhoria do Processo de *Software* Brasileiro
NATO	*North Atlantic Treaty Organization*
OO	*Object Oriented* - Orientado a Objeto
PF	Pontos por Função
PSP	*Personal Software Process*
SE	Saídas Externas
SLOC	*Source Lines Of Code*
SOFTEX	Excelência do Software Brasileiro
SWEBOK	*Software Engineering Body Of Knowledge*
TCF	*Technical Complexity Factor*
TRT	Tribunal Regional do Trabalho
TUCP	Pontos de Caso de Uso Técnicos
UAW	*Unadjusted Actor Weight*
UC	*Use Case*
UCP	*Use Case Points*
UML	*Unified Modeling Language*
UUCP	*Unadjusted Use Case Points*
UUCW	*Unadjusted Use Case Weight*
XFPA	*Extended Function Point Analysis*

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Neste capítulo serão apresentadas as questões mais importantes que levaram a realização deste trabalho, bem como os seus objetivos e a metodologia empregada.

1.1 Apresentação

Na medida em que a produção de software se torna mais competitiva, a melhoria da qualidade dos produtos de software passa a ser um diferencial ou uma exigência do mercado. Adotar uma abordagem visando assegurar a melhoria da qualidade do processo de produção e do próprio software passa a ser também uma questão de sobrevivência (FERRARI, 2007). Algumas propostas para tal situação seriam o controle de qualidade e o uso de padrões como o *International Standardization Organization* (ISO), o *Capability Maturity Model Integration* (CMMI), Melhoria do Processo de *Software* Brasileiro (MPS.BR), entre outros.

As empresas desejam aplicar os modelos de qualidade para obter rentabilidade e oferecer ao cliente satisfação na entrega de seus produtos. Podem-se mensurar algumas das características ou dos problemas enfrentados por grande parte das organizações no desenvolvimento de *software*, como afirmam Koscianski e Soares (2007):

1. Cronogramas não observados;
2. Projetos com tantas dificuldades que são abandonados;
3. Programas que não fazem o que era justamente esperado;
4. Programas tão difíceis de usar que são descartados; e
5. Programas que simplesmente param de funcionar.

Sendo assim, nota-se a importância do estabelecimento de práticas nas empresas que promovam a qualidade de software. Entre essas práticas, se enquadram as métricas de estimativa de software que, se aplicadas corretamente, auxiliam no processo de produção e de controle de software, minimizam ou extinguem os atrasos no cronograma, fornecem informações sobre o desenvolvimento do produto, indicam ações corretivas e propiciam a entrega dentro do prazo e do orçamento com um nível de qualidade que atenda aos requisitos do cliente (ANDRADE, 2004).

Existem várias formas de medir software, tais como Linhas de Código (LOC), Análise de Pontos por Função (APF - *Function Point Analysis*), MK II, Pontos de Função Completos (FFP - *Full Function Points*), COCOMO (*COnstrutive COst MOdel* – Modelo Construtivo de Custo), Pontos de Caso de Uso (UCP - *Use Case Points*) entre outras métricas e variações dessas.

Os métodos aplicados nesse trabalho, de forma detalhada e específica, serão Análise de Pontos por Função (APF) desenvolvido na década de 1970 por Allan Albrecht e Pontos de Caso de Uso (UCP) criado em 1993 por Gustav Karner. Existem diversas variações desses métodos na literatura, algumas delas serão ilustradas no final do capítulo 2.

A APF é uma forma de medir software que considera as funcionalidades criadas no sistema. Pode ser calculada antes de o código ser escrito, baseando-se na descrição arquitetural do projeto. Além disso, a APF é independente da tecnologia usada no desenvolvimento, conforme afirma Koscianski e Soares (2007). A APF é reconhecida como padrão internacional desde 2002 pela norma ISO/IEC 20926.

O método UCP proposto por Karner (1993) pode ser considerado uma melhoria da APF e destinado aos projetos orientados a objetos, pois se baseia nas especificações dos casos de uso, que são construídos na fase inicial do projeto, após a especificação dos requisitos de software. Quanto mais bem definidos os casos de uso, mais precisa será a estimativa, por isso o método é dependente dos casos de uso, que devem ser bem escritos e bem estruturados, com um nível de detalhamento apropriado. Pela inexistência de padrões para a construção de casos de uso não há padronização para o método que pode ser influenciado pela aplicação (MONTEIRO, 2005).

Implantar métricas de estimativa de software é uma das soluções para ações importantes e comuns ao desenvolvimento de projeto de software, tais como obedecer ao cronograma e atender aos requisitos propostos pelos usuários. Diante dessa verdade, esta pesquisa visa realizar uma análise comparativa das métricas APF (Análise de Pontos de Função) e UCP (Pontos de Caso de Uso) aplicadas a um estudo de caso do Tribunal Regional do Trabalho da 23ª Região do Estado de Mato

Grosso, em um Sistema de Gestão de Documentos – *DocFácil*, que ainda não utiliza qualquer métrica para estimar custo, prazo ou esforço.

Assim, espera-se por meio deste trabalho, gerar uma estimativa de tamanho de software correspondente a realidade da organização, estabelecer prazos coerentes com o desenvolvimento dos projetos, demonstrar a confiabilidade das métricas implantadas, desenvolver software de qualidade e garantir a satisfação do usuário.

1.2 Objetivos

1.2.1 Objetivo Geral

Este trabalho tem como objetivo realizar uma análise comparativa das métricas de software utilizando Análise de Pontos por Função (APF) e Pontos de Caso de Uso (UCP) para demonstrar a eficácia das métricas e, conseqüentemente, a obtenção da melhoria da qualidade do processo de desenvolvimento de software.

Para isso será feito um estudo de caso no "Módulo de Despachos" do sistema *DocFácil*, que está foi no Tribunal Regional do Trabalho da 23ª Região em Cuiabá, descrevendo algumas fases do projeto de desenvolvimento do sistema, aplicando as métricas citadas anteriormente.

1.2.2 Objetivos Específicos

Para atingir o propósito geral deste trabalho tem-se uma série de passos a serem percorridos ao longo do desenvolvimento, a saber:

i. Estudar os principais conceitos da qualidade de software e alguns dos modelos de maturidades como CMMI, PSP e MPS.BR;

ii. Identificar as principais métricas de qualidade de software*;*

iii. Investigar a Análise de Pontos por Função e os Pontos de Caso de Uso;

iv. Implementar um estudo de caso para o sistema *DocFácil;*

v. Especificar os casos de uso do sistema elaborando artefatos, tais como a descrição dos Casos de Uso;

vi. Efetuar a contagem dos pontos de função e dos pontos de casos de uso para o "Módulo de Despachos";

vii. Traçar um comparativo entre as métricas e os resultados obtidos.

1.3 Justificativa

O desenvolvimento de software pode de tornar sofisticado e complexo, os constantes avanços na indústria de hardware e a exigência cada vez maior do mercado por produtos com alto nível de qualidade, têm suscitado uma necessidade de maior empenho dos profissionais da área de engenharia de software, no que tange a estudos em busca de métodos e metodologias para apoiar o processo de produção de software (FERRARI, 2007).

Além de metodologias, outro aspecto que contribui é o uso de tecnologias e ferramentas durante a construção do produto de software. Existem várias tarefas que podem ser automatizadas, diminuindo a carga de trabalho das pessoas e, ao mesmo tempo, garantindo uniformidade (KOSCIANSKI, SOARES; 2007). O uso de métricas de software é uma das formas de contribuição para melhorar o processo de desenvolvimento, pois oferece informações que são representativas, objetivas, têm significado, são reproduzíveis, imparciais, facilmente implantadas na equipe e podem ser calculadas durante as várias fases de execução do projeto, levando a precisão dos resultados e cumprimento do cronograma estabelecido na fase de análise.

Como afirma Vieira (2007, p. 14), cada vez mais as empresas de desenvolvimento de software necessitam gerar estimativas para definir contratos com seus clientes. Por isso, o estabelecimento de métricas consistentes, objetivas e de fácil implantação na equipe, é fundamental para as organizações que desejam melhorar o processo de desenvolvimento de software, cumprir os cronogramas e oferecer produto/serviço de maior qualidade para seus clientes.

As métricas abordadas neste trabalho são a APF, que é mais antiga e padronizada internacionalmente, e a UCP, da década de 90 e que abrange projetos orientados a objeto, mas não possui um padrão.

A Análise de Pontos por Função (APF) é uma das métricas mais utilizadas em sistemas tradicionais desde 1979 e mede o tamanho das funções do software sob o ponto de vista do usuário, utilizando a documentação gerada durante todo o processo de desenvolvimento do produto, principalmente na fase de projeto. Apesar de ser a métrica mais utilizada no mercado, a APF não leva em conta as peculiaridades dos projetos orientados a objetos, pois foi criada em 1979, com base nos conceitos das técnicas de análise e projeto estruturados (ANDRADE, 2004).

Pontos de Caso de Uso (UCP) surgem como uma adaptação da APF, englobando os projetos orientados a objetos, são utilizados logo no início do ciclo de desenvolvimento, na fase de definição dos requisitos, com base no modelo de casos de uso. Ambos, APF e UCP, podem ser aplicados ao mesmo projeto, existem trabalhos que abordam o uso combinado para obter uma estimativa mais precisa.

Assim como na maioria das organizações, no Tribunal Regional do Trabalho (TRT), ainda não existe um padrão de qualidade ou metodologia específica para o desenvolvimento de software, os projetos são pouco modelados ou especificados devido a urgência com que as especificações são feitas pelos servidores e magistrados. Diante desse quadro, julga-se pertinente a aplicação das métricas APF e UCP em um estudo de caso no TRT. Essas métricas envolvem o mínimo de documentação (documento de requisitos, especificação de casos de uso e diagrama de classes, por exemplo), são simples, objetivas e de fácil implantação na equipe. Neste trabalho far-se-á a comparação de ambas, propondo uma estimativa coerente e mais detalhada para o cálculo do esforço do projeto.

1.4 Metodologia

A metodologia utilizada no trabalho em questão contempla uma revisão bibliográfica sobre: qualidade de software – garantia de qualidade, normas e organismos normativos, metodologias ágeis; engenharia de requisitos; métricas de software – orientadas ao tamanho, orientadas à função e medidas de qualidade, especificando as métricas de software Análise de Pontos por Função e Pontos de Casos de Uso. O estudo dos temas abordados oferece o referencial teórico necessário para subsidiar a proposição da abordagem pretendida.

Além disso, será realizado um estudo de caso visando avaliar a proposta defendida, ou seja, a importância da utilização de métricas de software. O estudo de caso adotado será feito com base no "Módulo de Despachos" do Sistema *DocFácil* que está sendo implementado no Tribunal Regional do Trabalho (TRT) da 23ª Região em Cuiabá. Para a sua realização são necessárias as especificações das funcionalidades e dos casos de uso referentes ao “Módulo de Despachos”, para cumprir as etapas dos métodos que serão aplicados.

Na etapa de estimativa de tamanho de software, serão realizadas as contagens de pontos de função e pontos de casos de uso, utilizando os métodos APF e UCP, para o "Módulo de Despachos".

Com base nos resultados obtidos será feita uma análise comparativa, ressaltando os pontos positivos e negativos do uso de métricas, a importância de estimar tempo, custo e esforço e, como isso está ligado diretamente à qualidade no planejamento, no desenvolvimento e na entrega do software.

1.5 Organização do Trabalho

Este trabalho é composto por 5 capítulos, incluindo este que faz uma introdução ao trabalho.

No capítulo 2, *Qualidade de Software*, são apresentados os conceitos de modelos de maturidade, entre eles o CMM (*Capacibility Maturity Model*), o CMMI (*Capacibility Maturity Model Integration*) e o MPS.Br (Melhoria de Processo do Software Brasileiro). Também são apresentados os principais conceitos de métricas de software, alguns métodos de estimativa de software, uma breve comparação entre as métricas APF e UCP e uma tabela que aponta algumas das variações desses métodos.

No capítulo 3, Estimativa de Software, descrevem-se de forma detalhada as métricas de software: Análise de Pontos por Função (APF) e Pontos de Caso de Uso (UCP).

No capítulo 4, Estudo de Caso, as métricas APF e UCP são aplicadas ao “Módulo de Despachos” do Sistema *DocFácil*.

No capítulo 5, Conclusão tem-se o fechamento do trabalho com a citação de algumas das limitações encontradas, algumas propostas para trabalhos futuros e as considerações finais.

No apêndice A, é apresentado o documento de visão do sistema - "Módulo de Despachos". No apêndice B, são apresentadas as descrições dos tipos de dados e registros das funcionalidades relacionadas ao cálculo da APF. E no apêndice C, são apresentadas as especificações dos principais casos de uso do sistema referente ao estudo de caso.

No Anexo A, são exibidas as características gerais do sistema para o cálculo do fator de ajuste da APF.

2 QUALIDADE DE SOFTWARE

Neste capítulo são apresentados os conceitos de qualidade de software, sua relevância no processo de desenvolvimento de software e alguns dos modelos de maturidade para a implantação da qualidade. E os conceitos de métricas de estimativa de software bem como os principais métodos de estimativa de software.

2.1 Engenharia de Software

O termo Engenharia de Software foi utilizado pela primeira vez, provavelmente, em uma conferência com esse nome, realizada por uma entidade que não possuía ligação com a área da computação – o Comitê de Ciência da *NATO* (*North Atlantic Treaty Organization* – Organização do Tratado do Atlântico Norte) realizada na Alemanha em 1968 (KOSCIANSKI; SOARES, 2007, p. 21).

Segundo Pressman (2002), a engenharia de software é como um rebento da engenharia de sistemas e de hardware. Pressman (2002) a divide em três elementos fundamentais: métodos, ferramentas e procedimentos, a fim de garantir ao gerente o controle do processo de desenvolvimento de software e uma base para a obtenção de software de alta qualidade produtivamente. De modo mais objetivo, pode-se dizer que a engenharia de software busca prover a tecnologia necessária para produzir software de alta qualidade a um baixo custo. Estes são fatores motivadores e essenciais para as empresas que procuram desenvolver software utilizando ambientes eficientes para produção.

2.2 Qualidade de Software

Qualidade é um conceito que, do ponto de vista da engenharia, é muito amplo. Um fator que exerce influência negativa sobre a qualidade de um projeto é a complexidade que está associada a duas características: o tamanho das especificações e sua volatilidade (KOSCIANSKI; SOARES, 2007). O custo é outra parte integrante do modelo de qualidade, tanto os custos relativos à não-conformidades (defeitos e suas correções) quanto os da manutenção da conformidade dos produtos e serviços a serem produzidos para garantir a qualidade.

O *Software Engineering Body of Knowledge* (SWEBOK) divide a Engenharia de Software em 10 áreas distintas (SWEEBOK, 2004): requisitos de software, design de software, gerência de engenharia de software, processo de engenharia de software, ferramentas e métodos, construção de software, teste de software, qualidade de software, manutenção de software, gerência de configuração de software. A Qualidade de Software assume posição importante, sendo um tema encontrado, de forma distribuída, em todas as outras áreas de conhecimento envolvidas em um projeto. Dessa forma, a garantia de qualidade torna-se um desafio ainda maior.

A satisfação do cliente é o principal objetivo a ser atingido na qualidade de software, para ter conformidade com os requisitos é primordial. Por meio da construção dos artefatos da engenharia de software, de metodologias tradicionais ou ágeis bem empregadas, as organizações almejam atingir um nível de maturidade e, conseqüentemente, rentabilidade e satisfação para os usuários.

2.2.1 Garantia da Qualidade de Software

A Garantia da Qualidade de Software (GQS) é uma das características de maturidade buscada pelas empresas e tem como objetivo fornecer aos vários níveis de gerência a adequada visibilidade dos projetos, dos processos de desenvolvimento e dos produtos gerados (KOSCIANSKI; SOARES, 2007). Obtendo a visibilidade desejada, a gerência pode atuar de forma pontual no sentido de atingir os quatro grandes objetivos de um projeto de desenvolvimento de software: desenvolver software de alta qualidade, ter alta produtividade da equipe de desenvolvimento, cumprir o cronograma estabelecido e não necessitar de recursos adicionais não previstos.

Como guias para a GQS aparecem os Modelos de Maturidade que procuram descrever como as organizações evoluem em um conjunto de níveis (KOSCIANSKI; SOARES, 2007). Em cada um deles, existem características específicas que definem a forma como a organização opera e qual é o estágio corrente de seu processo.

2.2.1.1 Modelos de Maturidade

O modelo de maturidade de processos fornece uma abordagem disciplinada para identificação dos processos críticos e definição de ações de melhoria. Existem diversos modelos na literatura, alguns deles são citados a seguir.

2.2.1.1.1 *Capacibility Maturity Model* (CMM)

O CMM (*Capacibility Maturity Model*) baseia-se nos cinco estágios de maturidade propostos por Crosby (1992), em *Quality is Free*. Conforme Koscianski e Soares (2007, p. 97), o objetivo do modelo é "que as organizações conheçam e melhorem seus processos de desenvolvimento de software com a implementação de práticas bem definidas".

Koscianski e Soares (2007) definem os níveis do CMM de acordo com as descrições abaixo:

- Nível 1: Inicial - não existem processos. O trabalho é realizado sem planejamento e as pessoas reagem às necessidades à medida que os problemas surgem.

- Nível 2: Repetitivo - a diferença em relação ao nível inicial é que já existem fórmulas prontas e as empresas são capazes de repetir as que deram certo – ainda que tais fórmulas possam estar longe do ideal.

- Nível 3: Definido - a característica chave é o esclarecimento das pessoas que compreendem os processos, sua utilidade e sua importância. A escolha dos procedimentos corretos não recai puramente sobre uma base empírica, mas, sobretudo sobre a sua adequação ao contexto que está sendo tratado. São capazes de detectar problemas mas, geralmente, quando os mesmos já estão instalados.

- Nível 4: Gerenciado – é um nível pró-ativo, capaz de antecipar os problemas. Para isso, sabem detectar tendências que apontam se o rumo seguido está correto ou não. As empresas deste nível efetuam medidas em seus processos e

procuram otimizá-los. Neste nível, mensurar faz parte do processo, pois só assim o acompanhamento se dá de forma mais efetiva.

- Nível 5: Otimizado – a qualidade está enraizada na filosofia das empresas, refletindo nos seus produtos e no seu funcionamento no dia-a-dia. A busca de alternativas para o aprimoramento de seus processos é um objetivo constante. Analisa constantemente, com bases históricas, seus processos, buscando melhoria contínua a fim de ganhar eficiência.

O CMM (*Capacibility Maturity Model*) foi criado para grandes projetos e organizações, porém isso não impede que as pequenas e médias empresas também o apliquem, pois os conceitos fundamentais do CMM são as boas práticas da engenharia de software. Uma prática adequada é não modificar tudo e repentinamente na empresa, mas adaptar gradativamente o modelo CMM conforme características da organização (KOSCIANSKI; SOARES, 2007).

2.2.1.1.2 *Capacibility Maturity Model Integration* (CMMI)

Nas palavras de Koscianski e Soares (2007, p. 162) "o objetivo do CMMI é servir de guia para a melhoria de processo nas organizações e também da habilidade dos profissionais em gerenciar o desenvolvimento, a aquisição e a manutenção de produtos e/ou serviços".

Existem duas representações para o CMMI: por estágio ou contínua (KOSCIANSKI; SOARES, 2007). A *abordagem por estágios* define 5 níveis de maturidade, expostos a seguir.

- Nível 1: Inicial – não existem processos ou padrões. Normalmente os prazos não são seguidos e há problema no cumprimento de requisitos.

- Nível 2: Gerenciado – os requisitos são gerenciados e os processos planejados, medidos e controlados.

- Nível 3: Definido – os processos são caracterizados e bem entendidos. A padronização de processos permite que os produtos tenham mais consistência.

- Nível 4: Gerenciado quantitativamente – os processos são controlados por meio de métodos estatísticos e outras técnicas quantitativas, por conta desse controle há um aumento da previsibilidade do desempenho de processos.

- Nível 5: Otimizado – os processos são continuamente melhorados com base em um entendimento quantitativo.

Para a representação do CMMI contínuo existem 6 níveis de capacitação e as áreas de processo são agrupadas por categorias afins (KOSCIANSKI; SOARES, 2007). Abaixo, têm-se as características de cada nível para o CMMI segundo Koscianski e Soares (2007).

- Nível 0: Incompleto – corresponde à não-realização de um processo.

- Nível 1: Realizado – cada processo deve cumprir todos os seus objetivos específicos em sua área.

- Nível 2: Gerenciado – o processo é monitorado, controlado e revisado, assim como os produtos resultantes.

- Nível 3: Definido – os processos padronizados são estabelecidos e melhorados continuamente.

- Nível 4: Gerenciado quantitativamente – os processos são definidos e controlados quantitativamente, há a identificação de medidas importantes para gerar ações corretivas.

- Nível 5: Otimizado – os processos são melhorados continuamente e modificados de forma a estabilizar as variações que podem comprometer o desempenho.

A diferença entre essas representações é que a representação por estágios utiliza os níveis de maturidade para medir a melhoria da capacitação da empresa, enquanto a contínua mede a melhoria de processos. A forma como cada abordagem é aplicada também difere, existem 05 níveis de maturidade para a representação por estágio e

06 níveis para a contínua, como descrito nesse tópico (KOSCIANSKI; SOARES, 2007).

De acordo como os dados do Ministério da Ciência e Tecnologia (MCT), o Brasil alcançou o número de 21 empresas certificadas até agosto de 2006 (KOSCIANSKI; SOARES, 2007).

2.2.1.1.3 *Personal Software Process* (PSP)

Nas palavras de Koscianski e Soares (2007, p. 116), o PSP:

> "O PSP possui o foco na melhoria dos processos do indivíduo, tornando sua forma de trabalho mais disciplinada. Para isso, são utilizados um conjunto de métodos. Formulários e *scripts* (roteiros) para planejar, medir e gerenciar o trabalho individual. O objetivo geral do PSP é a produção de produtos de software sem defeitos, respeitando prazos e custos planejados."

Assim como no modelo CMMI, o PSP é dividido em níveis. São eles (KOSCIANSKI; SOARES, 2007):

- Nível 0: fundamentos de medidas e formatos de relatórios que constituirão uma base para a a implantação do método. Também acrescenta padrões de programação, práticas de medidas de tamanho.

- Nível 1: planejamento de tarefas e estimativas de tamanho e tempo - elaboração de cronogramas.

- Nível 2: controle pessoal de qualidade de projeto. Técnicas de inspeção e revisão para auxiliar na detecção precoce de defeitos – coleta e análise de dados de erros de compilação e teste encontrados em programas anteriores.

- Nível 3: extensão a projetos grandes. Desenvolver iterativamente o programa. Em cada iteração, existe um ciclo completo de design, codificação e teste, como no nível 2.

Cabe ressaltar que, o foco do PSP é o trabalho individual, a fim de aumentar a eficiência de desenvolvedores e a qualidade dos produtos obtidos (KOSCIANSKI; SOARES, 2007).

2.2.1.1.4 Melhoria de Processo do Software Brasileiro (MPS.Br)

O Guia de Aquisição (2007, p. 04) apresenta o modelo da seguinte forma:

> "O MPS.Br é um programa para Melhoria de Processo do Software Brasileiro coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), contando com apoio do Ministério da Ciência e Tecnologia (MCT), Financiadora de Estudos e Projetos (FINEP) e Banco Interamericano de Desenvolvimento (BID)."

A descrição do modelo baseia-se em 3 guias: Guia Geral, Guia de Aquisição e Guia de Avaliação. O modelo possui algumas diferenças em relação aos outros citados neste capítulo, como por exemplo: são 7 os níveis de maturidade; tem compatibilidade plena com o CMMI e com a norma ISO/IEC 15504; foram criados para o Brasil, onde a maioria das empresas são micro, pequenas ou médias empresas; e tem custo acessível entre outras (KOSCIANSKY; SOARES, 2007).

Com relação ao MPS.Br, os níveis de maturidade estabelecem patamares de evolução de processos, caracterizando estágios de melhoria da implementação de processos na organização. A escala de maturidade se inicia no nível G e progride até o nível A. Eles são descritos abaixo, conforme o Guia Geral do MPS.Br (2007):

- Nível G (parcialmente gerenciado): composto pelos processos Gerência de Projetos e Gerência de Requisitos.
- Nível F (gerenciado): formado pelos processos do nível de maturidade anterior (G) acrescidos dos processos Aquisição, Gerência de Configuração, Garantia da Qualidade e Medição.
- Nível E (parcialmente definido): composto pelos processos dos níveis de maturidade anteriores (G e F), acrescidos dos processos Avaliação e Melhoria do Processo Organizacional, Definição do Processo Organizacional, Gerência de Recursos Humanos e Gerência de Reutilização. O processo Gerência de Projetos sofre sua primeira evolução retratando seu novo propósito: gerenciar o projeto com base no processo definido para o projeto e nos planos integrados.
- Nível D (largamente definido): formado pelos processos dos níveis de maturidade anteriores (G ao E), acrescidos dos processos Desenvolvimento

de Requisitos, Integração do Produto, Projeto e Construção do Produto, Validação, e Verificação.

- Nível C (definido): composto pelos processos dos níveis - de maturidades anteriores (G ao D), acrescidos dos processos Análise de Decisão e Resolução, Desenvolvimento para Reutilização e Gerência de Riscos. Neste nível, o resultado GRU 3 do processo Gerência de Reutilização (GRU) evolui para adequar esse processo aos resultados do processo Desenvolvimento para Reutilização (DRU).
- Nível B (gerenciado quantitativamente): composto pelos processos dos níveis de maturidade anteriores (G ao C), sendo que ao processo Gerência de Projetos são acrescentados novos resultados.
- Nível A (em otimização): composto pelos processos dos níveis de maturidade anteriores (G ao B), acrescido do processo Análise de Causas de Problemas e Resolução.

Para cada um destes sete níveis de maturidade é atribuído um perfil de processos que indicam como a organização deve aplicar o esforço de melhoria. O progresso e o alcance de um determinado nível se obtêm quando são atendidos os propósitos e todos os resultados esperados (Guia Geral do MPS.Br, 2007).

Segundo Koscianski e Soares (2007, p. 154), o modelo MPS.Br, por apresentar um número maior de estágios que o CMMI, permite que a implantação seja mais gradual e adequada para pequenas e médias empresas brasileiras.

2.3 Qualidade das Métricas

Todo o trabalho de avaliação é colocado em risco se não for possível garantir a obtenção de dados confiáveis (KOSCIANSKI; SOARES, 2007). Segundo Ferrari (2007), a precisão das estimativas de tamanho, por exemplo, torna-se fundamental para a elaboração de cronograma e orçamento realistas, por constituírem a base para a derivação das estimativas de esforço, prazo e custo. Nota-se, portanto, a necessidade da utilização de métricas ou medições para o planejamento e o gerenciamento de projeto de software.

Existem vários modelos de métricas de qualidade de software. O modelo SQWARE (*Software product Quality Requirements and Evaluation*) considera alguns critérios para elaborar ou escolher uma métrica para avaliar um produto. As métricas devem ser caracterizadas como (KOSCIANSKI; SOARES, 2007):

1. Significativas: os resultados obtidos devem agregar informação útil à avaliação de qualidade.

2. Repetíveis: caso uma medição seja realizada diversas vezes e não tenha nenhuma condição modificada, os resultados devem ser sempre os mesmos.

3. Reproduzíveis: os resultados de uma medição devem ser os mesmos para avaliadores diferentes.

4. Imparciais: deve-se fazer uma escolha criteriosa para os parâmetros de avaliação que serão utilizados.

5. Validadas: é preciso prover informação a respeito do grau de confiabilidade dos valores apresentados.

Métricas que tenham essas peculiaridades, provavelmente, serão mais eficientes e terão resultados consistentes e condizentes com a realidade da organização (KOSCIANSKI; SOARES, 2007).

2.3.1 Métricas de Software

Segundo Pressman (2002) a qualidade de software pode ser medida ao longo do projeto de engenharia de software e após a entrega do software ao cliente e aos usuários. A medição do software possibilita aos gerentes e projetistas maior compreensão do processo, podendo ser feita de diversas formas, utilizando métricas diretas e/ou indiretas. De acordo com Ferrari (2007), as métricas diretas utilizam-se das linhas de código enquanto as métricas indiretas derivam de medidas do domínio da informação e da avaliação subjetiva da complexidade do problema.

Medir software é uma das práticas que garante qualidade de software, cada projeto de software tem suas particularidades e exige da equipe maturidade ao escolher a

melhor forma de estimar as etapas ou os ciclos para o sistema. Além disso, a precisão dos resultados e o custo, são fatores que influenciam muito na escolha das métricas (KOSCIANSKI; SOARES, 2007).

As métricas de software referem-se a uma grande variedade de medições que são realizadas nos produtos de software (FERRARI, 2007). Segundo Pressman (2002, p. 61) as medições podem ser classificadas em medidas diretas e medidas indiretas, conforme descrito a seguir:

> "Entre as medidas diretas do processo de engenharia de software incluem-se o custo e o esforço aplicados. As medidas do produto incluem as Linhas de Código (LOC) produzidas, velocidade de execução, tamanho de memória e defeitos registrados ao longo de certo espaço de tempo. As medidas indiretas do produto incluem funcionalidade, qualidade, complexidade, eficiência, confiabilidade, manutenibilidade e muitas outras."

Pressman (2002) ainda faz a divisão das categorias de métricas de software em: métricas de produtividade, métricas de qualidade e métricas técnicas. Pressman (2002) e Somerville (2007) dividem as métricas de produtividade, ou estimativa de produtividade, em dois segmentos: métricas orientadas ao tamanho e métricas orientadas à função. Os conceitos e os métodos utilizados nesses segmentos serão apresentados nos itens a seguir.

2.3.2.1 Métricas orientadas ao tamanho

Segundo Pressman (2002, p. 62), métricas são medidas diretas do software e do processo por meio do qual ele é desenvolvido. Para Monteiro (2005, p. 79):

> "Estimar tamanho do produto de software é um dos primeiros passos para se efetivar uma estimativa. As fontes de informação com relação ao escopo do projeto iniciam-se na fase de concepção, com a descrição formal de requisitos. Por exemplo, uma solicitação de proposta do cliente, uma especificação de requisitos do sistema, ou uma especificação de casos de uso. Quanto maior for o conhecimento e o detalhamento das informações sobre o projeto, melhores serão os subsídios para a elaboração da estimativa de tamanho."

Para Koscianski e Soares (2007), utilizar medidas de tamanho de software exige bastante cautela. Os opositores ao uso destas medidas afirmam que as medidas de

tamanho de código LOC (*Lines Of Code*) são dependentes da linguagem de programação e que penalizam programas bem projetados (PRESSMAN, 2002, p. 63). Porém, segundo Koscianski e Soares (2007, p. 229) as medidas de tamanho ainda são usadas por motivos tais como:

- estão entre as métricas de aplicação mais simples;
- são de fácil interpretação quando não houver muita precisão em jogo; e
- possuem baixo custo de aplicação.

A métrica orientada a tamanho mais conhecida na literatura, Linhas de Código – LOC, será citada abaixo.

2.3.2.1.1 Medidas de Linhas de Código (*Lines Of Codes* - LOC)

Segundo Koscianski e Soares (2007), a medida mais simples para tamanho de um programa é a contagem de linhas de código. É uma das formas de medir mais antigas, mais simples e mais imprecisa, pois uma mesma solução pode ser desenvolvida de diversas maneiras e, ainda, um número pequeno de linhas de código (LOC) nem sempre significa velocidade e alto desempenho, como um alto valor de LOC pode não intuir a um baixo desempenho.

Somerville (2007, p. 407) aborda a origem da técnica, sua aplicação e sua atual situação no texto citado a seguir:

> "Essa abordagem foi inicialmente desenvolvida quando a maioria das programações era feita em FORTRAN, linguagem *Assembly* ou COBOL. Nessa época, os programas eram perfurados em cartões na caixa de programa. Contudo, os programas em linguagens, como Java e C++, consistem de declarações, comandos executáveis e comentários. Eles podem incluir instruções macro que se expandem para várias linhas de código. [...] Quanto mais expressiva for a linguagem de programação, menor será a produtividade aparente. Essa anomalia surge porque todas as atividades de desenvolvimento são consideradas quando se calcula o tempo de desenvolvimento, mas a métrica LOC se aplica somente ao processo de programação. Portanto, se uma linguagem requer mais linhas de código do que outra para implementar a mesma funcionalidade, a estimativa de produtividade será anormal."

Em Koscianski e Soares (2007), há duas variações da técnica: *SLOC* (*Source Lines Of Code* – Linhas de Código Fonte): lógico - na qual as linhas em branco e os comentários são desconsiderados, e físico - no qual há blocos de comandos. Atualmente, esta forma de medir não é tão utilizada, pois existem outras medidas mais precisas, mais eficientes e mais consistentes a serem aplicadas algumas delas citadas neste trabalho.

LOC foi uma métrica muito utilizada até a década de 1970 (ANDRADE, 2004). Com o surgimento de novas linguagens de programação, tem-se a necessidade de aplicar outras formas de medir software.

2.3.2.2 Métricas orientadas à função

As métricas para Somerville (2007) são relacionadas a funcionalidade geral do software a ser entregue. A produtividade é expressa em termos de quantidade de funcionalidade útil produzida. Essas são medidas indiretas de software e do processo de desenvolvimento.

Neste tópico, serão apresentadas algumas das métricas funcionais, tais como Análise de Pontos por Função (APF), COCOMO, COSMIC, FPP e Pontos de Caso de Uso (UCP).

2.3.2.2.1 Análise de Pontos por Função (APF)

A métrica Análise de Pontos por Função (*Function Point Analysis* - APF) foi desenvolvida em 1979 por Allan Albrecht, como uma forma de medir o software considerando as suas funcionalidades (KOSCIANSKI; SOARES, 2007).

O método foi apresentado à comunidade no final da década de 70. Após sucessivos trabalhos de Capers Jones destacando sua importância, houve um crescimento dos usuários de APF. Também foi importante a fundação em 1986 do *International Function Point User Group* (IFPUG). Juntamente com a aceitação crescente do mercado do método surgiram diversas variações da proposta apresentada por Allan Albrecht (VASQUEZ; SIMÕES; ALBERT, 2007, p. 37).

A contagem dos pontos de função é feita em 5 fases. Ressalta-se que o detalhamento de cada passo será feito no capítulo 3, abaixo se tem uma descrição resumida e objetiva de cada fase ou passo dessa métrica:

1. Determinação do tipo de contagem: definir se a contagem será para um sistema em desenvolvimento, para um projeto de melhoria ou para uma aplicação.

2. Delimitação da fronteira da aplicação: identificar o escopo do projeto.

3. Cálculo dos pontos de função não ajustados: verificar e classificar os arquivos que podem ser: arquivos lógicos internos, arquivos de interface externos, entradas externas, saídas externas e consulta externa, para calcular a complexidade de cada tipo de dado modelado para determinado sistema.

4. Cálculo do fator de ajuste: é calculado baseando-se em 14 características gerais do sistema.

5. Cálculo dos pontos de função ajustados: descreve o cálculo dos pontos de função ajustados, há uma contagem específica para cada tipo de projeto (desenvolvimento, melhoria ou aplicação).

A análise de pontos por função é normalizada pela ISO no documento ISO/IEC 14143-1:1998.

2.3.2.2.2 COSMIC FPP (*Full Function Points*)

A métrica FFP (*Full Function Points*) foi desenvolvida em 1997 por um grupo de pesquisadores da universidade de Quebec para medição funcional de sistemas em tempo real. Em 1998, um grupo de especialistas em métricas funcionais constitui o COSMIC (*Common Software Measurement International Consortion*), com o objetivo de reunir as melhores características da métricas existentes para desenvolver um novo método, o qual foi proposto no ano de 2000 (VASQUEZ; SIMÕES; ALBERT, 2007).

A FFP decompõe os requisitos funcionais em processos e subprocessos funcionais que equivalem às transações lógicas de um caso de uso, por exemplo. A métrica também estrutura os dados e a medição funcional da forma como descreve Souza (2006, p. 69):

> "Existem quatro tipos de dados previstos no modelo genérico de software da FFP: entrada, saída, leitura e gravação. Entradas movem dados dos usuários para o processo funcional da aplicação. Saídas movem dados no sentido contrário, ou seja, da aplicação para seus usuários. Leituras e gravações movem dados de e para o dispositivo de armazenamento.
>
> Para obtenção do dimensionamento funcional da aplicação, duas fases são previstas na FFP: a primeira, denominada Fase de Mapeamento, responsável por representar os requisitos da aplicação a ser medida na forma do modelo genérico de software. A segunda, conhecida como Fase de Medição, responsável pela avaliação e medição dos elementos dispostos no modelo, através de regras e procedimentos que quantificam o dimensionamento funcional do software."

A FPP foi projetada para medição de sistemas em tempo real, para sistemas gerenciais e para softwares orientados a objeto, características essas que a diferenciam das demais explicitadas neste trabalho. Atualmente está na versão 3.0, sendo regulamentada pela norma ISO/IEC 19761:2002.

2.3.2.2.3 *Construtive Cost Model* (COCOMO)

O modelo COCOMO (*COnstrutive COst MOdel* – Modelo Construtivo de Custo) é um modelo empírico que foi criado com base na análise de dados de um grande número de projetos de software. Esses dados foram usados para descobrir as fórmulas mais adequadas às observações, ligando o tamanho do sistema e do produto, fatores de projeto e de equipe ao esforço para desenvolver o sistema (SOMERVILEE, 2007).

O modelo COCOMO foi proposto por Boehm em 1981, a primeira versão do modelo é conhecida como COCOMO 81 e considera 3 modos de desenvolvimento, conforme López (2005):

- Orgânico: aplicável à ambientes de desenvolvimento estáveis, com pouca inovação e a projetos com equipes de dimensão relativamente pequena.
- Semi-destacado: aplicável à projetos com características que contemplam tanto o modo orgânico e quanto o embutido.
- Embutido: aplicável no desenvolvimento de sistemas complexos, com muita inovação, com restrições severas e/ou com requisitos muito voláteis.

Ocorreram muitas mudanças desde sua criação e o método no ano de 2000, sofreu algumas alterações para se adequar a nova realidade. O modelo COCOMO II é composto por 4 submodelos (SOMERVILLE, 2007): modelo de composição de aplicação, modelo de projeto preliminar, modelo de reuso e modelo de pós-arquitetura.

Segundo López (2005), o COCOMO II, busca medir esforço, prazo, tamanho e custo, necessários para o desenvolvimento de software, desde que se tenha, como premissa, a dimensão do mesmo. Para o cálculo do custo, deve-se conhecer o prazo e equipe de trabalho, para então chegar ao valor, sendo que para definir o tamanho do programa, torna-se necessário que se caracterize que medida será adotada (Linhas de Código, Pontos por Função ou Pontos de Caso de Uso).

2.3.2.2.4 Pontos de Caso de Uso (*Use Case Points* - UCP)

O método UCP (*Use Case Points)* foi desenvolvido por Gustav Karner em 1993, inspirando-se na Análise de Pontos por Função, com algumas adaptações para a aplicação do método para projetos orientados a objetos. Karner explora o modelo e a descrição de casos de uso, substitui algumas características técnicas da APF, cria fatores ambientais e propõe uma estimativa de produtividade (MONTEIRO, 2005).

Os Pontos de Caso de Uso fornecem uma estimativa de software na fase inicial do desenvolvimento de software, tendo como entrada os casos de uso. Como características têm-se a facilidade de uso, o processo de contagem simples, o levantamento de requisitos do projeto e o cálculo das estimativas, que pode ser feito com o auxílio de uma planilha eletrônica (VIEIRA, 2007).

Karner (1993) estabeleceu 06 passos para o processo de contagem dos Pontos de Caso de Uso, os quais serão abordados de forma simplificada neste tópico e detalhados no capítulo 3 deste trabalho.

Passos para a contagem dos Pontos de Caso de Uso (UCP):

- Classificação dos atores: os atores são classificados em simples, médio ou complexo, de acordo com a aplicação, e são atribuídos os respectivos pesos 1, 2 e 3 para calcular o somatório de pesos de atores não ajustados, o *UAW* (*Unadjusted Actor Weight*).

- Classificação dos casos de uso: os casos de uso (UC – *use case*) são classificados, assim como os atores, em simples, médio ou complexo dependendo do número de transações ou de classes contidas em cada descrição do UC. São atribuídos pesos para cada tipo de UC, sendo os valores 5, 10 e 15 respectivamente. Nessa etapa, tem-se o somatório dos pesos de casos de uso não ajustados *UUCW* (*Unadjusted Use Case Weight*).

- Cálculo dos pontos de casos de uso não ajustados: somam-se os valores de UAW e UUCW, obtidos nos itens anteriores, e têm-se os Pontos de Caso de Uso não ajustados, o UUCP (*Unadjusted Use Case Points*).

- Cálculo dos fatores técnicos de complexidades: calcula-se, de forma semelhante ao fator de ajuste da APF (Análise de Pontos por Função), o fator de complexidade técnica TCF (*Technical Complexity Factor*), que avalia 13 características referentes aos requisitos funcionais do sistema.

- Cálculo dos fatores ambientais: calcula-se o fator ambiental EF (*Environmental Factor*), somatório baseado em 6 itens relacionados aos requisitos não-funcionais do sistema, cujos pesos variam de 0 a 3.

- Cálculo dos pontos de caso de uso ajustados: faz-se o cálculo dos Pontos de Caso de Uso que é resultado da contagem dos Pontos de Caso de Uso não ajustado *versus* fator ambiental *versus* fatores técnicos de complexidade.

O método UCP, como esclarecido no início desse tópico, é de fácil aplicação nas organizações e simples, mas não é reconhecida como padrão. Um dos motivos é a falta de padronização das especificações dos casos de uso e do documento de requisitos, o que dificulta a aceitação do método.

2.3.2.3 Comparação entre APF e UCP: Trabalhos Relacionados

Para os métodos elicitados neste trabalho, serão traçados, de forma comparativa, os principais pontos relacionados à APF (Análise de Pontos por Função) e à UCP (Pontos de Caso de Uso) na Tabela 1, pois essas são as métricas aplicadas ao estudo de caso proposto e, portanto de maior relevância para o presente trabalho.

Tabela 1: Comparativo entre APF e UCP

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.
É amparada por diversos grupos, entre eles o IFPUG (*International function point users group*).	Não há um grupo ou organização responsável pela métrica.
Possui regras definidas e padronizadas para o processo de contagem de pontos de função.	Não há padrões para a descrição dos casos de uso, há dúvidas na contagem de casos de uso incluídos e estendidos.
Aplicada na fase de análise 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.

Fonte: Adaptado de Pereira *et al*. (2007).

Nesta pesquisa optou-se pela APF, método reconhecido internacionalmente e mais consolidado, e pelo UCP, método mais recente, semelhante à Análise de Pontos por Função, e que ainda não é utilizado e aprovado por todos, devido a falta de padronização nas especificações dos casos de uso, mas que oferece vantagens como a simplicidade e agilidade na aplicação do método.

Existem variações da APF e UCP, que são adaptações melhoradas para os métodos. Porém, não são amplamente difundidos e utilizados nas organizações, o que dificulta

a implantação dessas derivações em empresas que não têm experiência com métricas. Neste tópico serão apenas comentadas rapidamente algumas das variações dos métodos na Tabela 2.

Tabela 2: Variações da APF e da UCP

	Variação	***Descrição***
Análise de Pontos por Função (APF)	*Fast Count* (ANDRADE, 2004)	• Desenvolvido em 1993 para estimar ponto de função com base no método científico; • A estimativa é feita em menos tempo porque a contagem se baseia no modelo central de dados.
	Object Oriented Function Point (*OOFP*), (ANDRADE, 2004)	• Desenvolvido em 1998, faz mapeamento dos conceitos da APF para os conceitos Orientados a Objetos (OO); • Realiza a estimativa de tamanho em diferentes pontos do desenvolvimento de software.
	Predictive Object Points (*POPs*) (ANDRADE, 2004)	• Proposto em 1997 por Minkiewicz; • Baseado na contagem de classes e nos pesos dos métodos de cada classe.
	Object Oriented Design Function Points, (*OODFP*) (ANDRADE, 2004)	• Adaptado por Ram e Raju em 2000, após a definição dos procedimentos de contagem do IFPUG (*International Function Point User's Group*); • Estima o tamanho de software OO na fase de projetos, baseando-se na complexidade das classes e nas funções do software.
	Extended Function Point Analysis (*XFPA*), (SOUSA, 2006)	• Complementa e amplia as características gerais do sistema da APF, baseando-se em duas dimensões: tecnológica e ambiental ou contextual.
	Mark II (VIEIRA, 2007)	• Proposto por Charles Symons em 1991; • Vê o sistema como uma coleção de transações lógicas.

Pontos de Caso de Uso (UCP)	Pontos de Caso de Uso Técnicos (TUCP) (MONTEIRO, 2005)	• Desenvolvido em 2005 por Monteiro; • Propõe mudanças como: detalhar o conceito de transação; refinar a contagem de UCP complexos; desatrelar os fatores técnicos ambientais do cálculo do tamanho, entre outras.
	Variação	***Descrição***
	Mudanças segundo Ribu (2001) (RIBU, 2001 *apud* ANDRADE, 2004)	• Proposta de mudanças em 2001 para o método1 UCP, entre elas: descartar os fatores de ajustes técnicos; contar os casos de uso incluídos e estendidos, e utilizar um formulário padrão para descrição de casos de uso.
	Passos Obrigatórios para a contagem da UCP (VIERA, 2007)	• Determina que a partir da expansão dos casos de uso existe um conjunto de passos que podem ser diferenciados em obrigatórios ou complementares. Os passos obrigatórios são aqueles que determinam o esforço de desenvolvimento, enquanto que os passos complementares não têm influência sobre o desenvolvimento. • Propõe que os fatores ambientais sejam não apenas multiplicados, mas somados à complexidade dos casos de uso.

Essas variações, entre outras existentes, aprimoram os métodos originais, inserindo no processo de contagem dos Pontos de Função ou dos Pontos de Caso de Uso, mais exatidão, contribuindo então para a não subjetividade e para a especificidade do projeto de software a ser medido.

No próximo capítulo, serão detalhados os processos de contagem das métricas Análise de Pontos por Função e Pontos de Caso de Uso.

3 ESTIMATIVA DE SOFTWARE

Neste capítulo serão apresentados os conceitos das métricas Análise de Pontos por Função e Pontos de Caso de Uso.

3.1 Análise de Pontos por Função (APF)

As medidas de tamanho de software surgiram com o objetivo de estimar o esforço e o prazo associados ao desenvolvimento dos sistemas. Em 1979, Allan Albrecht desenvolveu a métrica Análise de Pontos por Função como forma de medir as funcionalidades do software e não o tamanho físico ou o número de linhas de código como era feito. De acordo com Vasquez, Soares e Albert (2006, p. 35), Análise de Pontos por Função e Pontos de Função (PF) podem ser definidos como:

> "Análise de Ponto por Função -APF, é uma técnica de medição das funcionalidades fornecidas por um software do ponto de vista de seu usuário. Ponto de Função é a unidade de medida desta técnica que tem por objetivo tornar a medição independente da tecnologia utilizada para a construção de software. Ou seja, a APF busca medir o que o software faz, e não como ele foi construído. Portanto o processo de medição, ou a contagem de pontos de função, é baseada em uma avaliação padronizada dos requisitos lógicos do usuário."

Segundo Vasquez, Soares e Albert (2006), a APF não mede diretamente o esforço, a produtividade ou o custo, é uma medida funcional. O processo de avaliação da Análise de Pontos por Função pode ser dividido em cinco etapas:

1. Identificação do tipo de contagem a ser utilizado;
2. Definição da Fronteira da Aplicação;
3. Contagem de pontos de função não ajustados;
4. Cálculo do Fator Ajuste; e
5. Contagem dos pontos de função ajustados.

Essas etapas serão descritas nos itens a seguir.

3.1.1 Tipo de Contagem

Seguindo as normas do IFPUG (*International Function Point Users' Group* – Grupo Internacional de usuários de pontos de função) existem três tipos de contagem para os pontos de função:

- **Contagem de um projeto de desenvolvimento** – mede as funcionalidades da primeira instalação e as eventuais funções de conversão necessárias a implantação do sistema;
- **Contagem de um projeto de melhoria** – mede as funções adicionadas, alteradas ou removidas do sistema e também as possíveis funções de conversão de dados;
- **Contagem de uma aplicação ou *baseline*** – mede a funcionalidade da aplicação (ou sistema) fornecida ao usuário.

Conforme o tipo de projeto a ser medido há uma fórmula final para calcular os pontos de função, mas independentemente do tipo de contagem escolhido, o cálculo é baseia-se nos requisitos do usuário e nas funcionalidades do sistema.

3.1.2 Definição da Fronteira da Aplicação

Vasquez, Soares e Albert (2006, p. 57) definem a fronteira da aplicação como: "a interface conceitual que delimita o software que será medido e o mundo exterior (seus usuários)". Trata-se do escopo do sistema, das funcionalidades requeridas pelo usuário. Sua determinação deve ser feita com base nos requisitos dos usuários através dos documentos do sistema, como documento de visão ou de especificação.

3.1.3 Contagem de pontos de função não ajustados

Para aplicar a APF considera-se, através do levantamento de requisitos, a complexidade de cinco fatores ou funções na terminologia do IFPUG (VASQUEZ; SOARES; ALBERT, 2006):

- Arquivos lógicos internos (ALI) – grupo de dados mantidos pelo sistema e relacionados logicamente.

- Arquivos de interface externos (AIE) – grupo de dados referenciados pelo sistema (dados de outra aplicação) e relacionados logicamente.

- Entradas externas (EE) – transação lógica que introduz dados (recebidos de outra aplicação) no sistema e que causam mudança nos ALI (alteração, exclusão ou inserção de dados).

- Saídas externas (SE) – transações lógicas em que dados são enviados além da fronteira do sistema (relatórios, telas de ajuda, consultas).

- Consulta externa (CE) – transação lógica que envolve um par consulta-resposta, não há qualquer cálculo matemático os dados são recuperados de ALI e AIE.

Sendo identificados e contados os tipos de dados, devem ser classificados segundo a complexidade funcional (baixa, média ou alta) para efetuar o cálculo dos pontos de função não ajustados. As Tabelas abaixo fornecem a classificação com relação à complexidade:

Tabela 3: Complexidade funcional para ALI (Arquivo Lógico Interno) e AIE (Arquivo Interno de Interface).

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

Fonte: Adaptado de Vasquez, Soares e Albert (2006) e Koscianski e Soares (2007).

Tabela 4: Complexidade funcional para EE (Entradas Externas)

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

Fonte: Adaptado de Vasquez, Soares e Albert (2006) e Koscianski e Soares (2007).

Tabela 5: Complexidade funcional para SE (Saídas Externas) e CE (Consultas Externas)

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

Fonte: Adaptado de Vasquez, Soares e Albert (2006) e Koscianski e Soares (2007).

Tabela 6: Cálculo dos pontos de função não ajustados

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

Fonte: Adaptado de Vasquez, Soares e Albert (2006) e de Koscianski e Soares (2007).

3.1.4 Cálculo do Fator Ajuste

O fator de ajuste é calculado baseando-se em 14 (catorze) características gerais do sistema, que definem as peculiaridades do sistema, listadas na Tabela 07.

Tabela 7: Características gerais do sistema

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

Fonte: Adaptado de Vasquez, Soares e Albert (2006) e de Koscianski e Soares (2007).

Essas características permitem diferenciar o sistema, pois afetam o sistema de forma peculiar. Elas possuem níveis de influência sobre a aplicação, variando de 0 a 5, como demonstra a Tabela 8. A especificação de cada uma das características gerais do sistema encontram-se no Anexo A.

Tabela 8: Níveis de influência das características gerais do sistema

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

Fonte: Adaptado de Vasquez, Soares e Albert (2006) e de Koscianski e Soares (2007).

Após determinar todos os níveis de influência o fator de ajuste (*FA*) pode ser calculado, e é dado por:

$$FA = 0{,}65 + ((n1 + n2 + n3 + ... + n14) \times 0{,}01)$$

Onde *n1, n2, ..., n14* representam os valores das 14 características do sistema.

3.1.5 Contagem dos pontos de função ajustados

A última etapa da contagem de pontos de função consiste em descrever o cálculo dos pontos de função ajustados. Há uma contagem específica para cada tipo de projeto (desenvolvimento, melhoria ou aplicação). Serão apresentadas as fórmulas como estão descritas no manual do IFPUG, inclusive neste trabalho serão utilizados mesmos nomes de variáveis.

3.1.5.1 Projeto em Desenvolvimento

Para uma aplicação em desenvolvimento o cálculo é feito da seguinte forma (VASQUEZ; SOARES; ALBERT, 2006):

$$\boldsymbol{DFP = (UFP + CFP) \times VAF}$$

A descrição para cada fator ou sigla segue abaixo:

- *DFP* é o número de pontos de função do projeto em desenvolvimento.
- *UFP* é o número de pontos de função não ajustados das funções disponíveis depois da instalação.
- *CFP* é o número de pontos de função não ajustados das funções de conversão.
- *VAF* é o valor do fator de ajuste.

3.1.5.2 Projeto de Melhoria

O projeto de melhoria engloba as alterações na aplicação para atender aos novos requisitos de negócios do usuário (VASQUEZ; SOARES; ALBERT, 2006). Existem várias especificidades para definir as funções do tipo dado e transação mas não serão detalhadas nesse trabalho, pois não se encaixa no estudo de caso feito. A fórmula para calcular os pontos de função do projeto de melhoria segue abaixo:

$$EFP = [(ADD + CHGA + CFP) \times VAFA] + (DEL \times VAFB)]$$

Fonte: Vasquez, Soares e Albert (2006).

Descrição dos fatores:

- *EFP* é o número de pontos de função do projeto de melhoria;
- *ADD* é o número de pontos de função não ajustados das funções incluídas no projeto;
- *CHGA* é o número de pontos de função não ajustados das funções modificadas no projeto;
- *CFP* é o número de pontos de função não ajustados adicionados pela conversão;
- *VAFA* é o valor do fator de ajuste após o projeto;
- *DEL* é o número de pontos de função não ajustados das funções excluídas no projeto;
- *VAFB* é o valor do fator de ajuste antes da aplicação do projeto.

3.1.5.3 Aplicação

O número de pontos de função de uma aplicação mede a funcionalidade fornecida ao usuário por uma aplicação instalada (VASQUEZ, SOARES, ALBERT, 2006). A fórmula para calcular os pontos de função da aplicação conforme Vasquez, Soares e Albert (2006) é a seguinte:

$$AFP = ADD \times VAF$$

Descrição dos fatores:

- *AFP* é o número de pontos de função ajustados da aplicação;
- *ADD* é o número de pontos de função não ajustados das funções implementadas;
- *VAF* é o valor do fator de ajuste da aplicação.

Essa fórmula representa as funcionalidades requisitadas pelo usuário de uma aplicação que pode ser um pacote de software, um software recentemente desenvolvido (VASQUEZ; SOARES; ALBERT, 2006). Neste trabalho, esse será o tipo de contagem para o estudo de caso, pois trata-se de um novo software a ser instalado na organização.

3.1.6 Análise do Resultado

Todas essas etapas são compostas de detalhes e especificidades que formam a métrica da APF. Para que o resultado da análise seja confiável é de suma importância que os requisitos tenham sido elicitados com clareza e corretude, pois a APF é feita de acordo com a especificação de requisitos de software.

A Análise de Pontos por Função oferece muitas vantagens para as organizações que a aplicam, podemos citar a amplitude da visão de projeto de desenvolvimento e/ou produção, a estimativa de gastos e de recursos requeridos, e o apoio ao gerenciamento do fator de qualidade.

3.2 Pontos de Caso de Uso (UCP)

A estimativa é feita com base nos casos de uso do projeto, por isso a eficácia do método depende de uma padronização para a especificação dos casos de uso, pois um

dos passos para a aplicação da métrica é a contagem de transações por caso de uso (MONTEIRO, 2005). Por ser um método simples, de fácil utilização e elaborado na fase inicial do projeto de software (a especificação de requisitos) tem sido utilizado por mais empresas no mercado e sido motivo de discussão e estudo.

O processo para o cálculo da UCP é descrito em 06 passos, como apresentado abaixo (KARNER, 1993).

1. Classificação dos atores;
2. Classificação dos casos de uso;
3. Cálculo dos pontos de casos de uso não ajustados;
4. Cálculo dos fatores técnicos;
5. Cálculo dos fatores ambientais; e
6. Cálculo dos Pontos de Caso de Uso ajustados.

3.2.1. Classificação dos Atores

O primeiro passo é classificar os atores envolvidos em cada caso de uso e multiplicar o total de atores de acordo com o tipo de complexidade (simples, médio ou complexo) pelo respectivo peso (1, 2 ou 3) conforme a Tabela 9.

Tabela 9: Classificação dos atores

Complexidade	***Descrição***	***Peso***
Simples	Aplicação com uma API (*Application Programming Interface* - Interface para Programação de Aplicativos) definida.	1
Médio	Aplicação com interface baseada em protocolo ou interação de usuário baseado em linhas de comandos, ou seja, sistema que interage através de um protocolo como TCP/IP ou FTP.	2
Complexo	Interação de usuário por meio de Interface Gráfica ou página Web.	3

Fonte: Adaptado de Monteiro (2005) e de Karner (1993).

O peso total dos atores do sistema, UAW (*Unadjusted Actor Weight* – Peso de atores não ajustados) é calculado pela somatória dos pesos de cada ator referente ao sistema modelado.

3.2.2. Classificação dos casos de uso

Nesse caso, deve-se calcular o peso bruto dos casos de uso que pode ser simples, médio, ou complexo, dependendo do número de transações contidas na descrição do caso de uso, o UUCW (*Unadjusted Use Case Weight* – Peso dos casos de uso não ajustados) (KARNER, 1993). O número de transações ou de classes deve ser multiplicado pelo respectivo peso de cada caso de uso conforme descrito na Tabela 10 (Classificação dos casos de uso).

Tabela 10: Classificação dos casos de uso

Complexidade	***Descrição***	***Peso***
Simples	Tem até 3 transações incluindo os passos alternativos ou deve ser realizado com até 4 classes de análise.	5
Médio	Tem de 4 a 7 transações incluindo os passos alternativos ou deve ser realizado com 5 a 10 classes de análise.	10
Complexo	Tem 7 ou mais transações. Deve ser realizado com pelo menos 10 classes de análise.	15

Fonte: Adaptado de Monteiro (2005), de Karner (1993) e de Ribu (2001).

3.2.3. Cálculo dos pontos de casos de uso não ajustados

Os Pontos de Caso de Uso não ajustados *UUCP* (*Unadjusted Use Case Points* – Pontos de Casos de Uso Não Ajustados) são calculados pela somatória dos atores (*UAW*) somados ao valor dos pesos de caso de uso não ajustados (*UUCW*), conforme a equação:

$$UUCP = \sum UAW + \sum UUCW$$

Fonte: Karner (1993).

3.2.4. Cálculo dos fatores técnicos

De forma semelhante a Análise de Pontos por Função no método UCP existem 13 fatores técnicos que são levados em consideração para calcular o fator de complexidade técnica que cobre uma série de requisitos funcionais do sistema para a estimativa dos Pontos de Caso de Uso.

Os fatores de complexidade técnica *TCF* (*Technical Complexity Factor*), são exibidos na Tabela 11, com seus respectivos pesos, que variam de 0 a 5. O valor 0 indica que este quesito é irrelevante para o projeto, de pouca criticidade e baixa complexidade; o valor 3 indica influência moderada; e o valor 5 indica uma forte influência, alta criticidade e complexidade (MONTEIRO, 2005).

Tabela 11: Fatores de complexidade técnica

F_i	***Descrição***	w_i
F_1	Sistemas Distribuídos	2
F_2	Desempenho da Aplicação	1
F_3	Eficiência do usuário-final (*on-line*)	1
F_4	Processamento interno complexo	1
F_5	Reusabilidade de código em outras aplicações	1
F_6	Facilidade de instalação	0,5
F_7	Usabilidade (facilidade operacional)	0,5
F_8	Portabilidade	2,0
F_9	Facilidade de manutenção	1,0
F_{10}	Concorrência	1,0
F_{11}	Características especiais de segurança	1,0
F_{12}	Acesso direto para terceiros	1,0
F_{13}	Facilidades especiais de treinamento	1,0

Fonte: Karner (2003).

O *TCF* é calculado multiplicando-se o valor de cada fator (F_1 – F_{13}) pelo respectivo peso e depois se calcula o somatório de todos esses valores no chamado *TFator,* conforme a fórmula a seguir (KARNER, 1993):

$$TFator = \sum w_i$$

O *TCF* é dado pela equação (KARNER, 1993):

$$TCF = (0{,}6 + (0{,}01 * TFator)$$

3.2.5. Cálculo dos fatores ambientais

Os fatores ambientais (*Environmental Factor – EF*) referem-se aos requisitos não-funcionais associados ao projeto, tais como a experiência da equipe, a estabilidade do projeto e a motivação dos programadores (MONTEIRO, 2005).

Cada fator ambiental é associado a um peso, Monteiro (2005, p. 94) explana detalhadamente os valores que devem ser aplicados:

> "Para fatores F1 a F4, 0 significa nenhuma experiência no assunto, 5 significa excelência no assunto, e 3 um conhecimento mediano sobre o assunto. Para o fator F6, 0 significa requisitos instáveis, 5 imutáveis, e 3 poucas mudanças. Para o fator F7, 0 significa nenhum colaborador por meio período, 5 significa que todos os colaboradores trabalham meio período, e 3 que alguns colaboradores trabalham meio período. Para o fator F8, 0 significa uma linguagem com pouca ou nenhuma dificuldade de programação, 3 significa com uma complexidade média, e 5 com uma alta complexidade e dificuldade."

A Tabela 12 exibe os fatores ambientais e seus respectivos pesos:

Tabela 12: Fatores de complexidades ambientais

F_i	***Descrição***	w_i
F_1	Familiaridade com o desenvolvimento de *software*	1,5
F_2	Experiência na aplicação	0,5
F_3	Experiência com orientação a objeto	1
F_4	Capacidade do analista ou líder do projeto	0,5
F_5	Motivação	1
F_6	Requisitos estáveis	2
F_7	Trabalhadores com dedicação parcial	-1
F_8	Dificuldade da Linguagem de Programação	-1

Fonte: Adaptado de Karner (1993) e de Andrade (2005).

O fator ambiental (*EF*) é calculado segundo a fórmula abaixo (KARNER, 1193):

$$EF = 1.4 + (- 0.03 \times EFator)$$

EFator é dado pela somatória dos pesos multiplicados pelos seus respectivos valores de influência.

$$EFator = \sum(\text{valor de influência} * w_i)$$

Fonte: Karner (1993).

3.2.6 Cálculo dos Pontos de Caso de Uso ajustados

Os pontos de casos de uso (*UCP*) ajustados são calculados após serem obtidos os UUCP (*Unadjusted Use Case Points* – Pontos de casos de usos não ajustados), o *TCF* (*Technical Complexity Factor* – Fator de Complexidade Técnica) e o *EF* (*Environmental Factor* – Fator Ambiental), segundo Karner (1993):

$$UCP = UUCP * TCF * EF$$

3.2.7 Análise do Resultado

Karner (1993) faz uma correlação entre os Pontos de Caso de Uso e a média de recursos para concluir a estimativa de um determinado projeto, e sugere um valor aproximado de 20 homens/hora por ponto de caso de uso (UCP).

O esforço é dado pela multiplicação dos Pontos de Caso de Uso obtidos e o fator de produtividade (número homens/hora), como demonstra a fórmula abaixo.

$$Esforço = UCP * PROD$$

Fonte: Karner (1993).

O valor de produtividade pode ser alterado de acordo com a equipe de desenvolvimento para uma organização específica e que se deve aplicar o método UCP em projetos pilotos para então definir o valor homens/hora (RIBU, 2001 *apud* VIEIRA, 2007).

No próximo capítulo serão apresentados o estudo de caso e a aplicação das métricas enunciadas nesse tópico.

4 ESTUDO DE CASO

Neste capítulo são apresentadas uma breve descrição do "Módulo de Despachos" do Sistema *DocFácil*, os cálculos dos Pontos de Função e dos Pontos de Caso de Uso para o módulo e a análise comparativa dos métodos.

4.1 Visão Geral do Sistema DocFácil – "Módulo de Despachos"

O *DocFácil* - Sistema de Gestão de Documentos tem como proposta a geração automatizada e o gerenciamento do ciclo de vida de documentos tais como: despachos, notificações, mandados, ofícios, cartas, votos, acórdãos, entre outros. Objetiva, sobretudo, a otimização, padronização, gerenciamento, controle de autoria e guarda de documentos. Pretende-se também que o *DocFácil* incorpore o processo de assinatura digital, publicação na internet e no DJE (Diário de Justiça Eletrônico), quando necessário.

O sistema permite a integração com o *OpenOffice* por meio de uma API (*Application Programming Interface*) a fim de inserir, alterar e excluir documentos em um repositório, possibilitando o uso de todas as funcionalidades dessa ferramenta de automação de escritório que, além de ser eficiente e poderosa, é gratuita. A aplicação deverá prover uma interface gráfica com botões, barra de ferramentas e menus que interagem com o *OpenOffice*.

Brevemente descrevendo o sistema, a criação de documentos no *DocFácil* funciona a partir de um conjunto de modelos existentes no repositório de dados. O gestor de modelos (um dos atores do sistema) solicitará à DTI (Departamento de Tecnologia de Informação) a criação de novos modelos de documentos os quais terão partes protegidas e outras editáveis. O usuário poderá preencher o corpo do documento e, usando o repositório de documentos, fazer a inclusão, alteração ou exclusão do documento. Caso o modelo procurado não exista no repositório, o usuário poderá recorrer ao Gestor de Modelos.

O "Módulo de Despachos" é parte do *DocFácil*, que é constituído por 18 módulos e tem como principais funcionalidades:

- Criação, edição e exclusão de Despacho;
- Criação de Modelo de Despacho;
- Assinatura de Despacho; e
- Publicação de Despacho.

Mais descrições sobre o "Módulo de Despachos" como escopo, usuários e envolvidos no projeto e a visão geral, encontram-se no Apêndice A desta monografia.

4.2 Cálculo dos Pontos por Função para o "Módulo de Despachos"

Os passos necessários para calcular os pontos de função do sistema são descritos de forma objetiva nos tópicos abaixo (VASQUEZ; SOARES; ALBERT, 2006):

- Identificação do tipo de contagem a ser utilizado;
- Definição da Fronteira da Aplicação;
- Contagem de pontos de função não ajustados;
- Cálculo do Fator Ajuste; e
- Contagem dos pontos de função ajustados.

4.2.1 Identificação do tipo de contagem

Existem 3 tipos de contagem: projeto de desenvolvimento, de melhoria (ou manutenção) e de aplicação (ou produção). O "Módulo de Despachos" é um sistema que está em produção e a fórmula para o cálculo é a contagem para uma aplicação, pois, segundo Vasques, Soares e Albert, fornece uma medida da atual funcionalidade obtida pelo usuário da aplicação, como descrita neste trabalho no item 3.1.5.3.

4.2.2 Definição da Fronteira da Aplicação e o Escopo da contagem

O sistema tem como proposta a geração automatizada e o gerenciamento do ciclo de vida do documento de despacho. Objetiva a otimização, padronização, gerenciamento do ciclo de vida, controle de autoria e armazenamento dos documentos.

O escopo do sistema compreende:

- Ambiente para criação, edição e exclusão de despachos; e
- Assinatura Eletrônica e publicação de despachos.

4.2.3 Contagem de pontos de função não ajustados

Para aplicar a APF considera-se, por meio do levantamento de requisitos, a complexidade de cinco fatores: arquivos lógicos internos (ALI), arquivos de interface externos (AIE), entradas externas (EE), saídas externas (SE) e consulta externa (CE). Todos esses fatores já foram detalhados no capítulo 3.

Após identificados e contados os tipos de dados e tipos de transação, os mesmos devem ser classificados segundo a sua complexidade funcional (baixa, média ou alta), para efetuar o cálculo dos pontos de função não ajustados. Para o "Módulo de Despachos" têm-se as funcionalidades citadas na Tabela 13.

Tabela 13: Descrição das Funcionalidades

Descrição	Tipo	TD	AR/TR	Comp.	Valor
Login	EE	04	01	Baixa	03
Criação de Despacho	ALI	07	05	Baixa	07
Edição de Despacho	ALI	07	05	Baixa	07
Criação de Despacho com base em Modelo	ALI	09	06	Média	10
Exclusão de Despacho	ALI	03	03	Baixa	07
Assinatura de Despacho	EE	13	08	Alta	07
Publicação de Despacho	CE	20	07	Alta	06
Impressão de Despacho	CE	04	04	Média	04
Criação de Modelo de Despacho	ALI	07	07	Média	10
Consulta de Despacho Fechado	CE	05	04	Média	04
Consulta de Despacho Aberto	CE	05	04	Média	04
Consulta de Despacho Excluído	CE	05	04	Média	04
Consulta de Despacho Modificado	CE	05	04	Média	04
Consulta de Despacho Disponibilizado na *WEB*	CE	05	04	Média	04
Consulta de Despacho Publicado	CE	05	04	Média	04
Consulta de Despacho do Processo	CE	12	07	Alta	06
Legenda: Tipo: Classificação da funcionalidade (ALI, AIE, EE, SE OU CE)					

Descrição	Tipo	TD	AR/TR	Comp.	Valor
Login	EE	04	01	Baixa	03
TD: Quantidade de tipos de dados AR: Quantidade de arquivos referenciados TR: Quantidade de tipos de registro Comp.: Complexidade					

O detalhamento desse resultado, ou seja, os tipos de dados e os arquivos referenciados, são apresentados no Apêndice B do presente trabalho.

4.2.4 Cálculo de Pontos de Função não ajustados

Os pontos de função não ajustados são calculados com base nas funcionalidades referenciadas no item acima. Para cada tipo de dado ou transação existem pesos de acordo com o número de arquivos referenciados e de tipos de registros. Na Tabela 14, tem-se a descrição para o total de pontos de função não ajustados (PFNA).

Tabela 14: Pontos de função não ajustados

Descrição	Complexidade	Total parcial	Total
ALI	3 Baixa x 7 2 Média x 10 0 Complexa x 15	21 20 0	41
AIE	0 Baixa x 5 0 Média x 7 0 Complexa x 10	0 0 0	0
EE	0 Baixa x 3 1 Média x 4 1 Complexa x 6	0 4 6	12
SE	0 Baixa x 4 0 Média x 5 0 Complexa x 7	0 0 0	0
CE	0 Baixa x 3 7 Média x 4 2 Complexa x 6	0 28 12	40
		Total PFNA:	***93***

O "Módulo de Despachos" é um projeto relativamente pequeno, observa-se isso no total de funcionalidades requeridas e no resultado obtido ao calcular os pontos de função não ajustados, que totalizaram 93 pontos.

4.2.5 Cálculo do Fator de Ajuste

O fator de ajuste é calculado baseando-se em 14 características gerais do sistema, que definem as peculiaridades do sistema. Essas características afetam o sistema de maneira geral e possuem níveis de influência sobre a aplicação, que variam de 0 a 5 pontos.

Após conhecer o valor dos fatores de influência, expressos na Tabela 15 que ilustra as especificações para cada característica, pode-se determinar o fator de ajuste aplicando a fórmula:

FA = 0,65 + ((n1 + n2 + n3 + ... + n14) x 0,01)
*FA = 0,65 + (21*0,01)*
FA = 0,86

Tabela 15: Características gerais do sistema

Característica	Descrição para o "Módulo de Despachos"	Valor
Comunicação de dados	A aplicação é mais que um *front-end*, mas suporta apenas um tipo de protocolo de comunicação.	4
Funções distribuídas	A aplicação não participa da transferência de dados ou processamento de funções entre os componentes do sistema.	0
Desempenho	Requisitos de projeto e desempenho foram estabelecidos e revisados, mas nenhuma ação especial foi tomada.	1
Configuração muito utilizada	Existem restrições operacionais, mas são menos restritivas do que aplicações típicas. Nenhum esforço extra será necessário para suplantar as restrições.	1
Volume de transações	Existem restrições operacionais, mas são menos restritivas do que aplicações típicas. Nenhum esforço extra será necessário para suplantar as restrições.	1
Entrada de dados *on-line*	Todas as transações são processadas em modo *batch*.	0
Eficiência do usuário final	De quatro a cinco dos itens descritos: menus; movimento automático do cursor; teclas de função preestabelecidas; utilização de mouse; o menor número possível de telas para executar as funções de negócio.	2
Atualizações *on-line*	Nenhuma atualização.	0
Complexidade Processamento	Nenhum dos itens é necessário.	0
Reusabilidade	A aplicação foi especificamente projetada e/ou documentada para ter seu código facilmente reutilizado por outra aplicação e aplicação é customizada para uso através de parâmetros que podem ser alterados pelo usuário.	5

Característica	Descrição para o "Módulo de Despachos"	Valor
Facilidade de instalação	Nenhuma consideração especial foi feita pelo usuário, mas um procedimento especial é requerido para a implantação.	1
Facilidade operacional	Nenhuma consideração especial de operação, além do processo normal de salvar foi estabelecida pelo usuário.	0
Múltiplos locais	Além da etapa 3 (três), um plano de documentação e manutenção deve ser elaborado e testado para suportar a aplicação em múltiplos locais.	5
Facilidade de mudanças	São fornecidos mecanismos de consulta flexível, que permitem manipulação de pedidos simples, dados de controle de negócio são mantidos pelo usuário por meio de processos interativos e as alterações têm efeito imediato.	3
	Total de fatores de influência:	23

4.2.6 Cálculo dos Pontos de função ajustados

O "Módulo de Despachos" se trata de uma aplicação e o cálculo dos pontos de função ajustados (PFA) são calculados da seguinte maneira:

*PFA = PFNA * FA*
*PFA = 93 * 0,86*
PFA = 79,98 PF
PFA = 80 PF

4.2.7 Análise do Resultado dos pontos de função

A projeção do esforço foi feita considerando o número pequeno da equipe e a produtividade na linguagem empregada (JAVA), ou seja, proposto o valor de 20 horas por ponto de função (PF). Têm-se então 1600 horas, fazendo a conversão para meses trabalhados obtêm-se a estimativa de 6,6 meses (seis meses e meio) para o desenvolvimento do projeto. Supondo que a hora trabalhada seja equivalente a R$ 200,00 - valor mínimo segundo o Serviço Federal de Processamento de Dados (SERPRO), o custo do projeto será de R$320.000,00. Resumindo tem-se:

- Tamanho do projeto: 80 PF
- Esforço: 20 horas/ PF x 80 PF = 1600 horas
- Prazo: 1600 / (30*8) = 6,6 meses
- Custo: 1600 horas x R$ 20,00 = R$320.000,00

4.3 Cálculo dos Pontos de Caso de Uso para o "Módulo de Despachos"

O processo segundo Karner (1993) para o cálculo da *UCP* é descrito em 06 passos, como apresentado abaixo.

- Classificação dos atores;
- Classificação dos casos de uso;
- Cálculo dos pontos de casos de uso não ajustados;
- Cálculo dos fatores técnicos;
- Cálculo dos fatores ambientais; e
- Cálculo dos Pontos de Caso de Uso ajustados.

4.3.1. Classificação dos Atores

Para o módulo de documentos de Despacho temos os atores gestor de *layout*, gestor de modelo e usuário final, citados na Tabela abaixo e o respectivo valor dos pesos de atores não ajustados, o *UAW (Unadjusted Actor Weight)*.

Tabela 16: Classificação para os atores do Módulo de Documentos

Descrição Ator	***Complexidade***	***Valor***	***Peso***	***Peso total***
Gestor de *layout*	Simples	1	1	1
Gestor de modelo	Simples	1	1	1
Usuário	Simples	1	1	1
			Total UAW:	***3***

4.3.2. Classificação dos casos de uso

Para o "Módulo de Despachos" foram escritos 21 casos de uso de acordo com a modelagem feita na fase de concepção do software. Conforme os requisitos coletados e as características do sistema, a complexidade foi calculada e na Tabela 17 observam-se a descrição do caso de uso e sua complexidade, o número de classes envolvidas para a sua realização e o peso referente, aplicando o método de Karner (1993), conforme detalhado no item 3.2.2 deste trabalho.

Tabela 17: Classificação para os casos de uso do Módulo de Documentos

Descrição UC	***Complexidade***	***Número classes***	***Peso***
UC Criar *layout* de despacho	Simples	04	5
UC Editar *layout* de despacho	Simples	04	5
UC Excluir *layout* de despacho	Simples	02	5
UC Criar modelo de despacho	Médio	07	10
UC Editar modelo de despacho	Médio	07	10
UC Excluir modelo de despacho	Simples	04	5
UC Criar despacho	Simples	04	5
UC Editar despacho	Simples	04	5
UC Assinar despacho	Simples	04	5
UC Publicar despacho	Simples	02	5
UC Imprimir despacho	Simples	02	5
UC Excluir despacho	Simples	02	5
		Total UUCW:	***70***

As especificações de casos de uso, os diagramas de caso de uso e demais documentação estão no Apêndice C deste trabalho.

4.3.3. Cálculo dos pontos de casos de uso não ajustados

Os Pontos de Caso de Uso não ajustados *UUCP* (*Unadjusted Use Case Points*) são calculados pela somatória dos pesos dos atores não ajustados (UAW) e dos pesos de casos de uso não ajustados (*UUCW*), conforme a equação:

$UUCP = \sum UAW + \sum UUCW$
$UUCP = 3 + 70$
UUCP = 73

4.3.4 Cálculo dos fatores técnicos

Os fatores de complexidade técnica, TCF (*Technical Complexity Factor*), são exibidos na Tabela 18 juntamente com seus respectivos pesos, que variam de 0 a 5. O valor 0 (zero) indica que esse quesito é irrelevante para o projeto, de pouca criticidade e baixa complexidade; o valor 3 indica influência moderada; e o valor 5 indica uma forte influência, alta criticidade e complexidade (MONTEIRO, 2005), conforme detalhado no item 3.2.4 deste trabalho.

Tabela 18: Fatores de complexidade técnica para o "Módulo de Despachos"

F_i	***Descrição***	***Peso***	***Valor***	***Total***
F_1	Sistemas Distribuídos	2	0	0
F_2	Desempenho da Aplicação	1	5	5
F_3	Eficiência do usuário-final (*on-line*)	1	5	5
F_4	Processamento interno complexo	1	0	0
F_5	Reusabilidade de código em outras aplicações	1	5	5
F_6	Facilidade de instalação	0,5	5	2,5
F_7	Usabilidade (facilidade operacional)	0,5	5	2,5
F_8	Portabilidade	2,0	3	6
F_9	Facilidade de manutenção	1,0	3	3
F_{10}	Concorrência	1,0	3	3
F_{11}	Características especiais de segurança	1,0	3	3
F_{12}	Acesso direto para terceiros	1,0	0	0
F_{13}	Facilidades especiais de treinamento	1,0	3	3
			TFator:	***38***

Sendo o *TFator* conhecido, pode-se calcular o *TCF*:

TCF = 0.6 + (0.01 * ***38***)
***TCF* = 0,98**

4.3.5. Cálculo dos fatores ambientais

Os fatores ambientais (*Environmental Factor – EF*) se referem aos requisitos não-funcionais associados ao projeto, tais como a experiência da equipe, a estabilidade do projeto e a motivação dos programadores (MONTEIRO, 2005).

Para o "Módulo de Despachos" tem-se a Tabela de fatores ambientais apresentada abaixo:

Tabela 19: Fatores de complexidades ambientais

F_i	**Descrição**	**w_i**	**Valor**	**Total**
F_1	Familiaridade com o desenvolvimento de *software*	1,5	3	4,5
F_2	Experiência na aplicação	0,5	3	1,5
F_3	Experiência com orientação a objeto	1	3	3
F_4	Capacidade do analista ou líder do projeto	0,5	5	2,5
F_5	Motivação	1	3	3

F_i	Descrição	w_i	Valor	Total
F_6	Requisitos estáveis	2	3	6
F_7	Trabalhadores com dedicação parcial	-1	3	-3
F_8	Dificuldade da Linguagem de Programação	-1	3	-3
			Tfator:	***12,5***

Conhecido o *TFator* obtém-se o valor do *EF*:

*EF = 1,4 + (-0,03 * TFator)*
*EF = 1,4 + (-0,03 * 12,5)*
EF = 1,025

4.3.6 Cálculo dos Pontos de Caso de Uso ajustados

Os pontos de casos de uso (UCP) ajustados são calculados após serem obtidos os UUCP (*Unadjusted Use Case Points*), o TCF (*Technical Complexity Factor*) e o EF (Environmental Factor) da seguinte forma:

*UCP = UUCP * TCF * EF*
*UCP = 73*0,98*1,025*
UCP = 73,32

4.3.7 Análise do Resultado dos Pontos de Caso de Uso

Considerando-se que a equipe de desenvolvimento deste projeto é pequena, para o sistema "Módulo de Despachos" será aplicado o valor padrão de 20 homens/horas. Têm-se então 1460 horas no total, convertendo esse resultado para meses trabalhados obtêm-se 6,08 meses (seis meses aproximadamente) para o desenvolvimento e o custo de R$ 292.000,00 ao atribuir o valor de R$ 200,00 - valor mínimo segundo o Serviço Federal de Processamento de Dados (SERPRO) - por hora trabalhada no projeto. Resumindo, têm-se os dados:

- Tamanho do projeto: 73 UCP
- Esforço: 20 homens/hora x 73 PF = 1460 horas
- Prazo: 1460 / (30*8) = 6 meses
- Custo: 1460 horas x R$ 20,00 = R$ 292.000,00

4.4 Análise Comparativa

Diante dos resultados demonstrados nos itens 4.2.7 e 4.3.7 e sumarizados na Tabela 20, nota-se que o número de Pontos por Função é relativamente maior que os Pontos

de Caso de Uso, isso porque a APF, segundo a maioria dos autores, é mais minimalista que o método UCP.

Tabela 20: Relação entre APF e UCP

Análise de Pontos por função	*Pontos de caso de uso*
Tamanho: 80 PF	Tamanho: 73 UCP
Esforço: 1600 horas	Esforço: 1460 horas
Prazo: 6,6 meses	Prazo: 6 meses
Custo: R$ 320.000,00	Custo: R$ 292.000,00

A APF é feita baseando-se nos requisitos do sistema, que se elicitados corretamente não sofrerão grandes alterações, propiciando um resultado objetivo. O cálculo da UCP, também pode sofrer influências relevantes devido a suas especificações não terem padronização, dependendo do analista são mais coesas ou não, ocasionando assim alterações significativas na contagem dos Pontos de Caso de Uso.

O "Módulo de Despachos", efetivamente, foi desenvolvido em 6 meses no Tribunal Regional do Trabalho. Comparando-se o tempo real do projeto e as estimativas feitas, pode-se afirmar que ambas são confiáveis. Acredita-se que houve um bom gerenciamento do projeto, pois mesmo sem ter feito a estimativa, a equipe alcançou o tempo pré-estabelecido na contagem dos métodos.

Caso a métrica fosse aplicada efetivamente pela equipe, e não somente neste trabalho monográfico as metas de qualidade seriam atingidas, pois o cronograma seria cumprido dentro do prazo, haveria corretude e satisfação do usuário.

Analisando os valores de custo, prazo e esforço na Tabela 20, observa-se que a APF oferece um pouco mais de prazo para a equipe de desenvolvimento da empresa contratante. Afirma-se que a Análise de Pontos por Função é mais estável que o método UCP, e para este caso, a UCP foi mais exata. Propõe-se o uso combinado dessas métricas em fases distintas de desenvolvimento do sistema, o que permitirá um resultado mais correto e fornecerá à equipe segurança, caso a discrepância dos valores seja irrelevante, caso contrário servirá de alerta, incentivando a uma revisão nos requisitos e/ou nos casos de uso.

Uma outra sugestão é o uso de uma das variações dos métodos citados na Tabela 20. A Análise de Pontos por Função Estendida, por exemplo, propõe que as características gerais do sistema, sejam particionadas em tecnológica e ambiental (ou contextual), o que auxilia na análise e na revisão dos requisitos não funcionais e até mesmo na reavaliação de requisitos funcionais do sistema (SOUZA, 2006). Tal fato proporciona maior especificidade ao cálculo da estimativa para o projeto e ao resultado obtido.

Uma segunda derivação que poderia ser aplicada ao "Módulo de Despachos", neste caso referente ao método de Pontos de Caso de Uso, é o uso do conceito de passos obrigatórios para aprimorar o processo de contagem do método Pontos de Caso de Uso proposto por Vieira (2007). A mudança proposta refere-se a contagem dos casos de uso obrigatórios e complementares e, afirma que os fatores ambientais sejam não apenas multiplicados, mas somados à complexidade dos casos de uso, proporcionando mais rigor ao resultado da contagem dos Pontos de Caso de Uso (Vieira, 2007).

Como afirmado anteriormente, para o "Módulo de Despachos", foram utilizadas as estimativas de tamanho de software APF e UCP, pois para uma empresa em que as métricas ainda não foram implantadas se torna mais viável a aplicação de modelos conhecidos e que fornecem uma base teórica consistente para a equipe se apoiar.

A APF é feita na fase de especificação de requisitos, e a UCP na especificação de casos de uso, esses são dois momentos diferentes do desenvolvimento. A aplicação das métricas nessas fases propicia o gerenciamento da qualidade de software, pois ao executar a contagem de pontos por função, inicia-se o desenvolvimento do software. Após, o início do projeto, gerente ou alguém responsável, poderá contar os Pontos de Caso de Uso, e analisar se os resultados são compatíveis ou não e tomar as decisões cabíveis.

5 CONCLUSÃO

No processo de desenvolvimento de software existem diversas práticas de gerenciamento. As métricas de software podem ser aplicadas na fase inicial e auxiliam no planejamento do cronograma e do custo efetivo do projeto.

Diante da relevância das métricas de software, este trabalho teve como objetivo propor a aplicação de métodos de estimativas de tamanho de software, ao Tribunal Regional do Trabalho (TRT), órgão em que o estudo de caso foi aplicado, no intuito de promover a melhoria da qualidade do processo de desenvolvimento de software. Dessa forma, atua-se diretamente no cronograma do projeto, proporcionando o cumprimento das metas estabelecidas inicialmente; no desenvolvimento dos requisitos coletados em conformidade com o usuário e, por conseguinte, na qualidade do processo, pois os prazos serão obedecidos e as funcionalidades exigidas pelo usuário estarão de acordo com os requisitos elicitados na fase de análise.

As métricas estudadas neste trabalho, Análise de Pontos por Função (APF) e Pontos de Caso de Uso (UCP) são conhecidas e, relativamente, difundidas nas organizações e academias. A APF, proposta por Allan Albrecht em 1979, é reconhecida e padronizada internacionalmente, tem no IFPUG (*International Function Point Users' Group* - que é um grupo internacional de usuários do método) um apoio para a difusão da métrica, também para a aplicação das provas de certificações aos profissionais e, na organização de congressos e iniciativas que propiciam mais conhecimento aos pesquisadores e maior segurança para as empresas que desejam implantar a APF.

A UCP, apresentada por Gustav Karner em 1993, é baseada na APF, mas tem suas peculiaridades. Tais como ser aplicada a projetos orientados a objetos, ser baseada nos casos de uso e fornecer uma estimativa na fase inicial do projeto. Pelo fato de ser uma métrica fundamentada nos casos de uso, que não são padronizados, para a UCP ainda não existe um padrão, pois é fortemente influenciada pelo analista que escreve as especificações de casos de uso e, que dependendo de sua experiência, pode levar a erros na contagem de Pontos de Caso de Uso.

As derivações das métricas APF e UCP podem ser aplicadas e alteradas (quando as equipes de desenvolvimento tiverem experiência com o processo de estimativas) de acordo com os projetos de software da organização, pois as especificidades serão respeitadas e as métricas serão adequadas ao ambiente do sistema, atendendo ao caráter minimalista e peculiar de cada projeto.

Neste trabalho foram feitas as estimativas de tamanho e de esforço seguindo as especificações das métricas APF e UCP devido a facilidade, simplicidade e eficiência dos métodos. A análise comparativa possibilitou a verificação de que há uma diferença no número de pontos de função e no número de Pontos de Caso de Uso e, especificamente para este estudo de caso (o "Módulo de Despachos"), o resultado da UCP foi o tempo real de execução do projeto. Não se pode afirmar, contudo, que a APF ou a UCP seja melhor ou menos confiável, pois essa análise deve ser feita com um número maior de testes.

As métricas, como ditas na seção 2.3, para serem de qualidade devem satisfazer alguns critérios. As métricas aplicadas nesta monografia podem ser consideradas significativas, pois os resultados agregam informação útil; ainda não são repetíveis, pois foram aplicadas apenas uma vez, portanto não há uma base sólida para esta afirmação; são reproduzíveis, pois os resultados obtidos devem ser os mesmos para diferentes avaliadores; são imparciais, ou seja, foram escolhidas criteriosamente, e são validadas, pois os valores apresentam confiabilidade da informação.

O estudo bibliográfico realizado e a aplicação das métricas a um estudo de caso possibilitaram a confirmação de que o uso de métricas auxilia e promove a qualidade de software, pois quando o esforço, o custo e o prazo são medidos inicialmente e são coerentes com a realidade, se ganha credibilidade, tem-se um produto estável e em conformidade com os requisitos do usuário, portanto o processo organizacional de uma empresa que aplica uma boa prática (como as métricas de software) difere da empresa que não investe em nenhuma técnica ou uma metodologia.

5.1 Limitações

Citam-se duas principais dificuldades encontradas durante o trabalho: informações escassas de projetos que de fato aplicam/aplicaram as métricas (APF ou UCP) e pouca bibliografia sobre o método Pontos de Caso de Uso, fato que condicionou o referencial teórico a um conjunto pequeno de autores.

5.2 Trabalhos Futuros

Um dos trabalhos que pode ser desenvolvido futuramente é a aplicação da APF com uma de suas variações, como a Análise de Pontos por Função Estendida, para obter uma contagem mais específica e um melhor resultado para o projeto.

Uma outra proposta é a confecção de *templates* de casos de uso para melhor estimar o método UCP, que possibilitem uma especificação mais precisa e mais intuitiva desses artefatos. Para tal, convém pesquisar as ferramentas disponíveis no mercado que permitem a especificação e gerência de casos de uso.

As métricas auxiliam no planejamento, na execução e no controle das atividades da gestão de projetos. Sendo assim, sugere-se a aplicação das métricas por parte das empresas que ainda não investem em métodos ou metodologias de qualidade de software. Após implantar tais estratégias, deve-se realizar uma avaliação dos resultados, com vistas a mensurar o desempenho obtido com a inserção dessas na realidade da empresa.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ANDRADE, E. L. P. **Pontos de Casos de Uso e Pontos de Função na gestão de estimativa de tamanho de projetos de software orientado a objetos**. 2004. 143 p. Dissertação de Mestrado (Mestrado em Gestão do Conhecimento e Tecnologia da Informação da Universidade Católica de Brasília). Universidade Católica de Brasília, 2004.

DIAS, R. **Análise por pontos de função: uma Técnica para Dimensionamento de Sistemas de Informações**. RESI. Revista Eletrônica de Sistemas de Informação, v. II, p. 1/3-1, 2003.

FERRARI, S. **Proposta de metodologia para controle de qualidade em uma fábrica de software**. 2007. 302 p. Tese de Doutorado (Programa de Pós-graduação em Engenharia de Produção) – Universidade Federal de Santa Catarina, Florianópolis, 2007.

ISO/IEC, ISO 14143-1. ***Software measurement***. *Functional size measurement. Definition of concepts.*1998.

ISO/IEC, ISO 15504. ***Software process assessment – part 1****: concepts and introductory guide*. 1998.

ISO/IEC, ISO 19761. **COSMIC-FFP**. *A functional size measurement method*. 2002.

ISO/IEC, ISO 20926. ***Function point counting practices manual***. 2002.

KARNER, G. ***Resource Estimation for Objectory Projects Objective Systems.*** Objective Systems SF AB (copyright owned by Rational/IBM), 1993.

KOSCIANSKI, A., SOARES, M. S. **Qualidade de Software**. 2. ed. São Paulo: Novatec, 2007.

LÓPEZ, P. A. P**. CoCoMo II Um modelo para estimativa de custos de Projetos de Software**. 2005. 98 p. Monografia. Universidade do Vale do Rio dos Sinos, 2005.

MONTEIRO, T. C. **Pontos de Caso de Uso Técnicos (TUCP):** Uma Extensão da UCP. 2005. 213 p. Dissertação de Mestrado (Mestrado em Informática Aplicada). Universidade de Fortaleza, 2005.

MPS.Br, 2008. Guia MPS.Br. Disponível em: www.softex.br/mpsBr. Acesso: 10 novembro 2008.

PEREIRA, R. ; TAIT, T. F. C. ; TRINDADE, D. F. G. ; CASTRO, C. Y. H. ; SILVA, J. V. . **Estimativas de Software: O Estudo de uma Aplicação Prática Utilizando a Técnica de *Use Case Points*.** In: [ERIPR] - Escola Regional de Informática - PR,

2007, Guarapuava. [ERIPR] - Escola Regional de Informática - PR, 2007. v. XIV. p. 24-35.

PRESSMAN, R. S. **Engenharia de Software**. 5.ed. São Paulo: McGraw Hill, 2002.

RIBU, K. ***Estimating Object-Oriented Software Projects with Use Cases***. Dissertação de Mestrado. Departament of Informatic, University of Oslo. Norway. Novembro, 2001.

SERPRO. **O que você pode esperar de uma Fábrica de Software**. Disponível em: http://www.serpro.gov.br/. Acesso em: 15 jan. 2009.

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

SOUZA, A. G. **Análise de Pontos de Função Estendida**: Métrica de Software baseada na abordagem das dimensões tecnológica e ambiental/contextual. Dissertação de Mestrado (Mestrado Interdisciplinar em Modelagem Computacional). Fundação Visconde de Cairu, 2006.

SWEBOK Guide. ***IEEE Swebok Group***, 2004. Disponível em: <http://www.swebok.org/pdfformat.html> Acesso em: 25 abr. 2008.

VASQUEZ, E.C.;SIMÕES, G.S.; ALBERT, R.M.. **Análise de Pontos de Função**. 7. ed. São Paulo: Érica, 2007.

VIEIRA, E. L. **Uso do conceito de passos obrigatórios para aprimorar o processo de contagem do método "Pontos de Caso de Uso".** 2007. 88 p. Dissertação de Mestrado (Mestrado em Ciência da Computação). Universidade Federal de Santa Catarina, 2007.

Apêndice A: Descrição do Projeto "Módulo de Despachos"

Sistema DocFácil
"Módulo de Despachos"

Introdução

O propósito desse documento é coletar, analisar e definir necessidades e características do **Sistema DocFácil**, focando nas potencialidades requisitadas pelos *stakeholders* e usuários-alvo justificando o seu desenvolvimento. Os detalhes de como o sistema atinge essas necessidades serão descritos nos casos de uso e especificações suplementares.

Escopo

O sistema deverá ser utilizado no Tribunal Regional do Trabalho na 23ª Região podendo também ser, futuramente, adotado por outros órgãos com necessidades semelhantes, sobretudo outras regionais. O escopo deste sistema compreende:

- Ambiente para criação, edição e exclusão de despachos;
- Assinatura Eletrônica e publicação de despachos.

Definições, Acrônimos e Abreviações

Consulte o documento Glossário.

Referências

Nenhuma.

Posicionamento

Oportunidade

A criação, edição e exclusão de despacho permitirão uma redução do tempo gasto com o controle e edição de despachos devido ao grande número de documentos, conseqüentemente conferirá mais celeridade ao processo.

O controle dos despachos publicados permitirá um melhor acompanhamento e gerenciamento por parte dos magistrados e servidores.

Descrição do problema

O problema	Gerenciar e padronizar os modelos de documentos de despachos existentes. Fazer o controle dos modelos de documentos de Despachos.
Afeta	A todos os usuários (servidores e magistrados) responsáveis pela criação, edição e/ou exclusão de documentos do tipo despacho.
O seu impacto é	Os servidores terão mais facilidade durante edição, assinatura e publicação dos despachos. Haverá maior controle sobre os despachos editados, assinados e publicados.
Um solução ideal	Uma solução é o desenvolvimento de um sistema que facilite o acesso a

seria	esses documentos de maneira rápida e personalizada para cada usuário de maneira otimizada, além de fornecer o gerenciamento do ciclo de vida de cada documento e a manutenção de forma organizada.

Sentença de Posição do Produto

Para	Magistrados e servidores.
Que	Necessitarem criar, assinar e publicar despachos.
O **"Módulo de Despachos"**	É um software.
Que	Permite a edição, assinatura, publicação e controle dos despachos.
Diferente de	O processo atual em que o despacho é gerado por meio do uso de editores de textos de contexto geral.

Descrições dos usuários e envolvidos

Os usuários do sistema serão todos os servidores e magistrados que lidam com criação, edição e publicação de documentos.

Descrição dos Envolvidos

Nome	Representa	Responsabilidades
Servidores e Magistrados	São os usuários finais da aplicação. Aqueles que criarão novos documentos a partir de modelos previamente criados.	Editar, incluir, remover documentos no sistema de acordo com as devidas permissões.
Gestor de Modelo	Desenvolvedores de novos *layouts* de modelos de documentos.	Criar *layouts* dos modelos.
Analista/ Desenvolvedor	Desenvolvedores da DTI que farão a criação de novos modelos de acordo com as solicitações dos usuários.	Confeccionar, juntamente com o usuário, os artefatos necessários para o novo modelo (ações, fluxo e estados).

Ambiente do Usuário

O Sistema disponibilizará um ambiente que possibilitará a edição, assinatura e publicação do despacho.

O ambiente do usuário será formado por um sistema instalado em sua máquina com acesso a banco de dados via tecnologias de comunicação.

Necessidades dos principais usuários e envolvidos

Necessidade	Prioridade	Solução Atual	Soluções Propostas
Confeccionar despacho	Alta	Uso de editores de textos de contexto geral.	Um ambiente de desenvolvimento específico para confecção de despachos.
Gerar automaticamente os modelos de	Alta	Cada servidor tem o seu modelo, não há qualquer controle ou modelo	Fornecer ao servidor um conjunto de modelos padrões.

documentos		padrão.	
Acrescentar novos modelos de documentos	Alta	Não há qualquer forma de inserir novos modelos. A inserção é feita de forma manual.	O desenvolvedor vai adicionar novos modelos de acordo com as necessidades dos servidores.
Assinar despacho	Alta	Nenhuma	Permitir a assinatura digital do despacho.
Publicar despacho	Alta	Uso do Sistema DJE.	Permitir a publicação da despacho de forma automática após a sua assinatura.
Retornar despacho para edição	Alta	Nenhuma	Permitir um fluxo do despacho entre servidor e magistrado.
Imprimir despacho	Baixa	Impressão usando ferramentas de visualização de textos.	Impressão usando as ferramentas pré-existentes (visualizadores e editores de textos).

Visão Geral do Produto

O *DocFácil* - sistema de Módulos de Documentos tem como proposta a geração automatizada e o gerenciamento do ciclo de vida de documentos tais como: despachos, notificações, mandados, ofícios, cartas, votos, acórdãos, entre outros. Objetiva, sobretudo, a otimização, padronização, controle de autoria, assinatura digital, publicação na internet e no DJE (Diário de Justiça Eletrônico), quando necessário.

O sistema fará a integração com o *OpenOffice* para usar a API (*Application Programming Interface*) desse software a fim de de usar todas as funcionalidade dessa ferramenta de automação de escritório que, além de ser extremamente poderosa, é gratuita. A aplicação deverá prover uma interface gráfica com botões, barra de ferramentas e menus que interajam com o *OpenOffice*.

O "Módulo de Despachos" é parte do *DocFácil* e tem como principais características as seguintes funcionalidades:

- Criação, edição e exclusão de Despacho;
- Criação de Modelo de Despacho;
- Assinatura de Despacho; e
- Publicação de Despacho.

Licenciamento e Instalação

Não há necessidade, pois as ferramentas e subsistemas utilizados são de propriedade do TRT ou são *open source.*

Restrições

Nenhuma especificada.

Limites de Qualidade

Nenhum especificado.

Precedência e Prioridade

Nenhuma especificada.

Outros Requisitos do Produto

Requisitos do Sistema

O sistema deve permitir uma conexão segura e estar disponível na maior parte do tempo, principalmente durante o expediente do Tribunal.

Requisitos de Desempenho

Nenhum especificado.

Requisitos Ambientais

Nenhum especificado.

Requisitos da Documentação

Manual do Usuário

Não é necessário, pois o sistema deve ser suficientemente fácil de usar para que não haja necessidade de um manual do usuário.

Guias de Instalação, Configuração, Arquivo Leia-Me

Apenas os necessários para futuras manutenções e instalações.

Apêndice B: Descrição dos tipos de dados e registros das funcionalidades para o "Módulo de Despachos"

Descrição dos tipos de dados e registros das funcionalidades para o cálculo dos pontos de função não ajustados na APF

Para a contagem dos pontos de função não ajustados considera-se a complexidade de cinco fatores: Arquivos lógicos internos (ALI), Arquivos de interface externos (AIE), Entradas externas (EE), Saídas externas (SE) e Consulta externa (CE). Para detalhar as funcionalidades e facilitar o entendimento do leitor, neste anexo têm-se o detalhamento dos tipos de dados e dos arquivos referenciados do "Módulo de Despachos".

Login (EE)

- Tipos de Dados: código usuário, senha, nome, grupo-usuário.
- Arquivos referenciados: usuário_despacho.

Criação de Despacho (ALI)

- Tipos de Dados: id_documento, id_situação_documento, cdgusr, data_documento, data_inserção documento, conteúdo, mimetype.
- Arquivos referenciados: documento, versão_documento, situação documento, usuário_despacho, modelo.

Edição de Despacho (ALI)

- Tipos de Dados: id_documento, id_situação documento, cdgusr, data_documento, data_inserção documento, conteúdo, mimetype.
- Arquivos referenciados: documento, versão_documento, situação documento, usuário_despacho, modelo.

Criação de Despacho com base em Modelo (ALI)

- Tipos de Dados: id_documento, id_situação_documento, cdgusr, data_documento, data_inserção documento, conteúdo, mimetype, id_versão_documento, id_situação documento.
- Arquivos referenciados: documento, modelo, usuário_despacho, versão_documento, tipo_documento, documento_processual.

Exclusão de Despacho (ALI)

- Tipos de Dados: data_documento, id_situação documento, cdgusr, id_usuário.

- Arquivos referenciados: situação documento, documento, usuario_despacho.

Assinatura de Despacho (EE)

- Tipos de Dados: id_documento, id_situação documento, cdgusr, data_documento, data_inserção documento, conteúdo, mimetype, id_assinatura, nome_assinatura, título_assinatura, data_assinatura, cargo_assinatura.

- Arquivos referenciados: documento, versão_documento, situação documento, usuário_despacho, modelo, assinatura.

Publicação de Despacho (CE)

- Tipos de Dados: id_documento, id_situação documento, cdgusr, data_documento, data_inserção documento, conteúdo, id_versão_documento, mimetype, título_documento, destinatário_documento, emissor_documento, ementa_documento, publicável_DJE, publicável_boletim interno, id_edição, id_assinatura, nome_assinatura, título_assinatura, data_assinatura, cargo_assinatura.

- Arquivos referenciados: documento, versão_documento, situação documento, usuário_despacho, modelo, assinatura.

Impressão de Despacho (CE)

- Tipos de Dados: id_documento, id_situação documento, conteúdo, cdgusr.

- Arquivos referenciados: documento, situação documento, versão_documento, usuário.

Criação de Modelo de Despacho (ALI)

- Tipos de Dados: id_documento, cdgusr, id_usuário texto, id_usuário_despacho, id_usuário modelo, id_tipo documento.

- Arquivos referenciados: documento, documento_processual, modelo, usuário_despacho, despacho, tipo_de_documento, usuário modelo, texto.

Criação de Modelo de Despacho (CE)

- Tipos de Dados: id_documento, id_versão_documento, id_situação documento, descrição_situação documento, cdgusr.

- Arquivos referenciados: documento, usuário_despacho, tipo_de_documento, situação documento.

Consulta de Despacho Fechado (CE)

- Tipos de Dados: id_documento, id_versão_documento, id_situação documento, descrição_situação documento, usuário.

- Arquivos referenciados: documento, usuário_despacho, tipo_de_documento, situação documento.

Consulta de Despacho Aberto (CE)

- Tipos de Dados: id_documento, id_versão_documento, id_situação documento, descrição_situação documento, usuário.

- Arquivos referenciados: documento, usuário_despacho, tipo_de_documento, situação documento.

Consulta de Despacho Excluído (CE)

- Tipos de Dados: id_documento, id_versão_documento, id_situação documento, descrição_situação documento, usuário.

- Arquivos referenciados: documento, usuário_despacho, tipo_de_documento, situação documento.

Consulta de Despacho Modificado (CE)

- Tipos de Dados: id_documento, id_versão_documento, id_situação documento, descrição_situação documento, usuário.

- Arquivos referenciados: documento, usuário_despacho, tipo_de_documento, situação documento.

Consulta de Despacho Disponibilizado na *WEB* (CE)

- Tipos de Dados: id_documento, id_versão_documento, id_situação documento, descrição_situação documento, usuário.

- Arquivos referenciados: documento, usuário_despacho, tipo_de_documento, situação documento.

Consulta de Despacho Publicado (CE)

- Tipos de Dados: id_documento, id_versão_documento, id_situação documento, descrição_situação documento, usuário.

- Arquivos referenciados: documento, usuário_despacho, tipo_de_documento, situação documento.

Consulta de Despacho do Processo (CE)

- Tipos de Dados: id_documento, id_versão_documento, id_situação documento, descrição_situação documento, usuário, nmrproc, anoprc, cdgprc, regprc, seqprc, digprc, fls.

- Arquivos referenciados: documento, usuário_despacho, tipo_de_documento, situação documento, documento processual, processo.

Apêndice C: Especificação dos principais casos de uso para o "Módulo de Despachos"

Caso de Uso:	***Criar Layout de Despacho***	*29/11/2008*

1. Descrição Resumida

Este caso de uso ocorre quando o gestor de *layout* deseja criar um *layout* para um documento despacho.

2. Status

Inicial.

3. Atores

Gestor de *layout* (Primário).

4. Acionadores

O gestor abre o gerenciador de *layout* e escolhe a opção desejada para o *layout* de Documento.

5. Fluxo de Eventos

5.1. Fluxo Básico

1. O gestor de *layout* cria um novo *layout* de Documento Despacho associando-o obrigatoriamente a um modelo de documento previamente criado.
2. O sistema exibe uma lista que contém os despachos elaborados e padrões de modelos de despachos, permitindo a visualização dos despachos ao gestor. O sistema também exibe um diretório que contém as famílias dos *layout*s para cada tipo grupo de usuário e uma lista de textos pré-definidos para serem inseridos no documento.
3. O gestor seleciona um modelo e indica para que grupo de servidores será o *layout* de despacho.
4. O sistema exibe um editor de texto (a tela inicial do OpenOffice) para dar início a confecção do *layout*.
5. O gestor confecciona o *layout* para o modelo de documento, ou seja, introduz as margens, cabeçalhos, rodapés, define as fontes, etc. Insere também os textos padronizados e faz as alterações necessárias até que o documento esteja em conformidade com os requisitos especificados para determinado grupo.
6. O gestor termina a edição do despacho e solicita que o sistema o salve.
7. O sistema salva o despacho.

5.2. **Fluxos Alternativos**:

1. O gestor de *layout* informa ao sistema que deseja criar um *layout* para um modelo de despacho que já possui um *layout* definido.
    1. O sistema avisa ao servidor (por meio de mensagem) que já existe um *layout* para o modelo de despacho selecionado e pede confirmação.

2. Caso o gestor selecione a opção inserir novo layout, o antigo será excluído. Caso o gestor mantenha a versão do layout do modelo, o fluxo segue o passo 2 do fluxo básico.

6. Pré-Condições

Deve existir um modelo de documento para que um novo *layout* seja criado.

7. Pós-Condições

7.1. Pós-Condição (ões) de sucesso

Um *layout* foi criado ou alterado.

7.2. Pós-Condição (ões) de falha

O sistema não permite salvar dois tipos de *layout* para um mesmo documento, caso o gestor tente salvar o um segundo *layout* ele irá excluir o antigo.

8. Pontos de Extensão

Não há.

Caso de Uso:	***Configurar Modelo de Despacho***	*29/11/2008*

1. Descrição Resumida

Esse caso de uso descreve as ações do gestor de modelo no *DócFácil*, de maneira específica o documento Despacho.

2. Status

Inicial.

3. Atores

Gestor de Modelo.

4. Acionadores

O gestor de modelo seleciona as operações utilizando o sistema DócFácil.

5. Fluxo de Eventos

5.1. Fluxo Básico

a. O gestor de modelo cria e nomeia um novo modelo relativo a um despacho.

b. O sistema apresenta a opção de inserir os estados, que são as várias possibilidades de tramitação de um documento, ou seja, o seu ciclo de vida. Pode-se dizer que o estado é a situação corrente do tipo de documento (salvo ou em edição por exemplo).

c. O gestor de modelo insere os estados pertencentes ao modelo de documento indicando a classe (um arquivo java) que implementa o conjunto de estados possíveis para o estado em questão. Permite-se também alterar ou excluir um estado existente ou modificar a ordem dos estados (caso esteja alterando um modelo).

d. O sistema passa para a próxima etapa que é a composição do conjunto de componentes para o modelo.

e. O gestor insere os componentes (ou exclui, caso queira alterar um modelo existente) completando campos como descrição, nome, tipo, ação, estado e ícone para cada componente. Nesse momento, o gestor irá definir a permissão ou não para um determinado grupo de usuários, habilitando ou desabilitando os componentes.

f. Após completar a etapa anterior, o gestor monta uma lista de documentos, a qual terá as especificidades de cada modelo de acordo com as permissões dos usuários.

g. O Gestor de modelo poderá definir um conjunto de textos que poderão ser usados tanto pelo Gestor de *Layout* como pelos usuários finais. Os textos-padrão são circundados pelos sinais de << e >> e são convertidos para determinados conteúdos de acordo com o documento que está sendo elaborado.

h. O Gestor de modelo finaliza o modelo e solicita que o sistema o salve.

i. O sistema salva o modelo de despacho.

5.2. Fluxos Alternativos

5.2.1. O Gestor de modelo informa ao sistema que deseja criar um despacho para um modelo que já tem um documento do mesmo tipo.

a. O sistema informa ao gestor (por meio de uma mensagem) sobre a existência de um documento e questiona se o gestor deseja substituí-lo.

1. Caso o gestor escolha a opção manter o documento despacho e retorna ao passo h do fluxo básico.
2. Caso o gestor escolha a opção excluir o modelo atual e criar um novo, o sistema exclui o modelo existente e retorna ao passo a do fluxo básico.

6. Pré-Condições

O modelo a ser criado deve ser para um tipo de documento pertencente ao fluxo de trabalho ou tramitação dos processos do tribunal.

7. Pós-Condições

7.1. Pós-Condições de sucesso

O modelo de documento foi criado ou alterado e finalizado com sucesso.

As informações foram completadas corretamente.

O modelo de documento corresponde ao planejado pelo gestor.

7.2. Pós-Condição de falha

O aplicativo não poderá finalizar dois modelos de um mesmo documento para um mesmo grupo de usuários, caso o gestor tente fazer isso o segundo modelo será sobrescrito.

8. Ponto de Extensão

Nenhum.

Caso de Uso:	*Montar Documento de Despacho*	*29/11/2008*

1. Descrição Resumida

Esse caso de uso descreve e controla operações para criar ou montar um Documento de Despacho.

2. Status

Inicial.

3. Atores

Usuário (primário), Sistema (secundário).

4. Fluxo de Eventos

4.1. Fluxo Básico

a. O usuário informa ao sistema que deseja criar um despacho.
b. O sistema exibe uma lista de modelos de documentos de acordo com a permissão do usuário.
c. O usuário cria um novo documento de despacho associando-o a um *layout* previamente criado.
d. O sistema exibe um editor de texto com o *layout* de documento escolhido inicialmente.
e. O usuário edita as partes necessárias refatorando os campos, inserindo as informações exigidas e referentes ao despacho.
f. O servidor termina a edição do despacho e solicita que o sistema o salve.
g. O sistema salva o documento despacho.

4.2. **Fluxos Alternativos**:

a. O usuário informa ao sistema que deseja criar um documento do tipo despacho para um determinado modelo.
    1. O sistema informa ao usuário que o mesmo não tem permissão para criar um despacho para os modelos existentes.
    2. O usuário escolhe um outro modelo de despacho e o fluxo retorna ao passo a do fluxo básico.

5. Pré-Condições

Deve existir um *layout* de documento de despacho para se criar um documento de despacho.

6. Pós-Condições

6.1. Pós-Condições de sucesso

O documento de despacho foi criado e poderá ser alterado/salvo/impresso/assinado seguindo os demais casos de uso.

7. Ponto de Extensão

Anexo A: Determinação das Características Gerais do Sistema referentes à APF

Características Gerais do Sistema

Neste Anexo são identificadas as diretrizes que determinam o valor de influência, para cada característica geral do sistema conforme as normas do IFPUG (*International Function Point Users' Group*).

As descrições foram extraídas de Vasquez, Soares e Albert (2006) e de Souza (2004). Para cada característica será dado um quadro com o valor de influência referente às possíveis opções.

1. Comunicação de dados: descreve o nível em que a aplicação se comunica com o processador (VASQUEZ; SOARES; ALBERT, 2006).

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

2. Processamento de dados distribuído: descreve o nível em que a aplicação transfere os dados de seus componentes (VASQUEZ; SOARES; ALBERT, 2006).

0: A aplicação não participa da 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 (planilhas eletrônicas, por exemplo).
2: Dados são preparados para transferência, então processados em outro componente do sistema.
3: Processamento distribuído e transferência de dados são feitos on-line e em apenas uma direção.
4: Processamento distribuído e transferência de dados são feitos on-line e em ambas as direções.
5: O processamento de funções é executado dinamicamente no componente mais apropriado do sistema.

3. Desempenho: descreve o nível de influência do tempo de resposta e taxa de transações no desenvolvimento da aplicação (VASQUEZ; SOARES; ALBERT, 2006).

0: O usuário não estabeleceu requisito para o desempenho.
1: Requisitos de projeto e desempenho foram estabelecidos e revisados, mas nenhuma ação especial foi tomada.
2: O tempo de resposta será crítico durante as horas de pico. Não há projeto especial para utilização de CPU. O intervalo de tempo limite do processamento é o dia seguinte.
3: O tempo de resposta será crítico durante as horas de pico. Não há projeto especial para utilização de CPU. O intervalo de tempo limite do processamento é crítico.
4: Além do descrito no item 3, os requisitos de desempenho estabelecidos pelo usuário são rigorosos o bastante para requerer tarefas de análise de desempenho na fase de análise e projeto da aplicação.
5: Adicionalmente, ferramentas de análise de performance devem ser utilizadas nas fases de projeto, desenvolvimento e/ou implementação para que os requisitos do usuário sejam atendidos.

4. Configuração altamente utilizada: descreve o nível de influência dos recursos computacionais necessários para a aplicação (VASQUEZ; SOARES; ALBERT, 2006).

0: Não existem restrições operacionais explícitas ou implícitas nos requisitos.
1: Existem restrições operacionais, mas são menos restritivas do que aplicações típicas. Nenhum esforço será necessário para suplantar as restrições.
2: Algumas considerações sobre tempo e segurança são especificadas.
3: Existem requisitos especiais de processador para uma parte específica da aplicação.
4: Restrições operacionais explícitas requerem atenção especial a nível de processador central ou processador dedicado para executar a aplicação.
5: Adicionalmente, existem limitações nos componentes distribuídos da aplicação.

5. Volume de transações: descreve o nível de influência para o projeto, desenvolvimento, instalação e suporte da aplicação (VASQUEZ; SOARES; ALBERT, 2006).

0: Não existem restrições operacionais explícitas ou implícitas nos requisitos.
1: Existem restrições operacionais, mas são menos restritivas do que aplicações típicas. Nenhum esforço será necessário para suplantar as restrições.

2: Algumas considerações sobre tempo e segurança são especificadas.
3: Existem requisitos especiais de processador para uma parte específica da aplicação.
4: Restrições operacionais explícitas requerem atenção especial a nível de processador central ou processador dedicado para executar a aplicação.
5: Adicionalmente, existem limitações nos componentes distribuídos da aplicação.

6. Entrada de dados *on-line*: descreve o nível de influência das transações interativas (VASQUEZ; SOARES; ALBERT, 2006).

0: Todas as operações 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*.

7. Eficiência do usuário final: nível de considerações sobre fatores humanos e facilidade de uso do usuário final da aplicação. As funções interativas que influenciam no projeto incluem (VASQUEZ; SOARES; ALBERT, 2006):

- Facilidades para navegação (teclas de função, saltos e geração dinâmica de menus);
- Menus;
- Ajuda *on-line* e documentação;
- Movimento automático do cursor;
- Paginação;
- Impressão remota por meio de transações *on-line;*
- Teclas de função pré-definidas;
- Execução de trabalhos *batch* a partir de transações on-line;
- Seleção de dados da tela pela movimentação do cursor;
- Uso intensivo de vídeo reverso, brilho, cores e outros recursos;
- Documentação impressa das transações;
- Interface para *mouse*;

- Janelas *pop-ups*;
- Número mínimo de telas para execução de funções de negócio;
- Suporte a dois idiomas (contar como quatro itens);
- Suporte a múltiplos idiomas, ou seja, suportar mais de dois idiomas (contar como seis itens).

O valor de influência é dado da seguinte forma:

0: Nenhum dos itens anteriores.
1: De 1 (um) a 3 (três) itens anteriores.
2: De 4 (quatro) a 5 (cinco) itens anteriores.
3: Apresenta 6 ou mais dos itens acima, mas não há requisitos do usuário relacionados à eficiência.
4: Apresenta 6 ou mais dos itens acima, e os requisitos estabelecidos para eficiência do usuário são fortes o suficiente para que a fase de projeto da aplicação inclua fatores humanos como: minimizar a digitação, maximizar os valores padrões, utilizar modelos.
5: Apresenta 6 ou mais dos itens acima, e os requisitos estabelecidos para eficiência do usuário são rigorosos o suficiente para que seja necessário o uso de ferramentas e processos especiais para demonstrar que os objetivos de eficiência foram alcançados.

8. Atualização *on-line*: nível de atualização *on-line* dos arquivos lógicos internos (VASQUEZ; SOARES; ALBERT, 2006).

0: Nenhuma atualização.
1: Atualização *on-line* de 1 (um) a 3 (três) arquivos. O volume de atualização é baixo e a recuperação de dados é fácil.
2: Atualização *on-line* de 4 (quatro) ou mais arquivos. O volume de atualização é baixo e a recuperação de dados é fácil.
3: A atualização da maioria dos arquivos lógicos internos é *on-line.*
4: Adicionalmente, a proteção contra perda de dados é essencial e foi especificamente projetada e codificada no sistema.
5: Adicionalmente, altos volumes de dados trazem considerações sobre custo para o processo de recuperação. Exigem procedimentos de recuperação totalmente automatizados com a mínima intervenção do operador.

9. Complexidade de Processamento: nível de processamento lógico que influencia o desenvolvimento da aplicação. Os componentes a seguir devem ser avaliados (VASQUEZ; SOARES; ALBERT, 2006):

- Controle sensível e/ou processamento específico de segurança da aplicação.

- Processamento lógico extensivo;
- Processamento matemático extensivo;
- Muito processamento de exceções, resultando em transações incompletas que necessitam de novo processamento;
- Processamento complexo para manipular múltiplas possibilidades de entrada e saída.

O valor de influência é dado da seguinte forma:

0: Nenhum dos itens anteriores.
1: Qualquer um dos itens anteriores.
2: Quaisquer dois dos itens anteriores.
3: Quaisquer três dos itens anteriores.
4: Quaisquer quatro dos itens anteriores.
5: Todos os itens anteriores.

10. Reusabilidade: nível de especificamente projetar, desenvolver e suportar o código da aplicação para ser reutilizado em outras aplicações (VASQUEZ; SOARES; ALBERT, 2006).

0: Não há código reutilizável.
1: Há código reutilizável na própria aplicação.
2: Menos de 10% (dez) da aplicação considera as necessidades de mais de um usuário.
3: 10% (dez) ou mais da aplicação considera as necessidades de mais de um usuário.
4: A aplicação será projetada e documentada para facilitar a reutilização de código e a aplicação será customizada pelo usuário a nível do código fonte.
5: A aplicação será projetada e documentada para facilitar a reutilização de código e a aplicação será customizada pelo usuário por meio de manutenção de parâmetros.

11. Facilidade de instalação: nível de conversão de ambientes anteriores que influenciam o desenvolvimento da aplicação (VASQUEZ; SOARES; ALBERT, 2006).

0: Nenhuma consideração especial foi feita pelo usuário e nenhum procedimento especial foi requerido para a implantação.
1: Nenhuma consideração especial foi feita pelo usuário, mas um procedimento especial é requerido para a implantação.

2: Requisitos de instalação e conversão de dados foram definidos pelo usuário, e roteiros de instalação e conversão de dados devem ser preparados e testados. O impacto da conversão de dados no projeto não é considerado importante.

3: Requisitos de instalação e conversão de dados foram definidos pelo usuário, e roteiros de instalação e conversão de dados devem ser preparados e testados. O impacto da conversão de dados no projeto é considerado importante.

4: Além do item 2 (dois), ferramentas automatizadas de implantação e conversão de dados foram preparadas e testadas.

5: Além do item 3 (três), ferramentas automatizadas de implantação e conversão de dados foram preparadas e testadas.

12. Facilidade de operação: descreve o nível em que a aplicação atende aos aspectos operacionais (VASQUEZ; SOARES; ALBERT, 2006).

0: Nenhuma consideração especial sobre facilidade operacional, além dos procedimentos normais.

1 - 4: Um, alguns, ou todos os itens seguintes aplicam a aplicação. Avalie todos que se aplicam. Cada item tem o valor de um ponto, exceto quando apontar em contrário.

6. Procedimentos de inicialização, backup e recuperação devem ser preparados, mas a intervenção do operador é necessária;
7. Procedimentos eficientes de inicialização, backup e recuperação devem ser preparados, mas nenhuma intervenção do operador é necessária (contar como dois itens);
8. A aplicação minimizará a operação de montagem de fitas magnéticas;
9. A aplicação minimizará a necessidade de manuseio de formulários.

5: A aplicação será projetada para não precisar de intervenção do operador no seu funcionamento normal. Apenas a inicialização e parada do sistema ficam a cargo do operador. A recuperação automática de erros será uma característica da aplicação.

13. Múltiplos locais: nível de desenvolvimento da aplicação para utilização em múltiplos locais e organizações (VASQUEZ; SOARES; ALBERT, 2006).

0: Os requisitos do usuário não consideram a necessidade de instalar a aplicação em mais de um local.

1: Necessidade de instalação em múltiplos locais foi considerada no projeto do sistema e a aplicação foi projetada para operar somente em ambientes *idênticos* de hardware e software.

2: Necessidade de instalação em múltiplos locais foi considerada no projeto do sistema e a aplicação foi projetada para operar somente em ambientes *similares* de hardware e software.

3: Necessidade de instalação em múltiplos locais foi considerada no projeto do sistema e a aplicação foi projetada para operar somente em ambientes *diferentes* de hardware e software.
4: Além das etapas 1 (um) e 2 (dois), um plano de documentação e manutenção deve ser elaborado e testado para suportar a aplicação em múltiplos locais.
5: Além da etapa 3 (três), um plano de documentação e manutenção deve ser elaborado e testado para suportar a aplicação em múltiplos locais.

14. Facilidade de mudanças: de desenvolvimento da aplicação em relação a facilidades de modificação da lógica de processamento e estrutura de dados (SOUZA, 2006). As seguintes características são avaliadas:

- São fornecidos mecanismos de consulta flexível, que permitem manipulação de pedidos simples.
- São fornecidos mecanismos de consulta flexível, que permitem manipulação de pedidos de complexidade média.
- São fornecidos mecanismos de consulta flexível, que permitem manipulação de pedidos de complexidade complexa.
- Dados de controle de negócio são mantidos pelo usuário por meio de processos interativos, mas as alterações serão efetivadas somente no próximo dia útil.
- Dados de controle de negócio são mantidos pelo usuário por meio de processos interativos, mas as alterações têm efeito imediato (conte como dois pontos).

O valor de influência é dado da seguinte forma:

0: Nenhum dos itens anteriores.
1: Qualquer um dos itens anteriores.
2: Quaisquer dois dos itens anteriores.
3: Quaisquer três dos itens anteriores.
4: Quaisquer quatro dos itens anteriores.
5: Todos os itens anteriores.