CENTRO PAULA SOUZA
FACULDADE DE TECNOLOGIA DE FRANCA
“Dr. THOMAZ NOVELINO”

MARIANA VACARI MUNARI SAMPAIO

Estudo de Caso: A utilização de métricas no processo de desenvolvimento de Software

FRANCA/SP

2013

MARIANA VACARI MUNARI SAMPAIO

Estudo de Caso: A utilização de métricas no processo de desenvolvimento de Software

Trabalho de Graduação apresentado à Faculdade de Tecnologia “Dr. Thomaz Novelino” - Fatec Franca, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Carlos Alberto Lucas

FRANCA/SP

2013

MARIANA VACARI MUNARI SAMPAIO

Estudo de Caso: A utilização de métricas no processo de desenvolvimento de Software

Trabalho de Graduação apresentado à Faculdade de Tecnologia "Dr. Thomaz Novelino" - Fatec Franca, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Carlos Alberto Lucas

Trabalho avaliado e aprovado pela seguinte Banca Examinadora:

Orientador: Prof. Me. Carlos Alberto Lucas
Assinatura: _____

Examinador 1: Prof. Esp. Fernando Martins
Assinatura: _____

Examinador 2: Profa. Dra. Jaqueline Brigladori Pugliesi
Assinatura: _____

Franca, 10 de Dezembro de 2013

Dedico este trabalho à minha família e a
todos os estudantes e profissionais da área de
Tecnologia da Informação.

AGRADECIMENTOS

Primeiramente a Deus, por ter proporcionado paciência e saúde para superar todos os desafios nesta missão de formação profissional.

À minha mãe Daniela e ao meu pai Danilo que sempre apoiaram em tudo, e que sempre disseram as palavras de incentivo que precisei.

Aos meus amigos de turma que estavam sempre presentes na minha vida e que de certa forma, influenciaram na minha formação.

“O que não se mede não se gerencia”.
Peter Drucker

RESUMO

A tecnologia está a cada dia sendo implantada dentro das organizações com maior frequência e isso está ocorrendo naturalmente, devido à necessidade de se manter no mercado e de superar o nível dos seus concorrentes, ou também a percepção de agilidade da execução com qualidade (prazo e custo) dos processos. Para o gerenciamento de projetos, por exemplo, as organizações devem utilizar alguns métodos e tecnologias visando maximizar o seu controle de todos os processos, e em se tratando de desenvolvimento de software, um minucioso controle das tarefas, como análise de requisitos, análise de portabilidade, modelagem, documentação, testes, implementação e implantação, sempre valorizando todos os requisitos funcionais ou não funcionais, que são extremamente valiosos para o sistema. Como a realidade é: o que não se mede não se gerencia, a gestão de projetos é aplicada à todo segmento de negócio. Sendo assim, para o desenvolvimento de software, é muito útil para as empresas contratantes e contratadas, pois com a utilização de métricas no início do projeto, logo após a conclusão da modelagem do banco de dados, facilitará a efetivação da proposta comercial entre estas empresas, pois com base nesta modelagem, os cálculos de prazo e custo serão reais, independentemente se o método utilizado para cálculo está baseado em Pontos por Função ou em Casos de Uso. A aplicação destas métricas irá contribuir para a transparência da aplicação (prazo e custo) de acordo com a necessidade do cliente, ou seja, conforme as especificações do sistema. O objetivo da utilização de um método é proporcionar para as partes envolvidas, um recurso que represente com exatidão, qual será a estimativa de prazo e custo do projeto, neste caso um sistema de Controle Bancário. A partir disto, este trabalho tem como objetivo demonstrar a transparência que a implantação e utilização de métricas podem trazer para as empresas envolvidas. Portanto, para atingir este objetivo foi necessário primeiramente desenvolver a pesquisa bibliográfica através de livros e artigos, para definir termos e explicar alguns processos que pertencem aos métodos de métricas e para melhor entendimento do projeto que foi desenvolvido em seguida, através de uma análise e comparação de dois métodos de cálculos de métricas, os Pontos por Função ou através de Casos de Uso.

Palavras-chave: Métricas – Enterprise – Gerenciamento – Estimativa – Prazo – Custo – Pontos de Função – Casos de Uso

ABSTRACT

The technology is being deployed every day in organizations with greater frequency and it is naturally occurring, due to the need to remain on the market and to overcome the level of its competitors, or even the perception of agility execution quality (deadline and cost) processes . For the management of projects , for example , organizations must use some methods and technologies to maximize their control of all processes , and is working software development , a thorough control of tasks, such as requirements analysis , portability analysis , modeling , documentation, testing , implementation and deployment , always valuing all functional or non-functional requirements that are extremely valuable to the system . Because the reality is, what is not measured does not manage, project management is applied to every business segment. Thus , for the development of software , it is very useful for contractors and contractors, because with the use of metrics early in the project , after the completion of the modeling database will facilitate the execution of the business proposal between these companies because based on this modeling, the calculations of deadline and cost will be real , regardless of the method used for the calculation is based on Function Points or use Case application of these metrics will contribute to the transparency of the application (deadline and cost) according to customer's requirement, as the system specifications . The purpose of the use is to provide a method for the parties involved, a feature that represents exactly what will be the estimated deadline and cost of the project , in this case a system of banking control . From this, this paper aims to demonstrate the transparency that the deployment and use of metrics can bring to the companies involved . Therefore, to achieve this goal it was necessary to first develop a literature search through books and articles, to define terms and explain some processes pertaining to methods and metrics for better understanding of the project that was developed and then through an analysis and comparison two methods of calculating metrics, Function Points or through Use-Case.

Keywords : Metrics - Enterprise - Management - Estimating - Deadline - Cost - Function Points - Use Cases

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS E QUADROS

LISTA DE ABREVIATURAS E SIGLAS

EA - *Enterprise Architect*
ISO - *International Organization for Standardization*
MPS-BR - Melhoria do Processo de Software Brasileiro
SOFTEX - Associação para Promoção da Excelência do Software Brasileiro
MCT - Ministério da Ciência e Tecnologia
FINEP - Financiadora de Estudos e Projetos
BID - Banco Interamericano de Desenvolvimento
RUP - *Rational Unified Process*
FP- Pontos de Função
IFPUG - *International Function Point User Group*
CMMI - *Capability Maturity Model Integration*
MPS-BR – Melhoria de Processos do Software Brasileiro

SUMÁRIO

INTRODUÇÃO

As organizações estão se rendendo ao uso da tecnologia e dos aplicativos para realizarem os seus processos, visando a fidelização e a satisfação dos clientes. Esta ação proporcionou o aumento de interesse em estudos na área de métricas e de métodos para controle, desenvolvimento e execução dos processos.

Em um projeto há vários processos, que devem ser realizados com qualidade, profissionalismo e baseados em normatizações que proporcionarão a confiabilidade e a garantia da conquista de melhores resultados.

Sendo assim, deve-se sempre lembrar a frase de Peter Drucker (consultor administrativo) que dizia: “O que não se mede, não se gerencia”. É partindo desta premissa que este trabalho de graduação será fundamentado.

Na área da Tecnologia da Informação e Comunicação há órgãos que regem as normatizações para controle de todos os processos, independentemente da área tecnológica, por exemplo, CMMI (*Capability Maturity Model Integration*) que significa Modelo Integrado de Maturidade em Capacitação, que têm a função de gerenciar a certificação de software na esfera internacional, e há também a certificação nacional MPS-BR (Melhoria do Processo de Software Brasileiro) .

Além destes órgãos, há também ótimas bibliografias que apresentam conceitos e técnicas para o gerenciamento de um projeto. É fato, que a medição deve ser realizada tendo como base principal um planejamento e um cronograma de acompanhamento.

Se referindo especificamente a projetos na área de tecnologia, é essencial uma minuciosa análise econômica, de requisitos e de portabilidade, que são fatores propulsores de sucesso em todos os projetos nesta área, e que validam a efetivação contratual, uma vez que a fase de estipular prazo e custo é essencial, pois contribuirão para a redução de riscos e gerenciamento de todas as fases do projeto.

Vale destacar que outras engenharias também têm suas formas de medir, como por exemplo, a medida de massa, volume, velocidade e outras inclusas nas áreas matemáticas. Porém, a engenharia de software trata a medição de uma forma mais indireta e vê essa quantificação como forma de avaliar o cálculo de estimativas, qualidade, produtividade e controle de um projeto (software).

Em pesquisa divulgada em Janeiro de 2013 pela *Maturity Project* [MPCM,2013], apontou que somente 49,7% dos projetos são concluídos com êxito.

É com base nestes percentuais que surgiram alguns questionamentos que darão sustentação para este trabalho de graduação.

Ressaltando que a área específica de ação deste trabalho será a de desenvolvimento de software, através de parametrizações dos processos de medição de prazo e de custo, baseada em uma modelagem de banco de dados.

No transcorrer deste trabalho, serão pesquisados, analisados e utilizados alguns recursos oferecidos pelos órgãos reguladores, já citados anteriormente, que subsidiarão nos esclarecimentos dos seguintes questionamentos:

Qual é a importância da utilização de métodos para controle, desenvolvimento e execução dos processos? Qual é a importância da utilização de métricas? Como deve ser feito esse processo de medição? Como podemos garantir que a métrica está de acordo com os processos que a empresa necessita?

Neste trabalho, serão exploradas duas das principais técnicas de métricas do mercado. Usa-se a métrica por pontos de função (MOF) e a métrica por casos de uso (utilizando o *Enterprise Architect*), que é um programa que contempla o desenvolvimento do ciclo de vida de um produto e têm entre suas funções, estimar prazo e custo a partir de uma modelagem e de diagramas.

No capítulo 1, será tratado o Gerenciamento de projetos e métricas e como as empresas e organizações tratam a questão de prazo e custo de projetos. No capítulo 2, serão tratadas as métricas e sua aplicação no desenvolvimento de sistemas.

No capítulo 3, serão aplicadas as métricas em um projeto, e demonstrado o seu desenvolvimento. No capítulo 4, será tratado o estudo de caso. Este é que vai nos possibilitar uma melhor análise e caracterização das métricas, bem como sua aplicação no decorrer de um projeto. No capítulo 5, serão demonstrados os resultados e comparações do uso de métricas por casos de uso e métricas orientadas à função.

1 GERENCIAMENTO DE PROJETOS E MÉTRICAS

De acordo com [MICHELAZZO,2006], as empresas, mesmo que de pequeno porte, buscam desenvolver seus produtos baseando-se em prazos, padrões e custos sem perder a qualidade. As organizações têm uma alta dependência em relação à tecnologia e buscam a satisfação do cliente através de produtos sofisticados e muitas das vezes desenvolvidos em prazos curtos. Isso aumentou o interesse em estudos na área de métricas e de métodos de desenvolvimento de projetos.

O desenvolvimento de software por sua vez é composto por diversos processos, dentre eles o gerenciamento. Tipicamente, o processo de gerência de projetos envolve três atividades: planejamento, acompanhamento e encerramento. No planejamento é feito um plano organizado e detalhado de como o projeto vai ser conduzido. O acompanhamento é o processo de refinação das atividades que são feitas no projeto. Deve-se ajustar e tomar maior conhecimento dos elementos que o compõe. Nesta fase é feito o processo de estimativa de prazo e custo, detalhando valores e esforços necessários. Já na parte de encerramento é feito a análise crítica do resultado final, da entrega, dos riscos e aprendizados obtidos com eles.

Na maneira mais indicada e considerada correta de gerenciar um projeto, é definido que a medição deve ser feita para que no planejamento se tenha um cronograma de acompanhamento, e considerações importantes de todas as tarefas do início ao fim. Uma vez que a fase de estipular prazo e custo esteja feita, ficará mais fácil de medir riscos e de gerir todas as fases do projeto.

Todo processo de gestão é feito de acordo com PMBOK(*Project Management Body of Knowledge*), e abrange 9 áreas do conhecimento: integração, escopo, custo, prazo, contratos, comunicações, riscos, recursos humanos e qualidade. Todas essas áreas possuem processos que se interagem e uma vez que o cliente solicite alterações de requisitos, pode acarretar em atrasos e aumento de custos. Por esse motivo e outros, muitos projetos são deixados de lado, cancelados e excedem em características informadas ao cliente anteriormente.

Em pesquisa divulgada em Janeiro de 2013 pela *Maturity Project* [MPCM,2013], apontou que 49,7% dos projetos são concluídos com êxito. Dos 50,3% restantes, 35,2% são concluídos com estouro de prazo e 15,1% dos projetos

iniciados são fechados e não concluídos. Dos 35,2%, 28% apresenta estouro de custo.

Também de acordo com os estudos realizados por [CAMPÊLO,2002], ficou claro que uma das maiores dificuldades encontradas para realizar o gerenciamento de projetos de software é saber a dimensão do que está sendo feito e que deve ser gerenciado. Junto com os processos de gerenciamento têm-se inúmeras questões pertinentes, mas as que se destacam na área e que ganham uma atenção maior dos gerentes são prazo e custo dos projetos. Antes mesmo de iniciar as atividades técnicas de um projeto, deve-se realizar as estimativas. O prazo de entrega se torna um objeto muito importante, bem como o custo que tem total aprovação ou não do cliente.

A atividade de estimar tem toda a aprovação dos gerentes e deve ser feita para que o gerenciamento seja possível de ser feito e é importante lembrar e destacar sobre sua importância no processo de gerencia de projetos.

Para o acompanhamento, andamento e finalização é necessário medir o progresso e comparar com o que foi estimado. Não se pode controlar e gerenciar aquilo que não se consegue medir, e não se consegue acesso às boas estimativas com dados anteriores e irrelevantes. Dados e estimativas devem ser feitos de forma criteriosa e que dê visibilidade de estudo do projeto em todo seu desenvolvimento. De acordo com Howard Rubin [CAMPÊLO,2002], esse processo de aceitação e implantação não é nada fácil. Mais de 80% das iniciativas de se implantar métricas de software nas organizações fracassam em menos de 02 anos. Então como se podem estipular os custos e prazos de um sistema, bem como gerenciar os mesmos? Como deve ser feito esse processo e como as métricas podem ser usadas?

As métricas ajudam na coleta de informações para que sejam usadas de forma a fornecer indicadores relevantes. O uso delas aproxima a comunicação e faz com que ela seja efetiva e integrada com os vários interessados no desenvolvimento, sendo que as técnicas de medição são como alavancas para melhoria da qualidade de desenvolvimento não só de sistemas, mas também de produtos que são medidos para avaliar a qualidade, enquanto que os processos são medidos para melhorias e estimativas que podem deduzir, e não prever.

A falta das métricas, por sua vez, pode levar as falhas no acompanhamento do projeto e gerenciamento. Isso porque uma vez que tenha problemas e que não

está sendo percebido por aqueles que podem direcionar esforços para a sua solução, pode levar ao cancelamento, atraso ou até fechamento de um projeto. De acordo com [TRAVASSOS,2008] em uma avaliação de empresas que adotaram as boas práticas de engenharia de software, resultaram em uma compreensão qualitativa das variáveis de desempenho como custo, prazo, produtividade, qualidade, satisfação do cliente e retorno do investimento.

Várias áreas também usam métricas para avaliação, continuidade e gerenciamento do negócio ou do projeto. Com o desenvolvimento de sistemas não poderia ser diferente, e deve-se medir e estimar.

Podem ser ajudados com várias organizações, inclusive a ISO (*International Organization for Standardization*), que cria padrões de normas para que a empresa desenvolvedora atenda as necessidades do cliente. Um exemplo bem comum e citado em vários livros é a ISO 9000 que foi criada com o intuito de garantir a qualidade para qualquer setor da indústria. Outro clássico relacionado à padronização para processos é a MPS-BR (Melhoria de Processo de Software Brasileiro) que é coordenada pela SOFTEX (Associação para Promoção da Excelência do Software Brasileiro) e conta ainda com a ajuda do MCT (Ministério da Ciência e Tecnologia), da FINEP (Financiadora de Estudos e Projetos) e do BID (Banco Interamericano de Desenvolvimento). A MPS-BR, padroniza, bem como adéqua os processos de software, e como tem compatibilidade com outros padrões, acaba sendo aceito internacionalmente como padrão de qualidade.

Por esse motivo verifica-se que métrica é algo válido e de extrema importância para as empresas que desejam se manter no mercado, sem perder a qualidade e sem dispersar recursos e tempo, e otimizando os gastos para o cliente (usuário final).

2 MÉTRICAS E SUA APLICAÇÃO NO DESENVOLVIMENTO DE SISTEMAS

No processo de gerenciamento, é feito a parte de informar o valor final do software e também medir algumas propriedades. Para isso usa-se métricas que podem ser baseadas principalmente em função ou em tamanho. Mas o que é métrica? Desde quando faz-se isso e como deve ser feita para obter o melhor resultado?

Em [MICHAELIS,2009], tem-se que métrica é definida como sendo:

> "1 .Arte de medir versos e que estuda os elementos de que eles são constituídos; 2. A estrutura de um verso em relação à medida."

A métrica atribui uma medida a algo do mundo real com objetivo de manipular e obter algumas conclusões através do mesmo. Deve-se lembrar que outras engenharias também têm suas formas de medir, como por exemplo, a medida de massa, volume, velocidade e outras inclusas nas áreas matemáticas. Porém a engenharia de software trata a medição de uma forma mais indireta e vê essa quantificação como forma de avaliar o cálculo de estimativas, qualidade, produtividade e controlar um projeto.

Como exemplo, vamos pegar um objeto simples e analisar: a empresa A oferece um produto X por R$ 80,00 reais, enquanto a empresa B fornece um produto Y por R$ 50,00 reais. Nesse caso pode-se concluir que a empresa B tem uma maior aceitação quando fazemos uma "métrica de custos".

Várias outras áreas da indústria e de serviços tem o uso de métricas. Em setores como a medicina, arquitetura, contabilidade e engenharias, as técnicas de medir são amplamente utilizadas das mais diversas formas e com os mais distintos objetivos.

Em [VASCONCELOS,2005] diz que traduzindo para o conceito em software, é possível verificar que métrica é uma medição das características ou propriedades de um programa, processo ou recurso a partir do qual decisões podem ser tomadas. Pode-se estimar o esforço, o custo e as atividades que serão necessárias para a realização do projeto.

O resultado da métrica segundo [CAMPÊLO,2002] ajuda em 4 pontos em um projeto: entender, avaliar, controlar e prever. Avaliar com o objetivo de obter padrões

e metas, entender para verificar o comportamento de processos, controlar processos, produtos e serviços, e prever valores e recursos em geral.

Além de ser usada para estimar dados, pode determinar uma base de estimativa posterior, acompanhar o progresso do projeto, determinar a complexidade, ajudar a determinar o estado aceitável de qualidade, analisar defeitos e riscos e ainda validar as melhores práticas e aprendizados.

Os objetivos de medir foram definidos também no RUP(Rational Unified Process) [RUP, 2000] e se dividem em:

- **Objetivos da organização:** conhecimento da organização sobre seus custos, recursos gastos, tempo de desenvolvimento e qualidade de produtos.
- **Objetivos dos projetos:** todos os projetos devem ser entregues no prazo estipulado e com o valor combinado. A métrica ajuda no processo de conhecimento de desempenho quanto aos requisitos funcionais e não funcionais, bem como permite à organização avaliar os resultados obtendo feedbacks positivos e/ou negativos.
- **Objetivos técnicos:** métricas enquanto medidas de necessidade técnica são caracterizadas estruturalmente e por seu comportamento. A frequência de mudanças de requisitos, completudes, validações e estabilidade de uma análise de requisitos são essenciais e, caso haja muitas falhas, a métrica ajuda no processo de tomada de decisão.

Na gestão de projetos, tem-se alguns outros objetivos que se integram com citados anteriormente. Pode-se destacar, por exemplo:

- Melhorar a qualidade do planejamento do projeto;
- Reduzir custos;
- Reduzir retrabalho;
- Melhorar processos de desenvolvimento;
- Aperfeiçoar métodos de gerenciamento, entre outros;

Temos que ter em mente que não é porque foi feito uma medição uma vez e que obtivemos um resultado, (mesmo que positivo), que devemos parar de estabelecer novas diretrizes e metas. O processo de medição deve ser contínuo e cada vez que preciso, deve-se alterar sua análise, até mesmo para auxiliar numa

nova decisão. Deve fornecer uma melhoria contínua e quantificar a qualidade e produtividade.

É importante lembrar que temos que medir processos e não pessoas, pois pessoas têm formas e métodos diferentes de programar, e para um mesmo trabalho podem ter visões diferentes. Deve-se lembrar também que um processo depende não somente da decisão de um gerente de projetos e sim de toda a organização e política do cliente e da empresa que está desenvolvendo. Não somente o gerente tem autonomia para colocar em discussão as mudanças do projeto, mas todos os membros da equipe devem dar opiniões que ajudem a melhoria de um sistema.

2.1 FORMAS DE MÉTRICAS

[CAMPOS,2013] define que, em geral, uma métrica pode ser definida como indireta ou direta.

A forma direta considera, por exemplo, o custo e o esforço no desenvolvimento, a quantidade de linhas de código e o total de erros ou instabilidades registradas, sendo que estes atributos são fáceis de serem observados. Já a qualidade, a segurança e a funcionalidade do software são medidas indiretas, pois cada pessoa pode considerar a medida como aceitável ou não, e pode ter formas de analisar diferentes umas das outras. Além disso, essas medidas indiretas podem ser calculadas com base em outras métricas específicas.

Dentre as principais características, deve-se considerar que a métrica deve ser válida e quantificar o que se quer medir. Deve ser confiável e dar um mesmo resultado quando informada as mesmas condições, facilitando o entendimento, cálculo e interpretação. Elas também devem mostrar dois indicadores básicos e que são muito importantes. São eles:

Produtividade: esta que depende de inúmeros fatores como a dimensão e complexidade dos sistemas a desenvolver, bem como suas especificidades. Ela deve mostrar que reduz o tempo gasto para executar um determinado serviço. As métricas de produtividade se concentram na saída do processo de Engenharia de Software avaliando justamente se obteve uma melhora nos mesmos.

Qualidade: a qualidade pode ser medida através de algumas características como a capacidade de correção, eficiência no processamento, confiabilidade,

portabilidade e facilidade de manutenção. Obter esses dados quantitativos e relativos a essas características é fundamental para melhorias no processo de desenvolvimento. Essa medida de qualidade indica o quanto o software atende aos requisitos e as necessidades definidas pelo usuário.

As duas propriedades estão sempre ligadas, visto que a produtividade não significa que deve realizar tudo em maior quantidade, mas sim que deve ser feito com mais qualidade de serviço.

Pode-se adicionar várias outras variáveis para calcular e estimar as métricas, isso dependendo da situação e das especificações da empresa.

Alguns autores definem que as métricas podem ser definidas também quanto à sua privacidade, segundo [IGLESIAS,2013] elas de dividem em:

- **Privadas para o indivíduo:** como exemplo, podemos citar as "taxas de defeitos e erros". Muitas empresas se negam a dividir os erros até com pessoas da própria organização. Os erros são cometidos por todos nós, e normalmente não gostamos de tê-los apontados, e nem divulgados.
- **Privada para o grupo do projeto:** um exemplo seria a "taxa de defeitos por módulo ou função", que a equipe de desenvolvimento poderá ficar sabendo e inclusive deve ter consciência para que possa melhorar e diminuir a taxa.
- **Pública para a organização:** "taxa de melhoria", prazo e custo de um projeto. Isso implica também numa possível cobrança de todo o corpo trabalhador da empresa, para que verifique o andamento do trabalho de todos os envolvidos.

A seguir, pode-se verificar a evolução das métricas e algumas das técnicas mais utilizadas.

2.2 A EVOLUÇÃO DAS MÉTRICAS

Métricas estabelecem medições nas diversas áreas de atuação de uma empresa. O conceito começou a ser utilizado visando uma melhoria das técnicas e da produtividade do software.

Nessa parte do estudo, serão abordadas as principais métricas de software, quando começaram a ser utilizadas e o quão amplo é a forma de se medir.

2.2.1 MÉTRICA DE HALSTEAD

Nos anos 70, a medição para estimar custos e outros atributos era feita com base no código fonte escrito. Dentre as principais formas de se fazer, destacavam-se a proposta de Halstead e McCabe [VASCONCELOS,2005].

A métrica segundo Maurice Halstead da Universidade de Purdue, foi introduzida em 1977 e era calculada para obter quantitativamente os resultados esperados. Usando o número de operadores e operandos de um módulo de um programa era possível fazer as medições. Consideramos que um operador é uma função, e algo que produz uma ação, já um operando é uma constante ou uma variável.

Ele calculava que o número de informações que um módulo poderia apresentar era baseado no seu comprimento e vocabulário. A fórmula utilizada era:

Volume = Comprimento *log2Vocabulário

onde o comprimento é somatório de instância entre operadores e operandos usados; e vocabulário é a somatória de operadores e operandos diferentes.

Devido ter como parâmetro o código, mais usado quando era preciso dar manutenção. Porém usava-se também para avaliar códigos durante o processo de desenvolvimento, estimando a qualidade e complexidade.

Esta métrica foi aplicada no cálculo de esforço de desenvolvimento de programas, manutenibilidade e na melhoria de complexidade durante a fase de testes.

A vantagem do uso dessa técnica é que ela funciona independente da linguagem de programação utilizada, mas ela está baseada na sintaxe dos programas e dessa forma não considera o conteúdo em si. Outra desvantagem é que ela é de difícil compreensão de usuários e por ser feita somente depois do código pronto, não é possível estimar o custo e prazo antes da implementação, somente depois.

Mas como tudo tem suas vantagens e desvantagens, a métrica recebeu críticas, principalmente por mensurar a complexidade textual, diferente do que ocorre na complexidade ciclomática, onde se mensura a complexidade estrutural e lógica. Essa métrica que utiliza complexidade ciclomática é a que vai ser discutida logo abaixo

2.2.2 MÉTRICA DE McCABE

Em 1976 Thomas McCabe definiu uma nova forma de analisar um programa. Sua técnica foi denominada de complexidade ciclomática [VASCONCELOS,2005].

Ela é considerada uma métrica importante que estima a complexidade lógica do trecho de um programa e analisa caminhos de execução independentes. McCabe usava um diagrama para representar o fluxo de execução e demostrar onde cada bloco do programa passava. Dessa forma, um programa sequencial e sem estruturas de controle como if e else teria sua complexidade calculada como 1, já que somente existe um caminho válido e que o programa pode ter uma saída [VASCONCELOS,2005].

A fórmula usada no cálculo é a seguinte:

M = (E – N) +P

Onde:

M – complexidade ciclomática

E – setas escritas

N – número de nós

P – quantidade de componentes ligados

O grafo é representado por nós e por arcos, onde os nós representam uma ou mais instruções sequenciais e os arcos indicam o sentido do fluxo entre várias instruções. Na figura 1 temos um exemplo:

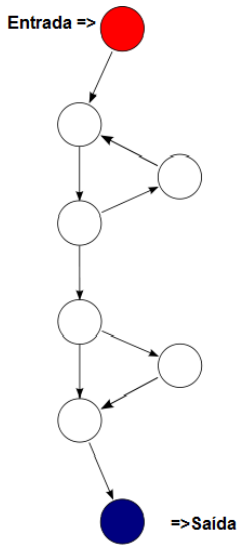

Figura 1 - Fluxo de Informação.

Fonte: VASCONCELOS, 2005

Analisando a figura 1 temos que E vale 9, P é 1 e N é 8 e assim colocando na fórmula chegamos ao seguinte resultado:

M = E –N+P

M = (9 – 8) +1

M = 2

Portanto, a complexidade ciclomática deste bloco é 2.

O uso desta métrica foi desestimulado por conta de alguns pesquisadores concluírem que a produtividade diminuía quando feita de forma não linear e adicionando decisões. Quanto mais pontos colocados de forma correta para minimizar erros e aumentar a resposta de um programa, seria melhor. Além disso, a construção e análise de grafos como os exemplificados na figura 1, gastava-se tempo.

Porém, a métrica apresenta vantagem quanto a identificar e minimizar códigos não estruturados e eliminar lógica redundante, isso também por meio de análise dos grafos.

A métrica ainda assim era usada nas seguintes áreas e com objetivos definidos:

I) Análise de risco no desenvolvimento de código – coleta de medidas sobre o código desenvolvido para antecipar e informar possíveis riscos.

II) Análise de risco para manutenção – medidas de complexidade antes e depois de uma alteração, podendo assim monitorar os riscos de mudança.

III) Planejamento de testes – investigação dos pontos do programa com maior complexidade e que precisam de maior esforço durante a fase de teste.

2.2.3 MÉTRICAS DE YIN E WINCHESTER

Em 1978 surgiram as métricas de complexidade de sistemas. A primeira delas foi publicada em 1978 por Yin e Winchester. Eles criaram e dividiram as métricas em dois grupos : primárias e secundárias [VASCONCELOS,2005].

As métricas primárias eram expressas extraindo-se valores específicos do design do sistema (como o número de arcos e o número de módulos). Já as

secundárias proviam um indicador sobre os principais módulos do sistema ou sobre tabelas do banco de dados.

2.2.4 MÉTRICAS NO CICLO DE VIDA

Na década de 80 surgiu a medição do software no início e durante o ciclo de vida. Dessa forma podia-se estimar o custo e o esforço necessário para se criar uma aplicação.

A vantagem é que sempre que preciso poderia medir uma fase do software. Podia-se também usar a métrica desde o início na fase de especificação, prevendo erros futuros, instabilidades e dificuldades que seriam encontradas. A desvantagem é que sempre alguém teria que parar o que estava fazendo para fazer os cálculos e chegar a uma decisão. Por esse motivo foi deixando de ser usada e foi substituída por outros tipos de métrica.

2.2.5 MÉTRICA DE HENRY E KAFURA

Em 1981, Henry e Kafura demostram outra métrica. Esta por sua vez era determinada: a complexidade do código do procedimento e a complexidade entre as conexões dos procedimentos no ambiente.

A métrica de Henry e Kafura era mais detalhada do que a métrica de Yin e Winchester e observava todo o fluxo de informação. Uma vantagem do método é de ter a possibilidade de ser completamente automatizado.

Após isso se destacaram conjuntos de pesquisas sobre a aplicação das métricas apresentadas além da explicação sobre o que seriam as métricas orientadas a função e tamanho. Estas que hoje são as mais usadas pelos desenvolvedores e gerentes de projeto, por ser mais fácil de ser calculada e compreendida [TIMOTEO,2008].

2.2.6 MÉTRICA ORIENTADA A TAMANHO

As métricas orientadas a tamanho são consideradas medidas diretas do software, que são passíveis de manter um padrão de medição. Basta a empresa ter um controle pequeno dos processos que faz e atribuir alguns valores para eles. Podemos considerar, por exemplo, esforço, custo com materiais, número de erros ou instabilidades ocorridas durante o desenvolvimento, entre outras, como medidas

para cálculo de métricas orientadas a tamanho. A empresa, porém, não deve se basear em código fonte de outros projetos que já foram concluídos, pois a medida depende de cada linguagem de programação e depende do critério de contagem, visto que existem empresas que não contam, por exemplo, linhas de comentário e linhas em branco, acreditando que elas não influenciam na execução do programas [5CQualiBR,2013]. Dessa forma, uma linha de código é considerada como sendo qualquer linha do texto de um programa, exceto comentários e linhas em branco e sem levar em consideração o número de comando ou fragmentos de um comando de uma linha. A definição inclui também o cabeçalho, declarações e comandos executáveis [BOAVENTURA,2001].

Pode-se analisar os dados brutos de um sistema e criar a métrica otimizando a qualidade, o esforço, o custo e principalmente aumentando a produtividade.

Podemos criar uma tabela geral para documentar e analisar cada valor de software. Na tabela 1 segue um exemplo:

Tabela 1- Exemplo de projetos orientados à tamanho

Projeto	Esforço	R$	Kloc/loc	Páginas	Erros	Pessoas
Projeto Loja	1h	0	9	1	0	1
Projeto Mercado	5 dias	200	5000	120	32	2

Fonte:O autor

Faremos agora alguns cálculos, levando em consideração o exemplo “Projeto Mercado” da tabela descrita:

Cálculo da produtividade: esse cálculo é feito dividindo a quantidade de linhas de código pela quantidade de dias gastos. Dessa forma temos que:

Kloc / esforço = 5000 / 5 = 1000.

Ou seja, o programador é capaz de criar (nesse caso) 1000 linhas de código por dia.

Cálculo da qualidade: cálculo feito dividindo a quantidade de erros pela quantidade de linhas de código. No nosso caso temos:

Erros / Kloc = 32 / 5000 = 0,0064

Cálculo do custo: dividindo o preço da hora de desenvolvimento pela quantidade de linhas de código. Dessa forma descobrimos qual valor por linha escrita. Temos então:

R$ / Kloc = 200 / 5000 = 0,04 por dia

Cálculo da produtividade por pessoa: é feito através da análise de quantas linhas de código cada programador escreveu. Basta dividir a quantidade de linhas por quantidade de envolvidos. No nosso exemplo temos:

Kloc / pessoas = 5000 / 2 = 2500

Cálculo do valor à ser pago por pessoa: é baseado na quantidade de linha de código produzida, dividida pela quantidade de pessoas e multiplicada pelo custo. Temos então:

(Kloc / pessoas) * custo = (5000 / 2) * 0,04 = R$ 100,00

Verifica-se então que cada programador deve receber R$ 100,00 por suas 2500 linhas de código cada um.

O que deixa um pouco vago é que se o contratante não tem nenhuma base ou referência de Kloc(mil linhas de código), de esforço e de custo, como poderá escolher um programador adequado para sua empresa e, mais que isso, como informar a um cliente que foi até a sua empresa? Qual valor ele terá que pagar pelo software e quanto tempo será gasto para isso?

Para isso temos a Métrica Orientada à Função.

2.2.7 MÉTRICAS ORIENTADAS À FUNÇÃO

Uma proposta da empresa IBM na década de 70 aboliu a contagem de linhas de código e começou a se concentrar na funcionalidade do software. A medida foi feita a pedido de um grupo de usuários, que tinha o trabalho de identificar as variáveis críticas que determinavam a produtividade. Eles então acabaram descobrindo que poderiam avaliar medindo o valor das funções executadas, ao invés de usar o volume, o tamanho ou a complexidade do código dos programas.

Logo em 1979, Allan Albrecht motivado por pesquisas, introduziu uma técnica de avaliação conhecida como Pontos por Função. Essa métrica é baseada na visão do usuário, e é independente da linguagem de programação e plataforma utilizadas, permitindo assim, calcular o esforço para a programação e auxiliar o usuário a melhorar a avaliação de um projeto [UFPUG, 2004].

No Brasil, a MPS-BR juntamente com a SISP(Sistema de Administração dos Recursos de Informação e Informática) fez uma Instrução Normativa SLTI de nº4, de 12 de novembro de 2010, que recomenda o uso das métricas em contratos de projetos de software, restringindo o uso da medida de esforço homem-hora

[SISP,2012]. Em outras regras e acordos, visa-se também: uso de Pontos de Função como forma de pagamento dos serviços de desenvolvimento e manutenção de sistemas em vez de converter em horas, adoção da técnica de medição por pontos de função sem ajustes pelas aplicações diferenciando as funcionalidades daqueles relativos a supressões ou alterações das mesmas, não vinculando a métrica de tamanho funcional com a de esforço e nem usar o fator de ajuste de qualquer forma.

Dentre os principais objetivos da métrica, podemos destacar:

- Medir o que foi solicitado e recebido do usuário;
- Medir independente da linguagem de programação utilizada;
- Prover uma forma de medição para apoiar a análise de produtividade e qualidade;
- Prover uma forma de estimar o tamanho do software;
- Prover um fator de normalização para comparação de software;

2.2.8 PROCESSOS DE CONTAGEM DE PONTOS DE FUNÇÃO

Assim que definido o banco de dados com todas as tabelas e campos, é possível iniciar a contagem de pontos de função e dessa forma estimar o custo e prazo final do sistema. A contagem e elaboração de uma métrica seguem algumas etapas que são descritas:

1ª etapa: Identificação e enumeração de funções

Na 1ª etapa é feito o levantamento e a identificação das funções da aplicação, bem como a contagem das mesmas. Devemos contar o número de entradas do usuário, número de saídas do usuário, número de consultas, número de arquivos e número de interfaces externas.

É importante definir o que seriam cada uma dessas funções da aplicação e como podemos avaliar a quantidade de cada uma delas. Dessa forma, definimos que a entrada é definida como sendo um processo elementar que processa os dados ou informações de controle. A intenção primária de uma entrada de dados é alterar ou manter um arquivo lógico interno. A saída é definida como um processo de envio de dados ou informações de controle para fora da aplicação. A principal

intenção é apresentar informações aos usuários através de lógica de processamento que não seja somente de recuperação de dados.

A questão externa de processos de procedimentos fica a cargo da consulta que é o envio de dados ou informações para fora da aplicação. A intenção primária de uma consulta é apresentar informações ao usuário através de recuperação de dados ou de informações de controle. Vale destacar que a consulta não contém nenhuma fórmula ou cálculo matemático e nem altera o comportamento do sistema.

Já os arquivos são definidos como sendo um grupo de dados ou informações de controle, relacionados logicamente e que são mantidos dentro da aplicação. Essa questão de manter os dados agrupados e armazenados num processo do programa é a principal intenção e definição de uso de um arquivo.

E por último, e não menos importante, o número de consultas externas. É definido como sendo um grupo de dados ou informações de controle logicamente relacionados, que são identificados pelo usuário e referenciado pela aplicação, bem como mantido dentro da fronteira externa de outra aplicação.

Para cada função citada, determinamos se é de nível simples, médio ou complexo de acordo com as tabelas a seguir.

Contagem e pontuação das entradas:

Tabela 2 - Contagem de entradas

CAMPOS ARQUIVOS	1-4 Itens	5-15 itens	16 ou mais itens
0 ou 1 tipo de arquivo	SIMPLES	SIMPLES	MÉDIO
2 tipos de arquivos	SIMPLES	MÉDIO	COMPLEXO
3 ou mais tipos de arquivo	MÉDIO	COMPLEXO	COMPLEXO

Fonte: O autor (De acordo com a ISO/IEC 9126)

Contagem e pontuação das saídas

Obs.:(deve-se contar mais uma tabela com todos os campos):

Tabela 3 - Contagem de saídas

CAMPOS ARQUIVOS	1-5 Itens	6-19 itens	20 ou mais itens
0 ou 1 tipo de arquivo	SIMPLES	SIMPLES	MÉDIO
2 ou 3 tipos de arquivos	SIMPLES	MÉDIO	COMPLEXO
4 ou mais tipos de arquivo	MÉDIO	COMPLEXO	COMPLEXO

Fonte: O autor (De acordo com a ISO/IEC 9126)

Contagem e pontuação de consultas:

Obs.:(deve-se contar mais uma tabela com todos os campos)

Tabela 4 - Contagem de consultas

CAMPOS ARQUIVOS	1-4 Itens	5-15 itens	16 ou mais itens
0 ou 1 tipo de arquivo	SIMPLES	SIMPLES	MÉDIO
2 tipos de arquivos	SIMPLES	MÉDIO	COMPLEXO
3 ou mais tipos de arquivo	MÉDIO	COMPLEXO	COMPLEXO

Fonte: O autor (De acordo com a ISO/IEC 9126)

Contagem e pontuação de arquivos:

Tabela 5 - Contagem de arquivos

CAMPOS REGISTROS	1-19 Itens	20-50 itens	1 ou mais itens
1 tipo de registro	SIMPLES	SIMPLES	MÉDIO
2 á 5 tipos de registro	SIMPLES	MÉDIO	COMPLEXO
6 ou mais tipos de registro	MÉDIO	COMPLEXO	COMPLEXO

Fonte: O autor (De acordo com a ISO/IEC 9126)

Contagem e pontuação de interfaces:

Obs. (deve-se contar mais uma tabela com todos os campos)

Tabela 6 - Contagem de interfaces

CAMPOS REGISTROS	1-19 Itens	20-50 itens	1 ou mais itens
1 tipo de registro	SIMPLES	SIMPLES	MÉDIO
2 á 5 tipos de registros	SIMPLES	MÉDIO	COMPLEXO
6 ou mais tipos de registros	MÉDIO	COMPLEXO	COMPLEXO

Fonte: O autor (De acordo com a ISO/IEC 9126)

2ª etapa: Obter pontos de função brutos (FP'S BRUTO)

Com base nos dados preenchidos e contabilizados na 1ª fase, seguimos com a contagem de pontos de função indicados na seguinte tabela:

Tabela 7 - Contagem de pontos de função brutos (FP's Brutos)

TIPO DE FUNÇÃO	COMPLEXIDADE FUNCIONAL	COMPLEXIDADE TOTAL	RESULTADO
ENTRADAS	___SIMPLES	x 3	
	___MÉDIA	x 4	
	___COMPLEXA	x 6	
SAÍDAS	___SIMPLES	x 4	
	___MÉDIA	x 5	
	___COMPLEXA	x 7	
CONSULTAS	___SIMPLES	x 3	
	___MÉDIA	x 4	
	___COMPLEXA	x 6	
ARQUIVOS	___SIMPLES	x 7	
	___MÉDIA	x 10	
	___COMPLEXA	x 15	
INTERFACES	___SIMPLES	x 5	
	___MÉDIA	x 7	
	___COMPLEXA	x 10	

Fonte: O autor (De acordo com a ISO/IEC 9126)

3ª etapa: Estimar o nível de influência do sistema

Para cada característica é dado um valor e realizado um total de contagem e qualificação do nível de influencia do sistema. Cada característica e pontuação dada está descrita no anexo I deste documento.

Tabela 8 - Características analisadas para estimativa do nível de influência

Comunicação de dados	
Performance	
Volume de transações	
Eficiência do usuário final	
Processamento complexo	
Facilidade de implantação	
Múltiplos locais	
Processamento distribuído	
Utilização de equipamentos	
Entrada de dados on line	
Atualização on line	
Reutilização de código	
Facilidade operacional	
Facilidade de mudanças	
TOTAL	

Fonte: O autor (De acordo com a ISO/IEC 9126)

O nível de influência deve ser multiplicado pela taxa de valor real obedecendo a seguinte formula:

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

Onde NI é o Nível de Influência e FA é o Fator de Ajuste.

4ª etapa: Multiplicação de pontos de função brutos pelo fator de ajuste

Para calcular o fator de ajuste, temos uma base de 14 características que valem a discussão e permitem avaliar a funcionalidade da aplicação. As características são as descritas no anexo I.

A técnica chamou a atenção das organizações e em 1986 foi formado um grupo internacional de usuários da técnica de pontos por função, chamado IFPUG (International Function Point User Group) destinado a implementar melhorias e disseminar a técnica.

Estas informações são disponibilizadas através do manual de práticas do IFPUG, cuja última versão 4.1 foi publicada em janeiro de 1999.

A técnica vem sendo melhorada seguindo uma evolução de versões, e neste caso, contagens realizadas com versões diferentes da técnica, não devem ser comparadas entre si. Cada estimativa feita, desde que em versões diferentes, não

deve ser levada em consideração quando comparada com uma versão mais antiga ou mais nova no manual prático.

2.2.9 MÉTRICAS POR CASOS DE USO

O uso de diagramas de casos de uso normalmente é feito na fase de levantamento e análise de requisitos, embora possa ser consultado durante todo o desenvolvimento e processo de modelagem. A linguagem apresentada nele é de simples interpretação e compreensão para facilitar o entendimento dos usuários. O diagrama de casos de uso identifica os atores que vão ter acesso ao sistema e as funcionalidades que o sistema irá disponibilizar aos atores [GUEDES,2009].

Devido à facilidade de entendimento e praticidade de uso, em 1993 Gustav Karner criou o modelo de métricas por casos de uso, que mais tarde foi incorporada em programas como o *Enterprise Architect* (EA) que iremos usar para a elaboração e análise de uma métrica.

Nesse tipo de métrica, a estimativa é feita conforme o modo que os usuários utilizam o sistema, bem como a complexidade das ações dos usuários e análise mais abstrata dos pontos de função.

O método começa com um caso de uso e depois é feita a análise parecida com a de Pontos de Função. No EA, o cálculo é feito através de uma funcionalidade que usa os parâmetros definidos na técnica e dos casos de uso definidos através de diagramas de Casos de Uso. Tal técnica permite a estimativa e dimensão de um sistema ainda na fase de levantamento de dados e descreve as funcionalidades do sistema de acordo com o uso por parte dos usuários.

Na forma manual de se fazer o cálculo, seguimos as seguintes etapas:

1ª etapa - Cálculo dos pesos dos atores do sistema

Trata-se da classificação dos envolvidos em cada caso de uso, sendo que a classificação e pontuação se dão de acordo com o peso dos atores do sistema juntamente com as quantidades desses atores.

Tabela 9 - Peso e qualificação dos atores do sistema

Tipo de Ator	Peso	Descrição
Simples	1	Sistema acessado através de uma API
Médio	2	Sistema interagindo através de um protocolo de comunicação como TCP/IP ou FTP
Complexo	3	Usuário interagindo com interface gráfica (stand-alone ou Web)

Fonte: O autor (De acordo com a ISO/IEC 9126)

2ª etapa: Cálculo do peso bruto dos casos de uso

Iniciamos fazendo o cálculo do peso bruto dos casos de uso dividindo-os em três níveis de complexidade de acordo com o número de transações envolvidas em seu processamento.

Tabela 10 - Peso e qualificação dos casos de uso

Tipo de Casos de Uso	Nº de transações	Peso
Simples	até 3	1
Médio	4 a 7	2
Complexo	7 ou mais	3

Fonte: O autor (De acordo com a ISO/IEC 9126)

Pode-se levar em consideração o número de classes envolvidas no processo e nesse caso o cálculo é como no caso anterior, porém somente é aplicável quando as entidades envolvidas em uma dada atividade já foram definidas. O peso total é o cálculo do peso dos atores e casos de uso:

$$PT = PA + PU$$

Onde PT é Peso Total, PA é o Peso dos Atores e PU é o Peso dos Casos de Uso.

3ª etapa: Cálculo do fator de ajuste

Essa etapa é bem parecida com o processo e cálculo do fator de ajuste na métrica por pontos de função.

É feita a análise de fatores técnicos abordando os requisitos funcionais, e os fatores do ambiente considerado os requisitos não funcionais. Os dois juntos geram multiplicadores distintos, que devem ser aplicados ao peso total calculado na etapa anterior.

Para cada requisito listado nas tabelas, deve ser atribuído um valor que determina a influência do requisito no sistema, que varia de 0 a 5. O valor 0 indica

que não tem nenhuma influência, 3 indica que há um influência moderada e 5 indica uma forte influência.

O fator de ajuste conforme a tabela 11, pontua em complexidade os requisitos:

Tabela 11 - Características analisadas para estimativa do nível de influência

REQUISITO	PESO
Sistema Distribuído	
Tempo de resposta	
Eficiência	
Processamento Complexo	
Código reusável	
Facilidade de instalação	
Facilidade de uso	
Portabilidade	
Facilidade de mudança	
Concorrência	
Recursos de segurança	
Acessível por terceiros	
Requer treinamento especial	

Fonte: O autor (De acordo com a ISO/IEC 9126)

Para completar o cálculo, seguimos a fórmula:

$$PFT = 0,6 + (0,01 * TF)$$

Onde PFT é o Peso de Fator Bruto e TF é o Total de Fatores ajustados.

Para os Fatores Ambientais, o nível de influência indica a disponibilidade de cada recurso no decorrer do projeto. Isso significa dizer que se um fator tem nível de influência alta deve ser atribuído um valor igual a 5 e que ele está presente no projeto como um todo, ao passo que atribuir um valor de influência zero indica que o mesmo não está presente no processo de desenvolvimento.

Para a pontuação seguimos a Tabela 12:

Tabela 12 - Características para estimativas do fator ambiental

REQUISITO	PESO
Familiaridade com RUP ou outro processo formal	
Experiência com a Aplicação em desenvolvimento	
Experiência em Orientação a Objetos	
Presença de analista experiente	
Motivação	
Requisitos estáveis	
Desenvolvedores em meio-expediente	
Linguagem de programação difícil	

Fonte: O autor

O fator ambiental é calculado pela seguinte fórmula:

EF = 1,4 + (-0,03 * EFactor)

Onde EFactor é a soma dos produtos do peso e dos fatores da tabela, e EF é o fator ambiental.

Segundo Karner, o tempo necessário para o desenvolvimento deve ser estimado calculando-se uma média de 20 horas de trabalho por Ponto de Caso de Uso (UCP), mas existem casos que demonstram uma variação entre 15 e 30 horas por ponto. Isso irá depender da empresa e dos desenvolvedores.

Como uma forma de automatizar o cálculo de uma métrica por casos de uso, o programa Enterprise Architect gera a estimativa de prazo e custo para o desenvolvimento, e isso será descrito e demonstrado nos capítulos seguintes.

3 DESENVOLVIMENTO E APLICAÇÃO DAS MÉTRICAS

Para a modelagem do caso de uso e execução das métricas usamos o programa *Enterprise Architect.*

O *Enterprise Architect* é um software para representação de sistemas baseado em UML(*Unified Modeling Language*). Tem como objetivo principal gerar documentos que auxiliem na construção e manutenção de softwares. Ele abrange desde requisitos iniciais, até modelos de testes e manutenção. Possui versões pagas, porém a que foi utilizada para desenvolvimento e aplicação das métricas foi uma versão de teste, disponível para uso em 30 dias.

Usuários consideram o E.A uma ferramenta robusta, porém de fácil utilização e que oferece um ótimo custo-benefício. Ela traz suporte total ao ciclo de vida de modelagem de processos, dados e sistemas, contempla inúmeras notações e técnicas: BPMN(*Business Processe Modeling Notation*) , UML(*Unified Modeling Languade*), Modelagem de Dados e outras, além e gerar arquivos em RTF e HTML.

O EA ainda permite o uso de técnicas de levantamento e documentação de Requisitos, tendo uma abordagem completa de Análise e Projeto de Sistemas, e conta com rastreabilidade de todos os elementos de modelagem (processos, casos de uso, componentes, tabelas, etc). Pode ser facilmente integrado com Visual Studio.NET e Eclipse, além de exportar ou trocar informações com outras ferramentas. Permite também gerar documentação e relatórios personalizados de um caso de uso completo ou de documentos “filhos” que são usados. Ele abrange os aspectos fundamentais do ciclo de vida de desenvolvimento de aplicativos, de gestão de requisitos através de fases de projeto, construção, testes e manutenção, com suporte para a rastreabilidade, gerenciamento de projetos e troca de controle desses processos. Ele também facilita o desenvolvimento orientado a modelos de código do aplicativo por usar uma plataforma de desenvolvimento integrado interno.

Dentre as características comuns de Gerenciamento de Requisitos suportados pelo Enterprise Architect, e definidas pela *SparxSystems*, incluem a personalização de como os requisitos são documentados, ligando os requisitos para a concepção e implementação dos detalhes e proporcionando rastreabilidade de requisitos ao longo das fases de construção e projeto. Os requisitos podem ser sujeitos a alterações de gestão e de processamento [CAMBIUCCI,2010].

O Enterprise Architect suporta métodos de modelagem de processos de negócio utilizando UML como linguagem de modelagem. Ele também suporta a definição de regras de negócio com a capacidade de gerar código executável a partir dessas regras.

Dentre os diagramas que podem ser feitos, temos como os principais:

- Máquinas de Estado
- Interação (diagramas de sequência)
- BPMN(*Business Processe Modeling Notation*), entre outros.

Além disso, o EA suporta diversas linguagens, dentre elas:

- ActionScript
- C
- C # (para ambos. NET 1.1 e. NET 2.0)
- + + (Padrão, plus. NET gerenciado extensões de C + +) C
- Delphi
- Java (incluindo Java 1.5, Aspectos e Genéricos)
- PHP
- Phyton
- Visual Basic

De acordo com o modelo de princípios *Driven Development*, o EA fornece um ambiente de desenvolvimento integrado que suporta edição de código (com destaque de sintaxe e *IntelliSense*), para a construção, depuração e código de teste tudo de dentro do modelo.

Compiladores e interpretadores suportados:

- Microsoft Windows nativo C
- Microsoft Windows nativo C + +
- Microsoft Windows Visual Basic
- Microsoft Família. NET (C #, J #, VB)
- Sun Microsystems Java
- PHP
- Compiladores GNU para C + +, C e Ada (GCC & GDB)

Diante de todas estas características para o entendimento do programa, temos que o EA é uma ferramenta robusta e de fácil manuseio. Por esse motivo, o escolhemos para trabalhar no estudo de caso e verificar a estimativa de métricas perante um caso de uso exposto nele.

No capítulo seguinte, será demonstrada a técnica de contagem por pontos de função e a técnica de métricas por casos de uso utilizando como apoio a ferramenta citada.

4 ESTUDO DE CASO

Para o estudo de caso, escolhemos um sistema de controle bancário.

Os requisitos desse sistema estão no Apêndice I.

No exemplo, o cliente solicitou um sistema que deve efetuar o cadastro de novos clientes de seu banco para abrir conta comum, conta especial e conta poupança. Deve ter a opção de manter o cliente para que possa atualizar seus dados ao invés de efetuar um novo cadastro.

O correntista (cliente) por sua vez poderá realizar as seguintes atividades:

- Realizar saque
- Emitir um extrato
- Encerrar uma conta
- Visualizar o seu saldo
- Realizar depósito ou transferência

Dentre os principais procedimentos da agência, o que o cliente nos pediu e destacou foi que o sistema deve:

- Registrar as principais ações do usuário
- Registrar a movimentação entre contas
- Possuir segurança para os clientes do banco
- Pedir a confirmação de senha e inserção de cartão magnético por 2 vezes
- Somente efetivar o saque quando confirmada a senha e inserido o cartão magnético quando solicitado

A listagem e especificação do caso de uso geral do sistema foi retirado e referenciado de [GUEDES,2009] e podem ser encontradas no Anexo II deste documento.

Devemos lembrar que, o mesmo projeto de controle bancário será colocado sob medição no contexto de ponto de função e casos de uso e em seguida iremos comparar os resultados obtidos.

Vale ressaltar também, que o objetivo não é impor o uso de um ou outro, mas sim de exemplificar e demonstrar que o uso de métricas é necessário e nos faz capazes de gerenciar com maior facilidade o tempo gasto nas atividades de desenvolvimento.

4.1 CASOS DE USO

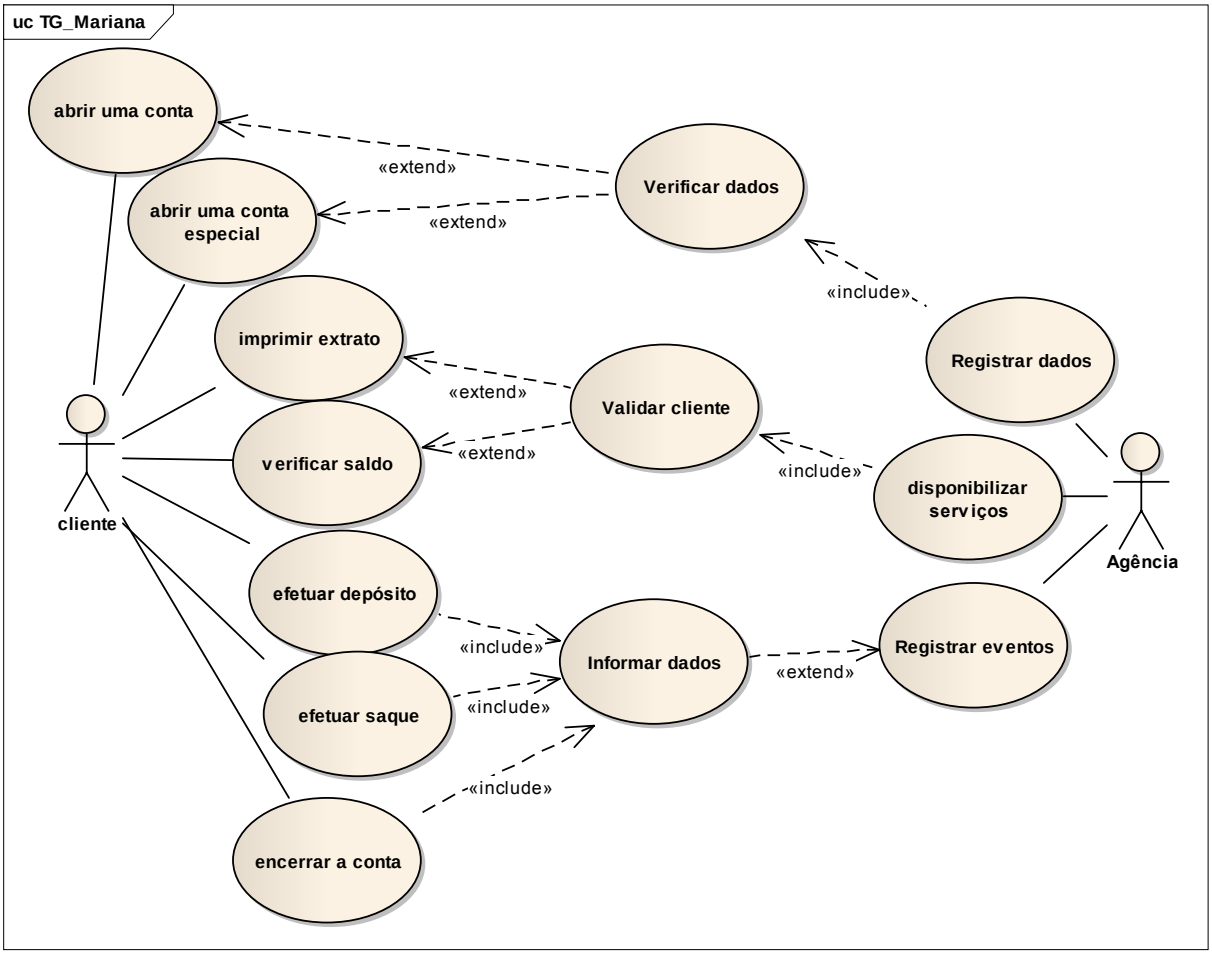

Figura 2 - Caso de uso: Sistema Bancário
Fonte:O autor

O caso de uso geral descrito acima, mostra as funções de cada ator. Nesse caso mostramos o ator “Cliente” e o “Agência”.

O “Cliente” representa as pessoas físicas ou jurídicas que mantêm ou mantiveram suas contas no banco, com direito a usar os serviços fornecidos enquanto suas contas estiveram ativas. Já a “Agência” é aquele funcionário que é contratado pelo banco para atender presencialmente os clientes. São basicamente os caixas da instituição e a própria instituição.

O cliente tem as opções de Encerrar conta, emitir saldo, realizar saque, realizar depósito, emitir extrato, abrir conta comum, enquanto que a agência tem as funções de registrar eventos, disponibilizar serviços e registrar dados.

Depois da definição do domínio e da construção do caso de uso geral, é possível calcular a métrica conforme veremos.

4.2 CALCULANDO PRAZO E CUSTO UTILIZANDO MÉTRICAS POR CASOS DE USO

Após a definição e elaboração do caso de uso, gera-se a estimativa de prazo e custo usando o Enterprise Architect.

Para tal processo devemos clicar no menu “Project”[1] e em seguida escolher a opção “Use Case Metrics”’[2].

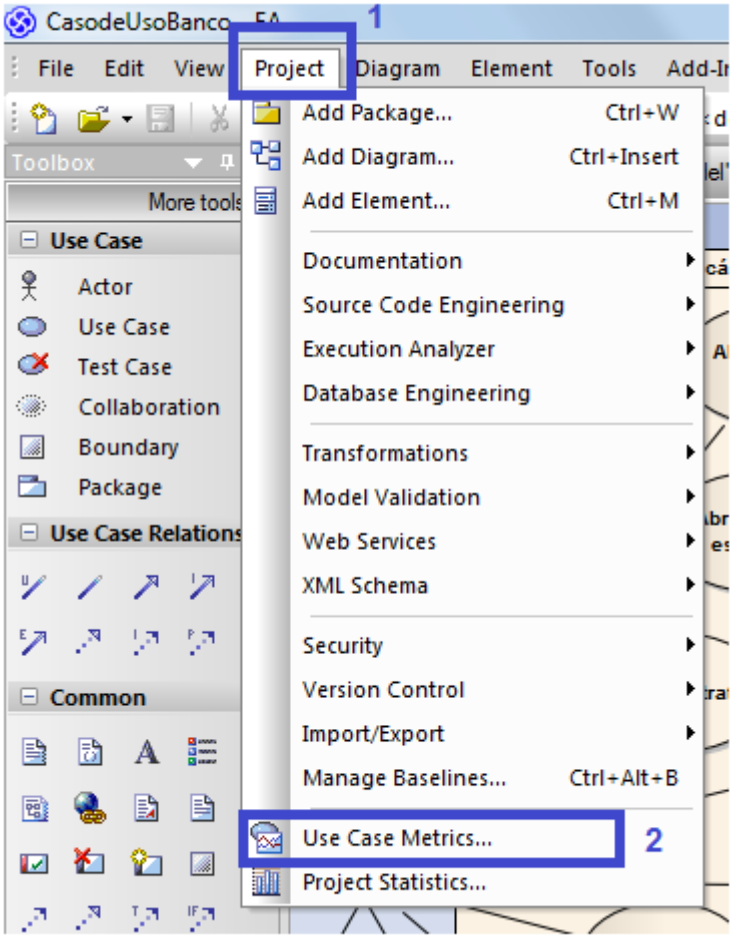

Figura 3 - Seleção de métricas por casos de uso

Fonte: O autor

Logo em seguida será exibida a tela conforme a Figura 4 com o resultado da métrica e que pode ser vista também na forma de um relatório como o que está no

Anexo II deste documento. Nele temos todas as pontuações feitas pelo programa, as classificações de características ambientais e de níveis de influência que o programa usou para o cálculo final do prazo e custo do sistema.

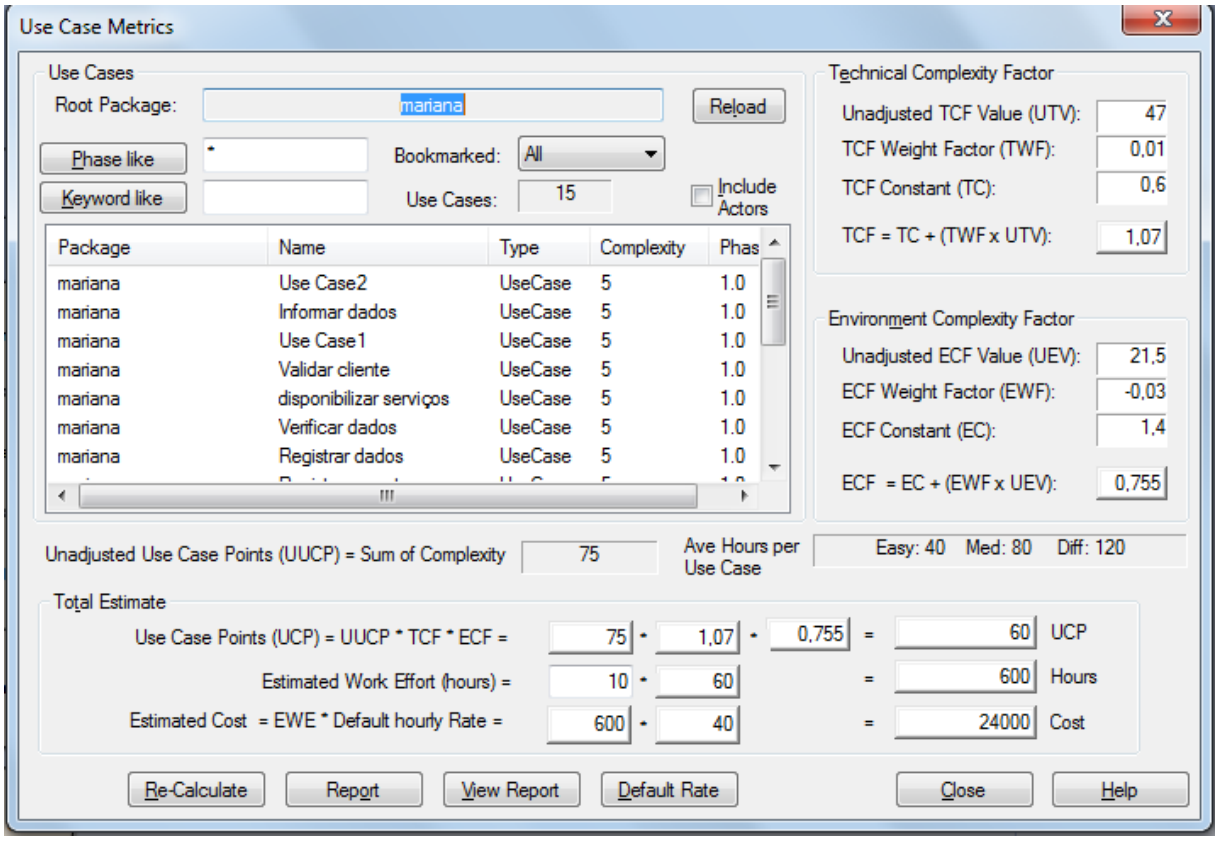

Figura 4 - Tela com resultados de métricas

Fonte: O autor

O programa faz o cálculo automático e o relatório é fácil de entender e de interpretar cada característica e como ele chegou a tal resultado.

No caso do nosso projeto, o Enterprise nos deu um resultado de prazo de 600 horas(4 meses e meio) para o desenvolvimento e um custo de R$ 24.000,00.

Isso pode ser diferente do cálculo usando pontos de função? Podemos obter resultados iguais ou diferentes? Isso é o que veremos a seguir.

4.3 CALCULANDO PRAZO E CUSTO UTILIZANDO MÉTRICAS POR PONTOS DE FUNÇÃO

Para este cálculo, analisamos o seguinte banco de dados com as tabelas e relacionamentos da Figura 5. As especificações do sistema são as mesmas que foram definidas no Apêndice I.

Usamos na construção e modelagem das tabelas do banco de dados o programa MySQL WorkBench 5.2 CE.

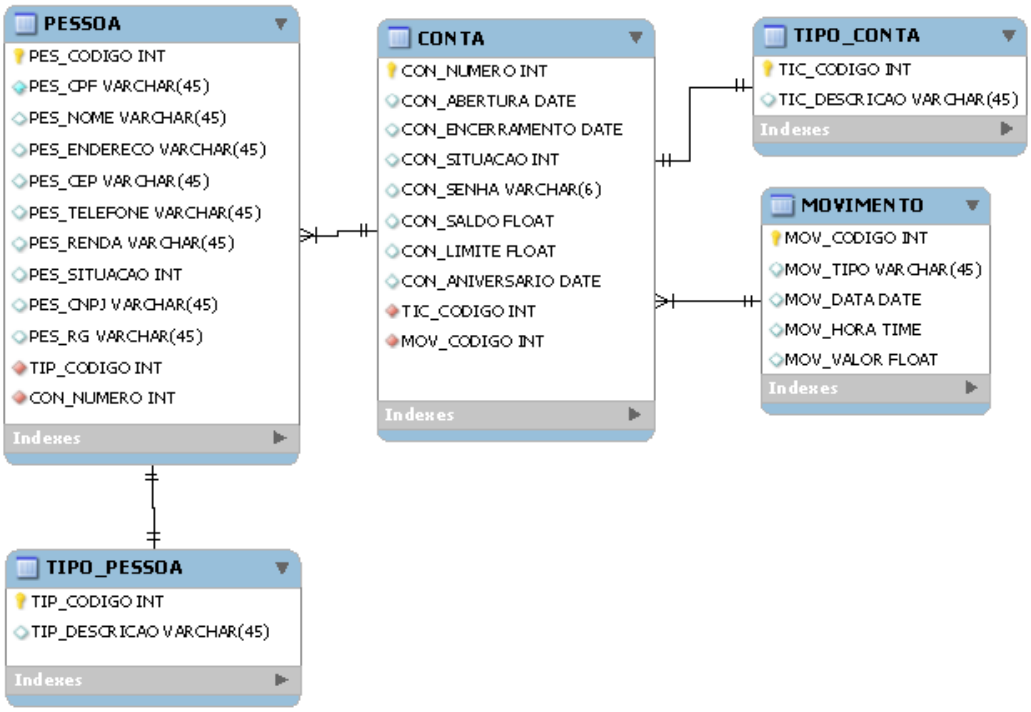

Figura 5 - Banco de Dados do sistema bancário.

Fonte: O autor

Iniciamos como explicado na seção 2.2.7 com a identificação e enumeração de funções juntamente com os pesos para obter os pontos de função brutos.

Devemos contar os campos das tabelas, e dessa forma temos:

PESSOA(12)+CONTA(10)+TIPO_PESSOA(2)+TIPO_CONTA(2)+MOVIMENTO(5)

Onde as tabelas TIPO_PESSOA e TIPO_CONTA têm 2 campos cada e por esse motivo são pontuados na tabela de entradas com campos de 1 a 4. Da mesma forma as tabelas PESSOA, CONTA e MOVIMENTO têm respectivamente 12 campos, 10 campos e 5 campos e são pontuados na tabela de entrada com campos de 5 a 15.

Tabela 13 - Contagem e qualificação de Entradas

Entrada							
	1 a 4	**5 a 15**	**16 ou mais**	**Total FP**		**Total FP * Peso**	
0-1				Simples	2	Simples	6
2	2			Médio	0	Médio	0
3 ou mais		3		Complexo	3	Complexo	18

Fonte: O autor

Tabela 14 - Contagem e qualificação de Saídas

Saída							
	1 a 5	**6 a 19**	**20 ou mais**	**Total FP**		**Total FP * Peso**	
0-1			1	Simples	3	Simples	12
02	3	2		Médio	3	Médio	15
4 ou mais				Complexo	0	Complexo	0

Fonte: O autor

Tabela 15 - Contagem e qualificação de Consultas

Consulta							
	1 a 4	**5 a 15**	**16 ou mais**	**Total FP**		**Total FP * Peso**	
0-1			1	Simples	2	Simples	6
2	2			Médio	1	Médio	4
3 ou mais		3		Complexo	3	Complexo	18

Fonte: O autor

Tabela 16 - Contagem e qualificação de Arquivos

Arquivo							
	1 a 19	**20 a 50**	**51 ou mais**	**Total FP**		**Total FP * Peso**	
1				Simples	5	Simples	35
2-5	5			Médio	0	Médio	0
6 ou mais				Complexo	0	Complexo	0

Fonte: O autor

Tabela 17 - Contagem e qualificação de Interfaces

Interface							
	1 a 19	**20 a 50**	**51 ou mais**	**Total FP**		**Total FP * Peso**	
1		1		Simples	6	Simples	30
2-5	5			Médio	0	Médio	0
6 ou mais			0	Complexo	0	Complexo	0

Fonte: O autor

Desses cálculos obtivemos um total de 144 pontos de função brutos que agora iremos ajustar calculando o nível de influência do sistema com base em algumas características. Para o sistema, definimos o nível de influência seguindo a tabela abaixo que totalizou o valor de 53. De acordo com a descrição de cada característica, foi dada a pontuação adequada

Tabela 18 - Características para estimativa do nível de influência

Nível de Influência do Sistema	
Comunicação de dados	5
Performance	5
Volume de transações	5
Eficiência do usuário final	1
Processamento complexo	5
Facilidade de implantação	3
Múltiplos locais	1
Processamento distribuído	3
Utilização de equipamento	5
Entrada de dados on-line	5
Atualização on-line	5
Reutilização de código	1
Facilidade operacional	4
Facilidade de mudanças	5
Total de NI	53

Fonte: O autor

Seguimos então com a fórmula :

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

E obtemos o fator de ajuste que nesse caso será:

0,65 + (0,01 * 53) = FA

FA= 0,65 + 0,53

FA= 1,18

Agora multiplicamos os pontos de função brutos pelo fator de ajuste, que no nosso exemplo obteremos:

144 * 1,18 = 169,92.

O valor é reajustado e arredondado para 170.

A partir de agora, qualquer cálculo feito será baseado no valor de 170 que é o valor dos pontos de função ajustados.

Conseguimos, por exemplo, calcular a quantidade de linhas de código usadas para o programa com base na tabela que informa o numero médio de LOC (linhas de código) por pontos de função a seguir:

Tabela 19 - Média de linhas de código das linguagens

LINGUAGEM	QTD MÉDIA DE KLOC/PF
Cobol	100
Pascal	90
Linguagens orientadas a objeto (C++)	30
Linguagens de 4ª geração (Java, Delphi,VB)	20
Geradores de código (SQL, HTML)	15

Fonte: O autor (De acordo com a ISO/IEC 9126)

Se definirmos, por exemplo, que o sistema de controle bancário será feito em cobol e em Java, temos os resultados:

COBOL = 170 * 100 = 17.000 Kloc (linhas de código)

JAVA = 170 * 20 = 3.400 Kloc (linhas de código)

Podemos escolher também como será o programa em sua implementação, e de acordo com a tabela a seguir e definida na NBR 13.596, temos os seguintes dados:

Tabela 20 - Produtividade de acordo com o tipo do sistema

Tipo de Sistema	Produtividade
Sistema Comercial	2.500 Kloc/Loc
Comércio Eletrônico	3.600 Kloc/Loc
Sistema Web	3.300 Kloc/Loc

Fonte: O autor (De acordo com a ISO/IEC 9126)

Para descobrirmos o prazo do sistema, considerando que o sistema será um sistema web, devemos calcular da seguinte forma:

COBOL = 17.000 / 3.300 = 5 meses e 15% (de 22 dias) = 5 meses e 3 dias

JAVA = 3.400 / 3.300 = 1 mês e 3% (de 22 dias) = 1 mês e 3 dias

De acordo com os padrões definidos em [UFPUG, 2004], temos que a produtividade individual é de 132h por mês (22 dias por mês x 6hs por dia), então dividimos e estipulamos a quantidade de horas trabalhadas por um programador:

COBOL = 5m * 132h = 660 horas + 3d x 6h = 18 horas = 678 horas

JAVA = 1 * 132 = 132 horas + 3 horas = 135 horas

Agora, definimos o valor da hora trabalhada do desenvolvedor. No nosso exemplo colocaremos cada hora com o valor de R$ 90,00 e teremos o projeto com o seguinte valor final:

COBOL = 678 * R$ 90,00 = R$ 61.020,00

JAVA = 135 * R$ 90,00 = R$ 12.150,00

CONSIDERAÇÕES FINAIS

Existem alguns motivos para querer medir as propriedades de um sistema. Dentre os principais, podemos destacar a necessidade de depois de analisar, poder aperfeiçoar os processos de desenvolvimento, melhorar a gerência de projetos, reduzir custos e gastos, reduzir dívidas com cronogramas, facilitar contratos, identificar melhores práticas de desenvolvimento, melhorar as estimativas, oferecer dados quantitativos de gerenciamento e desenvolvimento de software, entre outros. Podemos ver que os principais motivos são diretamente ligados ao controle e melhoria da própria empresa que produz o sistema.

Demonstra-se neste trabalho, técnicas de medição e ao escolhermos duas das tecnologias disponíveis no mercado, uma obsoleta (Cobol) e outra que está em alta (Java) podíamos de certa forma, prever, que a linguagem Cobol iria sair mais “cara” do que uma programação em Java. Isso implica não só no próprio desenvolvimento, mas no conhecimento da tecnologia, tempo, e produtividade.

No final, obtivemos que o sistema bancário feito em Cobol demoraria um tempo maior para ser feito e custaria mais caro do que o projeto feito em Java. Os resultados de prazo e custo estão descritos na Tabela 21:

Tabela 21 - Resultados da métrica por pontos de função

TECNOLOGIA/LINGUAGEM	PRAZO	CUSTO
COBOL	5 meses + 3 dias	R$ 61.020,00
JAVA	1 mês + 3 horas	R$ 12.150,00

Fonte: O autor

Nesse caso, caberia ao cliente escolher a tecnologia, bem como optar por um custo e prazo de desenvolvimento.

Acreditamos que comparando entre as duas linguagens de desenvolvimento, normalmente uma pessoa optaria pelo sistema feito em Java que demoraria 01 mês e 03 dias para o desenvolvimento e custaria cerca de R$ 12.150,00. Porém existem sistemas e clientes de todos os tipos e com as mais diversas necessidades. Qualquer um dos sistemas, com seus prazos e custos poderia ser escolhido para a implementação.

A estimativa feita pelo programa *Enterprise Archictect* foi rápida, porém tivemos que obrigatoriamente desenvolver o caso de uso geral do sistema. Isso no dia a dia das corporações seria de fácil aplicação, visto que eles precisam do caso

de uso para todo o desenvolvimento, até mesmo para a validação das funções e para facilitar a explicação das atividades para o usuário.

Percebemos que o software é de fácil manuseio e bastante interativo, o que facilita o uso e entendimento de suas principais funções, porém existem alguns termos técnicos que somente olhando o manual do programa conseguimos identificar o que significa. Algumas siglas e abreviações que sem a leitura das especificações do programa dificulta o entendimento do que está sendo feito. Isso implica num ponto de desvantagem para o uso de métricas por caso de uso usando o *Enterprise Architect*.

No EA, na tela de resultado principal da métrica, os valores ficam bem definidos e explícitos para análise. No projeto do sistema bancário, por exemplo, que escolhemos, o valor foi de R$ 24.000,00 e o total de horas foi de 600 horas. De acordo com a tabela NBR 13596 devemos trabalhar com o valor de 132 horas por mês e nesse caso se fizermos uma divisão simples, obteremos que o sistema será feito, em média em 4 meses e meio.

É interessante apresentar o valor em horas, pois o gerente de projetos pode controlar o desenvolvimento de acordo com a quantidade de funcionário e suas escalas, verificando o ponto citado anteriormente: métricas ajudam no gerenciamento de projetos.

O gerente de projetos pode inclusive dividir as atividades de acordo com a demanda de trabalho de cada desenvolvedor para que se faça cumprir com o prazo estipulado na métrica, reforçando ainda mais sua ajuda no gerenciamento.

Outro ponto que vale destacar é que o *Enterprise* gera um relatório com as especificações e as pontuações dadas para a estimativa do valor e prazo final do sistema. Permite também alterações em algumas das características ambientais e de nível de influência, bem como adicionar outras que achar relevante.

Já com as métricas por pontos de função, utilizamos duas tecnologias para exemplificar os cálculos, Cobol e Java.

A métrica por pontos de função implica na definição do diagrama de classes, visto que somente com ele conseguiremos definir a etapa inicial do cálculo que é a contagem de entradas, saídas, arquivos, consultas e interfaces. Isso, assim como a definição do caso de uso geral, não seria de grande dificuldade para os desenvolvedores, visto que a criação do banco de dados é parte do processo de

desenvolvimento. Pode-se encontrar dificuldade de fazer a estimativa logo no início, pois para a definição do banco de dados, é necessário o envolvimento de toda a equipe, porém depois de definido fica fácil de calcular.

A primeira parte é a que leva mais tempo, necessitando de uma minuciosa análise de cada componente, arquivos, consultas e possíveis interfaces do sistema.

Uma desvantagem é que não há um sistema que faça a estimativa de prazo e custo, assim que definido o banco de dados. Somente temos o cálculo feito à mão e com ajuda das fórmulas definidas anteriormente, porém como os cálculos não são complexos e conseguimos os resultados.

Existe toda descrição de cada característica e as típicas pontuações ajudam na contagem de cada uma delas, pontuando as métricas de pontos de função positivamente.

Dentro das vantagens podemos destacar também a opção de escolha da linguagem de programação a ser utilizada na implementação do sistema. Pode-se escolher qualquer tecnologia e pontuar dentro dos cálculos, obtendo uma resposta de prazo e custo diferente para cada uma delas.

No nosso exemplo, o sistema bancário feito em Cobol, demoraria 05 meses e 03 dias e custaria R$ 61.020,00, enquanto que o mesmo sistema desenvolvido em Java demoraria 01 mês e 03 horas para ser implementado e custaria cerca de R$ 12.150,00.

Da mesma forma que a métrica por casos de uso auxilia na gerência de projetos, a métrica por pontos de função ajuda a verificar e controlar a produtividade e mostra que dentre as áreas de estudo do PMBOK e gerenciamento de projetos a métrica abrange prazo e custo.

Outro ponto que deve ser levado em consideração é que de acordo com as pesquisas bibliográficas, vimos que somente sabemos a dimensão do que está sendo feito quando é gerenciado e depois de mensurado o prazo e custo. Definindo banco de dados ou casos de uso sabemos o tamanho e o que deve ser exigido para o projeto.

Ao final, conclui-se que independente da métrica que vamos usar para mensurar prazo e custo de desenvolvimento de um projeto, a métrica deve ser utilizada para gerenciar e controlar um projeto, e também medir produtividade e qualidade.

Conclui-se também que a métrica deve ser usada para gerenciamento e também para controle de prazos. Devemos usar independente de qual delas, independente de plataformas e de tamanho do projeto. Toda e qualquer medida auxiliará no ciclo de desenvolvimento.

Tabela 22- Resultados dos cálculos de métricas

TIPO DE MÉTRICA	LINGUAGEM	PRAZO	CUSTO
CASOS DE USO	Indefinida	4 meses e meio	R$ 24 000,00
PONTOS POR FUNÇÃO	Cobol(3ª geração)	5 meses e 3 dias	R$ 61 020,00
	Java (4ª geração)	1 mês e 3 dias	R$ 12 150,00

Fonte: O autor

A tabela 22 demonstra os resultados obtidos em cada estimativa. Usando a métrica por casos de uso com linguagem de programação indefinida, obteve-se o valor de R$ 24 000,00 reais e prazo de 4 meses e meio para o desenvolvimento, enquanto que na métrica por pontos de função, sendo o programa feito na linguagem Cobol, o programa demoraria 5 meses e 3 dias para o desenvolvimento e custaria R$ 61 020,00 e usando a linguagem Java iriam ser gastos R$ 12 150,00 e o prazo de desenvolvimento seria de 1 mês e 3 dias.

Independentemente se utilizar programas para automatizar ou se realizar os cálculos manualmente, medir é gerenciar. Métrica é importante em qualquer empresa e conforme o trabalho realizado, verifica-se que não só prazo e custo pode ser medido, mas em consequência pode-se medir produtividade e qualidade, que são variáveis de grande uso nas organizações. Com métricas prevemos, controlamos, analisamos e entendemos não somente o projeto em desenvolvimento, mas todos os processos.

REFERÊNCIAS

[5CQualiBR,2013] **Métricas,** 2013. Disponível em: <http://www.softwarepublico.gov.br/5cqualibr/xowiki/metricas> Acesso em 09 Out. 2013

[BERKENBROCK, 2006] **Qualidade, Confiabilidade e Segurança de Software**, 2006. Disponível em <www.comp.ita.br/~gian/ce230/projeto/relatorioFinalCe230.pdf> Acesso em 20 mai. 2013.

[BOAVENTURA,2001]Boaventura, Ines Aparecida. **Gerência de Projetos: Métricas de Software,**2001 Disponível em <www.dcce.ibilce.unesp.br/~ines/cursos/extensao/aula4.ppt> Acesso 12 Set. 2013

[CAMBIUCCI,2010] Cambiucci, Waldemir. **Enterprise Architecture: a arquitetura corporativa e o papel do arquiteto de TI,**2010. Disponível em <http://msdn.microsoft.com/pt-br/library/gg490650.aspx> Acesso em 12 Ago. de 2013

[CAMPÊLO, 2002] Campêlo, Gabriela Moreira Carneiro. **A utilização de métricas na gerência de projetos de Software**, 2002. Disponível em < http://www.cin.ufpe.br/hermano/download/dissertacoes/metricasGPCMM2.pdf> Acesso em 08 Out. 2013

[CAMPOS, 2013] Campos, Fábio Martinhos. **Métricas de Software Como Ferramenta de Apoio ao Gerenciamento de Projetos de Software,** 2013. Disponível em < http://www.linhadecodigo.com.br/artigo/453/metricas-de-software-como-ferramenta-de-apoio-ao-gerenciamento-de-projetos-de-software.aspx#ixzz2WLGpuj3t> Acesso em 10 jun. 2013

[COELHO,2009] COELHO, Hilda Simon. **Documentação de Software**,2009. Disponível em <http://www.periodicos.letras.ufmg.br/index.php/textolivre/article/download/24/24> Acesso em 08 mai. 2013

[GUEDES,2009] Guedes, Gilleanes T.A . **UML2. Uma abordagem prática,** 2009. 1ª ed. São Paulo. Novatec Editora,2009. 485 pág.

[IGLESIAS,2013] Iglesias, Susana. **Métricas do Processo e projeto de software**,2013**.** Disponível em <http://smiglesias.tripod.com/ES/aula06.pdf> Acesso em 07 Out. 2013

[MICHAELIS,2009] MICHAELIS. **Dicionário Online**, 2009. Disponível em <http://michaelis.uol.com.br/moderno/portugues/index.php?lingua=portugues-portugues&palavra=documenta%E7%E3o> Acesso em 08 mai. 2013.

[MICHELAZZO,2006] MICHELAZZO, Paulino. **A documentação de Software**, 2006. Disponível em <http://imasters.com.br/artigo/4371/gerencia-de-ti/a-documentacao-de-software/> Acesso em 09 mai. 2013.
[MPCM,2013] Maturity by Project Categoty Model - **Pesquisa: Maturidade em gerenciamento de Projetos,** 2013. Disponivel em <http://www.maturityresearch.com/novosite/index_br.html>. Acesso em 25/11/2013

[PRESSMAN,2006]Pressman, Roger, 2006. **Engenharia de Software**. 6. ed. Rio de Janeiro: McGraw-Hill.

[RUP, 2000]Rational Unified Process 2000, **Best Practices for Software Development Teams**. Disponível em <http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf> Acesso em 08 Out. 2013

[SILVA,2013] Silva, Mauro César. **CMMI para iniciantes**,2013. Disponível em <http://www.linhadecodigo.com.br/artigo/1401/cmmi-para-iniciantes.aspx> Acesso em 06 set. 2013

[SISP,2012] Secretaria de Logística e Tecnologia da Informação . **Roteiro de Métricas de Software do SISP**, 2012. Versão 2.0 Disponível em <www.governoeletronico.gov.br/.../roteiro-de-metricas-de-software-do-sis>Acesso em 01 Ago. 2013.

[TIMOTEO,2008] Timóteo, Aline Lopes. **Plano de dissertação de Mestrado**,2008. Disponível em < www.cin.ufpe.br/~alt/mestrado/alt-saap.doc > Acesso em 07 Out. 2013

[TRAVASSOS, 2008] Travassos, Guilherme Horta. **iMPS- Resultados de Desempenho de organizações que adotaram o modelo MPS**, 2008. Disponível em <http://www.softex.br/wp-content/uploads/2013/08/iMPS-Resultados-de-Desempenho-de-Organiza%C3%A7%C3%B5es-que-Adotaram-o-Modelo-MPS.pdf> Acesso em 04 out. 2013.

[UFPUG, 2004] International Function Point Users Group. **Manual de Contagem de Pontos de Função**,2004**.** Versão 4.2.1

[UFRS,2009]Universidade Federal do Ria Grande do Sul. **Métodos de Pesquisa**,2009. Disponível em <http://www.ufrgs.br/cursopgdr/downloadsSerie/derad005.pdf> Acesso em 19 jun. 2013

[VASCONCELOS, 2005] Vasconcelos, Alexandre. **Introdução a Métricas de Software**, 2005. Disponível em <www.cin.ufpe.br/~if720/slides/introducao-a-metricas-de-software.ppt> Acesso em 09 mai. 2013

ANEXO I – DESCRIÇÃO DOS NIVEIS DE INFLUÊNCIA

A estimativa do nível de influência deve ser baseada nas seguintes características definidas e disponíveis no site da SISP [SISP,2012]:

I) Comunicação de dados

Os aspectos relacionados aos recursos utilizados para a comunicação de dados do sistema deverão ser descritos de forma global. Devemos descrever se a aplicação usa protocolos diferentes para recebimento e envio das informações do sistema.

Para tal característica temos a seguinte pontuação:

0 - se aplicação é batch ou funciona stand-alone (funciona como uma estação de trabalho isolada);

1 - se aplicação é batch, mas utiliza entrada de dados ou impressão remota;

2- se aplicação é batch, mas utiliza entrada de dados e impressão remota;

3 -se aplicação tem entrada de dados on-line para alimentar processamento batch ou sistema de consulta;

4 -se aplicação tem entrada de dados on-line, mas suporta apenas um tipo de protocolo de comunicação;

5 -se a aplicação tem entrada de dados on-line e suporta mais de um tipo de protocolo de comunicação.

Vale lembrar que batch, significa que a aplicação não necessita de interação com o usuário e que Stand-Alone são programas autossuficientes, que não necessitam de nenhum interpretador ou software auxiliar para seu funcionamento. Outra definição importante e que devemos dar uma atenção é quanto a pontuação que normalmente é dada para as aplicações. As aplicações batch normalmente pontuam de 0 a 3, as online pontuam 4, as web de 4 a 5 e os sistemas real-time ou de controle de processos pontuam de 4 a 5.

II)Processamento distribuído

Esta característica refere-se a sistemas que utilizam dados ou funções distribuídas, tendo diversas CPU’s.

Pontuação para a característica:

0- se a aplicação não auxilia na transferência de dados ou funções entre os processadores da empresa;

1- se a aplicação prepara dados para o usuário final utilizar em outro processador (tal como planilhas);

2 -se a aplicação prepara dados para transferência e os transfere para serem processados em outro equipamento da empresa (não pelo usuário final);

3 - se o processamento é distribuído e a transferência de dados é on-line e apenas em uma direção;

4 -se o processamento é distribuído e a transferência de dados é on-line e em ambas as direções;

5 -se as funções de processamento são dinamicamente executadas no equipamento (CPU) mais apropriada.

Tipicamente, a maioria da aplicações, incluindo aplicações legadas, recebem 0 como pontuação, as distribuídas primitivas em que os dados não são transferidos online pontuam de 1 a 2, as cliente-servidor ou web recebem 3 ou 4. Raramente é pontuado a nota 5.

III) Performance

Trata-se de parâmetros estabelecidos pelo usuário como aceitáveis, e relativos a tempo de resposta.

Pontuação:

0- se nenhum requerimento especial de desempenho foi solicitado pelo usuário;

1- se os requerimentos de desempenho foram estabelecidos e revistos, mas nenhuma ação especial foi requerida;

2- se o tempo de resposta e volume de processamento são itens críticos durante horários de pico de processamento. Se nenhuma determinação especial para a utilização do processador foi estabelecida, e se a data limite para a disponibilidade de processamento é sempre o próximo dia útil;

3- Se tempo de resposta e o volume de processamento são itens críticos durante todo o horário comercial. Se nenhuma determinação especial para a utilização do processador foi estabelecida e se a data-limite necessária para a comunicação com outros sistemas é limitante;

4- Se os requerimentos de desempenho estabelecidos requerem tarefas de análise de performance na fase de planejamento e análise da aplicação;

5- Se além do descrito no item 4, as ferramentas de análise de performance foram usadas nas fases de planejamento, desenvolvimento e/ou implementação para atingir os requerimentos de performance estabelecidos pelos usuários;

Normalmente as aplicações batch recebem notas de 0 a 4, as online de 0 a 4, as web de 4 a 5, os online gerenciais recebem 2, os real-time e de telecomunicações e controle de processos de 0 a 5, sendo a maior nota (5) deve ser analisada com base em ferramentas de desempenho.

IV) Utilização de equipamento

Trata-se de observações quanto ao nível de utilização de equipamentos requerido para a execução do sistema. Este aspecto é observado com vista a planejamento de capacidades e custos.

Pontuação para essa característica:

0- se nenhuma restrição operacional explícita ou mesmo implícita foi incluída;

1- se existem restrições operacionais leves. Não é necessário esforço especial para atender às restrições;

2- se algumas considerações de ajuste de performance e segurança são necessárias;

3- se são necessárias especificações especiais de processador para um módulo específico da aplicação;

4- se as restrições operacionais requerem cuidados especiais no processador central ou no processador dedicado para executar a aplicação;

5- se além das características do item anterior, há considerações especiais que exigem utilização de ferramentas de análise de performance, para a distribuição do sistema e seus componentes, nas unidades processadoras.

Nesse caso, a maioria das aplicações recebe nota 2, enquanto as aplicações cliente-servidor, web, e sistemas de controle pontuam de 3 à 5.

V) Volume de transações

Volume de transações consiste na avaliação do nível de influência do volume de transações no projeto, desenvolvimento, implantação e manutenção do sistema.

A pontuação para essa característica é:

0- caso não estejam previstos períodos de picos de volume de transação;

1- caso estão previstos picos de transações mensalmente, trimestralmente, anualmente ou em certo período do ano;

2- caso esteja previstos picos semanais;

3- caso tenha picos diários;

4- caso tenha um alto volume de transações foi estabelecido pelo usuário, ou o tempo de resposta necessário atinge nível alto o suficiente para requerer análise de performance na fase de projeto;

5- caso tenha a pontuação 5,e além disso seja necessário utilizar ferramentas de análise de performance nas fases de projeto, desenvolvimento e/ou implantação.

Normalmente as aplicações batch recebem notas de 0 a 3, as online de 0 a 4, as real-time e de telecomunicações e controle de processos pontuam de 0 a 5, sendo que a pontuação máxima requer uma análise em uma ferramenta de desempenho.

VI) Entrada de dados on-line

A análise desta característica permite quantificar o nível de influência exercida pela utilização de entrada de dados no modo on-line no sistema.

A pontuação é a seguinte:

0- se todas as transações são processadas em modo batch;

1- Se 1% a 7% das transações são entradas de dados on-line;

2- Se 8% a 15% das transações são entradas de dados on-line;

3- Se 16% a 23% das transações são entradas de dados on-line;

4- Se 24% a 30% das transações são entradas de dados on-line;

5- Se mais de 30% das transações são entradas de dados on-line.

Normalmente as aplicações batch recebem notas de 0 a 1, as online, real-time e de telecomunicações e controle de processos pontuam 5, as cliente-servidor também pontuam 5.

VII) Eficiência do usuário final

A análise desta característica permite quantificar o grau de influência relativo aos recursos implementados com vista a tornar o sistema amigável permitindo incrementos na eficiência e satisfação do usuário final.

Dentre os componentes que podemos destacar a existência, temos:

a) Auxílio à navegação (teclas de função, acesso direto e menus dinâmicos);

b) Menus;

c) Documentação e help on-line;

d) Movimento automático do cursor;

e) Movimento horizontal e vertical de tela;

f) Impressão remota (via transações on-line);

g) Teclas de função preestabelecidas;

h) Processos batch submetidos a partir de transações on-line;

i) Utilização intensa de campos com vídeo reverso, intensificados, sublinhados, coloridos e outros indicadores;

j) Impressão da documentação das transações on-line através de Hard copy;

k) Utilização de mouse;

l) Menus pop-up;

m) menor número possível de telas para executar as funções de negócio;

n) Suporte bilíngue (contar como 4 itens);

o) Suporte multilíngue. (contar como 6 itens).

A pontuação para essa característica é a seguinte:

0- Nenhum dos itens descritos;

1- de um a três itens descritos;

2- de quatro a cinco dos itens descritos;

3- mais de cinco dos itens descritos, mas não há requerimentos específicos do usuário quanto a amigabilidade do sistema;

4- mais de cinco dos itens descritos, e foram estabelecidos requerimentos quanto à amigabilidade fortes o suficiente para gerarem atividades específica envolvendo fatores, tais como minimização da digitação, para mostrar inicialmente os valores utilizados com mais frequência;

5- mais de cinco dos itens descritos, e foram estabelecidos requerimentos quanto à amigabilidade fortes o suficiente para requerer ferramentas e processos especiais para demonstrar antecipadamente que os objetivos foram alcançados.

VIII) Atualização on-line

Mede a influência no desenvolvimento do sistema face a utilização de recursos que visem a atualização dos Arquivos Lógicos Internos no modo on-line.

Pontuação:

0- Nenhuma;

1- Atualização on-line de um a três arquivos lógicos internos. O volume de atualização é baixo e a recuperação de dados é simples;

2- Atualização on-line de mais de três arquivos lógicos internos. O volume de atualização é baixo e a recuperação dos dados é simples;

3- Atualização on-line da maioria dos arquivos lógicos internos;

4- Em adição ao item anterior, é necessário proteção contra perdas de dados que foi projetada e programada no sistema;

5- Além do item anterior, altos volumes trazem considerações de custo no processo de recuperação. Processos para automatizar a recuperação foram incluídos minimizando a intervenção do operador.

IX) Processamento complexo

A complexidade de processamento influencia no dimensionamento do sistema, e portanto deve ser quantificado o seu grau de influência, com base nas seguintes categorias:

•Processamento especial de auditoria e/ou processamento especial de segurança foram considerados na aplicação;

•Processamento lógico extensivo;

•Processamento matemático extensivo;

•Processamento gerando muitas exceções, resultando em transações incompletas que devem ser processadas novamente. Por exemplo, transações de autoatendimento bancário interrompidas por problemas de comunicação ou com dados incompletos;

•Processamento complexo para manusear múltiplas possibilidades de entrada/saída. Por exemplo: multimídia;

Pontuação:

0- Nenhum dos itens descritos;

1- Apenas um dos itens descritos;

2- Dois dos itens descritos;

3- Três dos itens descritos;

4- Quatro dos itens descritos;

5- Todos os cinco itens descritos.

X) Reutilização de código

A preocupação com o reaproveitamento de parte dos programas de uma aplicação em outras aplicações, implica em cuidados com padronização

Pontuação:

0- Nenhuma preocupação com reutilização de código;

1 - Código reutilizado foi usado somente dentro da aplicação;

2- Menos de 10% da aplicação foi projetada prevendo utilização posterior do código por outra aplicação;

3 -10% ou mais da aplicação foi projetada prevendo utilização posterior do código por outra aplicação;

4 -A aplicação foi especificamente projetada e/ou documentada para ter seu código reutilizado por outra aplicação e é customizada pelo usuário em nível de código –fonte;

5 -A aplicação foi especificamente projetada e/ou documentada para Ter seu código facilmente reutilizado por outra aplicação e é customizada para uso através de parâmetros que podem ser alterados pelo usuário.

XI)Facilidade de implantação

A quantificação do grau de influência desta característica é medida observando-se o plano de conversão e implantação e as ferramentas utilizadas durante a fase de testes do sistema.

Pontuação:

0- Nenhuma consideração especial foi estabelecida pelo usuário e nenhum procedimento especial é requerido na implantação;

1 - Nenhuma consideração especial foi estabelecida pelo usuário, mas procedimentos especiais são necessários na implantação;

2 - Requerimentos de conversão e implantação foram estabelecidos pelo usuário e roteiro de conversão e implantação foram providos e testados. O impacto da conversão no projeto não é considerado importante;

3 - Requerimentos de conversão e implantação foram estabelecidos pelo usuário e roteiro de conversão e implantação foram providos e testados. O impacto da conversão no projeto é considerado importante;

4 - Além do item 2, conversão automática e ferramentas de implantação foram providas e testadas;

5 - Além do item 3, conversão automática e ferramentas de implantação foram providas e testadas.

XII)Facilidade operacional

A análise desta característica permite quantificar o nível de influência na aplicação, com relação a procedimentos operacionais automáticos que reduzem os procedimentos manuais, bem como, mecanismos de inicialização, recuperação, e armazenamento, verificados durante os testes do sistema.

Pontuação:

0- Nenhuma consideração especial de operação, além do processo normal de salva foi estabelecida pelo usuário;

1 a 4 - Verifique quais das seguintes afirmativas podem ser identificadas na aplicação. Selecione as que forem aplicadas. Cada item vale um ponto, exceto se definido explicitamente:

•Foram desenvolvidos processos de inicialização, salva e recuperação, mas a intervenção do operador é necessária;

•Foram estabelecidos processos de inicialização, salva e recuperação, e nenhuma intervenção do operador é necessária.

•A aplicação minimiza a necessidade de montar fitas magnéticas;

•A aplicação minimiza a necessidade de manuseio de papel.

5 - A aplicação foi desenhada para trabalhar sem operador, nenhuma intervenção do operador é necessária para operar o sistema além de executar e encerrar a aplicação. A aplicação possui rotinas automáticas para recuperação em caso de erro.

XIII)Múltiplos locais

Esta característica consiste na observação da arquitetura do projeto, observando-se a necessidade de instalação do sistema em diversos lugares.

Pontuação:

0- Os requerimentos do usuário não consideraram a necessidade de instalação em mais de um local;

1 - A necessidade de múltiplos locais foi considerada no projeto, e a aplicação foi desenhada para operar apenas em ambiente de software e hardware, idênticos;

2 - A necessidade de múltiplos locais foi considerada no projeto, e a aplicação está preparada para trabalhar apenas em ambientes similares de software e hardware;

3 - A necessidade de múltiplos locais foi considerada no projeto, e a aplicação está preparada para trabalhar sob diferentes ambientes de hardware e/ou software;

4 - Plano de documentação e manutenção foram providos e testados para suportar a aplicação em múltiplos locais, além disso, os itens 1 ou 2 caracterizam a aplicação;

5 - Plano de documentação e manutenção foram providos e testados para suportar a aplicação em múltiplos locais, além disso, o item 3 caracteriza a aplicação;

XIV)Facilidade de mudanças

A preocupação com a manutenção influencia no desenvolvimento do sistema. Esta influência deve ser quantificada observando-se os seguintes atributos:

a) Estão disponíveis facilidades como consultas e relatórios flexíveis para atender necessidades simples (conte como 1 item);

b) Estão disponíveis facilidades como consultas e relatórios flexíveis para atender necessidades de complexidade média (conte como 2 itens);

c) Estão disponíveis facilidades como consultas e relatórios flexíveis para atender necessidades complexas (conte 3 itens);

d) Dados de controle são armazenados em tabelas que são mantidas pelo usuário através de processos on-line, mas mudanças têm efeitos somente no dia seguinte;

e) Dados de controle são armazenados em tabelas que são mantidas pelo usuário através de processos on-line, as mudanças tem efeito imediatamente (conte como 2 itens).

Para essa característica temos a seguinte pontuação:

0-Nenhum dos itens descritos;

1-Um dos itens descritos;

2-Dois dos itens descritos;

3-Três dos itens descrito;

4-Quatro dos itens descritos;

5-Todos os cinco itens descritos

ANEXO II – DOCUMENTO GERADO PELO ENTERPRISE ARCHITECT

Estimativa geral baseada em casos de uso

Item	Valor
Data da estimativa	**16-nov-2013 18:30:39**
Fase	*****
Total de casos de uso	**13**
Pontos de casos de uso geral (UUCP)	**65,00**
Complexidade técnica	**1,07**
Complexidade de desenvolvimento	**0,75**
Pontos de casos de uso	**52,00**
Estimativa de horas por casos de uso	**10,00**
Total de horas	**520,00**
Custo total	**20800,00**

Fatores de ajustes- Características

Métrica	Descrição	Tamanho	Valor	Total
TCF01	Sistema distribuído	2,00	5,00	10,00
TCF02	Tempo de resposta	1,00	4,00	4,00
TCF03	Eficiência do usuário final	1,00	2,00	2,00
TCF04	Complexidade de processamento interno	1,00	4,00	4,00
TCF05	Reuso de código	1,00	2,00	2,00
TCF06	Facilidade de instalação	0,50	5,00	2,50
TCF07	Usabilidade	0,50	3,00	1,50
TCF08	Portabilidade	2,00	3,00	6,00
TCF09	Facilidade de mudanças	1,00	3,00	3,00
TCF10	Concorrência	1,00	2,00	2,00
TCF11	Recursos de segurança	1,00	2,00	2,00
TCF12	Acessível por terceiros	1,00	5,00	5,00
TCF13	Requer treinamento especial	1,00	3,00	3,00
			Total:	**47,00**

Fator	Valor
Fatores não ajustados (UTV)	47,00
Peso(TWF)	0,01
Peso ajustado(TC)	0,60
Fator de influência (TCF) = TC + (UTV * TWF)	**1,07**

Fatores ambientais de desenvolvimento

Métrica	Descrição	Tamanho	Valor	Total
ECF01	Familiaridade com RUP ou outro processo formal	1,50	4,00	6,00
ECF02	Experiência com a Aplicação em desenvolvimento	0,50	3,00	1,50
ECF03	Experiência em orientação á objetos	1,00	4,00	4,00
ECF04	Presença de analista experiente	0,50	4,00	2,00
ECF05	Motivação	1,00	3,00	3,00
ECF06	Requisitos estáveis	2,00	4,00	8,00
ECF07	Desenvolvedores em meio expediente	-1,00	0,00	-0,00
ECF08	Linguagem de programação difícil	-1,00	3,00	-3,00
			Total:	**21,50**

Fator	Valor
Valores não ajustados(UEV)	21,50
Peso(EWF)	-0,03
Peso Ajustado (EC)	1,40
Fatores ambientais de desenvolvimento (ECF) = EC + (UEV * EWF)	**0,75**

Casos de uso e atores

Pacote	Nome	Tipo	Fase	Custo	Custo corrente	Saída %
Primary Use Cases	Use Case2	UseCase	*	1600,00	0	0
Primary Use Cases	Use Case1	UseCase	*	1600,00	0	0
Use Case Model	Manter clien-te	UseCase	*	1600,00	0	0
Use Case Model	Realizar saque	UseCase	*	1600,00	0	0
Use Case Model	Registrar movimento	UseCase	*	1600,00	0	0
Use Case Model	Realizar depósito	UseCase	*	1600,00	0	0
Use Case Model	Manter clien-te	UseCase	*	1600,00	0	0
Use Case Model	Abrir conta poupança	UseCase	*	1600,00	0	0
Use Case Model	Abrir conta especial	UseCase	*	1600,00	0	0
Use Case Model	Abrir conta comum	UseCase	*	1600,00	0	0
Use Case Model	Emitir Saldo	UseCase	*	1600,00	0	0
Use Case Model	Encerrar conta	UseCase	*	1600,00	0	0
Use Case Model	Emitir extrato	UseCase	*	1600,00	0	0

APÊNDICE I – DOCUMENTAÇÃO DOS CASOS DE USO

Descrição de casos de uso do sistema de controle bancário.

[...] primeiramente um cliente, representado aqui por um ator, solicita a abertura de uma conta, a qual pode ser uma conta comum, que não permite a retirada de mais dinheiro do que está depositada, uma conta especial, que permita o saque extra até um determinado limite, ou uma conta poupança, que rende juros enquanto o dinheiro depositado permanecer sem ser movimentado.[...]

[...]Como os processos de abertura de cada tipo de conta têm características muito semelhantes entre si, com poucas distinções, resolveu-se colocar o caso de uso **"Abrir conta comum",** como uma generalização(embora ele seja efetivamente utilizado) e os outros dois tipos de abertura como especializações do primeiro, detalhando-se em sua documentação as características particulares que elas têm em relação ao caso de uso geral "**Abrir conta comum".**[...]

[...] embora o caso de uso **"Manter cliente"**, possa ser utilizado independentemente pelos funcionários do banco, a criação de uma conta bancária implica o registro do novo cliente, ou se já estiver cadastrado, uma possível atualização. [...]

[...] **"Realizar depósito"** porque é obrigatório depositar algum valor no momento em que o processo de abertura da conta for concluído [...]

[...] Um cliente pode eventualmente querer encerrar uma conta, porém, antes de efetuar o encerramento, algumas operações devem ser levadas a efeito. Em primeiro lugar, é preciso verificar o saldo da conta para determinar se o banco precisa devolver algum dinheiro ao cliente ou, caso a conta seja especial e estiver negativa, se o cliente precisa depositar algum dinheiro para encerrar a conta[...]

[...]serviços necessitam registrar o movimento realizado, razão pela qual ambos obrigatoriamente utilizam o caso de uso **"Registrar Movimento"**, que para fins de histórico bancário, registrar qualquer saque ou depósito porventura realizado em uma conta.[...]

[GUEDES,2009] Guedes, Gilleanes T.A . "UML2. Uma abordagem prática". 1ª ed. São Paulo. Novatec Editora,2009. Páginas 73 à 75.

Caso de Uso “Abrir Conta Comum”

Nome do caso de uso	Abrir conta comum
Caso de uso geral	
Ator Principal	Cliente
Atores secundários	Funcionário
Resumo	Este caso de uso descreve as etapas percorridas por um cliente para abrir uma conta comum
Pré condições	É necessário verificar os dados cadastrais
Pós condições	É necessário realizar um depósito inicial
Ações do ator	**Ações do sistema**
1-Fornecer dados cadastrais 4-Fornecer valor de depósito de no mínimo R$25,00	2-Verificar e validar dados 3-Registrar dados
Restrições/Validações	1-Para abrir uma conta corrente é preciso ser maior de idade
	2- É necessário comprovar estar empregado e o salário e ser maior que 500,00
	3- O valor mínimo de depósito deve ser de R$ 25,00

Caso de Uso “Abrir Conta Especial”

Nome do caso de uso	Abrir conta Especial
Caso de uso geral	Abrir conta comum
Ator Principal	Cliente
Atores secundários	Funcionário
Resumo	Este caso de uso descreve as etapas percorridas por um cliente para abrir uma conta corrente especial
Pré condições	É necessário verificar os dados cadastrais
Pós condições	É necessário realizar um depósito inicial
Ações do ator	**Ações do sistema**
Iguais as do “abrir conta comum”, exceto por fornecer a informação de que o cliente deve estar empregado e seu salario ser superior á R$ 500,00	Idênticas ao do caso de uso” Abrir conta comum”, exceto por definir o limite do cheque especial após a aprovação do depósito inicial
Restrições/Validações	1-Para abrir uma conta corrente é preciso ser maior de idade
	2- É necessário comprovar estar empregado e o salário ser maior que R$ 500,00
	3- O valor mínimo de depósito deve ser de 25,00

Caso de Uso “Validar clientes”

Nome do caso de uso	Validar clientes
Caso de uso geral	
Ator principal	Funcionário
Atores secundários	
Resumo	Este caso de uso descreve as possíveis atividades de manutenção do cadastro de clientes, ou seja, fazer a verificação do cadastro e confirmação de dados do cliente
Pré condições	Agência disponibilizar serviços
Pós condições	
Ações do Ator	**Ações do sistema**
1-Informar CPF ou CNPJ do cliente 4-Se necessário, alterar ou inserir os dados do cliente	2-Consultar cliente por seu CPF ou CNPJ **3**-Se já houver um cliente cadastrado com o CPF ou CNPJ informado, apresentar seus dados 5-Se necessário, gravar as atualizações
Restrições/Validações	1-O CPF ou CNPJ precisam ser validados
	2-Os campos: nome, endereço e data de nascimento são obrigatórios

Caso de Uso “Verificar Saldo”

Nome do caso de uso	Verificar Saldo
Caso de uso geral	
Ator Principal	Cliente
Atores secundários	
Resumo	Descreve os passos necessários para um cliente obter o saldo referente a uma determinada conta
Pré condições	Validar o cliente/Disponibilizar serviços
Pós condições	
Ações do ator	**Ações do sistema**
1-Informar o numero da conta 1- Informar a senha	1- Verificar a existência da conta 2- Solicitar a senha da conta 5-Verificar se a senha está correta 6-Emitir o saldo
Restrições/Validações	1-A conta deve existir e estar ativa
	2 – A senha deve estar correta

Fluxo Alternativo I - Conta não encontrada	
Ações do ator	**Ações do sistema**
	1-Comunicar ao cliente que o número da conta informada não foi encontrada
Fluxo Alternativo II - Senha inválida	
Ações do ator	**Ações do sistema**
	1- Comunicar ao cliente que a senha fornecida não combina com a senha da conta

Caso de Uso “Imprimir Extrato”

Nome do caso de uso	Imprimir Extrato
Caso de uso geral	
Ator Principal	Cliente
Atores secundários	
Resumo	Descreve os passos necessários para um cliente obter o extrato referente a uma determinada conta em um determinado período
Pré condições	Validar cliente
Pós condições	
Ações do ator	**Ações do sistema**
1-Informar o numero da conta 2- Informar a senha 7-Informar o período inicial e final do extrato	2-Verificar se a conta existe e está ativa 3-Solicitar a senha 5-Verificar se a senha está correta 6-Solicitar períodos 8-Listar os movimentos da conta dentro do período informado
Restrições/Validações	1-A conta deve existir e estar ativa
	2 – A senha deve estar correta
	3- Os períodos precisam estar corretos e o período inicial não pode ser maior que o final
Fluxo Alternativo I - Períodos Inválidos	
Ações do ator	**Ações do sistema**
	1-Informar ao cliente que as datas fornecidas são inválidas

Caso de Uso “Efetuar Depósito”

Nome do caso de uso	Efetuar Depósito
Caso de uso geral	
Ator Principal	Cliente
Atores secundários	Funcionário
Resumo	Descreve os passos necessários para um cliente depositar algum valor em uma determinada conta
Pré condições	Informar dados
Pós condições	
Ações do ator	**Ações do sistema**
1-Informar o numero da conta 3-Fornecer o valor a ser depositado	2-Verificar a existência da conta 4- Executar caso de uso “Registrar movimento
Restrições/Validações	1-A conta deve existir e estar ativa

Caso de Uso “Efetuar Saque”

Nome do caso de uso	Efetuar Saque
Caso de uso geral	
Ator Principal	Cliente
Atores secundários	
Resumo	Descreve os passos necessários para um cliente retirar algum valor em uma determinada conta
Pré condições	
Pós condições	
Ações do ator	**Ações do sistema**
1-Informar o numero da conta 4-Informar a senha 6-Informar o valor a ser retira-do	2-Verificar a existência da conta 3-Solicitar a senha da conta 5- Verificar se a senha da conta está correta 7-Se o valor solicitado for válido, entregar a importância ao cliente 8-Executar caso de uso “Registrar movimento”
Restrições/Validações	1-A conta deve existir e estar ativa
	2- A senha precisa estar correta
Fluxo Alternativo I - Conta comum e poupança	

Ações do ator	**Ações do sistema**
Restrições/Validações	1-Se o valor solicitado for igual ou menor que o saldo da conta, sacar o valor
	1-O valor a ser retirado precisa ser igual ou menor que o saldo da conta
Fluxo Alternativo II - Conta Especial	
Ações do ator	**Ações do sistema**
Restrições/Validações	1- se o valor solicitado for igual ou menor que o saldo da conta somado ao limite, sacar valor
	1-O valor a ser retirado precisa ser igual ou menor que o saldo da conta somado ao limite
Fluxo Alternativo III – Saldo Insuficiente	
Ações do ator	**Ações do sistema**
	1-se o valor solicitado for superior ao que o cliente pode sacar, emitir uma mensagem informando que o saldo é insuficiente e recusar o pedido

Caso de Uso “Registrar eventos”

Nome do caso de uso	Registrar Eventos
Caso de uso geral	
Ator principal	Cliente
Atores secundários	
Resumo	Descreve os passos necessários par registrar um movimento referente á um saque ou depósito
Pré condições	
Pós condições	
Ações do Ator	**Ações do sistema**
	1-Receber o numero da conta movimentada, o tipo do movimento (se é depósito ou saque), a data e o valor do movimento 2-Registrar o movimento
Restrições/Validações	

Caso de Uso “Encerrar Conta”

Nome do caso de uso	Encerrar conta
Caso de uso geral	
Ator Principal	Cliente
Atores secundários	Funcionário
Resumo	Este caso de uso descreve as etapas percorridas por um cliente para encerrar uma conta
Pré condições	Informar dados
Pós condições	
Ações do ator	**Ações do sistema**
Iguais ao do caso de uso abrir conta comum.	Idênticas ao do caso de uso Abrir conta comum, exceto por definir a data de aniversário da conta, que não necessariamente será a data de abertura da conta, uma vez que a abertura de uma conta poupança não obriga depósito.

Caso de Uso “Verificar dados”

Nome do caso de uso	Verificar dados
Caso de uso geral	
Ator Principal	Funcionário
Atores secundários	
Resumo	Este caso de uso descreve as possíveis atividades de manutenção do cadastro de clientes ou seja, fazer a verificação do cadastro e confirmação de dados do cliente.
Pré condições	Agência disponibilizar os serviços
Pós condições	
Ações do ator	**Ações do sistema**
1-Informar CPF ou CNPJ do cliente	2-Consultar cliente por seu CPF ou CNPJ
4- Se necessário, alterar ou inserir os dados do cliente	3-Se já houver um cliente cadastrado com o CPF ou CNPJ informado, apresentar seus dados
	5-Se necessário, gravar as atualizações
Restrições/Validações	1- O CPF e CNPJ precisam ser validados
	2 - Os campos nome, endereço, e data de nascimento são obrigatórios