ESTIMATIVAS E MÉTRICAS: (Medidas de Software)

Definição de um Método para a Engenharia de Requisitos baseado nas Necessidades de Informação

- "Quanto custa a revista Super Interessante?"
- "Quais as tarefas a serem entregues pelos alunos no dia 09/MAR/2010?"
- "Qual a carga energética horária, gerada pela CPFL, no dia 02/FEV/2010?"

Objetivos:

(a) Identificar as medidas e medições de software
(b) Medir a complexidade dos sistemas de software
(c) Estimar os pontos fracos e fortes das medições de software

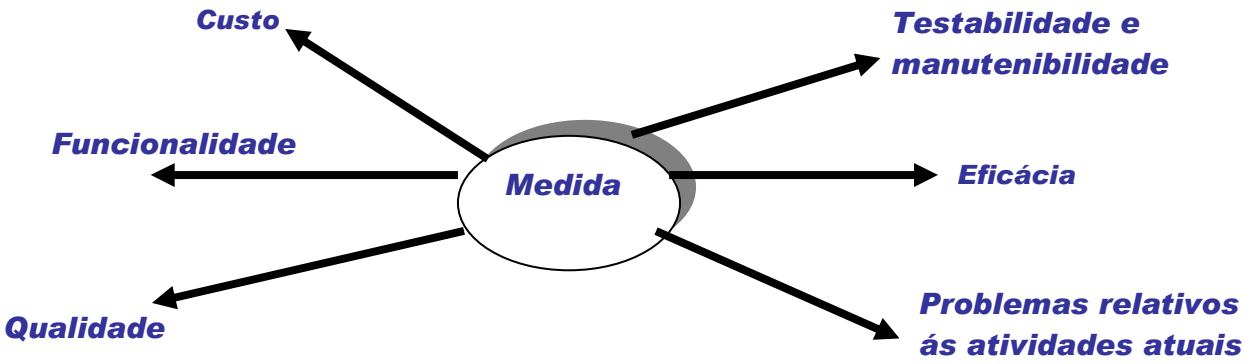


Figura A, conhecimento obtidos a partir das medições de software

Um medidas de software é o mapeamento de um conjunto de no mundo da engenharia de software, em um conjunto de construções matemáticas , tais como números, símbolos ou vetores de números (McClure, 1994) .

O conhecimento obtido da medição de software é demostrado na Figura A, resumida abaixo:

(a) **Custo**, que afeta um planejamento relevante para os projetos futuros
(b) **Testabilidade e manutenibilidade** de processo e produtos atuais
(c) **Eficácia** de um produto de software
(d) **Qualidade** do processo-produto direcionando atributos, como :
    1. Confiabilidade
    2. Portabilidade e
    3. manutenibilidade do sistema de software fornecido
(e) **Funcionalidade e facilidade** de utilização de um produto do ponto de vista do usuário

Um medição de software é uma técnica ou método que aplica medidas de software a uma classe de objetos de engenharia de software de forma a atingir um objetivo predefinido. Cinco característica da medição podem ser identificadas(Basili, 1988 e 1989)

1) **Objeto de medição** variando de produtos(por exemplo, código – fonte, projeto de software, requisitos de software, casos de teste de software) a processos(e.i: processo de projeto arquitetônico, processos de codificação e teste unitário, processo de teste de sistema) e projetos.
2) **Propósito da medição** tal como caracterização, avaliação, analises e previsão.
3) **Fonte de medida** tal como projetista, testadores e gerentes de software.
4) **Propriedade medida** tal como custo de software, confiabilidade, manutenibilidade, tamanho e portabilidade
5) **Contexto da medição** onde os artefatos de software são medidos em ambientes diferentes(incluindo pessoa, tecnologia, recursos disponíveis), que são especificados antecipadamente antes da aplicação de algumas medidas de software.

Métodos próprios para determinar, de maneira quantitativa, a extensão dos atributos caracterizadores de determinado projeto, processo ou produto, incluindo fórmulas e diretrizes para o cálculo de seu próprio valor, conforme define Pressmam <1995>, as métricas de software têm suas proposições feitas já há algumas décadas sem ter, ainda, sido devidamente assumidas como técnicas condizentes com o desenvolvimento profissional na indústria informática.

O termo é extremamente amplo. Um universo de itens e elementos podem ser medidos tanto no software quanto na sua relação com o ambiente onde esteja implantado.

Quanto ao produto software, é possível estabelecer medições na linearidade de sua vida útil (ciclo de vida) ou em seu espaço tridimensional, onde, conforme demonstra a figura 1, as direções dimensionais englobam:

- Complexidade algorítmica, na busca da eficiência de processamento;
- Qualidade do produto, na busca pela capacidade de atendimento do cliente;
- Produção, visando o melhor mecanismo de desenvolvimento que venha a atender o cliente dentro dos limites de tempo e valor negociados.

Quanto ao ciclo de vida, devido à necessidade de medir todos os segmentos da vida útil de um software, há ao menos um método apropriado para cada um desses segmentos, desde o momento inicial de sua criação (a concepção) até sua operacionalização (passando por sua construção), momento em que as medições servem para decidir quando de sua manutenção. A figura 2 demonstra um ciclo clássico.

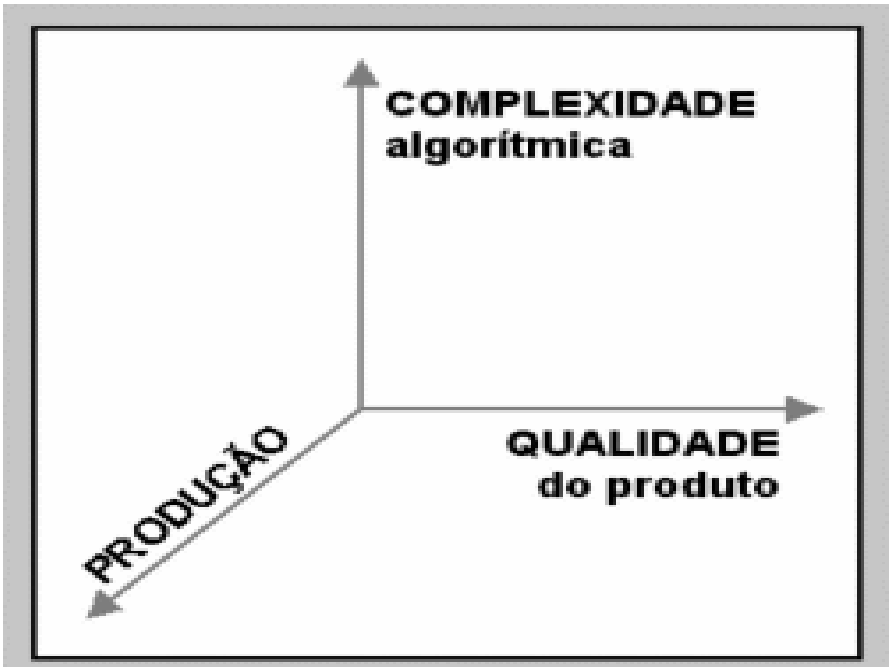


Figura 1 - As 3 dimensões do software, passíveis de mensuração

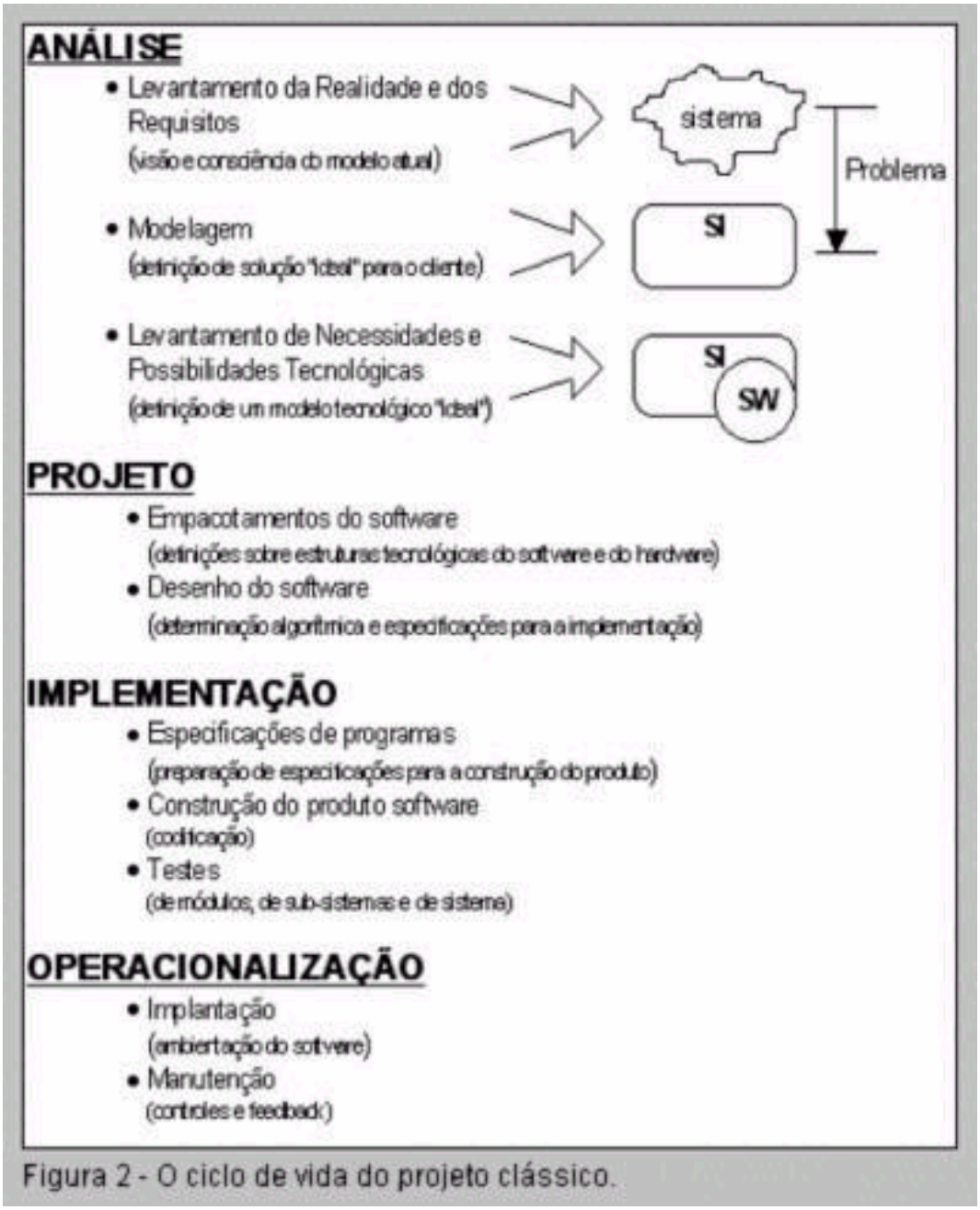


Figura 2 - O ciclo de vida do projeto clássico.

Então, o que se pode resumir é que não importando a visão que se tenha do software, multidimensional ou linear, cada uma de suas partes, etapas e subdivisões podem e devem ser medidas. E somente com métricas apropriadas consegue-se realizar tais estimativas com significativa precisão e confiabilidade.

O software e sua produção têm características muito próprias de intangibilidade. Tal condição é oriunda de fatores como complexidade implícita e alto grau de subjetividade em sua concepção, conforme denota SOFTWARE ENGINEERING; EUA: John Wiley & Sons, 1990, ;Jones <1990>.

Tais características dificultam muito uma mensuração precisa e absoluta. Todos os autores envolvidos e dedicados a tal estudo bem o sabem. Orçar produto e produção e auferir padrões de qualidade nesta área é bem mais difícil que em outras. É bom ressaltar que os métodos são apropriados para estimar, resguardando toda a imprecisão intrínseca de processos tão próximos da criatividade, do estado de espírito e do humor humanos.

Esses métodos de estimação se aplicam, enfim, desde o orçamento até o controle de desempenho dos produtos software e todas elas, as estimativas de desenvolvimento, etapa a etapa, as estimativas de gerenciamento da produção e as estimativas do gerenciamento do desempenho do produto são igualmente importantes para adequar tempo e esforço de trabalho às expectativas profissionais exigidas pela engenharia de software.

ENGENHARIA DE SOFTWARE:

O software, por estar envolvido com todas as disciplinas, vem se tornando o componente principal de muitas atividades complexas, o que aumenta, a cada dia, o desafio de produção e a necessidade de técnicas poderosas e altamente especializadas.

Exige, então, engenharia peculiar, específica, minuciosa, que consiga abranger parâmetros e conceitos suficientes para o domínio de todas suas derivações. Esta é a Engenharia de Software.

Segundo ENGENHARIA DE SOFTWARE; São Paulo: Makron Books, 1995',' ;Pressman <1995>, é derivada das engenharias de sistemas e de hardware, abrangendo elementos básicos como o método, o ferramental e o procedimento.

Conforme explica PRINCIPLES OF SOFTWARE ENGINEERING MANAGEMENT; Inglaterra: Addison Wesley Longman, 1990',';Gilb <1990>, ela é um processo, de projeto, construção e manutenção de algo, que visa especificações determinadas de custo e qualidade. Ele também aceita a idéia de que a engenharia é a aplicação da heurística para resolver problemas e considera “engenharia de software” um termo genérico para um conjunto de disciplinas especializadas de engenharia, que envolve a da lógica, a da informação e da pessoa, cada qual com suas próprias áreas avançadas de especialização. PFLEEGER, Shari L.; SOFTWARE METRICS: A RIGOROUS AND PRATIC APPROACH; 2a edição; Inglaterra: International Thomson Computer, 1997',';Fenton & Pfleeger <1997> descrevem a engenharia de software como a coleção de técnicas que aplicam uma abordagem de engenharia à construção e ao suporte de produtos software e comentam que as atividades desta incluem administração, orçamento, planejamento, modelagem, análise, especificação, projeto, implementação, testes e manutenção.

Como um ramo integrante da disciplina de engenharia, a ciência da produção de software não pode se furtar às necessidades de mensuração, afinal de contas, ninguém concebe a idéia de práticas de engenharia civil, elétrica ou mecânica - só para citar algumas - sem suas regras básicas de mensuração.

Produção eficaz de software (não importando a dimensão deste ou o porte do desenvolvedor) é uma questão de engenharia e esta é uma ciência e a ciência não se pratica, ou efetiva, sem que haja métodos apropriados para medir fenômenos.

ESTIMATIVA AO ALCANÇE DE TODOS:

É bom que se ressalte: Aplicação de métricas não é privilégio de alguns poucos desenvolvedores de grande porte. Todo desenvolvedor pode e deve utilizar recursos apropriados para orçar, gerenciar e dominar seus projetos.

Existem diversos modelos para mensuração de sistemas computacionais, mais ou menos empíricos, comercialmente viáveis, desde algumas formas resumidas ao uso de experiências passadas como único

guia, até métodos complexos e relativamente completos, que associam experimentação, inferência e formulação matemática na elaboração de formas seguras de se medir software. Autores como ENGENHARIA DE SOFTWARE; São Paulo: Makron Books, 1995;Pressman <1995> e USE OF METHODOLOGIES: AN EMPIRICAL ANALYSIS OF THEIR IMPACT ON THE ECONOMICS OF THE DEVELOPMENT PROCESS; in: European Journal of Information Systems; no 6/97; Inglaterra: Operational Research Society, 1997',Chatzoglou <1997> concordam que métricas nunca conseguirão ser uma ciência exata, de um padrão único, pois muitas variáveis - humanas, técnicas, ambientais e políticas - afetam custos e recursos de produção. Não há um método absoluto, ou seja, um método que consiga ser melhor que todos os outros, sob todos os aspectos, em qualquer situação. O método deve ser escolhido dependendo das características particulares do sistema que se pretenda desenvolver e de circunstâncias especiais que envolvam o problema a solucionar.

Este site traz uma relação de métodos voltados a orçamento ou mensuração de qualidade, apresenta o USO INTEGRADO DE MÉTRICAS, como alternativa viável de aplicação de tais métodos para desenvolvedores de todos os portes e oferece serviços de consultoria para aqueles que pretendam ingressar um universo definitivamente profissional no contexto dos projetos de software.

As métricas podem ser qualificadas, segundo aos tipos de aplicações que se faz delas, em:

- Orçamento de produto, necessária para planejamento de produção. Proporciona conhecimento de esforços, prazos e custos de desenvolvimento;
- Qualidade, uma referência genérica sobre a capacidade de atendimento às expectativas que se possa fazer do produto, que pode ser medida tanto no próprio produto quanto em seu desenvolvimento. Qualidade, então, divide-se em:
  - Qualidade do produto, necessária para estabelecer a devida relação de mercado, como informação competitiva;
  - Qualidade do processo produtivo, necessária à determinação de grau de maturidade e competência do desenvolvedor.

Esta página, e suas subseqüentes, enfocam qualidade de processo e apresentam uma relação de métodos, técnicas e métricas destinadas a tal. Algumas considerações sobre o tema também se apresentam, ao final.

Medições - Controvérsias e argumentações;

a) Qual é a mas apropriada?;
b) Como os dados que são compilados deve ser usados?;
c) é justo usar medições para comparar pessoas, processos e produtos?

Eng. Software (Processo)	Software (Produto)

Estimativas

1. esforço humano (pessoas-mês)
2. Duração cronológica do projeto(tempo-calendário).
3. Custo (moeda- R$)
4. Experiência passada (Referencia -guia)
5. Analise de riscos,
6. impactos
7. determinação criteriosa de prazos
8. atribuições e responsabilidades
9. monitorarão e controle

Objetivo:

proporcionar uma idéia do custo do sistema, programa ou módulo que estivermos dimensionando.

Projeto sem medidas (inspirado do céu)

Qualquer técnica de gerenciamento de projeto não é capaz de salvar o projeto mal dimensionado Compromete :

- relacionamento AS vs. usuário
- qualidade

Medidas de Software

Medição é comum na nossa vida

Exemplos :peso, temperatura, voltagem, dimensões físicas, etc.)

Unidades de medida:

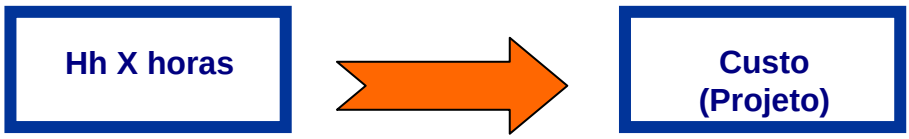
os projetos, sejam sistemas, programas ou módulos, necessitam de uma unidade de medida para que possam ser mensuráveis.

Prazo: (calendario-dias-mês)

- depende do dimensionamento do sistema.
- As técnicas
- fórmulas para calcular o dimensionamento

Custo: (Homem/hora -Hh)

- Hh a ser gasto e o índice de produtividade da equipe que irá trabalhar no projeto
- índice de produtividade



Escopo

- item principal para o sucesso do projeto.
- mal definidos deixam abertura para o usuário solicitar modificações constantemente.
- Relacione as tabelas , arquivos, funções/processos

Ex: Um erro muito comum "relatórios"

o Analista relaciona todos os arquivos, tabelas, funções de (cadastros, consulta) em tela e coloca um item chamado "relatórios, etc".

DICA(Solução)

- Coloque os nomes dos relatórios,
- Relatórios tela, papel ou meio magnético a sua saída,
- Caso o usuário não souber quais relatórios ele quer,
- abstraía 3 ou 4 com os nomes definidos e suas funções
- (Ex.: Lista de Materiais, Demonstrativo financeiro com o total de gastos e vendas por período),

deixe bem claro no escopo que qualquer relatório além desses descritos será objeto de outra proposta.

Observação:

- muitas técnicas de dimensionamento independem da plataforma (hardware, software, linguagens e etc.,
- só irão fazer diferença no cálculo do prazo e custo do sistema.

Dificuldades: Eng. Software

1) concordar sobre o que medir; e
2) para avaliar as medidas que são obtidas

Razões para medir um software:

a) Medir a qualidade do produto
b) Avaliar a produtividade das pessoa que produzem o produto
c) Avaliar os benefícios (qualidade) derivados dos novos métodos e ferramentas utilizados
d)Formar uma linha básica para estimativas
e) ajudar a justificar os pedidos de novas ferramentas ou treinamento adicional.

Tipos de medidas físicas

1) Medidas diretas:

Ex. do mundo real: Comprimento de um parafuso

Software (Produto)

a) Custo;
b) esforço aplicado;
c) LOC (Line of Code) Linhas de código produzidas
d) velocidade de execução;
e) Tamanho da memória ;
f) defeitos registrados ao longo de certo espaço de tempo

2) Medidas indiretas:

Ex. do mundo real: Qualidade dos parafusos produzidos, medido pela contagem de parafusos rejeitados

Eng. Software

a) funcionalidade;
b) qualidade;
c) complexidade;
d) eficiência;
e) manutenibilidade.

Divisão das métricas em categorias

1) Métricas de produtividade:

- concentra-se na saida do processo de Eng. Software

2) Métricas de qualidade:

- Oferecem um indicação de quão estreitamente e o software(conforme as exigências do cliente)

3) Métricas Técnicas:

- concentram-se na características do software (ex: complexidade lógica, grau de modularidade

4)Métricas orientadas ao tamanho

- Usadas para compila as medições diretas da saída e da qualidade de Eng. Software

5) Métricas orientadas para a função:

- Oferecem medições indiretas

6) Métricas Orientadas ás pessoa;

- compilam informações sobre as pessoas que desenvolvem o software(técnicas, ferramentas, etc.)

NECESSIDADE DE ESTIMAR:

Com uma analogia bem delineada com a engenharia civil, ANÁLISE DE PONTOS DE FUNÇÃO; Rio de Janeiro: Infobook,1996';Braga <1996> afirma que não se admite a idéia de construir uma estrada sem antes desenhá-la, executar análises ambientais, dimensioná-la, planejar transposições de obstáculos, calcular custos e analisar indicadores. O autor se pergunta por que empresas ainda insistem em desenvolver projetos de software sem compilar um conjunto de informações essenciais.

Por vários motivos, os desenvolvimentos de projetos de software vêm fracassando em expectativas, prazos e custos propostos. O principal deles é que, muitas vezes, não haverá sequer um planejamento de como a idéia modelada pelos levantamentos de requisitos e necessidades dos clientes possa ser transformada em um produto. Muitas vezes, sequer haverá um modelo e/ou projeto que permita tal planejamento.

Projetos de software, assim como qualquer outro projeto, de qualquer ramo da engenharia, precisam de planejamento para conquistar sucesso.

É certo que esta engenharia é diferenciada das outras por desenvolver um tipo de produto tão particularizado e intangível. Mas, dentro dos padrões atuais de complexidade de produto e competitividade de mercado, não se pode imaginar, sequer aceitar a idéia de, tal produção sem projeto, sem medida ou sem planejamento.

Esse planejamento necessário só se pode dar com o domínio de informações como dimensão, esforço, prazo, valor e disponibilidade de recursos para a realização.

Essa realidade é bem expressa por CONTROLE DE PROJETOS DE SOFTWARE; Rio de Janeiro: Campus, 1991';DeMarco <1991>, quando afirma que “não se pode controlar o que não se pode medir” e comenta que, apesar de ser óbvio o forte elo entre medição e controle, é uma idéia nova para a maioria dos gerentes de software que acalentam a ilusão de que se consegue levar adiante um projeto, sob total controle, sem nunca terem medido coisa alguma. Tal afirmação foi feita em 1991; será que este cenário mudou, evoluiu de lá para cá?

ODELL, James J.; ANÁLISE E PROJETO ORIENTADOS A OBJETO; São Paulo: Makron Books, 1995',;Martin & Odell <1995> dizem que a capacidade de criar software não está acompanhando a evolução do hardware e clamam por uma revolução industrial do software, pois os projetos ficam cada vez mais difíceis de execução à medida em que os sistemas tornam-se mais complexos em face às novas condições criadas pelo desenvolvimento da tecnologia do hardware. Ainda que mais complexo, esse software também precisa ser mais confiável, ter um prazo menor para ser desenvolvido e não ter seu custo muito elevado.

FLEURY, Afonso C. C.; INDICADORES DA QUALIDADE E PRODUTIVID_ADE NA INDÚSTRIA BRASILEIRA;

in: revista Indicadores da Qualidade e Produtividade; vol.1 - no 1 - fev/92; São Paulo: depto. Engenharia de Produção da EPUSP, 1992',';Muscat & Fleury <1992> definem que estratégias competitivas podem ser focadas em níveis de competitividade, tais como:

(a) custo;
(b) qualidade;
(c) tempo;
(d) flexibilidade e
(e) inovação.

Na área de software, pode-se afirmar que inovação e flexibilidade são prerrogativas indiscutíveis para a integração em um mercado onde as evoluções tecnológicas acontecem em velocidades quase inacreditáveis; ambas são uma questão de projeto bem elaborado e equipe produtiva bem estruturada. Custo e qualidade só tornam-se possíveis com bons projetos. Tempo é uma questão de planejamento de produção. Projetos competentes só são possíveis se bem dimensionados.

Diante de tal cenário, algumas observações podem ser feitas:

- A indústria de software, de uma maneira geral, tem sido desenvolvida sem grandes critérios profissionais e os novos paradigmas de software: complexo; de grandes proporções; manutenível; seguro; interativo; completo; confiável e econômico, permitem cada vez menos o amadorismo e a produção artesanal;
- A manufatura do software precisa de infra-estrutura confiável, transparente, ágil, harmônica, gerenciável e evolutiva;
- Esse universo de produção precisa associar cérebro, método e criatividade e isso só é possível com engenharia;
- A engenharia necessária precisa de mensurações;
- Mensurações exigem métricas, e estas podem ser apresentadas segundo uma cronologia de desenvolvimento:
  1) **LOCs (Lines of Codes - Linhas de Código):** décadas de 1950 e 60)_O maior problema estava na linguagem e na experiência do programador. Um programador experiente pode ter uma produtividade 3 vezes ou mais que um programador iniciante.
  2) **Regressão Linear** - década de 1960
  3) **Delphi** - final da década de 1960, início da de 70
  4) **Halstead (Maurice Halstead, Univ. de Purdue)** 1972:_Baseada nos operandos (itens de dados) e operadores (comandos de linguagem). O problema era o mesmo que o LOCs, além disso, só tendo o código pronto, poderemos medir, não serve por não ter sido feito o dimensionamento por esse método. Não podemos utilizar duas unidades de medida diferentes, teríamos que ter cálculos de compensação e isso é inviável comercialmente.
  5) **FPA (Allan J. Albrecht, IBM) 1974 :_APF (Análise de Pontos de Função):** um dos mais utilizados atualmente. Baseia-se na visão do usuário, apesar do pessoal de informática ter dominado a técnica. Contabilizam-se dados (Arquivo Lógicos Internos e Arquivo de Interface Externa) e funções (Entrada Externa, Saída Externa e Consulta Externa). Atualmente a técnica de APF contém diversas ramificações, mas a linha base continua ser a determinada no CPM 4.1 (versão atual Jan/2000) pelo IFPUG (International Function Point Users Group).
  6) **Número Ciclomático (Thomas J. McCabe) 1976**
  7) **Tabela de conversão para Linguagens de Programação (Capers Jones) 1976**
  8) **Putnam's Slim Model (W. Putnam) 1978**
  9) **COCOMO (Barry Bohem, Univ. do Sul da Califórnia) 1981**
  10) **Feature Points (Capers Jones, Software Productivity Research) 1986**
  11) **COCOMO II (Barry bohem e outros, Univ. do Sul da Califórnia) 1994**
  12) **OO-FPA (Thomas Fetcke e outros, Univ. de Quebec em Montreal) 1997**

13) **Bang:** Criada por Tom de Marco (conhecido como um dos "papas" da Análise Estruturada), baseia-se no cálculo de dados e funções, mas tem o incoveniente de termos o anteprojeto para dimensionamento do sistema e posterior comparação.
14) Uso Integrado de Métricas (André L. Trindade, USP-CEETEPS) 1999

LINHAS DE CÓDIGO:

A primeira e mais natural das formas de medir o tamanho de um sistema computacional, criando a unidade de medida mais comum e mais utilizada, até então, para retratar dimensões de software: as Linhas de Código - **LOC** (Lines Of Code). A unidade de medida em si, é bastante utilizada, ainda hoje, em outras métricas.

Não se pode dizer que chega a ser um método de mensuração; apenas uma técnica de contagem, bastante empírica por sinal, que consegue traduzir dimensão em número de linhas de instruções de linguagens específicas de programação.

Inicialmente, a estratégia seria de uma mensuração em programas já construídos e uma associação de elementos - como especificidade de rotinas e determinação de linguagem - com a experiência profissional e a capacidade organizacional de quem aplica tal recurso.

Só se consegue uma aplicação razoável se houver uma cultura de documentação histórica de dados, constantemente reavaliados e atualizados, alta padronização dos tipos de rotinas cadastradas e mensuradas e trabalho com linguagens específicas e não mutáveis.

Características do KLOC

- Medidas diretas
- criar uma tabela: relaciona cada projeto de desenvolvimento de softeste que foi concluído no decorrer dos últimos anos.
- A partir dos dados brutos contidos na tabela, podem ser extraídos as seguintes formulas:

Produtividade = KLOC pessoa-mês
Qualidade = Defeitos/KLOC
Custo = $ / LOC
Documentação = Paginas de doc / LOC

Projeto	Esforço	$	KLOC	Pags. (Doc)	Erros	pessoas
aaa-01	24	168	12,1	365	29	3
ccc-04	62	440	27,2	1224	86	5
fff-03	43	314	20,2	1050	64	6
.....	.....	....	.....	.....	.....	....
.....	.....	....	.....	.....	.....	....
.....	.....	....	.....	.....	.....	....

Segundo a Figura para o projeto aaa-01 temos :

KLC = 12,1 (Mil linhas de código)
Esforço[a] = 24 pessoas-mês
Custo[a] = 168 mil dólares

[a]: Representa todas as ativ. de Eng. Software: (analise, projeto, codificação e teste)

Informações adicionais:

Documentos = 365 páginas
Erros = 29 (após a entrega - 1º ano de operação)
Pessoas = 3 (trabalharam no projeto)

Desvantagens: (KLOC)

Linhas de código (KLOC) cria Controvérsias
Grupos
a**) A favor;**

- Podem ser facilmente contados
- Todos os software têm

b) Em contra

- Depende da ling. de Prog. e Experiência do Prog.
- Penalizam programas bem projetados(mais curtos)
- Não acomodam ling. não procedimentais
- as LOC devem ser estimadas antes que a analise e o projeto tenham sido concluídas

Linhas de Código - *VANTAGENS*

Pode-se citar, como ponto alto, o fato da concepção de uma unidade perceptível de mensuração, para aqueles que trabalham com criação dos programas de computador. Também não se pode ignorar a característica pioneira de tal técnica.

Linhas de Código - *DESVANTAGENS*

Tal forma de medir não admite comparações entre tecnologias, ou seja, não há sentido em comparar linhas de código escritas em Cobol, com outras escritas em Visual Basic, por exemplo. **Trindade** <mailto:andre@tw.eng.brmailto:andre@tw.eng.br> pergunta: Não há? E responde: É exatamente esse o campo pesquisado por Capers Jones já há alguns anos, que culminou na publicação da tabela de conversão para linguagens de programação.

Mesmo quando há comparação entre duas rotinas de mesma função, escritas com a mesma linguagem, pode haver, e é praticamente certo que haverá, diferenças entre as dimensões das rotinas criadas, pelo simples fato de que as experiências dos programadores não são as mesmas, e um terá um algoritmo mais limpo que outro.

Não considera fatores de interferência possíveis, como capacitação técnica da equipe desenvolvedora, condições ambientais para o trabalho, recursos de hardware e software utilizados, etc.

Por fim, mesmo que os itens anteriores possam ser equacionados convenientemente, ainda resta a exigência pela manutenção de históricos sobre as dimensões já realizadas, devidamente tratadas em suas inferências, para que se possa utilizar tais informações como base estimativa para projetos vindouros. A cultura desse tipo de expediente ainda é insípida, principalmente em empresas de menor porte e, ainda, em países de menor desenvolvimento tecnológico e científico.

Bibliografia:

a) Engenharia de Software Teoria e Prática. James Peters / Witold Pedrycz. Editora Campus(2001)

b) Engenharia de Software – Fundamentos, Métodos e Padrões, Wilson de Pádua Paula Filho. Editora LTC (2003)

c) Engenharia de Software, David Gustafson. Coleção Schaum, Editora Bookman(2003)

d) Teste de Software, Produzindo Sistemas Melhores e Mais Confiáveis, Leonardo Molinari, Editora Erica-2003