Métricas e Estimativas de Software

Imagine que você faça parte de uma equipe de Rally, e que você e sua equipe tenham que atravessar um deserto enorme e cheio de obstáculos e que 50% dos fatores críticos de sucesso dependam do seu navegador para estimar o tempo do percurso e a distância. Agora imagine-se diante de um projeto de Sistemas de Informação onde 50% dos fatores críticos de sucesso estão nos prazos e custos. Certamente você terá que fazer uma eficiente métrica e estimativa de software.

As métricas e estimativas de software vem se tornando um dos principais tópicos na Engenharia da Informação com a crescente exigência de seus consumidores pela qualidade, rapidez, comodidade e baixo custo de implantação e manutenção, é impossível não enxergar tais técnicas como alavanca para um produto de melhor qualidade, com custos adequados. Mas existem ainda muitas barreiras que impedem os profissionais da área de utilizarem tais técnicas ou de o fazerem de maneira errônea, embora a literatura disponível atualmente sobre engenharia da informação seja relativamente ampla e variada, o que nos leva a questionar: Por que as métricas e estimativas de software propostas para o desenvolvimento de sistemas não são fiéis à realidade e à dimensão do problema? Tais técnicas acompanharam a rápida evolução do setor?

Tais questões nos levam a crer que algumas métricas e estimativas de software ficaram obsoletas e outras ganharam vigor nas épocas atuais, quando a busca da qualidade de software vem crescendo com grande rapidez.

Acredito que o ato de medir e estimar é a parte mais importante de um projeto de sistema bem-sucedido e alguns fatos como: a falta de maturidade, o desinteresse das empresas de desenvolvimento de sistemas e a baixa popularidade deste assunto entre os profissionais da área de informática são algumas das principais causas para o insucesso e o alto custo dos sistemas de informação.

O termo métrica de software refere-se à mensuração dos indicadores quantitativos do tamanho e complexidade de um sistema. Estes indicadores são, por sua vez, utilizados para correlatar contra os desempenhos observados no passado afim de derivar previsões de desempenho futuro.

A métrica de software tem como princípios especificar as funções de coleta de dados de avaliação e desempenho, atribuir essas responsabilidades a toda a equipe envolvida no projeto, reunir dados de desempenho pertencentes à complementação do software, analisar os históricos dos projetos anteriores para determinar o efeito desses fatores e utilizar esses efeitos para pesar as previsões futuras. Estes princípios nos permitem prever o resto do processo, avaliar o progresso e reduzir a complexidade, como numa prova de rally, onde a cada corrida ficamos mais esclarecidos da condições e limites da equipe.

As Métricas Orientadas ao Tamanho

Métricas de software orientadas ao tamanho são medidas diretas do software e do processo por meio do qual ele é desenvolvido. Se uma organização de software mantiver registros simples, uma tabela de dados orientada ao tamanho poderá ser criada. A tabela relaciona cada projeto de desenvolvimento de software que foi incluído no decorrer dos últimos anos aos correspondentes dados orientados ao tamanho deste projeto. A partir dos dados brutos contidos na tabela, um conjunto de métricas de qualidade e de produtividade orientadas ao tamanho pode ser desenvolvido para cada projeto. Médias podem ser computadas levando-se em consideração todos os projetos.

As métricas orientadas ao tamanho provocam controvérsias e não são universalmente aceitas como a melhor maneira de se medir o processo de desenvolvimento de software. A maior parte da controvérsia gira em torno do uso das linhas de código (LOC) como uma medida-chave.

Os proponentes da afeição de linhas de código afirmam que as mesmas são o "artefato" de todos os projetos de desenvolvimento de software que podem ser facilmente contados, que muitos modelos existentes usam LOC ou KLOC (milhares de linhas de código) como entrada-chave e que já existe um grande volume de literatura e de dados baseados nas linhas de código. Por outro lado, os opositores afirmam que as medidas LOC são dependentes da linguagem de programação utilizada na codificação do projeto, que elas penalizam programas bem projetados, porém mais curtos, que elas não podem acomodar facilmente linguagens não-procedurais e que seu uso em estimativas requer um nível de detalhes que pode ser difícil de conseguir (isto é, o planejador deve estimar as linhas de código a ser produzidas muito antes que a análise e o projeto tenham sido construídos).

Essa forma de medida foi uma herança do modelo de manufatura em que os CIO'S podiam determinar os recursos necessários para uma "corrida", contando o número de produtos manufaturados necessários. Essa métrica não leva em consideração o fato de que o desenvolvimento envolve um custo relativo ao ambiente ou linguagem de programação utilizada. Por exemplo, em orientação a objeto (OO), a flexibilidade da ferramenta no uso de mecanismos de herança dilui o resultado final da contagem de linhas.

A contagem de linhas de código pode ser uma medida do que foi feito, e não uma medida a ser utilizada para previsão.

Métricas Orientadas à Função

Consiste em um método para medição de software do ponto de vista do usuário, que determina de forma consistente o tamanho e complexidade de um software, sob a perspectiva do usuário. Ele dimensiona um software, quantificando a funcionalidade proporcionada ao usuário a partir do seu desenho lógico. Ou seja, são medidas indiretas do software e do processo por meio do qual ele é desenvolvido. Em vez de contar linhas de código, a métrica orientada à função concentra-se na funcionalidade ou utilidade do programa. Uma abordagem foi sugerida baseada nesta proposta chamada de pontos-por-função (function point). Os pontos-por-função (FP's) são derivados usando-se uma relação empírica baseada em medidas de informações e complexidade do software.

Um dos princípios da análise de pontos-por-função focaliza-se na perspectiva de como os usuários "enxergam" os resultados que um sistema produz. A análise considera as várias formas com que os usuários interagem com o sistema, com os seguintes objetivos:

1. Fornecer medidas consistentes;

2. Medir funcionalidades que o usuário solicita ou recebe;

3. Independência da tecnologia;

4. Método simples.

As métricas orientadas à função apresentam vários benefícios, dentre eles podemos citar o seguintes:

1. Uma ferramenta para dimensionar aplicações;

2. Um veículo para quantificar custo, esforço e tempo;

3. Um veículo para calcular índices de produtividade e qualidade;

4. Um fator de normalização para comparar software.

Tal métrica parece ser útil e funcional para o desenvolvimento tradicional, mas apresenta algumas falhas com o modelo de desenvolvimento em orientação a objeto (OO), pois alguns atributos do design em OO invalidam o cálculo de alguns pontos-por-função. As características fundamentais de OO têm efeito de reduzir a validade da contagem de funções para a avaliação de esforço e recursos necessários para a execução de um projeto.

A métrica de pontos por função foi originalmente projetada para sistemas de informação comerciais. Para acomodar estas aplicações, a dimensão dos dados foi enfatizada para a exclusão de dimensões funcionais e de controle. Por esta razão, a medida de pontos por função era adequada para muitos sistemas de engenharia. Um número de extensões para a medida básica de pontos por função tem sido proposto para remediar esta situação.

Uma extensão de pontos por função chamada "feature points" (ou, pontos característicos) é uma evolução da medida de pontos por função que pode ser aplicada a sistemas e aplicações de engenharia de software. A medida "feature points" acomoda aplicações em que a complexidade algorítmica é alta. Sistemas real-time de controle de processos e outros apresentam alta complexidade algorítmica, e são receptivos a métrica de "feature points".

Para computar o "feature point", valores do domínio são contados e ponderados. A métrica "feature point" conta uma nova característica de software, os algoritmos. Um algoritmo é definido como "um problema computacional que é incluído com um programa de computador específico". Inverte uma matriz, decodificar um bit de string ou manusear uma interrupção são exemplos de algoritmos.

Outra extensão de pontos por função para sistemas real-time e produtos de engenharia tem sido desenvolvido por Boeing. A aproximação de Boeing integra a dimensão dos dados de software com dimensões funcionais e de controle para obter uma medida orientada à função, chamada pontos por função 3D, que é receptiva a aplicações que enfatizem capacidades de função e controle. Características de todas as três dimensões dão "contadas, quantificadas e transformadas" em uma medida que fornece uma indicação da funcionalidade fornecida pelo software.

Contagem de dados retidos (a estrutura de dados interna do programa, isto é, arquivos) e dados externos (entradas, saídas e referências externas) são usados com medidas de complexidade para derivar uma contagem da dimensão de dados.

A dimensão funcional é medida considerando "o número de informações internas requeridas para transformar entradas em dados de saída". Para os propósitos da computação de pontos por função 3D, uma transformação é vista como uma série de passos de processamento que são limitados por regras semânticas estabelecidas. Como uma regra geral, a transformação é concluída com um algoritmo que resulta em uma mudança fundamental para dados de entrada como são processados para se transformarem em dados de saída. Passos de processamento que adquirem dados de um arquivo e simplesmente os coloca na memória do programa não poderia ser considerado uma transformação. O dado não sofreu nenhuma mudança.

O nível de complexidade associado a cada transformação é uma função do número de passos de processamento e o número de regras semânticas é que controla os passos de processamento.

A dimensão de controle é medida pela contagem do número de transições entre os estados. Um estado representa algum modo internamente observável de comportamento e uma transição ocorre como resultado de algum evento que força o software ou sistema a mudar seu comportamento, isto é, mudar seu estado.

Quando pontos por função 3D são computados, transições não são associadas a valores de complexidade.

Nota-se que pontos por função, "feature points" e pontos por função 3D representam a mesma coisa – "funcionalidade" ou "utilidade" fornecida pelo software. De fato, cada uma destas medidas resulta no mesmo valor se somente a dimensão de dados de uma aplicação é considerada.

Para sistemas real-time mais complexos, a contagem "feature points" é de 20 a 35% mais alta que a contagem determinada usando somente pontos por função.

Pontos por função (e suas extensões), como a medida LOC, é controversa. Os proponentes acham que FP é independente da linguagem de programação, tornando-se ideal para aplicações usando linguagens convencionais e não procedurais, e que ela é baseada em dados que são conhecidos muito cedo na evolução do projeto, fazendo a FP mais atrativa como uma estimativa mais próxima.

Oponentes acham que o método requer alguma prestidigitação em que a computação é baseada em dados subjetivos, que a contagem das informações de domínio (e outras dimensões) podem ser difíceis de coletar após terminado o projeto, e que FP não tem significado físico direto. É só um número.

Métricas Voltadas para Orientação a Objeto

Muitas métricas já foram desenvolvidas para gerações passadas de tecnologia e, em muitos casos, são usadas até para desenvolvimento OO, porém não são muito coerentes, pois a diferença entre sistemas tradicionais e sistemas OO são muito grandes.

Existem várias propostas para métricas OO que levam em consideração as características básicas e interações do sistema como: número de classes, número de cases, número de métodos, médias de métodos, médias de métodos por classe, linhas de código por método, profundidade máxima da hierarquia de classes, a relação existente entre métodos públicos e privados, entre outros.

Tais métricas baseiam-se na análise detalhada do design do sistema. Como na técnica de pontos-por-função, faz sentido adicionar um peso às métricas das classes para produzir uma medida de complexidade do sistema. A maioria das medidas examina atributos em termos dos conceitos de OO, como herança, polimorfismo e encapsulamento. Para tanto, seria necessário coletar um número significativo de contagens, ou seja, seria necessário tomar valores de vários projetos e dimensioná-los selecionando as classes, os métodos e os atributos desejáveis para medir o tamanho e a complexidade de um novo software, o que nos tomaria um longo tempo.

Estimativa de Tempo

Após desenvolver uma estimativa do volume de trabalho a ser feito, você pode pensar que é fácil estimar a extensão do tempo que o projeto exigirá. Afinal, se você tem um projeto estimado em dez pessoas-mês e há cinco pessoas disponíveis ele deve levar dois meses para ser concluído. Mas, e se somente duas pessoas estiverem disponíveis? O projeto leva cinco meses para ficar pronto? De um modo geral, a nossa preocupação aqui é com a relação tempo/pessoal. Muitos anos de dolorosa experiência ensinaram-nos que tal relação não é simples.

Duplicar o número de pessoas em um projeto não reduz necessariamente a duração do projeto pela metade (muito pelo contrário, se colocarmos mais pessoas num projeto em andamento isso apenas retardará ainda mais o processo, uma vez que estas pessoas deverão receber treinamento adequado e "aprender" todo o projeto desde seu início até a fase atual, e isso consome muito tempo).

A estimativa do esforço é a técnica mais comum para se levantar os custos de qualquer projeto de desenvolvimento de engenharia. Um número de pessoas-dia, pessoas-mês ou pessoas-ano é aplicado à solução de cada tarefa do projeto. Um custo em dólares é associado a cada unidade de esforço e um custo estimado será derivado. Como a técnica LOC (linhas de código) ou FP (pontos-por-função), a estimativa de esforço inicia-se com um delineamento das funções do software obtidas a partir do escopo do projeto. Uma série de tarefas de engenharia de software - análise de requisitos, projeto, codificação e teste - deve ser executada para cada função.

O planejador estima o esforço (por exemplo, pessoas-mês) que seria exigido para se concluir cada tarefa de engenharia de software para cada função de software. Taxas de mão-de-obra (isto é, custo/esforço unitário) são aplicados em cada uma das tarefas de engenharia de software. Muito provavelmente, a taxa de mão-de-obra irá variar para cada tarefa. O pessoal de nível sênior envolver-se-á fortemente na análise de requisitos e nas primeiras tarefas de realização de projeto; o pessoal de nível júnior (que é inerentemente menos dispendioso) envolver-se-á nas últimas tarefas de projeto, codificação e nos primeiros teste.

O custo e o esforço de cada função e tarefa de engenharia de software são computados como o último passo. Se a estimativa do esforço for realizada independentemente da estimativa LOC ou FP, teremos então duas estimativas para o custo e para o esforço que podem ser comparadas e reconciliadas. Se os dois conjuntos de estimativas demonstrarem razoável concordância, haverá uma boa razão para acreditar que as estimativas são confiáveis. Se, por outro lado, os resultados dessas técnicas de decomposição exibirem pouca concordância, será necessário levar a efeito a investigação e análise adicionais.

Estimativa de Custo

O objetivo desta análise é calcular de maneira antecipada todo e qualquer custo que esteja associado ao sistema, tais como: construção, instalação, operação e manutenção.

O custo da construção é um tópico importante, visto que é graças a ele que sabemos o total de todas as pessoas envolvidas no desenvolvimento do sistema, tais como: burocratas, diretores, membros da comunidade usuária, consultores e programadores, membros da auditoria, do controle de qualidade ou da equipe de operações.

O custo de instalação do sistema é um projeto simples que podemos entregar em disquetes ou CD-ROMs e a instalação fica por conta do próprio usuário. Porém, em caso de sistemas grandes, o processo de instalação é mais complexo e envolve outros fatores, tais como: custo de treinamento do usuário, custo de conversão de banco de dados, custo de instalação do fornecedor, custo da aprovação legal, custo do processamento paralelo, custo da equipe de desenvolvimento durante a instala-

ção. Naturalmente, toda a comunidade de usuários precisará de treinamento para maior familiarização com o sistema. Este tipo de capital pode ser obtido através de empréstimo ou de reservas próprias. Dependendo da organização temos que expressá-lo como sendo custo do empréstimo feito ou em termos de juros que o dinheiro poderia estar rendendo se tivesse sido investido em lugar de ser utilizado no projeto. Esta área deve estar sempre em sintonia com o departamento de contabilidade.

O custo operacional entra em ação após a instalação do sistema. Haverá um custo para o usuário manter sua operação. Contudo, isso também deve representar uma área em que seu novo sistema economizará dinheiro, pois ele presumivelmente será mais barato que o atual sistema. Os custos operacionais mais comuns são: custos de hardware e de suprimentos, custos de software, custo de pessoal, custo de manutenção e custo de recursos.

O custo de falhas, ou manutenção, como podemos imaginar, diz respeito às diversas formas de erros que podem tornar o sistema completamente indisponível até que este erro seja corrigido, enquanto que em outros casos o sistema continua funcionando, porém uma ou mais de suas saídas podem estar incorretas. Este custo é importante pelo simples fato de não termos sistemas perfeitos. Devemos checar sempre as estatísticas disponíveis acerca da confiabilidade do software.

O Fator Humano

Quando os objetivos para o desenvolvimento de sistemas não são claros, as pessoas passam a deduzir e criar o produto dentro de suas próprias visões, levando a sistemas inadequados para a função do negócio a ser atendida e, consequentemente, a métricas falhas, gerando uma expectativa divergente entre o cliente e os técnicos ressonáveis, isto é, uma estimativa irreal.

As pessoas são sensíveis aos estímulos externos e por meio de tais estímulos são influenciadas suas atitudes e pensamentos. Um analista, ou um grupo de analistas, disposto a estimar o tempo e custo de um projeto não poderia deixar de dar a devida relevância a este fato. Um grupo de pessoas motivado, trabalhando em um ambiente agradável, sem sofrer qualquer tipo de pressão por parte da empresa ou organização produziria muito mais do que um grupo de pessoas sujeitas a condições adversas a estas. Mas não é apenas o ambiente de trabalho, são as relações familiares e pessoais, os estudos e uma outra série de fatores que podem ser aqui classificados como estímulos externos.

Para tentar amenizar a dificuldade e se estabelecer critério para a estimativa em relação a pessoal (fator humano) surge o conceito de "Engenharia Humana", que consiste em aplicar conceitos de psicologia para se projetar uma interação homem-computador de alta qualidade. Do ponto de vista do especialista em engenharia humana, o homem e a máquina são partes integrantes de todo um sistema homem-máquina. Ele vê o homem como um elo de coleta e processamento de dados.

Mesmo com técnicas como esta, o fator humano continuará sendo uma incógnita praticamente indecifrável na prática da estimativa de software.