

OS CINCO OBJETIVOS DE DESEMPENHO DA PRODUÇÃO E UM COMPARATIVO COM O DESENVOLVIMENTO DE *SOFTWARE*

Autores:

1. MARÍLIA LITWAK NEVES (UPE/FCAP)

Rua Alcina Coelho de Carvalho, 225 - apto 503

CEP 53130-400, Casa Caiada, Olinda

Pernambuco - PE

2. ANTONIO DE SOUZA SILVA JÚNIOR (UPE/FCAP)

Resumo

Este trabalho visa realizar um estudo comparativo dos cinco objetivos de desempenho da Administração da Produção aplicados à indústria de software. O processo de produção de software é um ramo atípico da Administração da Produção. Software é um misto de "bem intangível" e serviço, com características peculiares. Fabricar software implica em ter como saída do processo produtivo um bem intangível, baseado nos requisitos (necessidades) dos clientes, para os quais o conceito de qualidade é extremamente subjetivo. O conceito de "qualidade de software" possui portanto extrema dependência a percepção do usuário final do sistema. A produção de software com qualidade é uma das metas básicas da Engenharia de Software, que para atingir este objetivo disponibiliza técnicas, ferramentas e métodos. O artigo mostrará os principais modelos para auxiliar o processo de produção de software a atingir os cinco objetivos de desempenho. Entende-se por modelo, no contexto de produção de software, o processo de estabelecer um conjunto de "melhores práticas" que devem ser utilizadas no desenvolvimento de software com base em estudos históricos e conhecimento operacional. O enfoque será a produção de software por encomenda, muito embora os conceitos aqui mostrados sejam extensíveis à produção de software em larga escala.

Palavras chave: Administração da Produção, Objetivos de Desempenho, Software.

Abstract

This work aims to realize a comparative study of the five production performance goals applied to the software industry. The software production process is an atypical brunch of Production Administration. Software is a compound of "intangible good" and service, with peculiar characteristics. Making software involves having an intangible output of the productive process, based in requirements (needs) of clients for who the quality concept is considerably subjective. Thus the concept of "quality of software" possesses extreme dependence with the perception of the system final user.

The production of software with quality is one of the basic goals of Software Engineering. So it disposes techniques, tools and methods to reach this objective. However the Software Engineering still meets in the period of training of development and many of its studies are based on empirical results.

Recife, Pernambuco – PE.

The article will show the main models for help the process of software production to reach the five production performance goals. The concept of model is, in the software production context, the process of establishing a set of best practices to be used in software development based in historical studies and operational knowledge. This article will focus software produced by request although the concepts showed here can be extended to the software production in large scale.

Key Words: Production Administration, Performance Goals, Software.

GLOSSÁRIO

BENCHMARK - marca comparativa; marca de referência. Padrão real de dados econômicos de um período específico, usado como base ou referência para um estudo comparativo com dados da mesma espécie.

DOWNLOAD - Carregar um programa de computador central para uma estação.

KNOW-HOW - Conhecimento de técnicas ou detalhes práticos de alguma coisa que permite mais eficiência e melhores resultados em uma operação ou processo.

MÍDIA - materiais magnéticos usados para armazenar sinais, como disco, fitas, cd-roms, etc.

SETUP - arranjo, organização, configuração.

1. Introdução

Garantir a satisfação dos clientes e fidelizá-los ao máximo é o desejo de qualquer empreendedor, independente de seu ramo de atividade. Atualmente, este é um objetivo cada vez mais difícil de ser atingido. Características que no passado serviam como diferencial no mercado, ou seja, fatores ganhadores de pedido, hoje são apenas fatores qualificadores.

Os cinco fatores principais de desempenho competitivos, também conhecidos como os cinco objetivos de desempenho, são: qualidade, confiabilidade, rapidez, custo e flexibilidade. A importância de cada um destes fatores na organização depende da sua atividade-fim. Por exemplo, para um hospital, o mais importante é a rapidez e a qualidade no atendimento dos doentes, mesmo que para isso o custo do serviço seja alto. Já em uma joalheira, rapidez pode não ser um objetivo de desempenho tão relevante (Nigel Slack, 1997).

Um dos aspectos importantes para a retenção de clientes é a capacidade da empresa em medir o desempenho dos fatores ou atividades importantes para a sua retenção. Tão importante quanto analisar quais os objetivos de desempenho críticos para a organização é a necessidade de quantificá-los. Para melhorar o desempenho de qualquer fator-chave, é indispensável que ele seja medido, e as técnicas de métricas são bastante diferentes para cada um dos cinco objetivos de desempenho. Todavia encontrar métricas adequadas para aferir o desempenho ainda é um problema sem uma solução considerada ótima na Administração da Produção e Operações.

Quando a atividade-fim da empresa é a produção de *software*, o processo produtivo é ainda mais complexo. *Software* é um bem intangível, que possui como insumo o trabalho intelectual, e cujas métricas de desempenho são ainda insipientes apesar dos constantes avanços da Engenharia de Software. Há muita subjetividade no desenvolvimento de *software*, pois o produto final é extremamente dependente dos requisitos iniciais levantados pelo cliente. Inúmeras vezes nem o próprio cliente é capaz de perceber suas verdadeiras necessidades. O conceito de “qualidade de software” possui extrema dependência a percepção do usuário final do sistema.

Conclui-se então a pertinência em realizar um estudo sobre técnicas que auxiliem na obtenção dos objetivos de desempenho na área de software. Neste artigo, será realizada uma análise de como cada um dos objetivos de desempenho pode ser identificado no processo de produção de software.

2. Definição de *software* e os conceitos de “*Software de prateleira*” e “*Software por encomenda*”

Software é um bem intangível que faz parte da propriedade imaterial das empresas, semelhante a outros elementos como marcas, *know-how*, patentes, direitos autorais, dentre outros. Ao adquirir-se um *software*, não é a mídia (exs.: CD-ROM, DVD) ou o arquivo recebido por meio de *downloads* que se está comprando. O que se recebe mediante pagamento é o direito de usar um ou vários produtos de terceiros cujo valor depende do potencial intelectual empregado em seu desenvolvimento.

Segundo o Dicionário Michaelis, software pode ser definido como “qualquer programa ou grupo de programas que instrui o hardware sobre a maneira como ele deve executar uma tarefa”. Um *software* é desenvolvido ou projetado por engenharia, não sendo manufaturado no sentido clássico. Daí os custos de desenvolvimento são concentrados no trabalho de engenharia e os projetos não podem ser geridos como

projetos de manufatura. A engenharia relacionada ao processo de desenvolvimento de software é denominada “Engenharia de *Software*”. A Engenharia de *Software* é o estabelecimento e uso de princípios sólidos de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais.

Quanto à escala de produção, pode-se dizer que *software* pode ser produzido em escala (*software* de prateleira) ou por encomenda. *Software* "de prateleira" (*off the shelf*) são programas de computador produzidos em série, destinados a uma pluralidade de usuários e comercializados no varejo. Seu processo de produção requer um prévio e forte trabalho intelectual por parte dos projetistas antes de sua produção em larga.

Software por encomenda são programas elaborados especialmente para certo usuário.

Neste artigo, quando falar-se em produção de software, entenda-se produção de software por encomenda, cujo processo de produção é um exemplo de produção por projeto.

2.1 O ciclo de vida do processo de produção de software

Produzir software é uma atividade que obedece a uma série de etapas, ordenadas de forma lógica. Existem inúmeras definições para o ciclo de vida do processo de produção de software. O processo de software é “um conjunto de atividades e resultados associados que levam a produção de produto de software” (Ian Sommerville, 2003).

Dentro deste contexto, existem quatro atividades básicas dentro de todos os processos de software:

- especificação - etapa em que é definido o escopo de atuação do software, bem como suas funcionalidades e restrições – nesta etapa são analisados minuciosamente os requisitos do cliente e as necessidades dos usuários finais;
- desenvolvimento - onde o software é construído, de acordo com as especificações identificadas na fase anterior;
- validação – atividade onde se deve garantir que o software faz realmente o que se propôs a fazer desde sua concepção;
- evolução - manutenção do software, para que o mesmo esteja sempre atualizado com as necessidades do cliente.

3. Os cinco objetivos de desempenho da produção no contexto de projetos de software

Os objetivos de desempenho tratados neste artigo são os cinco objetivos definidos por Slack (1997): qualidade, confiabilidade, rapidez, custo e flexibilidade. Para cada um dos objetivos, será feito um paralelo com projetos de software.

3.1 Qualidade

A qualidade de um produto ou serviço pode ser analisada por meio de duas óticas: a do produtor e a do cliente. Do ponto de vista do produtor, a qualidade relaciona-se com a concepção e produção de um produto que satisfaça as necessidades do cliente. Do ponto de vista do cliente, a qualidade está associada ao valor e à utilidade reconhecida no produto, estando em alguns casos associada ao preço.

O objetivo de desempenho de qualidade envolve um aspecto externo que lida com a satisfação do consumidor e um aspecto interno que lida com a estabilidade e a eficiência da organização.

Bom desempenho de qualidade em uma operação não apenas leva à satisfação de consumidores externos, como também torna mais fácil a vida das pessoas envolvidas na operação. Logo, satisfazer os clientes internos é quase tão importante quanto satisfazer aos consumidores externos.

A qualidade reduz riscos e aumenta a confiabilidade. Quanto menos erros cometidos em cada micro operação ou unidade de produção, menos tempo será necessário para a correção e, conseqüentemente, menos confusão e irritação. Se a empresa raramente cometer erros, os empregados não precisarão gastar tempo corrigindo-os ou conferindo se o restante das operações está sendo feito corretamente. Podem gastar tempo concentrando-se em fazer bem suas tarefas.

O desenvolvimento de *software* com qualidade é um dos grandes desafios da Engenharia de *Software*. Qualidade de *software* (Dos Santos, Adriana Delfino & Chain, Marcos Lordello, 2003) (Gomes, Nelma da Silva, 2000) (Rickili, Maria das Graças, 2004) envolve o cumprimento de prazos, atendimento aos requisitos do *software* e estimação correta de custos e recursos. É necessário um controle muito grande dos processos que envolvem a fabricação do software, desde a sua criação até a sua completa instalação no cliente.

Muitas empresas ainda realizam o controle de qualidade sobre o produto de software apenas quando este já está sendo utilizado e o cliente já está sofrendo as conseqüências dos seus problemas e defeitos. Tal prática acontece porque estas empresas acreditam que enquanto o produto (software) não estiver "funcionando", não há maneiras de avaliar sua qualidade.

Quando o foco da qualidade de software é colocado no processo completo de desenvolvimento. Sua construção é controlada em todas as fases e a sua qualidade é medida antes que ele seja entregue aos usuários finais. Dentre os modelos de qualidade para processos de software serão destacados neste artigo os modelos CMMI, ISO 9001:2000 e SPICE.

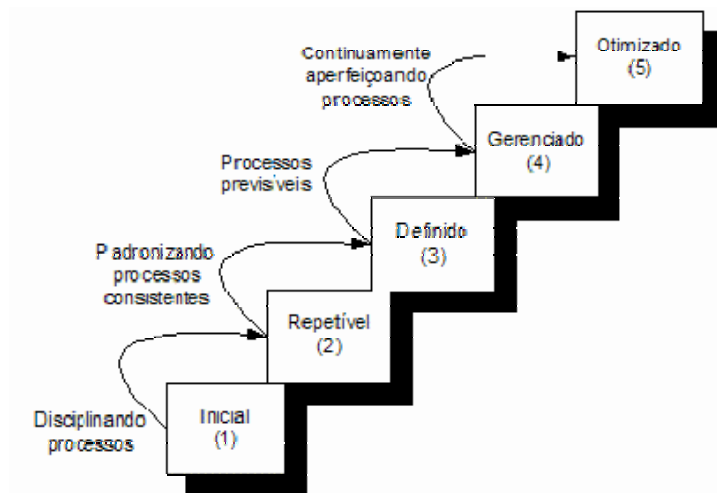
3.1.1 CMMI

O modelo CMMI (*Capability Maturity Model Integration*) (<http://pt.wikipedia.org/wiki/CMMI>) é um conjunto de processos desenvolvido pelo SEI (*Software Engineering Institute* - www.sei.cmu.edu) a fim de melhorar o desenvolvimento de aplicações em organizações cujo produto final é o software. Trata-se de uma evolução do modelo CMM (*Capability Maturity Model*), também desenvolvido pelo SEI, e procura estabelecer um único modelo para o processo de melhoria corporativo, integrando diferentes modelos e disciplinas.

Para entender o CMMI, é necessário antes compreender os conceitos básicos do CMM (Fagundes, Eduardo Mayer, 2004). No modelo CMM, o processo da organização é dividido em cinco níveis de desenvolvimento: inicial, repetível, definido, gerenciado e otimizado. Estes cinco níveis provêm uma escala mensurável de maturidade, como também a capacidade de execução de uma organização produtora de software. Os níveis também auxiliam na definição de prioridade dos esforços de uma organização. Em uma organização considerada madura, a gerência monitora a qualidade dos produtos de software e os

processos que são utilizados para produzi-los. O objetivo é julgar a qualidade dos produtos, baseados em métricas, e analisar os problemas dos produtos e processos.

A figura abaixo apresenta os cinco níveis de maturidade do CMM possíveis para uma organização que utiliza processos de software.



Fonte: (<http://www.efagundes.com/Artigos/CMM.htm>)

Figura 1 – Cinco níveis do modelo CMM

A principal mudança do CMMI em relação ao CMM é a possibilidade de utilização de duas diferentes abordagens para a melhoria de processos. Estas abordagens são denominadas “modelo contínuo” e o “modelo em estágios”. O modelo CMM tradicional é um modelo em estágios. No modelo “modelo contínuo”, cada área-chave de processo possui características relativas a mais de um nível. Assim, uma área-chave que, no modelo em estágios, pertence exclusivamente ao nível 2, no modelo contínuo pode ter características que a coloque em outros níveis. Os modelos "contínuo" e o "em estágios" se diferenciam na maneira de implementar, contudo possuem basicamente o mesmo conteúdo.

3.1.2 ISO 9001:2000

O modelo ISO 9001:2000 é um conjunto de normas criadas para especificar os requisitos de um sistema de gestão da qualidade em uma organização. No modelo ISO 9001, a organização tem como objetivo aumentar a satisfação dos clientes através de uma aplicação efetiva do sistema, incluindo processos para melhoria contínua do negócio. Trata-se de uma metodologia genérica, que pode ser aplicada a qualquer tipo de organização, inclusive organizações desenvolvedoras de software.

A ISO 9001:2000 sugere que a aplicação e a gestão de um sistema de processos seja uma forma de garantir a boa gestão da qualidade. Para adotar esta “abordagem de processo”, a ISO 9001 inclui uma metodologia denominada PDCA (“Planejar-Fazer-Checkar-Agir”), que pode ser aplicada a todos os processos. Segue-se uma breve descrição das etapas desta metodologia:

- Planejar - estabelecer os objetivos e processos necessários para gerar resultados de acordo com os requisitos dos clientes e com as políticas da organização;

- Fazer - implementar os processos;
- Verificar - monitorar e medir os processos e produtos em relação às políticas, objetivos e requisitos para o produto e relatar os resultados;
- Agir - tomar ações para melhorar continuamente o desempenho dos processos.

O ISO 9001:2000 se baseia na gestão de processos individuais, os quais quando tomados como um todo, ajudam a gerenciar efetivamente o negócio inteiro. O modelo PDCA ajuda a gerenciar os processos individuais.

A certificação ISO 9001:2000 autentica os padrões de qualidade de uma organização. Utiliza um modelo de gestão da qualidade como o ISO 9001:2000 além de beneficiar os funcionários, favorece em consequência a própria organização, ao ordenar e organizar os seus processos.

3.1.3 SPICE

O modelo SPICE, cujo nome é formado por uma espécie de abreviatura do termo *Software Process Improvement and Capability Determination*, é um projeto mundial oriundo da comissão internacional de Eletrotécnica da ISO e controlado por diversos países baseado no CMM sendo, porém, mais flexível e aplicável por um universo maior de empresas. Este modelo da série ISO visa à geração de normas particulares para a indústria de software. Trata-se de um conjunto de normas que envolvem as etapas de planejamento, gerência, acompanhamento, monitoração, controle, fornecimento, desenvolvimento, operação, evolução e suporte do software.

O SPICE é um projeto para avaliar processos de empresas de software buscando a melhoria constante.

O processo de software é dividido em instâncias, que são unidades ativas detalhadas em separado. São elaborados guias de introdução, melhoria, determinação de maturidade, treinamento, qualificação e construção dos processos. Avaliações são feitas com base nestas guias.

O modelo SPICE define duas dimensões:

1) dimensão de processo

A dimensão de processo agrupa os processos em cinco categorias, de acordo com o tipo de atividades que executam. Cada processo é descrito em termos de um propósito que exprime um único objetivo funcional. Em todas as tarefas, atividades, práticas e as características dos artefatos produzidos são utilizadas como indicadores que demonstram se determinado processo é praticado em um determinado nível de capacidade. Esses indicadores são classificados de acordo com uma escala que expressa justamente seu nível de capacidade. Esta dimensão, não contemplada no modelo CMM, é fundamental para que as organizações possam selecionar os processos considerados por elas como de maior prioridade, e levá-los para patamares mais elevados de capacidade, enquanto outros, de menor importância, possam se manter em níveis mais baixos.

2) dimensão de capacidade

Esta dimensão visa identificar um conjunto de atributos que permitam determinar o nível de maturidade do processo de desenvolvimento de software. A dimensão de capacidade é extremamente similar ao

proposto pelo modelo CMM e estabelece níveis de maturidade.

A abordagem utilizando duas dimensões influenciou decisivamente o aparecimento do modelo CMMI, em sua variante contínua, cujo modelo possui diversos pontos em comum com a norma SPICE.

3.2 Confiabilidade

O conceito de confiabilidade em Administração da Produção e Operações está relacionado ao princípio de realizar as atividades em tempo para os consumidores receberem seus bens ou serviços quando foram prometidos.

Ao tratar-se da produção de software, existem duas vertentes para o conceito de confiabilidade. A primeira, semelhantemente aos processos produtivos tradicionais, focaliza o cumprimento dos cronogramas do projeto. Tal interpretação de confiabilidade pode ser encarada como o estudo de confiabilidade de serviços (visto que o projeto de software é um serviço, com grande capital intelectual englobado) e não será o alvo de estudo deste artigo, visto que as técnicas tradicionais da Administração da Produção de serviços já tratam deste assunto.

A outra visão do conceito de confiabilidade em software relaciona-se à ocorrência de falhas no "produto". A confiabilidade é definida então como a probabilidade do software operar sem ocorrência de falhas durante certo período de tempo em um determinado ambiente. Todavia, a confiabilidade de software é extremamente ligada ao conceito de disponibilidade.

Disponibilidade consiste na probabilidade de, em qualquer instante de tempo, um sistema funcionar satisfatoriamente em um determinado ambiente. Ou seja, a probabilidade de um sistema estar disponível quando necessário. A disponibilidade pode ser calculada da seguinte maneira:

$$Disponibilidade = ((MTTF)/(MTTF + MTTR)).100\%$$

Onde:

MTTF - *Mean Time to Failure* - é o tempo médio até a ocorrência de falha;

MTTR - *Mean Time to Repair* - é o tempo médio de reparo.

Os atributos confiabilidade e disponibilidade consistem na base da Engenharia de Confiabilidade de Software (ECS). A Engenharia de Confiabilidade de Software é definida como o estudo quantitativo do comportamento operacional de sistemas de software, tendo como base os requisitos dos usuários relativos à confiabilidade, visto que cada sistema de software requer diferentes níveis de confiabilidade. Por exemplo, para uma pequena padaria, caso o software de emitir pedidos dos clientes não esteja disponível em um determinado momento, o impacto não é tão crítico, pois o próprio atendente pode despachar os pedidos manualmente. Já no caso de um sistema de bolsa de valores, alguns minutos do sistema fora do ar implicam em significativos prejuízos financeiros. O nível de confiabilidade e disponibilidade necessária para estes dois sistemas é completamente diferente.

3.3 Rapidez

Rapidez em Administração da Produção e Operações é um conceito relacionado a quanto tempo os

consumidores precisam esperar para receber seus produtos ou serviços. O tempo de espera começa a ser contado desde o momento do pedido até a sua entrega.

Há uma grande dificuldade das organizações desenvolvedoras de software de cumprirem suas metas de prazo e orçamento. De acordo com o *Standish Group* - instituição norte-americana independente dedicada à investigação e consultoria, que trabalha com pesquisas e projetos na área de software – em pesquisa realizada em 2004 (*The Standish Group Report*, 2004), apenas um percentual de 16,2% dos projetos de software terminaram dentro do prazo e orçamento inicialmente estimados. Em inúmeros casos, o prazo de entrega pode implicar na sobrevivência ou não de uma empresa.

A utilização de *software* em condições mais diversas e para atender a um leque de funções cada vez amplo é crescente. A complexidade dos sistemas também aumenta com o decorrer do tempo. Hoje clientes desejam softwares produzidos em menos tempo e com maior qualidade. A produção de software enfrenta atualmente o desafio de encontrar uma forma de atender estas exigências aparentemente conflitantes.

A complexidade do software vem aumentando consistentemente. Porém, a maneira como o software é produzido não tem se adaptado a esta nova situação. Ocorrem então que sistemas são entregues fora do prazo, ou são de difícil utilização e até mesmo não cumprem a sua função.

Neste contexto, o estudo dos processos de software torna-se de suma relevância. O uso de um processo com boa definição e gerência mostra-se a solução atual para o dilema da produção de software (qualidade x rapidez). Um processo adequado fornece:

- a orientação quanto à ordem das tarefas que serão executadas;
- o direcionamento das tarefas de cada desenvolvedor e da equipe como um todo;
- especifica que artefatos serão produzidos e provê critérios para monitoração e medição do andamento do projeto.

Processos de software podem melhorar significativamente a qualidade dos produtos de uma empresa produtora de software, e com eles é possível medir e avaliar o desempenho dos setores ligados ao desenvolvimento.

Como foi visto na Seção 3.1, referente à qualidade, o uso de um processo de software é largamente apontado como um fator para sucesso de uma empresa de desenvolvimento de software.

A partir do uso de boas práticas de desenvolvimento, preconizadas pelos modelos de desenvolvimento de software (CMMI, SPICE e assim por diante), os sistemas são desenvolvidos de maneira cada vez mais padronizada, melhorando continuamente a rapidez no desenvolvimento. As boas práticas de desenvolvimento utilizam técnicas como o reuso de código para desenvolver os sistemas de maneira cada vez mais rápida.

Ao utilizar-se de modelos de qualidade como os vistos na Seção 3.1, obtém-se um ganho significativo de rapidez no processo de desenvolvimento de *software*.

3.4 Flexibilidade

Em Administração da Produção, flexibilidade é a capacidade de alteração da produção, seja no que faz, como faz ou quando faz. No caso de produtos e serviços, é a capacidade da operação introduzir no mercado novos itens. Especificamente quanto a *software*, é a habilidade de modificar e adaptar o projeto para atender a uma mudança nas necessidades do cliente.

A mudança em projetos de software, sejam eles concluídos ou em andamento, é inevitável. Diversos são os fatores que impulsionam a necessidade de mudanças, por exemplo:

- novos requisitos emergem quando o software é implantado;
- o surgimento de mudanças no ambiente do negócio;
- há erros no software que devem ser reparados;
- um novo equipamento deve ser incorporado ao sistema;
- o desempenho do software deve ser melhorado.

Um grande problema para as organizações é implementar e gerir mudança também em seus sistemas legados.

A flexibilidade do software pode ser medida em termos da sua expansibilidade e versatilidade.

A expansibilidade de um software consiste na capacidade de um software de absorver mais funções ou uma maior quantidade de dados, sem a necessidade de alterações estruturais.

Para avaliar a versatilidade de um software, em relação ao hardware ou a outros softwares, é preciso determinar se, e em que medida, ele exibe as seguintes características:

- a) capacidade de ser executado em diferentes tipos de hardware;
- b) compatibilidade com vários tipos de interface e protocolo;
- c) flexibilidade para adaptar a diferentes tipos de ambientes de computação, por exemplo, redes locais;
- d) capacidade de realizar transferência de dados on-line e de trocar dados com software executável em outros equipamentos;
- e) capacidade de gerar dado que possam ser acessados por outros softwares e de acessar dados gerados por outros softwares.

3.5 Custo

O objetivo de desempenho denominado "custo" em uma organização está relacionado ao desejo de produzir com o menor custo possível. Isto é, produzir bens e serviços a custo que possibilite estabelecer preços apropriados ao mercado e ainda permitir um retorno financeiro para a empresa. Se tal objetivo for alcançado, a organização proporciona a vantagem de custo para a instituição.

Todos os objetivos de desempenho propostos por Slack apóiam de maneira direta ou indireta a redução do custo da organização. A qualidade mais alta funciona como um redutor de custos. Quando a organização comete menos erros dentro das operações, isto é refletido na redução direta dos refugos, retrabalhos e

desperdícios.

A confiabilidade interna também reduz custos. Se todas as partes, materiais e informações fluem, dentro da operação, conforme planejado, as despesas indiretas com o seguimento das entregas atrasadas serão eliminadas. Da mesma forma são eliminadas as despesas indiretas com o esforço das reprogramações.

A flexibilidade operacional reduz custos, pois a redução dos tempos de *setup* reduz despesas indiretas, além de incrementar a confiabilidade interna. A flexibilidade permite optar-se por roteiros alternativos de processo a fim de evitar indisponibilidades inevitáveis de máquinas, o que por consequência reduz custos.

O cálculo de custo em projetos de software merece atenção especial, devido ao fato de que software é um produto singular, diferente das saídas de um processo padrão de manufatura. As estimativas de esforço, custo, prazo e qualidade são necessárias ao longo de todo o projeto, embora mais críticas no momento de formulação da proposta ou orçamento.

Em um processo de desenvolvimento de software é preciso medir custo, produtividade e qualidade não só do produto final mas também de todo o processo.

Com a implantação de um sistema de coleta de métricas, os desenvolvedores podem avaliar melhor a sua produtividade e adaptabilidade ao processo de desenvolvimento e, com isso, estimar melhor o tempo necessário para executar cada tarefa.

A princípio, é necessário determinar o que se quer medir, a fim de se definir como os dados serão coletados. Essas decisões devem ser compatíveis com o processo de desenvolvimento. Uma metodologia de métrica e estimativa não vai solucionar de imediato os problemas de um processo de desenvolvimento. No entanto, esta deve ser utilizada para tornar possível o entendimento do processo, para facilitar a previsão de suas fases e mostrar como controlá-las.

As estimativas jamais poderão ser precisas e exatas, pois não são apenas fatores técnicos e "contáveis", palpáveis que fazem parte de um projeto, mas também existem pessoas, sentimentos, política, crenças, ambiente e outros mais que não se podem medir, são absolutamente variáveis. Afinal, não seria possível estabelecer um padrão, ou ainda, transformar em números os sentimentos de uma pessoa, ou provar a ela que, matematicamente, suas superstições não são válidas.

Neste artigo, será feita uma análise sobre uma das metodologias mais utilizadas para cálculo da estimativa de custos em projetos de software: o modelo COCOMO.

3.5.1 O modelo COCOMO

A metodologia COCOMO [12] é um modelo de estimativa do tempo e cálculo de custo e prazos de desenvolvimento de um produto. Foi desenvolvido pelo Doutor Barry Boehm, um dos profissionais mais influentes da indústria de software. Em seu livro *Software Engineering Economics* lançado em 1981, Dr. Barry introduziu o modelo COCOMO, proporcionando uma base bem definida o cálculo de custo e prazos de uma solução de software. Com pouco tempo o COCOMO se tornou um dos modelos de custo de software mais conhecidos. COCOMO é uma espécie de abreviatura formada pelas primeiras sílabas da expressão de língua inglesa *CO*nstructive *CO*st *MO*del. O COCOMO é baseado no estudo de sessenta projetos, onde os pesquisadores examinaram de 2.000 a 100.000 linhas de código de programas de

Assembly a PL/I.

O COCOMO possui três formas de implementação:

1 – Básico

2- Intermediário

3 – Avançado

O COCOMO básico é um modelo estático que calcula o esforço de desenvolvimento de software e seu custo, em função do tamanho de linhas de códigos desenvolvidas.

$$E = ab(KLOC)^{bb}$$

$$D = cb(E)^{db}$$

$$P = E / D$$

Nas equações acima temos que **E** é o **esforço aplicado pela pessoa** no mês, **D** é o **tempo de desenvolvimento** em meses cronológicos, **KLOC** é o **número calculado de linhas de código** para o projeto (expressado em milhares), e **P** é o **número das pessoas necessário**. Os coeficientes **ab**, **bb**, **cb** e **db** são dados na Tabela 1:

Projeto de Software	Ab	bb	cb	Db
Início	2.4	1.05	2.5	0.38
Meio	3.0	1.12	2.5	0.35
Fim	3.6	1.20	2.5	0.32

Fonte: (http://pt.wikipedia.org/wiki/M%C3%A9todo_COCOMO#Veja_tamb.C3.A9m)

Tabela 1 – Coeficientes ab, bb, cb e db no modelo COCOMO básico

O COCOMO básico possui como principal vantagem ser rápido em estimativas e custos de software, mas sua exatidão é limitada devido à sua falta de fatores para explicar as diferenças entre ferramentas, qualidade de pessoal e experiência, utilização de técnicas e ferramentas modernas, dentre outras variáveis de projeto que influenciam nos custos de software.

Já o COCOMO em seu nível intermediário calcula o esforço de desenvolvimento de software em função do tamanho do programa, que inclui custo, avaliação subjetiva do produto, ferramentas, pessoal e atributos de projeto:

$$E = ai(LOC)^{(bi)}.EAF$$

Onde na equação **E** é o **esforço aplicado em pessoas** por mês, **LOC** é o **número de linhas de código** para o projeto e **EAF** é o **fator calculado** acima. Os coeficientes **ai** e o **bi** são dados na Tabela 2:

Projeto de Software	Ab	bb
Início	3.2	1.05
Meio	3.0	1.12
Fim	2.8	1.20

Fonte: (http://pt.wikipedia.org/wiki/M%C3%A9todo_COCOMO#Veja_tamb.C3.A9m)

Tabela 2 – Coeficientes ai e bi no modelo COCOMO intermediário

O método intermediário é uma extensão do método básico, porém com o acréscimo de categorias de controle como atributos do produto, atributos de hardware, atributos pessoais e atributos do projeto.

Em sua terceira forma de implementação, ou seja, o COCOMO avançado são incorporadas características da versão intermediária com uma avaliação de impacto de custo em cada etapa do projeto.

3.5.2. O modelo COCOMO II

O COCOMO II (versão II do modelo COCOMO) é um modelo objetivo de cálculo de custos para o planejamento e execução de projetos de software. Trata-se de uma importante metodologia no gerenciamento dos projetos de software, bem como em linhas de negócio de software.

O COCOMO II inclui diversas melhorias, de modo a ampliar sua aplicabilidade e aperfeiçoar a acurácia de suas estimativas nas abordagens modernas para desenvolvimento de software. É o resultado do trabalho de muitos especialistas em economia do software, que utilizaram o COCOMO por incontáveis horas em laboratórios de pesquisa e em aplicações no campo, abrangendo um vasto espectro de domínios e de organizações de software.

A metodologia COCOMO II é utilizável para uma coleção mais ampla de tecnologias que o modelo COCOMO Original. Ela proporciona um suporte atualizado aos projetos de software orientados a objeto, softwares criados usando modelos de desenvolvimento em espiral ou incremental e softwares criados utilizando componentes de prateleira. Pontos de Objeto são usados para estimar tamanho ao invés da tradicional métrica de linhas de código (LOC). Uma estimativa inicial de tamanho é determinada pela contagem do número de telas, relatórios e componentes de terceira geração que irão ser usados na aplicação.

4. Conclusões Finais

O artigo mostrou que os cinco objetivos de desempenho da produção são aplicáveis de uma maneira geral a outros processos produtivos distintos dos sistemas de manufatura. Foi escolhido um ramo singular de produção: a construção de software.

Software é um misto de "bem intangível" e serviço, com características peculiares e para o qual aparentemente seria difícil estabelecer uma analogia com objetivos de desempenho de produção. Porém, nos estudos aqui realizados verificou-se que é possível mapear os conceitos da Administração da Produção e Operações para este tipo de produto.

A Engenharia de Software surgiu em meados da década de 1970, em uma tentativa de contornar a crise do software e dar um tratamento de engenharia - ou seja, mais sistemático e controlado - ao desenvolvimento

de sistemas de software. Esta ciência envolve os estudos de tecnologias e práticas usadas para criar software, melhorando a produtividade e qualidade. Porém a Engenharia de Software ainda se encontra no estágio de desenvolvimento e muitos de seus estudos baseiam-se em resultados empíricos. Daí a relevância deste artigo, que visa mostrar a aplicação da teoria consagrada de Administração da Produção na Engenharia de Software.

Como trabalhos futuros, sugere-se um aprofundamento no estudo do mapeamento dos cinco objetivos de desempenho para a produção de software. Pode ser realizado um trabalho de crítica dos modelos de processo de desenvolvimento de software já existentes, por exemplo, CMMI e SPICE.

Bibliografia

- [1] A Avaliação de Software para EAD via Internet, Eduardo O C Chaves. Disponível em:
<<http://www.edutec.net/Textos/Self/EDTECH/softEAD.htm>>. Acesso em: 19 set. 2006.
- [2] Avaliação de Bens Intangíveis. Disponível em :
<<http://www.patentnet.com.br/intangiveis.htm>>. Acesso em: 19 set. 2006.
- [3] CMM - Capability Maturity Model. Eduardo Mayer Fagundes (Engenheiro, Mestre e Professor da Universidade Mackenzie). Disponível em:
<<http://www.efagundes.com/Artigos/CMM.htm>>. Acesso em: 19 set. 2006.
- [4] CMMI – Wikipédia. Disponível em:
<<http://pt.wikipedia.org/wiki/CMMI>>. Acesso em: 19 set. 2006.
- [5] Qualidade de software. Adriana Delfino dos Santos e Marcos Lordello Chaim. CNPTIA - Embrapa Informática Agropecuária. Disponível em:
<<http://www.cnptia.embrapa.br/modules/sections/index.php?op=viewarticle&artid=9>>. Acesso em: 19 set. 2006.
- [6] Dicionário Michaelis – UOL – Português-Inglês-Espanhol/Versão 2000
- [7] Engenharia de confiabilidade de software. Antonio Mendes da Silva Filho (Professor do Departamento de Informática da UEM. Doutor em Ciência da Computação). Revista Espaço Acadêmico, Ano III, Nº 27 - Agosto/2003. Disponível em:
<<http://www.espacoacademico.com.br/027/27amsf.htm>>. Acesso em: 19 set. 2006.
- [8] Engenharia de Software - Wikipédia. Disponível em:
<http://pt.wikipedia.org/wiki/Engenharia_de_software - último acesso em 19/09/2006
- [9] Implantação de uma fábrica de software. Cristine Elaine Dias Rocha, Thayssa Águila da Rocha e Thiago José Miranda Almeida. Centro Universitário do Estado do Pará - Curso de Especialização em Engenharia de Sistemas, 2003. Disponível em:
<http://www.cin.ufpe.br/~tar/trabalhos/especializacao.htm#_Toc558187>. Acesso em: 19 set. 2006.
- [10] Ética e Computação. Selma Shin Shimizu Melnikoff. Escola Politécnica da USP, Setembro de 2004. Disponível em:
<http://www.lps.usp.br/lps/arquivos/conteudo/grad/dwnld/etica_comp.ppt>. Acesso em: 19 set. 2006.
- [11] INDG - Instituto de Desenvolvimento Gerencial. Glossário. Disponível em:
<<http://www.indg.com.br/info/glossario/glossario.asp?r>>. Acesso em: 19 set. 2006.
- [12] Método COCOMO – Wikipédia. Disponível em:
<http://pt.wikipedia.org/wiki/M%C3%A9todo_COCOMO#Veja_tamb.C3.A9m>. Acesso em: 19 set.

2006.

[13] Métricas e Estimativas de Software - O início de um rally de regularidade. Alvaro Eduardo Gomes. Disponível em:

<<http://www.apinfo.com/artigo44.htm>>. Acesso em: 19 set. 2006.

[14] Qualidade de Software. Empresa MSW Métricas e Software. Disponível em:

<<http://www.mswconsult.com.br/qualidade.html>>. Acesso em: 19 set. 2006.

[15] Princípios de Engenharia de Software 2004-2. Disponível em:

<www-di.inf.puc-rio.br/~julio/0114838Rel01.pdf>. Acesso em: 19 set. 2006.

[16] Qualidade de Software – Uma Necessidade. Nelma da Silva Gomes, especialista em sistemas de informação, com pós-graduação em Gestão Estratégica da Informação, e consultora da UCP/PNAFM/MF. Disponível em:

<www.fazenda.gov.br/ucp/pnafe/cst/arquivos/Qualidade_de_Soft.pdf>. Acesso em: 19 set. 2006.

[17] Implantação e levantamento dos custos da qualidade. Revista Científica da FAL, ANO 2 VOL. 3 No. ABR/2004. Disponível em:

<<http://www.falnatal.com.br/revista/artigos/edicao3/artigos.asp?artigo=2>>. Acesso em: 19 set. 2006. [18]

Garantia da Qualidade de Software no Serpro. Maria das Graças Bezerra Rickli. Disponível em:

<<http://www.serpro.gov.br/publicacao/tematec/2002/ttec62>>. Acesso em: 19 set. 2006.

[19] SLACK, Nigel. Administração da produção (edição integral). Atlas, São Paulo, 1997

[20] SOMMERVILLE, I. Engenharia de Software. 6 ed. Ver. Tec. Kechi Hiram. São Paulo. Addison Wesley, 2003.

[21] Empresa ti MÉTRICAS. Disponível em:

<<http://www.metrics.com.br/estimativas.htm>>. Acesso em: 19 set. 2006.

[22] Usabilidade e a maturidade na produção de software. Professor Clarindo Isaías P. S. Pádua e Aloísio Antônio Ribeiro Júnior (mestrando). Departamento de Ciência da Computação da UFMG. Disponível em:

<<http://www.comp.ufla.br/~zambalde/usabilidade%20maturidade.htm>>. Acesso em: 19 set. 2006.

[23] Um Processo de Software para Rapid Application Development (RAD) . Disponível em:

<<http://www.async.com.br/~kiko/papers/rad/>>. Acesso em: 19 set. 2006.

[24] THE STANDISH GROUP. *The Standish Group Report - CHAOS. The Standish Group, 2004.*