



MPS.BR – Mitos e Verdades a Respeito de um Modelo de Maturidade



Andriele Ferreira Ribeiro

Bacharel em Ciência da Computação (UFMG). Mestre em Administração de Empresas (UFMG). Pós-graduado em Melhoria de Processo de Software (UFLA). Certificado como Project Management Professional (PMP) pelo Project Management Institute (PMI). Certificado como implementador e avaliador oficial do modelo MPS.BR. Professor de disciplinas relacionadas à gerência de projetos e administração de empresas. Vice-Presidente de Filiação do PMI-MG no ano de 2006. Gerente de Modernização do Desenvolvimento de Sistemas da PRODEMGE, tendo atuado como gerente de projetos, consultor em escritório de projetos e melhoria de processos no DCC – UFMG, PRODEMGE e FUMSOFT.

Qualquer área de conhecimento atualmente é controversa. Fra-se de efeito para se iniciar um artigo, não é mesmo? Mas pense bem se ela não faz sentido. A quantidade de informações relativas a qualquer ramo de atuação – administração, medicina, nutrição, publicidade, preparação física, etc – é muito grande. Vamos tomar como exemplo a nutrição. Existem dietas dos mais diferentes tipos visando o emagrecimento: as que envolvem proteínas predominantemente, as que dosam os diversos tipos de alimento, as que excluem quase todos os alimentos, as que associam seu signo à rotina alimentar e muitas outras. A presença de muitas informações tem como consequência direta a existência de pontos de vista diferentes.

Quando falamos de desenvolvimento de software e qualidade, a realidade não poderia ser diferente. Abordagens técnicas e gerenciais são criadas, discutidas, modificadas com o objetivo de se produzir software de qualidade. Existem

metodologias, modelos, normas, corpos de conhecimento, enfim, muitas linhas escritas em diversos formatos e com diversos objetivos tratando do tema. Isto gera divergências, naturalmente. Às vezes porque elas existem de fato. Mas em muitos casos por desconhecimento de quem critica.

Ultimamente tenho lido muitos textos, depoimentos, e-mails em listas de discussão e até ouvido podcasts em que modelos de maturidade tais como o MPS.BR (a denominação correta do modelo é MR-MPS, mas o nome do programa que lhe deu origem – MPS.BR – tem sido mais comumente utilizado para designá-lo) e o CMMI (Capability Maturity Model Integration) são questionados. Diria que em alguns casos eles são até mesmo execrados. Até aí, problema nenhum. A liberdade de expressão garante o direito de todos se manifestarem livremente e críticas dirigidas a qualquer coisa podem, em muitos casos, trazer benefícios à entidade sendo criticada. Estas são as

chamadas críticas construtivas, baseadas em argumentos sólidos e consistentes. Infelizmente não é exatamente isto o que tenho visto por aí...

Para que vocês possam ter uma idéia melhor do que estou dizendo, vou dar alguns exemplos de comentários absolutamente equivocados que têm circulado na Internet. Escrevo algumas palavras ao fim de cada um deles, tentando mostrar o porquê do equívoco.

1. Modelos de maturidade só se preocupam com processos. A competência das pessoas é desconsiderada.

Este é um engano recorrente. Os processos são de fato bastante valorizados nos modelos. Mas há também resultados / práticas / objetivos específicos, tanto no MPS.BR quanto no CMMI, referenciando diretamente a necessidade de competência e conhecimento das pessoas em relação às atividades que executam. Mais à frente ainda neste artigo falarei mais a respeito deste assunto.

2. Modelos de maturidade estão ressuscitando o taylorismo. Um grupo de pessoas completamente alheias à realidade de quem “põe a mão na massa” cria um bando de processos burocráticos.

Qualquer iniciativa de melhoria de processo de software está condenada ao fracasso se os executores das atividades que estão sendo descritas no processo são alijados das discussões. Portanto, não há cabimento em se associar este tipo de comportamento ao modelo A, B ou C. Se isto ocorre dentro de alguma empresa, a falha é de quem está conduzindo o projeto de melhoria ou de quem está orientando a sua execução (consultores, implementadores, etc) e não do modelo.

3. Scrum ou XP são preferíveis ao CMMI / MPS.BR porque usam uma abordagem iterativa e não cascata.

Não há a definição de uso de um ciclo de vida de projeto específico nos modelos de maturidade. O que é necessário para que haja aderência aos modelos é a definição de qual ciclo de vida será utilizado para cada um dos projetos. Pode ser cascata, iterativo, misto, incremental, entrega evolutiva ou qualquer outro.

4. Dezoito meses é o tempo necessário para se avaliar com um sucesso uma empresa no nível G do MPS.BR.

Não há um tempo padrão para se implementar os processos de um ou outro nível. Isto varia de empresa para empresa em função de uma série de fatores tais como grau apoio da alta direção, investimento direto no projeto de melhoria de processos e nível de conhecimento dos colaboradores. A Softex (instituição mantenedora do modelo) tem dado apoio financeiro para cobertura de parte dos custos de consultoria para empresas que atendam a alguns requisitos. Entre eles está o compromisso de se realizar uma avaliação após 12 meses e não 18. Mas isto não quer dizer que o tempo seja este ou aquele. Este compromisso existe apenas para empresas que participam de algum grupo que recebe este apoio financeiro.

5. O MPS.BR é burocrático demais. Se eu for gerar todos os documentos que são solicitados pelo modelo, não tenho tempo para fazer o mais importante: software de qualidade.

O MPS.BR não obriga a geração de nenhum documento específico. O modelo apenas exige que algumas formalizações sejam realizadas. E estas formalizações, em projetos não são mera burocracia. A obtenção de compromissos por parte de clientes e fornecedores, por exemplo, é muito mais segura se for formalizada. Às vezes leio ou escuto algo do tipo: "Se alguma mudança ocorreu no projeto, não preciso formalizar com meu cliente. No meu método de trabalho isso é apenas falado e pronto". Eu gostaria muito de trabalhar em um ambiente em que as relações funcionassem desta forma e em que a memória das pessoas fosse boa o suficiente a ponto de lembrar de todas as mudanças e impactos no projeto acordados apenas oralmente. Mas creio que o mundo real seja bem diferente disto na maioria absoluta das situações.

Agora que já falamos do que é errado em relação ao CMMI / MPS.BR, vamos falar do que os modelos realmente preconizam. Aqui cabe uma ressalva: alguns de vocês podem estar se perguntando por que estou tratando CMMI e MPS.BR de forma indistinta. A resposta é simples: porque tecnicamente falando eles são completamente compatíveis.

Não estou entrando no mérito do reconhecimento internacional de ambos. Nesse ponto o CMMI é muito mais forte. Mas tecnicamente há apenas algumas pequenas diferenças, a saber: 1) Há alguns processos e, portanto, requisitos adicionais no MPS.BR ligados às áreas de gestão do conhecimento e gerência de reuso. 2) O MPS.BR apresenta maior flexibilidade na exclusão de processos a serem considerados na avaliação.

Estas diferenças não impedem a compatibilidade entre eles. Para que após uma avaliação MPS.BR a empresa avaliada esteja apta a realizar uma avaliação CMMI em nível similar com sucesso, basta que parte da flexibilidade apresentada pelo MPS.BR não seja exercitada. Ou seja, se um determinado processo pode ser excluído em uma avaliação MPS.BR e tal exclusão não pode ocorrer no CMMI, a empresa deve se preocupar em atender também aos resultados esperados deste processo, mesmo que ele possa não ser avaliado oficialmente no caso do MPS.BR. Feita a ressalva, voltemos ao que os modelos realmente objetivam. Para mostrar o pragmatismo da melhoria de processos de software via MPS.BR serão citados e comentados alguns resultados esperados de alguns processos do modelo. Ficará fácil de visualizar que não há nada de burocrático ou sobrenatural sendo proposto. Apenas para fins de esclarecimento, as siglas no início de cada resultado são abreviações dos nomes dos processos aos quais eles pertencem e o número representa a posição do resultado dentro do processo. Nos exemplos abaixo são citados resultados dos processos Gerência de Projetos (GPR), Gerência de Requisitos (GRE) e Medição (MED).

GPR1. O escopo do trabalho para o projeto é definido;

Responda com sinceridade: há algo errado em se definir o escopo de um projeto? Eu não acho que haja. E mesmo aqueles que defendem um escopo "flutuante" em que as solicitações de mudanças são tratadas como benéficas e naturais não podem prescindir de algum nível de definição inicial deste escopo. Para que a mudança seja possível é preciso saber o que se quer mudar.

GPR 6. Os riscos do projeto são identi-

ficados e o seu impacto, probabilidade de ocorrência e prioridade de tratamento são determinados e documentados;

Antes de comentar este resultado, vamos recordar a definição de projeto presente no PMBOK: "Um projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo." A organização do trabalho em projetos, sejam eles de desenvolvimento de software ou não, se dá fundamentalmente para que seja possível lidar com os desafios inerentes à criação de um resultado novo, exclusivo, diferente em algum aspecto de tudo que já foi criado anteriormente. Para gerenciar a rotina, métodos mais simples são suficientes. Com projetos a coisa é diferente. Como sempre há algum grau de novidade, incertezas são inerentes ao conceito de projeto e é fundamental que o gerenciamento dos riscos ocorra em algum nível. A única certeza é que os riscos existem e se os mesmo não forem devidamente tratados, a probabilidade de que algum deles ocorra e prejudique algum aspecto do projeto é de quase 100%.

Esse resultado, presente no processo Gerência de Projetos, é o mínimo que se espera em relação ao gerenciamento de riscos no MPS.BR. Exige-se que os riscos sejam identificados (nada mais natural), classificados em relação à sua probabilidade e impacto e priorizados em função destas características. A priorização é importante para que aquilo que é mais importante tenha um cuidado maior do que aquilo que potencialmente pode trazer menos danos ao projeto.

GPR 7. Os recursos humanos para o projeto são planejados considerando o perfil e o conhecimento necessários para executá-lo;

Falamos de pessoas agora há pouco, no comentário sobre uma das críticas infundadas ao modelo. Este resultado é um dos pontos de valorização das pessoas dentro

do modelo. O que ele pede? Simples. Que as alocações dos recursos humanos aos projetos não se dêem apenas por disponibilidade, ou seja, que alguém não seja alocado simplesmente porque está livre ou menos atarefado. Cada papel ou função dentro de um projeto requer uma série de conhecimentos e habilidades e o não atendimento destes "pré-requisitos" normalmente traz consigo muitos problemas. É razoável alocar alguém que nunca desenvolveu em Java como programador em um projeto que só usará esta linguagem, sem que nenhuma ação de capacitação seja realizada? Faz sentido promover alguém a gerente de projetos se esta pessoa não conhece sequer o conceito de projeto? A resposta NÃO parece óbvia, mas surpreendentemente isto ocorre dentro das empresas. Este resultado visa justamente mudar esta cultura, fazer com que as empresas prestem mais atenção ao tratamento dispensado às pessoas que nela trabalham.

Certa vez, quando eu estava em uma reunião de consultoria em uma empresa, meu interlocutor deu gargalhadas quando expliquei o significado deste resultado para ele. E comentou: "Isto é piada! Nossos funcionários são estes e pronto. Se eles não têm conhecimentos ou habilidades necessárias, durante o projeto eles dão um jeito de adquiri-los". É claro que não há uma rigidez tão grande no modelo a ponto de impedir que a falta de um conhecimento num primeiro momento inviabilize a alocação de uma pessoa ao projeto. Mas a coisa também não pode se dar de maneira tão informal. Se um analista está acostumado a especificar requisitos usando alguma técnica da análise essencial, por exemplo, ele poderia ser alocado à função de especificador usando casos de uso, desde que algum treinamento ou qualquer outra ação de capacitação que suprisse suas necessidades fosse planejada e executada. O que o modelo quer evitar é que "vá se

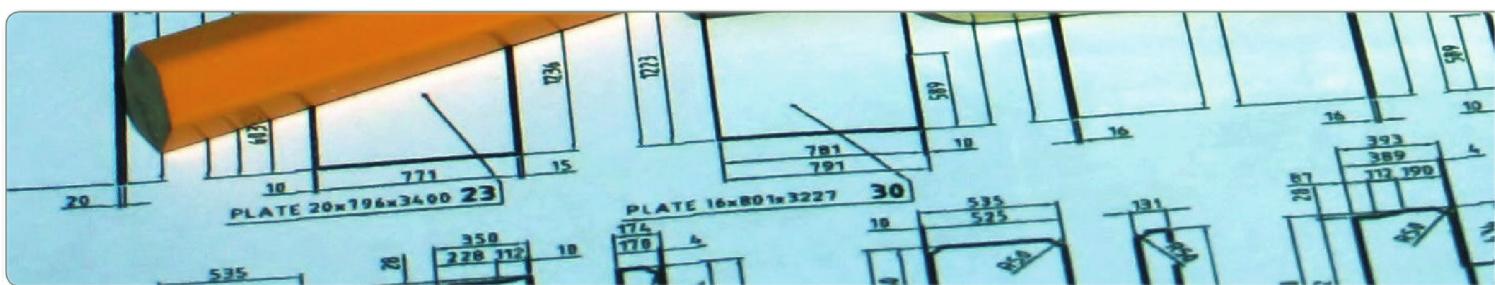
dando um jeitinho" que a pessoa com um déficit de conhecimento importante vá adquirindo este conhecimento de qualquer forma.

GPR 13. O progresso do projeto é monitorado com relação ao estabelecido no Plano do Projeto e os resultados são documentados;

Planejamento e controle são dois lados de uma mesma moeda. Não há muito sentido em se planejar se não houver um monitoramento constante que avalie se o plano está de fato sendo concretizado ou se necessita de alguma ação de correção de rota – ação esta que tanto pode estar ligada à execução quanto à modificação do próprio plano original, desde que haja a concordância dos diversos envolvidos no projeto.

Já ouvi muito a pergunta: pra que planejar se meu projeto sofre mudanças o tempo todo? Apesar de freqüente, não vejo muito sentido neste questionamento. Se há muitas mudanças, o planejamento poderá mudar mais. Só isso. Mas dependendo do contexto do projeto, se o monitoramento não for disciplinado o suficiente, tais mudanças irão ocorrer de forma totalmente desordenada e aí sim o fracasso é certo. Planejamento, controle e alguma dose de formalização são aliados e não inimigos dos projetos ditos mais "dinâmicos". Para evitar um trabalho muito grande na construção do plano inicial e nas modificações subsequentes, uma abordagem que pode ajudar muito é o planejamento por ondas sucessivas. Apesar do nome diferente, planejar desta forma nada mais é do que refinar o plano de forma progressiva. Ou seja, detalhamos apenas aquilo que é um pouco mais conhecido, aquilo que está próximo do momento atual. Etapas mais avançadas do projeto são planejadas de forma mais genérica dado que é impossível detalhar o que não se conhece.

Para ilustrar, vamos a um exemplo prático do que está sendo dito acima: se



ainda estou definindo os requisitos de um software, é inútil tentar prever datas de início de fim para a construção de cada um destes requisitos em separado. Ainda não há informação suficiente para este nível de detalhamento. Planejar desta forma nada mais é do que um exercício de futurologia. Entretanto, baseado em alguma medida de tamanho de software, pode-se planejar de forma menos detalhada e estimar que a etapa de construção de um conjunto de requisitos que totalizam N pontos de função, se iniciará no dia X e finalizará no dia Y.

GRE 1. O entendimento dos requisitos é obtido junto aos fornecedores de requisitos;

O texto deste resultado não parece nada absurdo, não é mesmo? Entender o que tem que ser feito (requisitos) junto a quem sabe o que tem que ser feito (fornecedor de requisitos) é algo tão óbvio que ficamos até desconfiados, e nos perguntamos se é isto mesmo o que o modelo quer dizer. Mas não há nenhuma pegadinha não. O que se espera é que os requisitos sejam entendidos, definidos e que tal definição tenha a participação de uma pessoa (ou de mais pessoas) chamada fornecedor de requisitos, que nada mais é do que o eleito por outros stakeholders do projeto (normalmente o patrocinador) para ser aquele que diz o que o software deve fazer para quem irá desenvolvê-lo. A única necessidade adicional não presente no texto acima é o fato de que tais requisitos devem ser documentados. Alguns críticos dos modelos de maturidade fazem críticas ferrenhas às exigências de documentação, mas convenhamos: algum nível de registro de requisitos é necessário haver. Mais uma vez, dizer que não se documenta requisitos porque eles mudam demais não me parece uma boa justificativa por motivos já expostos anteriormente – para que se possa mudar sem levar o projeto para o buraco

é preciso saber e se ter controle do que está sendo mudado. Além disso, não há um formato específico obrigatório para se documentar os requisitos. Deve-se documentar o necessário para que possa ser possível entender o que deve ser feito. Cada empresa pode e deve definir a sua forma de fazer isto.

MED2 - Um conjunto adequado de medidas, orientado pelos objetivos de medição, é identificado e/ou definido, priorizado, documentado, revisado e atualizado.

“Não se gerencia o que não se mede, não se mede o que não se define, não se define o que não se entende, não há sucesso no que não se gerencia.”

A frase acima é de William Edwards Deming, um dos papas da qualidade. Sua conclusão é clara: a empresa precisa se conhecer objetivamente para alcançar o sucesso. E medidas, bem entendidas, definidas, coletadas e analisadas são a base para este auto-conhecimento. Como saber se a produtividade da minha empresa está baixa ou alta? Ou até antes disto: como saber qual é a produtividade de minha empresa? Como avaliar se medidas tomadas para reduzir o número de defeitos em um produto foram ou não eficazes? Como aumentar a previsibilidade de meus projetos? Medindo, medindo e medindo – e claro, analisando e tirando o melhor proveito destas medições.

O resultado MED2 fala também em objetivos de medição. Estes objetivos são derivados das necessidades de informação da organização. Ou seja, devemos medir aquilo que é importante para a empresa. É importante que o processo operacional de medição esteja completamente alinhado com aquilo que a empresa precisa para melhorar sua gestão. Isso é verdade na indústria de software, mas também em qualquer outro ramo de atuação.

Os resultados citados e explicados

neste artigo são apenas uma pequena amostra do modelo. Eles foram extraídos de processos ligados aos níveis G e F apenas (primeiros níveis numa escala que se inicia no G e vai até o A). Entretanto, esta amostra, apesar de pequena, representa bem o propósito de um modelo de maturidade: ser um guia de referência para a aplicação de boas práticas dentro das empresas que desenvolvem software. É interessante observar que dentre os resultados que apresentei não há nada ligado a processos de engenharia de software. Não citei nada referente a design, codificação, ferramentas, testes automatizados, etc. Isso costuma gerar uma certa frustração no profissional essencialmente técnico. Mas há uma explicação para esta ausência. Os níveis iniciais do MPS.BR (G e F) se preocupam em arrumar a casa do ponto de vista gerencial. Processos mais técnicos são também bastante relevantes, mas é impossível aplicá-los de forma consistente se a gestão dos projetos de software é caótica. Esses processos mais técnicos são tratados mais à frente, em níveis superiores depois que uma base sólida já foi construída.

Conclusão

Qualidade de software não é uma ciência exata. Discussões e divergências sempre ocorrerão. A internet potencializa as discussões e gera um fluxo maior de informações. O problema é que há informações certas e erradas, fundamentadas e levianas, racionais e apaixonadas. É cada vez mais importante saber separar o joio do trigo.

Modelos de maturidade não são garantia de sucesso absoluto para nenhuma empresa. Mas certamente servem muito bem como um guia de referência, um caminho a ser seguido por empresas que desejam melhorar seu desempenho operacional de forma consistente ao longo do tempo. ●

