Engenharia de Software :
Adequação de perfis de personalidades aos papéis funcionais no desenvolvimento de software

engenharia de software magazine

Edição 18 :: Ano 2

ISSN 1983127-7

Qualidade de Software
Entenda como fatores humanos podem impactar a qualidade

Qualidade de Software
Conheça alguns modelos de apoio à avaliação de interfaces com usuário

Gerência de Configuração
Saiba como implementar a gerência de configuração a partir do zero

GESTÃO DE PROJETOS

- Entenda como os modelos de maturidade apóiam o planejamento e a gerência
- Conheça os diferentes tipos de Escritório de Projetos (PMO)

Validação, Verificação & Teste
Descubra as competências exigidas de um profissional de Testes

Aulas desta edição:

- Definições Iniciais sobre Validação, Verificação e Testes – Partes 9 a 11
- Diagrama de Classes na Prática – Partes 4 a 8

Conhecimento faz diferença!

Mais de 60 mil downloads na primeira edição!

Faça já sua assinatura digital! | *www.devmedia.com.br/es*

Faça um *up grade* em sua carreira.
Em um mercado cada vez mais focado em qualidade ter conhecimentos aprofundados sobre requisitos, metodologia, análises, testes, entre outros, pode ser a diferença entre conquistar ou não uma boa posição profissional. Sabendo disso a DevMedia lança para os desenvolvedores brasileiros sua primeira revista digital totalmente especializada em Engenharia de Software. Todos os meses você irá encontrar artigos sobre Metodologias Ageis; Metodologias tradicionais (*document driven*); ALM (*application lifecycle management*); SOA (aplicações orientadas a servicos); Analise de sistemas; modelagem; Métricas; orientação à objetos; UML; testes e muito mais. **Assine já!**

DevMedia GROUP

engenharia de software magazine

Ano 2 - 18ª Edição 2009 **Impresso no Brasil**

Corpo Editorial

Colaboradores
Rodrigo Oliveira Spínola
rodrigo@sqlmagazine.com.br

Marco Antônio Pereira Araújo
Eduardo Oliveira Spínola

Revisão e Supervisão
Thiago Vincenzo Ciancio - thiago.v.ciancio@devmedia.com.br

Coordenação Geral
Daniella Costa - daniella@devmedia.com.br

Diagramação
Janete Feitosa

Capa
Romulo Araujo - romulo@devmedia.com.br

Na Web
www.devmedia.com.br/esmag

DevMedia group

Apoio

ASSESPRO PARANÁ

Parceiros:

kali software

Atendimento ao Leitor

A DevMedia conta com um departamento exclusivo para o atendimento ao leitor. Se você tiver algum problema no recebimento do seu exemplar ou precisar de algum esclarecimento sobre assinaturas, exemplares anteriores, endereço de bancas de jornal, entre outros, entre em contato com:

Cristiany Queiróz – Atendimento ao Leitor
www.devmedia.com.br/mancad
(21) 2220-5338

Kaline Dolabella
Gerente de Marketing e Atendimento
kalined@terra.com.br
(21) 2220-5338

Publicidade

Para informações sobre veiculação de anúncio na revista ou no site entre em contato com:

Kaline Dolabella
publicidade@devmedia.com.br

Fale com o Editor!

É muito importante para a equipe saber o que você está achando da revista: que tipo de artigo você gostaria de ler, que artigo você mais gostou e qual artigo você menos gostou. Fique a vontade para entrar em contato com os editores e dar a sua sugestão!

Se você estiver interessado em publicar um artigo na revista ou no site SQL Magazine, entre em contato com os editores, informando o título e mini-resumo do tema que você gostaria de publicar:

Rodrigo Oliveira Spínola - Colaborador
editor@sqlmagazine.com.br

EDITORIAL

Nesta edição a Engenharia de Software Magazine destaca como tema de capa o planejamento e a gerência de projetos. Para isto, trazemos duas matérias muito interessantes. No primeiro deles, **Modelos de Maturidade na Gestão de Projetos**, será apresentado o funcionamento dos principais modelos de maturidade em gerenciamento de projetos, além de uma análise comparativa, onde serão abordados aspectos como semelhanças, pontos fortes e limitações.

No segundo artigo, **Project Management Office - Conceito e evolução desta entidade tão atual**, veremos que o PMO pode ser entendido como uma estrutura organizacional de apoio ao gerenciamento de projetos e portfólio nas organizações – Escritório de Projetos. O escritório de projetos, a depender da maturidade organizacional, pode fornecer suporte metodológico, coaching aos gerentes da organização, além de sugerir uma administração centralizada do portfólio de projetos da organização permitindo um ponto de contato único e de apoio a decisão. No decorrer desse artigo, será mostrada a evolução do PMO, os tipos já propostos e os aceitos na atualidade com suas principais características.

Além destas matérias, esta edição traz mais cinco artigos:

- GCO: Por onde começar?
- Quais as competências exigidas de um Profissional de Testes?
- Fatores humanos: A influência na Qualidade de Software
- Modelo de Avaliação de Interfaces
- Adequação de Perfis de Personalidades aos Papéis Funcionais no Desenvolvimento de Software

Desejamos uma ótima leitura!

Desejamos uma ótima leitura!
Equipe Editorial Engenharia de Software Magazine

Rodrigo Oliveira Spínola
rodrigo@sqlmagazine.com.br
Doutorando em Engenharia de Sistemas e Computação (COPPE/UFRJ). Mestre em Engenharia de Software (COPPE/UFRJ, 2004). Bacharel em Ciências da Computação (UNIFACS, 2001). Colaborador da Kali Software (www.kalisoftware.com), tendo ministrado cursos na área de Qualidade de Produtos e Processos de Software, Requisitos e Desenvolvimento Orientado a Objetos. Consultor para implementação do MPS.BR. Atua como Gerente de Projeto e Analista de Requisitos em projetos de consultoria na COPPE/UFRJ. É Colaborador da Engenharia de Software Magazine.

Marco Antônio Pereira Araújo
(maraujo@devmedia.com.br)
Doutor e Mestre em Engenharia de Sistemas e Computação pela COPPE/UFRJ - Linha de Pesquisa em Engenharia de Software, Especialista em Métodos Estatísticos Computacionais e Bacharel em Matemática com Habilitação em Informática pela UFJF, Professor e Coordenador do curso de Bacharelado em Sistemas de Informação do Centro de Ensino Superior de Juiz de Fora, Professor do curso de Bacharelado em Sistemas de Informação da Faculdade Metodista Granbery, Professor e Diretor do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Fundação Educacional D. André Arcoverde, Analista de Sistemas da Prefeitura de Juiz de Fora, Colaborador da Engenharia de Software Magazine.

Eduardo Oliveira Spínola
(eduspinola@gmail.com)
É Editor das revistas Engenharia de Software Magazine, SQL Magazine, WebMobile. É bacharel em Ciências da Computação pela Universidade Salvador (UNIFACS) onde atualmente cursa o mestrado em Sistemas e Computação na linha de Engenharia de Software, sendo membro do GESA (Grupo de Engenharia de Software e Aplicações).

Caro Leitor

Para esta edição, temos um conjunto de **8 vídeo aulas**. Estas vídeo aulas estão disponíveis para download no Portal da Engenharia de Software Magazine e certamente trarão uma significativa contribuição para seu aprendizado. A lista de aulas publicadas pode ser vista abaixo:

Tipo: Validação, Verificação e Teste
Título: Definições Iniciais sobre Validação, Verificação e Testes – Partes 9 a 11
Autor: Rodrigo Oliveira Spínola
Mini-Resumo: Atividades de VV&T são fundamentais ao longo de todo o processo de desenvolvimento de software. Entretanto, ainda é comum o pensamento de que só devemos avaliar a qualidade do software ao final do processo através de atividades de testes. Veremos nesta série de vídeo aulas o porquê desta perspectiva não estar correta e entenderemos os principais conceitos associados às atividades de validação, verificação e testes. Nestas 3 aulas, serão apresentados conceitos associados à revisão por checklist e baseada em técnicas de leitura.

Tipo: Projeto
Título: Diagrama de Classes na Prática – Partes 4 a 6 – Estudo de Caso 1
Autor: Rodrigo Oliveira Spínola
Mini-Resumo: Esta aula é parte de uma série sobre a construção de diagrama de classes da UML. O objetivo do conjunto de aulas é apresentar de forma prática como elaborar o diagrama de classes a partir de diferentes estudos de caso. Nestas aulas, o diagrama de classes do primeiro estudo de caso é finalizado.

Tipo: Projeto
Título: Diagrama de Classes na Prática – Partes 7 e 8 – Estudo de Caso 2
Autor: Rodrigo Oliveira Spínola
Mini-Resumo: Esta aula é parte de uma série sobre a construção de diagrama de classes da UML. O objetivo do conjunto de aulas é apresentar de forma prática como elaborar o diagrama de classes a partir de diferentes estudos de caso. Nestas aulas, é dado início à elaboração do diagrama de classes para o segundo estudo de caso. Nesta etapa inicial, serão identificadas as classes e os atributos do modelo.

ÍNDICE

Você não está mais sozinho.

GCO: Por onde começar?

Como implementar a gerência de configuração a partir do zero

Samuel Diniz Casimiro
samuel.casimiro@gmail.com
É servidor público e trabalha como analista de sistemas em um grande órgão do governo; trabalhou cerca de 10 anos na área de desenvolvimento de sistemas de um grande banco brasileiro; é certificado SCJP e SCWCD; especialista em soluções J2EE para intranets; consultor de TI para pequenas e médias empresas; e crítico ferrenho dos modelos de desenvolvimento adotados atualmente nas grandes empresas.

Imagine que, de repente, o seu gerente entra na sala e te diz: "A Diretoria decidiu que vai implementar o MPS.BR aqui na nossa empresa. Vamos montar uma equipe e você foi um dos escolhidos. Resolvemos que você ficará encarregado do processo Gerência de Configuração (ler **Nota 1**)."

De que se trata o artigo?

Veremos como se pode começar a implementação da Gerência de Configuração com o objetivo de obter os níveis G e F do MPS.BR. O escopo deste artigo é, portanto, a gerência de configuração de software, focada no ciclo de desenvolvimento – e não a gerência de configuração como abordada pelo ITIL, que é mais abrangente.

Para que serve?

O propósito do processo Gerência de Configuração – GCO – é registrar e garantir a integridade dos produtos de trabalho dos demais processos relacionados com o ciclo de desenvolvimento de software. Sem o processo GCO, o software evolui sem controle e isso pode gerar um caos nas bibliotecas de software e na documentação de projetos e processos de forma geral.

Em que situação o tema é útil?

O processo GCO, como descrito no MPS.BR, descreve uma série de boas práticas para a implementação de um Sistema de Gerência da Configuração e de atividades que vão garantir um melhor controle dos produtos de trabalho dos processos relacionados com o desenvolvimento de software.

Nota do DevMan 1

Gerência de Configuração

Sob a perspectiva de desenvolvimento, a Gerência de Configuração de Software abrange três sistemas principais: controle de modificações, controle de versões e controle de gerenciamento de construção.

O sistema de controle de versões permite que os artefatos sob Gerência de Configuração evoluam de forma distribuída, concorrente e disciplinada, evitando perdas ou sobreposições durante o desenvolvimento e a manutenção do artefato. Podemos citar como exemplos de ferramentas de mercado: CVS, Subversion, IBM Rational ClearCase e Microsoft Visual Source Safe.

Um item, ao ser desenvolvido, evolui até que atinja um estado em que atenda aos propósitos para o qual foi criado. Isso implica em diversas alterações, gerando uma versão do item a cada estado (Munch, 1996). Para estabelecer o controle sobre as diversas versões, todas as versões devem ser armazenadas e identificadas. Isso, geralmente, é feito com o auxílio de uma ferramenta.

A versão do item pode ser incluída no esquema de identificação ou ser acessível a partir de uma tabela à parte. É conveniente que o esquema de identificação das versões dos itens seja feito em forma de árvore, pois ao mesmo tempo em que mantém um histórico das versões dos itens, permite identificação única e ramificações a partir de qualquer versão.

O sistema de controle de modificações armazena todas as informações geradas durante o andamento das solicitações de modificação e relata essas informações aos participantes interessados e autorizados. Podemos citar como exemplos de ferramentas de mercado: Bugzilla, Jira, Trac e IBM Rational ClearQuest.

O objetivo dessa tarefa de gerência de configuração é relatar a todas as pessoas envolvidas no desenvolvimento e na manutenção do software as seguintes informações sobre as alterações na configuração de software:

O que aconteceu?

Quem o fez?

Quando aconteceu?

O que mais será afetado?

Para isso, deve ser criado um banco de dados sobre as ocorrências na gerência de configuração. Esse banco de dados deve estar disponível aos desenvolvedores com acesso através de palavras-chave. Além disso, deve ser gerado regularmente um relatório de situação para informar as alterações mais importantes. O acesso rápido às informações sobre a configuração agiliza o processo de desenvolvimento e melhora a comunicação entre as pessoas, o que é uma maneira de eliminar muitos problemas relativos à modificação do mesmo item de informação, com intenções diferentes e conflitantes.

O sistema de gerenciamento de construção automatiza o processo de transformação dos diversos artefatos do software que compõem um projeto em um sistema executável propriamente dito. Este processo é nomeado construção do software que, por exemplo, testa e empacota a aplicação java como um arquivo jar. Este processo ocorre de forma aderente às normas, procedimentos, políticas e padrões definidos para o projeto. Podemos citar como exemplos de ferramentas de mercado: Maven e Apache Ant.

As vantagens da utilização da Gerência de Configuração de Software são inúmeras. Dentre elas, podemos listar: (1) ganho de produtividade e eficiência; (2) diminuição do retrabalho e dos erros; (3) aumento da disciplina no processo de desenvolvimento; (4) aumento da memória organizacional; (5) acesso às informações qualitativas e quantitativas referentes ao processo de desenvolvimento, como por exemplo, medida de esforço para efetuar uma alteração e freqüência de modificações por componente; (6) possibilidade de estabelecer uma trilha de auditoria indicando por que, quando e por quem um artefato foi alterado; (7) auxílio à gerência de projetos e (8) garantia de ambiente estável no qual o produto deve ser desenvolvido.

Qualquer semelhança com a realidade é mera coincidência. Também, quem mandou você ficar enviando e-mails para os gerentes com sugestões sobre melhoria de processos? Agora eles imaginam que você é alguém com perfil para trabalhar na implementação do MPS.BR. E mais, assumiu um dos maiores desafios: a gerência de configuração.

Mas não precisa se desesperar. Afinal, você se considera muito inteligente, não é verdade? Então vai dar conta do recado (e nós da Engenharia de Software estamos aqui para te ajudar, não é verdade?). Passado o susto dessa notícia, é bem provável que a sua empresa providencie algum tipo de treinamento para você. Além disso, a essa altura do campeonato, você provavelmente já sabe o que é gerência de configuração (confira o artigo "Gerência de Configuração de Software", da 2ª. Edição da revista Engenharia de Software). Mais daí surge a pergunta: Por onde começar?

Bom, você que leu os Guias de Implementação da Softex (veja os links no final do artigo), já percebeu que o processo Gerência de Configuração – GCO – não é exigido no primeiro Nível do modelo, o Nível G. A GCO só é exigida do Nível F em diante. Mas não fique excessivamente tranqüilo. É um erro achar que você só deve se preocupar com GCO depois que a sua empresa obter o Nível G. Na verdade, esse é um dos maiores erros que levam as empresas a sofrerem tanto no momento de implementar a GCO. Como melhores práticas, dispare ações de GCO imediatamente, pois o tempo corre contra você. Há muito trabalho a se fazer antes de seus colegas responsáveis pela Gerência de Projetos – GPR – e pela Gerência de Requisitos – GRE – obterem o Nível G. A GCO precisa estar lado a lado dos demais processos, desde o seu início. Por quê? Porque o processo GCO se cruza com todos os demais processos, inclusive com ele mesmo, por meio do atributo de processo RAP 13: "Os produtos de trabalho são colocados em níveis apropriados de controle". Isso é gerência de configuração. Por isso, esteja preparado desde o início. É mais fácil começar certo do que arrumar depois. Como diz o ditado, é melhor prevenir do que remediar!

E vamos logo ao trabalho! Veremos a seguir uma série de pontos importantes por onde você pode começar, focando apenas na GCO. Este artigo não tem a pretensão de avaliar cada cruzamento da GCO com todos os demais processos. Quando falarmos de cruzamentos, vamos nos concentrar nos processos dos níveis G e F, que é por onde você vai começar e é quando a GCO precisa ser consolidada na organização. Além do mais, a curva de aprendizado é muito mais alta no início. Começando certo fica mais fácil de implementar a GCO nos próximos níveis.

No mais, vale ressaltar que, ao decidir implementar o MPS.BR, sua empresa provavelmente já tomou uma série de providências. Coisas tais como iniciar um projeto, definir patrocinadores, alocar pessoas como você, providenciar treinamentos, talvez contratar uma consultoria, definir um cronograma, designar responsabilidades e garantir a colaboração de todos na empresa. Não é o objetivo deste artigo tratar desses assuntos, que estão muito mais relacionados com gestão do que com a GCO propriamente dita. Vamos imaginar que os gerentes da sua empresa fizeram seu dever de casa. Em outros artigos, talvez possamos nos concentrar melhor no papel dos gerentes com relação à implementação desse modelo de melhoria. Aqui, por enquanto, vamos nos preocupar nos aspectos práticos e operacionais, ok?

Comece pelos RAPs

Como você já deve saber, os resultados esperados exigidos pelo modelo se dividem em Resultados dos Atributos de Processo – RAPs – e em Resultados Esperados do Processo – que aqui vamos chamar simplesmente de "resultados esperados". Ao passo que os RAPs se referem a atributos genéricos que se aplicam a todos os processos e indicam a capacidade deles, os resultados esperados são específicos de cada processo.

Se você já teve a chance de acompanhar uma avaliação MPS.BR, ou se pelo menos já participou do preenchimento das planilhas de indicadores (também chamada "planilha de evidências"), então deve ter reparado que os RAPs são avaliados por último, depois dos resultados esperados de cada processo. Nos relatórios de GAP-Analysis (para mais detalhes, veja o **quadro** "Avaliação preliminar: Vale a pena?"), os RAPs também ficam por último. E o mesmo ocorre nos guias de implementação de cada nível. Até o próprio RAP 1, "O processo atinge seus resultados esperados", coloca os RAPs em segundo plano. O que se pretende dizer com isso é: se você não tomar cuidado, vai acabar caindo na armadilha de deixar a implementação dos RAPs para depois dos resultados esperados. E isso, prezado leitor, é um grande erro! Como assim? Vamos esclarecer melhor essa questão.

Os RAPs, especialmente do 2 ao 10 e do 15 em diante, fornecem a base para a implementação de todo o processo. Estamos falando da institucionalização do processo, ou seja, o seu alicerce. Por exemplo, o RAP 2 fala de uma "política organizacional" para o processo. Pergunta: como você vai implementar a operacionalização de um processo sem uma política organizacional estabelecida para ele? Outro exemplo, como você vai definir os pontos em que os resultados esperados vão ser produzidos se o processo ainda não foi planejado, conforme exige o RAP 3?

É claro que nem todos os RAPs devem ser observados antes dos resultados esperados, mas provavelmente você vai ter mais facilidade se começar por eles, especialmente pelo RAP 2. Uma boa dica é: dê uma estudada geral nos RAPs, implemente o RAP 2 para GCO e depois comece a implementar os resultados esperados um a um sempre repassando os demais RAPs e verificando o que se aplica ao momento. Vamos falar mais sobre isso a seguir.

Avaliação preliminar: Vale a pena?

Uma prática que se tem visto em algumas empresas, especialmente durante a implementação dos níveis G e F, é a contratação de uma avaliação preliminar – também chamada de GAP-Analysis (análise das lacunas). Essa avaliação também é feita por uma instituição avaliadora autorizada pela Softex e consiste em verificar se a empresa apresenta alguma lacuna com relação a um nível de maturidade previamente escolhido pela empresa. Assim como a avaliação oficial, essa avaliação preliminar gera, ao final da análise, um relatório que aponta os resultados esperados, gerais ou específicos, para os quais não foram encontradas evidências. Por isso, não espere algo parecido com uma consultoria. Uma consultoria de implementação vai indicar não só as lacunas, mas também *meios* de sanar tais lacunas. O máximo que você pode conseguir de uma avaliação preliminar é uma breve reunião com os avaliadores para tirar algumas dúvidas sobre o relatório.

Por isso, devido ao fato de ser tão precária e não indicar os *meios* de se sanar as lacunas, alguns especialistas acreditam que essa avaliação preliminar seja totalmente dispensável. Assim, se a sua empresa não está disposta a contratar uma consultoria de implementação, sugiro também não contratar essas avaliações preliminares. Talvez seja mais interessante contratar um profissional de mercado com boa experiência em avaliações. Um profissional assim, dentro da sua empresa, vai não só apontar as lacunas, como também propor meios para evidenciar todos os resultados que se esperam.

Primeiro a política

A política organizacional é de suma importância para que você saiba o que a alta administração da sua empresa espera do processo GCO. Tudo bem se você fizer parte do *staff*: um analista ou talvez um técnico. Mas, se você está conduzindo e coordenando a implementação do processo GCO, é sua responsabilidade provocar a alta administração para construir essa política e se comprometer com ela. Só não pense que eles vão fazer todo o trabalho para você. Na prática, provavelmente você terá que construir essa política, propô-la, e eles vão apenas analisá-la e aprová-la. É importante lembrar que essa aprovação tem que ser registrada, pois isso é exigido pelo MPS.BR. Essa aprovação pode ser uma ata de reunião, um comunicado da alta administração para os seus funcionários divulgando a política, etc.

Caso você esteja tendo dificuldades em construir uma política de GCO, pode-se fazer um *benchmarking* com outras empresas. Mantenha contatos com pessoas da área de engenharia de software. Esses contatos podem te ajudar com informações e outras dicas. Uma teia de relacionamentos assim constitui um excelente fórum para discutir assuntos relacionados com MPS.BR. Sites de relacionamento, como o LinkedIn, Facebook e Orkut são ótimas ferramentas para isso. Seja membro das comunidades mais populares que tratam de MPS.BR e CMMI. Existem grupos de discussão muito pertinentes também no Yahoo! Groups e no Google Groups. Outra fonte de informação para a sua política de GCO são as demais políticas que porventura existam na sua organização, como a política de segurança da informação. Aproveite a estrutura e o formato que a sua organização utiliza para essas políticas – é um bom começo.

Depois o plano

Feita a política, é hora de passarmos para o RAP 3. Esse RAP fala do plano do processo. O plano nada mais é do que um documento que descreve o processo e define como ele deve ser executado, ou seja, estamos falando de planejamento. Isso envolve desenhar o processo, suas atividades e seus produtos ou artefatos. O plano também deve especificar os recursos que serão utilizados na execução do processo, as pessoas e os papéis que desempenharão as atividades (RAP 5 e 6). Por fim, o plano do processo deve deixar claro o momento em que as atividades serão executadas. Por tudo isso, muitos consideram o plano como o artefato mais importante do processo.

Ainda no plano e já pensando no RAP 4, aproveite para definir atividades de monitoramento do processo, ou seja, formas de verificar se o processo está sendo executado conforme as especificações do plano. São as famosas atividades de verificação. Pode-se aproveitar o plano também para atender ao RAP 12. Esse RAP, quando aplicado ao processo GCO, trata da gerência de configuração do próprio processo. Se você já definiu os produtos de trabalhos das atividades do processo, agora é o momento de definir os níveis de controle de cada produto ou artefato. A definição dos níveis de controle responde às seguintes perguntas: O artefato será armazenado em meio controlado? Onde? Quem poderá acessar? Alterações nesse artefato precisam ser controladas? Como? Será utilizado um sistema de versionamento? Esse artefato ficará sob baseline? Essas perguntas podem ser respondidas no próprio plano do processo de GCO. O MPS.BR sugere vários níveis de controle (para uma descrição de alguns deles, veja a **Tabela 1** – Níveis de controle). Mas nada impede que você defina outros níveis que melhor se adaptem à realidade da sua organização.

Treinamento não é tempo perdido

Outra coisa importante que não se deve deixar para depois é o treinamento das pessoas (RAP 7). Comece solicitando treinamento para você mesmo, caso ainda não tenha feito. Em seguida, promova treinamento para aqueles que foram indicados pela administração para executar as atividades do processo definidas no plano de GCO. Nessa fase, pode ser de ajuda encorajar que algumas pessoas se certifiquem. Apesar de as certificações não terem muito valor prático, pois elas auferem basicamente apenas conhecimento teórico, a busca por elas acaba gerando alguma empolgação das pessoas pelo assunto da certificação. E empolgação é muito importante nessa fase da implementação dos processos. Precisamos de pessoas engajadas, dispostas a promover a mudança organizacional necessária para implementar o processo GCO. Precisamos de evangelistas. Quanto mais gente do seu lado, melhor.

Reparou como tanta coisa depende dos RAPs 2 e 3? Vamos agora passar para outro aspecto muito relacionado com os RAPs 12 e 13.

Preparando processos e artefatos

Os RAPs 12 e 13 estão diretamente relacionados com GCO. O próprio guia de implementação deixa isso bem claro. O RAP 12 exige que sejam definidos requisitos de GCO para todos os produtos de trabalho, isto é, para todos os artefatos consumidos e produzidos nas atividades dos demais processos. Lembra quando falamos do plano de GCO, que teríamos que definir níveis de controle para os produtos de trabalho do próprio processo GCO? Pois é. Vamos precisar fazer isso para os demais processos também. Em que plano você vai fazer isso? No plano de GCO? Ou nos planos de cada processo? Bom, essas são questões que você terá que responder. Uma sugestão: defina isso em um único catálogo para todos os processos, inclusive para o próprio processo GCO (vamos falar mais sobre esse catálogo abaixo).

É nos RAPs 12 e 13 que a GCO se cruza com os demais processos. Daí a importância de já nos preocuparmos com isso desde o início da implementação dos demais processos, mesmo que você esteja ainda no Nível G.

Nível de controle	Descrição
Observação: Os níveis de controle descritos abaixo podem ser combinados conforme a necessidade.	
Controle de acesso por autenticação	Esse é, talvez, o nível de controle mais precário. Nesse nível, o sistema não faz qualquer tipo de restrição ao acesso do recurso – apenas identifica o usuário mediante autenticação, geralmente gravando um log de acesso.
Controle de acesso por autorização	Esse nível de controle é geralmente implementado em conjunto com a autenticação e consiste em liberar o acesso do usuário apenas a recursos previamente autorizados. Essa autorização geralmente é concedida por usuários de níveis hierárquicos superiores.
Controle pessimista	O procedimento de check-out de um recurso impede que ele seja acessado por outro usuário até que seja realizado o procedimento de check-in. Em outras palavras, as alterações só podem ser feitas por um usuário de cada vez.
Controle otimista	O procedimento de check-out não impede que ele seja acessado por outro usuário. Entretanto, ao realizar o procedimento de check-in, o sistema é capaz de verificar conflitos nas alterações e oferecer sugestões para resolvê-los. Esse nível de controle é muito utilizado e está disponível em muitas ferramentas de controle de versões.
Apenas armazenar	Envolve apenas armazenar o recurso em local pré-estabelecido. Por exemplo, em uma pasta na rede local ou uma pasta de um repositório do controle de versões. O local de armazenamento, geralmente é descrito no plano do processo ou em um catálogo de artefatos, se houver.
Colocar sob baseline	Envolve armazenar o recurso em um repositório do controle de versões e definir procedimentos para proteger a baseline. Isso geralmente envolve o trabalho com branches e a utilização de recursos de rotulação de versões (tags). A colocação de itens sob baseline geralmente envolve um processo formal de verificação da qualidade e aprovação das alterações.

Tabela 1. Níveis de controle

Defina logo cedo os dados de controle

Mesmo que a sua empresa ainda esteja implementando o Nível G, é interessante já começar definindo os níveis de controle para os artefatos do processo GRE e GPR. E, para definir os níveis de controle, a documentação dos processos e dos artefatos precisa estar preparada para isso. Quando vocês estiverem no Nível F, bastará observar os níveis de controle que vocês já definiram anteriormente.

Para começar, toda a documentação dos processos e os templates dos artefatos precisam conter dados suficientes para as atividades de controle de versões e controle de modificações. A GCO é a base para o gerenciamento de mudanças.

Convém que os documentos do processo tenham um número de versão e que a documentação do processo indique claramente o período de vigência de cada versão do processo e quais versões dos templates fazem parte daquela versão do processo. Isso é importante para que aqueles que vão executar o processo saibam exatamente quais versões da documentação e dos templates devem ser utilizadas em um dado momento.

Além do controle de versões, a GCO trata também do controle de modificações. Assim, é importante identificar na documentação do processo e nos templates não só um número de versão, mas também quem foi responsável por aquela versão, quando ela passou a vigorar e o quê motivou cada alteração. No futuro, quando você estiver implementando o Nível F, vai utilizar esses dados para realizar o controle de alterações nos artefatos. Você deve ter informação suficiente para responder às seguintes perguntas: Que versão do processo ou template devo utilizar? Qual a diferença básica entre a versão atual e as anteriores?

Catálogo de artefatos

Entretanto, não basta identificar a versão dos artefatos e registrar aquilo que motivou cada modificação. Para que um sistema de GCO se torne viável, é necessário definir padrões para identificar cada artefato de forma inequívoca. Em outras palavras, você precisa definir padrões de nomenclatura e um documento que descreva de forma simples, porém suficiente, o propósito de cada artefato. Para isso, a nossa sugestão é que você desenvolva uma espécie de sistema catálogo que serviria de repositório para toda a documentação dos processos e os seus respectivos templates de artefatos. Esse catálogo implementaria mecanismos de classificação dos documentos, ferramentas de pesquisa e controle de versões e de modificações. Um catálogo assim poderia ser implementado com base em alguma ferramenta para controle de versões, como o CVS. Esse catálogo seria parte integrante do sistema de GCO. Vamos falar mais sobre isso depois do próximo tópico.

Resultados esperados mais críticos

Vamos agora supor que você já implementou o Nível G e que os processos GPR e GRE estão preparados para implementar definitivamente os RAPs 12 e 13, que tratam de GCO, conforme sugerido acima. O próximo passo agora é implementar o Nível F e o processo GCO propriamente dito. O que fazer agora?

O que é baseline?

Este é um dos conceitos mais importantes da GCO. Se formos traduzir ao pé da letra, baseline significa literalmente "linha base". De início, isso pode parecer estranho, mas a tradução utilizada nos textos em português sobre ITIL talvez possa ajudar: "Configuração de referência". Essa definição representa melhor o real significado do termo.

Baseline é justamente isso, a configuração de referência de um ou mais ICs. E, nesse ponto, a configuração se traduz em versão. Em outras palavras, uma baseline representa um ou mais ICs cuja versão sirva de referência para o processo de desenvolvimento de software. Vamos exemplificar.

Imagine que a sua empresa possua três ambientes: desenvolvimento, testes/homologação e produção. Concorda que as versões que estão rodando em produção servem de referência para futuras alterações? Os ICs de produção, em geral, compõem o que chamamos de "baseline de sistema". Se, por exemplo, alguma implantação apresentar problemas, podemos restaurar a última baseline de sistema que, teoricamente, tudo voltará ao normal. A baseline de sistema serve de base também para futuros projetos. Aplicações cliente/servidor, em geral, possuem apenas uma baseline de sistema. Já aplicações desktop podem possuir várias (ex.: Word 97, Word 2000, Word XP).

Vale a pena também falar de outro tipo de baseline, a "baseline de projeto". Digamos que você esteja desenvolvendo um software de forma iterativa em um projeto. A cada iteração, novos ICs são desenvolvidos, testados e homologados. De forma que, ao final de uma iteração, você terá um conjunto de ICs mais ou menos estáveis, que foram testados e aprovados. Concorda que esses ICs servirão de base para a próxima iteração do projeto? É justamente isso que chamamos de baseline de projeto: Um conjunto de ICs cuja configuração (leia-se "versão") servirá de base para a continuação do projeto.

Portanto, baseline é um conjunto de ICs em versões de referência, isto é, que servem de base para outras versões, novos ICs, etc. E, em se tratando de versões de referência, elas devem ser controladas com rigor. Pressupõe-se que ICs que compõem uma baseline foram de alguma forma testados e aprovados para se garantir certo grau de confiança nessa baseline.

Vamos repassar alguns resultados esperados mais críticos e ver como, na prática, poderemos alcançar esses resultados e conseguir as evidências necessárias à certificação.

Sistema de GCO

Sem dúvida, o resultado esperado mais crítico é o primeiro, GCO 1. Afinal, se um sistema de gerência da configuração (SGCO) for bem implementado, vários resultados esperados – e até alguns RAPs – serão obtidos como conseqüência disso. O problema é que não existe ainda um único sistema que implemente plenamente a GCO como o MPS.BR exige. O que temos são alguns sistemas que implementam subconjuntos das funções do GCO.

Como já foi explicado em artigos anteriores (confira o artigo "Gerência de Configuração de Software", da 2ª. Edição), o sistema de GCO, como descrito no MPS.BR, deve contemplar três subconjuntos principais de funções: controle de versões, controle de modificações e controle de construção ou liberação. Mas, como acabamos de dizer, não existe ainda um único

sistema que ofereça todas essas funções num único pacote. O que existe são alguns sistemas para cada um desses conjuntos de funções, que podem ou não ser integrados.

Por exemplo, para o controle de versões, temos o conhecido CVS, o Subversion, o IBM Rational ClearCase e o Microsoft Visual Source Safe (VSS), dentre outros. Para o controle de modificações, temos o Bugzilla, Jira, Trac e o IBM Rational ClearQuest, só para citar alguns. Para o controle de construção, a oferta é menor, pois esse tipo de ferramenta depende muito da sua infraestrutura de TI e, em geral, é desenvolvido em cada empresa. É claro que quando o assunto é construção (build), não podemos deixar de lado o Apache Ant. Contudo, cabe à empresa desenvolver seus próprios scripts de build e, assim, automatizar as atividades de liberação de software. O difícil é integrar tudo isso para garantir a rastreabilidade.

Diante dessa dificuldade, o quê fazer? Bom, se você trabalha em uma grande empresa, sugiro que você avalie a aquisição de ferramentas proprietárias, como as da IBM. As ferramentas já citadas se integram perfeitamente e podem, juntas, fornecer todas as funções de um sistema de GCO. Entretanto, se você trabalha em uma pequena empresa, adquirir uma ferramenta assim talvez não seja viável. Nesse caso, sugiro que você desenvolva seu próprio sistema de GCO, provavelmente utilizando ferramentas opensource como o CVS e o Apache Ant para o controle de versões e liberações, respectivamente, e implemente as funções de controle de modificações por conta própria, baseado em banco de dados mesmo e alguma linguagem de programação da sua preferência. Se a sua empresa utiliza o Eclipse como IDE, existem alguns plugins muito interessantes que podem ser integrados ao CVS e ao Apache Ant como o Synergy ALM, TD/OMS e o Aldon Application Lifetime Management Suite.

Identificação dos itens de configuração

Vamos agora ao GCO 2, que em minha opinião também é crítico (afinal, o que NÃO é crítico?). Esse resultado esperado trata da identificação dos itens de configuração. Como o propósito deste artigo é falar da implementação de GCO, não vamos nos deter aqui na definição do que é um item de configuração – IC. Se você solicitou o treinamento sugerido no início deste artigo, já deve saber muito bem do que se trata. Para o que pretendemos aqui, vamos considerar como itens de configuração todos os artefatos críticos para os projetos de desenvolvimento de software. O MPS.BR exige que esses ICs sejam identificados (elencados) com base em critérios e que os níveis de controle sejam definidos.

Bom, já falamos sobre níveis de controle. Mas o que vem a ser esses critérios de identificação? É tudo aquilo que o levou a tratar um artefato como IC. Em geral, isso é feito com base na criticidade do artefato. Por exemplo, a especificação do software é algo crítico, não concorda? Modificações na especificação geram impactos em todo o processo de desenvolvimento. Portanto, sem dúvida, a especificação deve ser encarada como um IC. Casos de uso, diagramas, modelos de dados e o próprio código fonte também devem ser tratados como ICs, só para citar alguns produtos de trabalho críticos. Mas nem só os produtos de trabalho dos projetos são ICs. Os produtos de trabalho organizacionais, se críticos, também devem ser encarados como tal. Como exemplo, podemos citar as normas, padrões, guias, templates e tudo aquilo que compõe a documentação dos processos.

Mas como é feita a identificação desses ICs? Lembra do catálogo que mencionamos anteriormente? Pois é justamente lá. Se você não puder implementar um catálogo sistematizado, terá que criar algum documento escrito com essa informação. Qual informação? Nome padronizado dos artefatos, nível de controle, local de armazenamento, papéis responsáveis pelo artefato e assim por diante.

Baselines

Os resultados esperados GCO 1 e 2 falam de ICs em quaisquer níveis de controle, mas do GCO 3 em diante vamos tratar de ICs em um nível de controle muito especial: colocados sob baseline (para mais informações, consulte o **quadro** "O que é baseline?").

O que deve ser colocado sob baseline? O MPS.BR não responde. Entretanto, como se trata do nível de controle mais rigoroso, é de se imaginar que sejam colocados sob baseline apenas ICs muito, mas muito críticos. Pense nos seus artefatos mais importantes e no principal produto do seu trabalho: o código fonte. Esses são fortes candidatos a comporem a sua baseline. Especificação de requisitos, diagramas de projeto detalhado e acordos de nível de serviço são outros fortes candidatos.

As baselines estão diretamente relacionadas com o seu sistema de controle de versões. Baselines de projetos geralmente são implementadas na forma de *branches*. Enquanto que baselines de sistemas geralmente são implementadas via mecanismos de *tag* (rotulação) comumente oferecidos por ferramentas de controle de versão, como o CVS e o VSS.

O GCO 4 é uma conseqüência do GCO 3, pois trata da descrição de cada versão que é disponibilizada. Para tal, as ferramentas de controle de versão oferecem campos para descrever textualmente um dado IC ou um pacote de ICs quando da sua disponibilização (check-out, commit, etc.).

Integre as funções do sistema

Até aqui percebemos que o controle de versões é o cerne da GCO. Mas não é tudo. O GCO 5 trata justamente da integração do controle de versões com o controle de modificações. Apesar de não ser a principal função do sistema de GCO, talvez seja a mais difícil de implementar. Muitas empresas já realizam muito bem o controle de versões, mas poucas conseguem integrar os seus sistemas de controle de modificações com o de versões. Isso significa dizer que algumas empresas não sabem dizer, de pronto, quais versões surgiram de um pedido de modificação, ou vice-versa. Estamos aqui falando de rastreabilidade bidirecional – algo que vai lhe ser muito útil no processo GRE.

Só que o GCO 5 vai além de simplesmente documentar as modificações e prover a rastreabilidade. Esse resultado esperado tem a ver também com o controle das modificações. Quando falamos em controle, estamos querendo dizer que existe um processo formal de aprovação de modificações que envolve desde a documentação da modificação, passando pela análise de impacto, avaliação das modificações e verificação para saber se o que foi implementado está de acordo com o pedido. Nesse processo, uma das coisas mais importantes é a definição dos papéis que serão responsáveis por cada passo. A ferramenta vai apenas facilitar o trabalho das pessoas que assumirem esses papéis.

Tentaremos descrever em síntese como poderia ocorrer esse processo de controle de modificações. Começa com alguém documentando uma modificação qualquer e sua necessidade. Em seguida, alguém vai verificar quais funcionalidades são afetadas por essa modificação e avaliar o impacto. A avaliação de impacto vai servir de subsídio para alguém aprovar ou não a modificação. Uma vez aprovada, considerando que seja uma pequena alteração, o analista vai fazer check-out dos ICs afetados pela alteração. Nesse momento, o check-out será vinculado à modificação. Após as alterações, o analista vai fazer um check-in dos ICs. Depois disso, as alterações serão revisadas por alguém e, uma vez testadas e aprovadas, a liberação é autorizada. Feita a liberação, as novas versões dos ICs passam a integrar a baseline do projeto ou sistema.

Bom, estamos quase concluindo a análise dos resultados esperados. Ainda faltam os GCOs 6 e 7.

Não vamos esmiuçar aqui o GCO 6, pois ele é uma conseqüência do GCO 5. O GCO 6 trata da segurança da execução do GCO 5, o que é inerente ao processo de controle de modificações – ainda mais quando estamos tratando do nível de controle mais formal que é colocar sob baseline.

Auditorias de configuração

Mas o que dizer do GCO 7, que fala de auditoria da configuração? Este é um assunto ainda polêmico. Alguns avaliadores consideram essa auditoria como parte do processo Garantia de Qualidade. O próprio Guia de Implementação dá margem para isso, quando menciona, no último parágrafo de GCO 7, que as verificações podem ser feitas em parceria com a equipe de qualidade. Se assim for, basta implementarmos o processo Garantia da Qualidade. Já outros avaliadores encaram a auditoria como algo adicional à garantia da qualidade. Seja como for, a sua empresa vai ter que providenciar pessoas para executar as verificações exigidas por esse resultado.

É exigido pelo MPS.BR que essas auditorias sejam realizadas objetivamente. O que é isso? O guia podia ter sido mais claro, mas, nesse caso, objetivamente é sinônimo de independentemente. Ou seja, as pessoas que participaram do desenvolvimento do projeto não podem ser as mesmas que farão as auditorias. Objetivamente também significa que as auditorias devem ser o menos subjetivas possível. Isso envolve utilizar métodos objetivos, tais como questionários fechados e check-lists. O ideal é que as auditorias sejam feitas de forma automatizada. Contudo, isso é muito difícil de implementar para as auditorias chamadas funcionais, que têm o objetivo de verificar a qualidade do conteúdo dos ICs.

Na prática você terá que fazer o seguinte. No plano de GCO, defina as atividades de auditoria, os papéis responsáveis pela sua execução e o momento em que as auditorias serão realizadas. O foco das auditorias é, sem dúvida, as baselines de projeto e as de sistema. Daí, podemos inferir que o melhor momento para a execução das auditorias é logo após a atualização dessas baselines e antes das liberações. Se um projeto for desenvolvido em cascata, isso significa dizer que ele terá apenas uma baseline e uma liberação, ou seja, apenas uma auditoria de baseline. Já projetos desenvolvidos de forma iterativa ou incremental, podem gerar várias baselines e, portanto, talvez tenham várias auditorias. A documentação do processo GPR pode definir que essas atividades de auditoria sejam incluídas no cronograma dos projetos.

O MPS.BR também exige que as não conformidades encontradas nas auditorias sejam acompanhadas até a sua

conclusão. É por isso que é tão importante ter uma equipe de auditoria independente dos projetos. Do contrário, as atividades de auditoria vão onerar ainda mais os projetos por roubar-lhes mão de obra. As auditorias podem ser onerosas, mas o que se quer é garantir a qualidade das baselines que, como o próprio termo já indica, são a base do desenvolvimento.

Podem ser feitas auditorias em ICs que não fazem parte de baseline? Claro! Isso é até desejável. Entretanto, isso pode se tornar muito oneroso e a auditoria de baselines já basta para o grau de maturidade exigido no Nível F. Ao passo que sua empresa for sistematizando as atividades de auditoria, pode-se aumentar o seu escopo de verificação.

Próximos passos

Como o processo GCO não está sozinho no Nível F, os próximos passos, sem dúvida, estarão relacionados com os processos de Garantia da Qualidade (GQA) e Medição (MED).

O cruzamento entre GCO e MED é implementado na forma do RAP 4. Já o cruzamento entre GCO e GQA está na forma dos RAPs 10 e 14.

Infelizmente, não se pode considerar aqui esses cruzamentos, pois seria necessário destrinchar vários conceitos de qualidade e medição. Mas sem dúvida eles devem ser seus próximos desafios. Na prática, você vai se reunir com as pessoas responsáveis pela implementação desses processos e acordar as atividades de medição e qualidade que devem permear o processo GCO, e vice-versa.

Falando só um pouco sobre medição, você vai precisar de indicadores para esse processo. Uma dica é utilizar os resultados das auditorias de GCO para calcular indicadores exigidos pelo processo Medição.

Conclusão

Vimos aqui alguns aspectos práticos para começar uma implementação do processo GCO a partir do zero. É importante que todos os demais processos já comecem se preocupando com GCO, mesmo no Nível G, para se evitar dificuldades futuras. Vimos também a importância de construir um bom alicerce para o processo por meio dos RAPs. Com esses pontos em mente, fica muito mais fácil implementar esse processo tão importante que se cruza com todos os demais processos. Em artigos futuros, vamos ver quais são as evidências típicas dos resultados esperados e RAPs de GCO. ●

Links

Softex > MPS.BR > Guias
http://www.softex.br/mpsbr/_guias/default.asp

IBM Rational ClearCase
http://www-01.ibm.com/software/awdtools/clearcase/

IBM Rational ClearQuest
http://www-01.ibm.com/software/awdtools/clearquest/

Synergy ALM
http://www.eclipseplugincentral.com/Web_Links-index-req-viewlink-cid-1173.html

Aldon Application Lifetime Management Suite
http://www.aldon.com/prod/alm/ov/

TD/OMS
http://www.remainsoftware.com/

Referências

PRESSMAN, R. S. Software Engineering: a practitioner´s approach. Mc Graw Hill Higher Educational, 6ª. Edição. 2005.

MAHLER, A. Variants: Keeping things together and telling them apart. In Configuration Management, Vol. 2 of Trends in Software, Wiley, New York, 1994.

BERSOFF, E. H.; Henderson, V. D. e Siegel, S.G. Software Configuration Management: A tutorial. Los Alamos, Califórnia. IEEE Computer. v.12, n.1, 1979.

IEEE for Software Configuration Management Plans. 1998.

TUSCANY. P. A. Software development environment for large switching projects. In Proceedings of Software Engineering for Telecommunications Switching Systems Conference, 1987.

PACHECO, R. F. Uma Forma de Implantação de Gerenciamento de Configuração de Software em Empresas de Pequeno Porte. Dissertação (Mestrado) – Instituto de Ciências Matemáticas de Computação, Universidade de São Paulo, São Carlos, 1997.

SOFTEX, 2006b, MPS.BR - Melhoria de Processo do Software Brasileiro - Guia de Implementação (Versão 1.1), Associação para Promoção da Excelência do Software Brasileiro.

ISO, 1995b, ISO/IEC 12207 - Information technology - Software life cycle processes, International Organization for Standardization.

Profissional da Área de Testes

Conhecimentos, habilidades e atitudes requeridas para a área de Testes de Software

Daniel Scaldaferri Lages

dlages@gmail.com

Possui MBA em Gerência de Projetos pela Fundação Getúlio Vargas, é pós-graduado em Gerência de T.I. pela Universidade FUMEC e Bacharel em Ciência da Computação pela UFMG. Atualmente é Coordenador da equipe de Quality Assurance da CPM Braxis, filial BH. Sua experiência profissional inclui o cargo de Analista de Testes no Synergia (núcleo de engenharia de software do Departamento de Ciência da Computação da UFMG), Gerente da fábrica de software na Unitech (hoje CPM Braxis) e docência no curso de graduação de Sistemas de Informação na PUC-MG. Possui a certificação ITIL para gerenciamento de serviços de T.I. É certificado em testes de software pela ISTQB e em qualidade de software pela IBM.

Quem participa de listas de discussão e fóruns na Internet sobre Testes de Software (ler **Nota 1**) sabe que é frequente o ingresso de novos integrantes solicitando orientações e conselhos para iniciarem a carreira na área. Normalmente, perguntam o que devem saber e fazer para poderem se especializar em testes e, consequentemente, encontrar um emprego com maior facilidade. Este artigo poderá ajudá-los no sentido de orientar sobre quais são as competências desejáveis para um profissional de testes. Dessa forma, o iniciante poderá utilizá-lo como guia para montar um plano de estudos e assim aumentar seu conhecimento. Poderá, também, aperfeiçoar-se nos aspectos comportamentais necessários, preparando-se para gerar bons resultados na sua empresa, ou mesmo para uma entrevista de emprego na área. Da mesma forma, uma empresa poderá se basear nas competências indicadas para realizar um processo de seleção para uma vaga específica de testes, ou montar uma equipe de testadores sob medida.

De que se trata o artigo?

Este artigo apresenta um conjunto de conhecimentos, habilidades e atitudes que o profissional de testes (ou o que quer iniciar sua carreira na área) deve possuir para que seja capaz de destacar-se na realização do seu trabalho. É apresentada também a estrutura de equipe mais comum encontrada na área de testes.

Para que serve?

O artigo visa orientar profissionais que desejam ingressar na área de Testes de Software, assim como orientar os profissionais que já atuam a complementarem suas competências. Serve também para empresas que desejam montar uma equipe de testes na medida certa para atender suas necessidades.

Em que situação o tema é útil?

Empresas e profissionais que possuem interesse em conhecer quais são as competências que um profissional e/ou equipe de testes de software devem possuir para obterem sucesso na função.

As competências necessárias para o desempenho das atividades da área de Testes de Software são formadas por características **comportamentais** e **técnicas**. A área de testes exige um equilíbrio maior entre essas duas características,

diferentemente das outras áreas da Engenharia de Software, como por exemplo, a área de Requisitos. Nesta área, o comportamento tende a ter maior importância frente ao conhecimento técnico, pois o profissional passará mais tempo em contato com o cliente, realizando o levantamento dos requisitos. Nos testes, apenas conhecimentos técnicos não são suficientes, pois a área envolve muitos aspectos de psicologia, como serão apresentados adiante. Da mesma forma, apenas um comportamento adequado não credencia uma pessoa a ser um bom profissional de testes. Este deve possuir grandes conhecimentos técnicos para atuar com precisão.

O termo **competência** no contexto desse artigo seguirá o conceito de [GRAMIGNA], que é definido como o "repertório de comportamentos e capacitações que algumas pessoas ou organizações dominam melhor que outras, fazendo-as eficazes em uma determinada situação". No próximo item, o repertório para os profissionais de testes será apresentado.

O que saber e como agir?

Como dito na introdução, para que um profissional de testes possa começar a atuar, deverá possuir algumas competências técnicas e comportamentais básicas. Naturalmente, ao longo do tempo e com boa orientação, o profissional irá desenvolver essas competências. Não é aconselhável ter muita pressa e sair estudando tudo que aparece pela frente. A prática é fundamental. A experiência virá com o tempo.

Mas para que possa se destacar é preciso ter conhecimentos técnicos amplos, e alguns deles mais profundos. Assim como possuir atitudes exemplares. Segundo [LOVELAND], o profissional de testes é uma "raça especial" por uma série de razões. Alguns traços de personalidade são necessários para que se tenha sucesso, felicidade e longa carreira na função. O "Qualquer um pode testar" é um mito. O bom testador deve atuar com precisão e segurança, falhando o mínimo possível. Deve preocupar-se ao máximo com sua **credibilidade**, que é um fator determinante para o seu sucesso. Um testador que falha sucessivamente perde a confiança dos colegas de trabalho, que passam a desconfiar da sua competência, por exemplo, duvidando das falhas registradas ou ignorando sugestões, alertas e questionamentos.

Para alcançar essa credibilidade, o testador deverá ser persistente na busca pelas competências necessárias (mas lembrando, sem correria). Para desmembrar as competências, um conceito bastante conhecido entre os profissionais da área de recursos humanos é bem útil, o CHA. Este é o acrônimo das palavras **conhecimentos, habilidades e atitudes**.

Conforme [GRAMIGNA], as competências do ser humano podem ser comparadas a uma árvore:

- A raiz corresponde às *atitudes*: A raiz corresponde ao conjunto de valores, crenças e princípios, formados ao longo da vida, e que determinam nossas atitudes. Ela é o início de tudo e o principal componente da competência. Está relacionada com o *querer ser* e o *querer agir*;
- O tronco corresponde ao *conhecimento*: É o segundo componente de uma competência. Trata-se do conjunto de informações que a pessoa armazena e lança mão quando precisa. Quanto maior este conhecimento, mais a competência se fortalece. E permite que o profissional enfrente com flexibilidade e sabedoria os diversos desafios de seu dia-a-dia;

Nota do DevMan 1

Teste de Software

Teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado. O seu objetivo é revelar falhas em um produto, para que as causas dessas falhas sejam identificadas e possam ser corrigidas pela equipe de desenvolvimento antes da entrega final. Por conta dessa característica das atividades de teste, dizemos que sua natureza é "destrutiva", e não "construtiva", pois visa ao aumento da confiança de um produto através da exposição de seus problemas, porém antes de sua entrega ao usuário final.

O conceito de teste de software pode ser compreendido através de uma visão intuitiva ou mesmo de uma maneira formal. Existem atualmente várias definições para esse conceito. De uma forma simples, testar um software significa verificar através de uma execução controlada se o seu comportamento corre de acordo com o especificado. O objetivo principal desta tarefa é revelar o número máximo de falhas dispondo do mínimo de esforço, ou seja, mostrar aos que desenvolvem se os resultados estão ou não de acordo com os padrões estabelecidos.

Já o planejamento dos testes deve ocorrer em diferentes níveis e em paralelo ao desenvolvimento do software. Os principais níveis de teste de software são:

- Teste de Unidade: também conhecido como testes unitários. Tem por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente. O universo alvo desse tipo de teste são os métodos dos objetos ou mesmo pequenos trechos de código.
- Teste de Integração: visa provocar falhas associadas às interfaces entre os módulos quando esses são integrados para construir a estrutura do software que foi estabelecida na fase de projeto.
- Teste de Sistema: avalia o software em busca de falhas por meio da utilização do mesmo, como se fosse um usuário final. Dessa maneira, os testes são executados nos mesmos ambientes, com as mesmas condições e com os mesmos dados de entrada que um usuário utilizaria no seu dia-a-dia de manipulação do software. Verifica se o produto satisfaz seus requisitos.
- Teste de Aceitação: são realizados geralmente por um restrito grupo de usuários finais do sistema. Esses simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado.
- Teste de Regressão: Teste de regressão não corresponde a um nível de teste, mas é uma estratégia importante para redução de "efeitos colaterais". Consiste em se aplicar, a cada nova versão do software ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste anteriores do sistema. Pode ser aplicado em qualquer nível de teste.

• A copa (com frutos, flores e folhas) corresponde às *habilidades*: Agir com talento, capacidade técnica, obtendo resultados positivos é o que chamamos de habilidade. Uma competência só é reconhecida quando disponibilizada. Não adianta alguém saber que sabe. Para obter o reconhecimento, precisa demonstrar.

As competências de um bom profissional devem ser formadas por uma combinação de conhecimentos, habilidades e atitudes. Apenas o conhecimento, sem habilidades e atitudes, nos passa a sensação de um profissional por demais teórico, sem capacidade de externalizar e colocar em prática todo o seu conhecimento. Só habilidades, sem conhecimentos e boas atitudes nos traz a imagem de um bom operador, com conhecimentos restritos à sua função, com pouca capacidade de desenvolvimento e de tomada de atitudes. Este tipo de profissional faz bem o que o mandam. Pode ser facilmente substituído, mas em muitos casos, são extremamente úteis. Da mesma maneira, as atitudes sozinhas, não são suficientes para qualificar um grande profissional, pois este poderá apresentar boa índole, boa postura profissional, mas pouco conseguirá produzir, por falta de conhecimento.

Conhecimentos

Conhecimentos em Engenharia de Software são essenciais para um testador. [DIMAGGIO] divide esse conhecimento em três áreas: *processo de desenvolvimento de software*, *programação de computadores* e *sistemas operacionais (SO)*.

O testador deve conhecer o **processo de desenvolvimento** no qual está inserido, pois suas atividades fazem parte desse processo. Para desempenhar bem a função, deve dominar o processo. Ele deve conhecer quais são os artefatos de entrada e saída de cada fase do processo, principalmente das que envolvem atividades de testes. Dessa forma, poderá antever problemas e sugerir soluções antecipadamente, além de planejar as atividades com maior segurança, uma vez que terá uma visão ampla de como funcionam as interfaces das fases do processo. Portanto, deve procurar entender todo o processo de desenvolvimento da sua empresa. Caso não esteja trabalhando, pode estudar processos modelos, como por exemplo, RUP, PRAXIS, XP, SCRUM, V-MODEL, TMAP, pois com certeza, se os processos das empresas não são fiéis a algum desses, são bem próximos. Conhecer a fundo o processo da sua empresa faz grande diferença entre os profissionais da organização.

O bom testador também se destacará caso saiba **programação**. Isso não quer dizer que tenha que dominar linguagens, arquiteturas e programar algoritmos elaborados. Mas deve conhecer lógica de programação, características de linguagens e *frameworks*. Esse conhecimento fará com que ele saiba o ponto fraco dessas tecnologias, uma vez que já as utilizou e sabe onde errou mais. Um testador que já programou em C sabe que muitos erros são cometidos na manipulação de memória, por exemplo, causados pelo esquecimento em desalocá-las, o que poderá gerar um estouro de memória. Já em Java, essa manipulação é automática. Em contrapartida, falhas de acesso concorrente (acessos a objetos que já foram eliminados) em Java são facilmente reproduzidas, caso essa situação não seja tratada.

Entender dos **sistemas operacionais** é outro aspecto importantíssimo. Conhecer as políticas de permissões de acessos, configurações de variáveis de ambientes é importante para saber diferenciar o que são falhas na aplicação ou apenas configurações desajustadas. Esse conhecimento traz produtividade para os testes, evitando o registro de falhas e investigações sem necessidade. Comandos básicos e intermediários, como por exemplo, os comandos *diff*, *wc* e *tail –f* do UNIX são ferramentas "poderosas" que facilitam os testes dos testadores. Saber UNIX (e seus "familiares"), com seus processos e *daemons*, fará com que o testador seja mais completo, se diferenciando daquele que testa apenas sistemas com interfaces amigáveis do *Windows*. Esse conhecimento em SO deve abranger a criação de scripts, sejam eles *shellscripts*, ou os usuais *.bat*. Isto torna o testador apto a automatizar tarefas simples, que geram produtividade caso sejam automatizadas. Por exemplo, um *script* que obtém o código-fonte de uma aplicação em um repositório, compila esse código, realiza o *deploy*, chama a ferramenta de execução de testes automáticos, e depois envia o resultado por e-mail, será bastante útil para executar os testes de regressão executados no período da noite.

Além dos conhecimentos destacados por [DIMAGGIO], saber realizar consultas em **banco de dados** é essencial. Muitas vezes não é possível, através da interface de usuário da aplicação, verificar todos os resultados esperados. Neste caso, realizar consultas SQL é a única saída. Não é requerido que o profissional saiba consultas complexas, mas logicamente, quanto mais souber, mais irá se destacar.

Figura 1. Mapa Mental dos Conhecimentos

Além dos conhecimentos principais ilustrados na **Figura 1**, é interessante que o testador conheça as características dos ***browsers*** mais utilizados no mercado e suas configurações, como por exemplo, configurações de *cache*, de segurança e privacidade. Muitas vezes os problemas estão nessas configurações, e não na aplicação *Web*. Principalmente para testes de performance (*stress*, volume, carga etc.), é importante o conhecimento em protocolos de rede, como o *http*. Para esse tipo de testes também é importante que o profissional tenha noção do funcionamento dos itens que comumente são monitorados, como é o caso da memória e CPU. Mas o conhecimento não precisa ser profundo, uma vez que esse tipo de teste envolve (ou deveria envolver) todas as áreas de apoio possíveis, como suporte de rede, SO, DBA, arquitetos, e não apenas testadores.

Ferramentas de testes são fundamentais, mas o profissional de testes não precisa dominá-las para iniciar sua carreira. Com os conhecimentos apresentados aqui, torna-se fácil o aprendizado. Por exemplo, quem sabe programar não terá dificuldades em aprender a automatizar testes. Quem conhece o funcionamento básico da Internet (*browser*, requisições *http*), não terá dificuldades em aprender a utilizar uma ferramenta para testes de *stress*. Mas, logicamente, o profissional que já conhece essas ferramentas poderá ser mais bem avaliado em uma entrevista de emprego.

Interessante citar que conhecer a **língua inglesa** também é uma vantagem, pois os melhores livros e grandes artigos estão escritos nessa língua, além das melhores ferramentas. Dessa forma, o profissional de testes terá maior facilidade para evoluir no que tange aos seus conhecimentos.

Habilidades

Em seu artigo, [DIMAGGIO] enfatiza duas características que serão aqui categorizadas como habilidades: *comunicação* e *organização*. Além dessas, a *capacidade analítica* e a *facilidade de aprendizado* são outras habilidades importantes.

A **comunicação** pode ser dividida em comunicação escrita e comunicação oral. Ambas devem ser uma comunicação clara e objetiva. O analista de testes deve escrever bem, pois uma de suas atividades principais é elaborar os casos de testes. Ele deverá escrever, no mínimo, a descrição do caso de testes, o passo a passo para execução, os pré-requisitos e os resultados esperados. Durante a execução dos testes, a todo o momento deverá registrar as falhas, transcrevendo para a ferramenta as ações para que a falha seja reproduzida. Caso não estejam claros, tanto a execução dos testes, quanto a reprodução das falhas identificadas, poderá ser comprometida, uma vez que na maioria das vezes, outras pessoas que irão consumir os textos escritos. Além desses casos citados, a comunicação por e-mail também é muito utilizada. O analista de testes que não sabe escrever, não passa credibilidade. Portanto, não é um bom testador. Planos de testes, relatórios de execuções são outros exemplos de documentos que também são gerados pela equipe de testes, e que devem ser bem escritos.

Quanto à comunicação oral, ela também deve ser clara. Muitas situações só são resolvidas através de diálogos, e muitas dúvidas através de perguntas bem elaboradas. Saber o que perguntar transmite credibilidade. A interação entre os desenvolvedores e os testadores deve ser constante para evitar mal-entendidos. Apesar de um e-mail ser bem escrito, ainda sim é uma comunicação impessoal. A comunicação com o cliente em uma homologação segue as mesmas idéias. Além dessas situações básicas, é normal a necessidade de treinamento por parte da equipe de testes, seja de ferramentas ou de processos. O profissional que sabe falar em público e transmitir seu conhecimento de forma clara possui um diferencial.

Possuir **organização** é fundamental para executar atividades de testes. [DIMAGGIO] cita um cenário comum: é quase sempre impossível executar os testes planejados em sistema onde o testador não seja obrigado a interrompê-los devido à identificação de um *bug* que bloqueie o restante da execução e o obrigue a continuar os testes em outra parte do sistema até o *bug* seja removido. O testador deverá ser organizado o suficiente para interromper um teste em uma parte do sistema, começar os testes em outra parte, e depois voltar para a parte anterior de onde ele parou, sem prejuízos para a qualidade do sistema. A organização é uma habilidade essencial para todas as atividades.

A **capacidade analítica** é uma habilidade que se refere à capacidade de analisar dados, muitas vezes não relacionados entre si, e após essa análise, gerar informações de valor. O testador deverá analisar o sistema que está sendo testado entendendo não apenas suas partes isoladamente, mas também como se relacionam. Não deve apenas seguir um roteiro de testes, mas sim entender o sistema e seu contexto. É no momento da elaboração dos casos de testes que essa habilidade é mais requerida. O testador deve ser capaz de imaginar testes válidos e inválidos que teste todo o sistema.

Existem diversas ferramentas, gratuitas e pagas, para apoiar e/ou viabilizar os testes de software. Um bom testador deve ter **facilidade de aprendizado** para poder avaliá-las, utilizá-las e disseminá-las dentro da sua empresa. Portanto, deve ser capaz de realizar comparações entre ferramentas que possuem a mesma finalidade de forma a encontrar a melhor delas. Essa facilidade de aprender e trabalhar com ferramentas é muito bem-vinda. A **Figura 2** apresenta o mapa mental para as habilidades.

Figura 2. Mapa Mental das Habilidades

Atitudes

No seu tradicional livro, *The Art of Software Testing*, [MYERS] dedica grande parte de um capítulo para discutir questões da psicologia que envolve as atividades de testes. Ele diz que o teste de software é uma tarefa técnica, mas que envolve grandes questões da psicologia humana. Portanto, algumas atitudes devem ser direcionadas a essas questões para que o relacionamento entre desenvolvedores, gerentes e testadores seja saudável. Essas atitudes serão úteis, por exemplo, caso uma notícia não tão boa tenha que ser comunicada pela equipe de testes ao gerente/desenvolvedor. Dependendo da forma como essa informação é passada poderá ou não "sobrar" para o testador: "Se não gosta da mensagem, bate no mensageiro".

O relacionamento entre o desenvolvedor e o testador é um barril de pólvora prestes a explodir caso os desenvolvedores não entendam o papel dos testadores, e caso os testadores não entendam a posição de um desenvolvedor. Essa última justificada por [SPILLNER], que diz que todos cometem erros,

mas ninguém gosta de admiti-los. Quando um produto gerado pelo desenvolvedor é questionado por outros, ou seja, colocado à prova, a tendência é que ele fique na "defensiva", cheio de justificativas, pois não quer que suas falhas venham à tona. A natureza humana é assim.

Para justificar essa "guerra fria" basta pensar: o "sucesso" de um é o "fracasso" do outro. Ou seja, quanto mais falhas forem identificadas, melhor será visto o trabalho do testador, mas não o do desenvolvedor. O contrário também é verdadeiro. Quanto menos falhas identificadas, melhor será visto o trabalho do desenvolvedor, mas não o do testador, principalmente se essas falhas forem percebidas pelo cliente em homologação ou em produção. Essa competição é saudável, mas de maneira nenhuma poderá ser levada para o lado pessoal. Ambos devem lembrar que possuem o mesmo objetivo: a qualidade do produto. Quando essa maturidade é atingida, o projeto e a empresa ganham.

Uma atitude que vale para todos em todas as profissões, e dessa forma para os testadores, é a **pró-atividade**. Esta tem um peso muito grande para os testadores, pois o teste costuma ser uma das últimas, senão a última atividade a ser realizada antes do produto entrar em produção. Além disso, geralmente, já começa com atraso e prazo reduzido devido ao acúmulo de atrasos das atividades anteriores. Por conta desses fatores, o profissional de testes não pode ficar parado quando encontra obstáculos, pois não existe tempo hábil para recuperação.

Voltando à questão da credibilidade, uma atitude do testador que proporciona o aumento da confiança do desenvolvedor nos testes de software é a **autoconfiança**. Se questionado, o testador deverá mostrar segurança em suas ações e decisões. Mas a autoconfiança não pode ser confundida com confiança exagerada, pois essa não possui garantia. O testador deve estar balizado para poder apresentar seus questionamentos e justificativas. O "eu acho", o "não sei direito" devem ser evitados. Normalmente, a falta dessa atitude vem acompanhada do acanhamento, o que pode passar aos desenvolvedores, e principalmente aos clientes, fraquezas e incertezas.

A **paciência**, acompanhada da **persistência**, deve fazer parte do repertório de atitudes dos testadores. Muitas vezes eles são obrigados a realizarem tarefas repetitivas e "manuais", e também testar a mesma funcionalidade várias vezes. A paciência também é um componente da "**diplomacia"**. Conforme dito anteriormente, o relacionamento entre desenvolvedor e testador, muitas vezes, é tenso. O bom desenvolvedor possui um ego muito forte. O testador deve ser diplomático, ou seja, utilizar bons modos e delicadeza para conduzir uma questão, pois, querendo ou não, na maioria das vezes, é portador de más notícias. Conforme [LOVELAND], terá que dizer ao desenvolvedor *"The baby is ugly!"*.

[DIMAGGIO] diz que o profissional de testes deve possuir alto grau de **ceticismo**, mas sem hostilidades. Nunca deve acreditar nas coisas na primeira vez que as vê. Deve questionar até que tudo seja provado. "Mostre-me!". Deve usar a sua teimosia para **investigar** o correto funcionamento do sistema. Essa atitude investigativa também é ressaltada por [DIMAGGIO]. Muitas vezes vários artefatos devem ser consultados para que se chegue a uma conclusão.

A próxima atitude é a **criatividade destrutiva**. O profissional de testes não pode ter medo de "derrubar" o sistema durante os testes. Nos testes de software os limites existem para serem ultrapassados. É normalmente ali onde os programadores cometem mais erros. "O que irá acontecer caso eu faça isso?". Esse tipo de pergunta deve estar na cabeça do testador. Apesar de existirem casos de testes a serem seguidos, essa atitude destrutiva é um *plus* no momento dos testes, podendo identificar várias falhas que, se corrigidas, podem tornar o sistemas bem mais robusto.

As duas últimas atitudes que [DIMAGGIO] cita em seu artigo, e que são bastante interessantes são: **ânsia por novas tecnologias** e **valorização da perspectiva do usuário**. A primeira é mais global, aplicando não só aos profissionais de testes e se refere à vontade de aprender e trabalhar com as novas tecnologias que surgem constantemente na área. Não se pode parar no tempo. O profissional deve se manter atualizado, estudando novas tecnologias, arquiteturas, padrões de mercado, processos etc. Sempre poderão aplicar algum aprendizado. Já a segunda é mais específica para os testadores. Eles devem se colocar no lugar do usuário quando estão executando os testes, tornado-se um "advogado" do usuário. Nesta perspectiva, e na falta de uma equipe de usabilidade, sugestões de melhorias levantadas pelos testadores podem render novos serviços e um produto de melhor qualidade.

[LOVELAND] ainda lembra da **curiosidade**. Perguntas como "Porque?", "E se...?" devem ser comuns. Os testadores não devem ser tímidos em perguntar. A **Figura 3** apresenta o mapa mental para as atitudes.

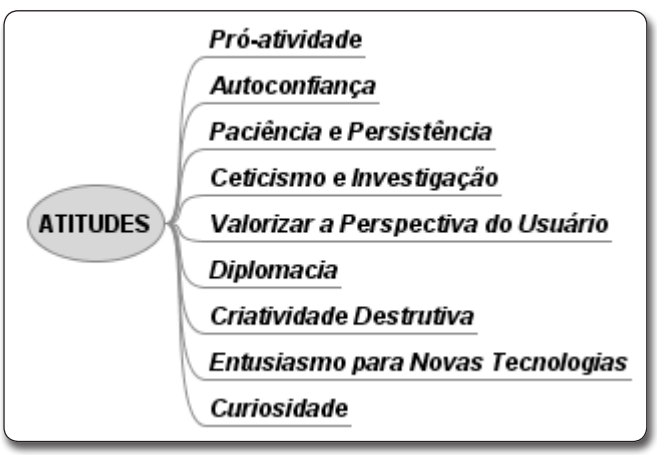


Figura 3. Mapa Mental das Atitudes

O que fazer?

Após a avalanche de conhecimentos, habilidades e atitudes descritas, o profissional deve estar se perguntando como e por onde começar a estudar e colocar em prática isso tudo. Não existe um roteiro para isso. Para atacar os conhecimentos citados aqui, o profissional poderá estudar sozinho, ou mesmo se matricular em cursos específicos. Atualmente existem vários

sobre Testes de Software, que poderão ser úteis para complementar o currículo das faculdades. Já as disciplinas de Banco de Dados, Programação e Sistemas Operacionais, que fazem parte da grade do curso de graduação, essas os profissionais devem levar muito a sério, aproveitando o momento na universidade. Para quem gosta de estudar sozinho, a Internet proporciona vários sites sobre testes com fóruns e listas de discussão onde é possível encontrar materiais bastante úteis. Os profissionais devem estar sempre lendo artigos, reportagens e *cases* de sucessos relacionados à área de Testes de Software.

As certificações são muito bem vindas. A preparação para elas envolve muito estudo. Além disso, muitas empresas já valorizam e até exigem que o profissional seja certificado para que se torne apto a preencher uma vaga. Algumas licitações já exigem das empresas participantes que possuam profissionais certificados.

Quanto às habilidades e atitudes, essas não são adquiridas apenas estudando. Devem ser praticadas. Redija textos e peça para algum amigo revisar. Apresente oralmente trabalhos na faculdade, ou na empresa onde trabalha. Seja candidato para avaliar ferramentas de testes. Não evite essas situações, ao contrário, vá de encontro a elas. Pense e haja de acordo com as atitudes citadas e comece a perceber os resultados. Repare nas outras pessoas que não possuem as mesmas atitudes, e perceba as diferenças.

Estrutura de uma Equipe de Testes

Uma equipe onde todos os membros possuem as mesmas características, ou seja, o mesmo conjunto de CHA, não é uma equipe desejável. A não ser, é claro, que todos eles possuam **todos** os conhecimentos, habilidades e atitudes necessárias. Esse cenário seria o ideal, mas na prática, isso é muito difícil de acontecer. Segundo [CHIAVENATO], a variabilidade humana é enorme. As pessoas apresentam profundas diferenças individuais. Cada pessoa tem a sua própria personalidade, a sua história pessoal, seus conhecimentos e habilidades, seus objetivos e motivações, suas limitações pessoais etc. Elas reagem de forma diferente a uma mesma situação. Além desses aspectos, a experiência dos profissionais também é diferente, tornando-os melhores que os outros em uma determinada competência.

Por essa razão, deve-se buscar montar uma equipe de testes equilibrada, formada por membros que se complementam. Devem possuir pessoas experientes (consequentemente, com salários mais altos), e outras nem tanto. Até mesmo para tornar o custo da equipe viável. As competências são identificadas e agrupadas para satisfazerem os **papéis** dentro de uma equipe de testes. Os livros da área costumam propor estruturas de equipes. As mais comuns, conforme citado em [SPILLNER], possuem *testadores*, *analistas de testes*, *arquitetos de testes*, *automatizadores de testes*, *líder e/ou coordenador de testes*.

Os *testadores* são os responsáveis pela execução dos testes. Seguem um roteiro de casos de testes e registram as falhas quando encontradas. São responsáveis também por verificar as correções realizadas pelos desenvolvedores. Podem também revisar os casos de testes. Normalmente são os profissionais que estão começando na área.

Já os *analistas de testes* são mais experientes que os testadores e já realizaram esta função. São os responsáveis por criar os casos de testes, baseando-se nos requisitos do usuário ou na especificação funcional do sistema. Devem identificar os pontos a serem testados e o resultado esperado; extrair os pré-requisitos e os dados de entrada; elaborar o passo-a-passo, criando os casos de testes para que o testador possa executá-los. Os casos de testes devem ser elaborados de forma que o sistema seja coberto da melhor forma possível, levando em consideração o tempo e os recursos à disposição. O analista de testes também deverá indicar a ordem dos casos a serem executados.

Os *arquitetos de testes* são especialistas nas técnicas de testes. Eles sabem identificar quais são os tipos de testes que devem ser realizados em um determinado projeto, arquitetando todas as questões referentes aos testes, como por exemplo, preparação do ambiente, geração da massa de dados, seleção da melhores ferramentas e a identificação dos casos de testes a serem automatizados.

O *automatizador de testes* deve possuir bons conhecimentos em programação, pois a automação envolve intervenções manuais nos *scripts* gerados pelas ferramentas de automação. Esses profissionais devem, por exemplo, inserir no código pontos de verificações, criar rotinas para leitura de dados de entrada proveniente de arquivos ou banco de dados, e implementar mecanismos de *logs* para facilitar a análise da execução do testes. A criação de bibliotecas com funções relativas às intervenções citadas é uma boa prática, pois gera produtividade no momento da automação de testes, devido ao reuso. Esse profissional também seria o responsável pelos testes de performance, que também são testes automatizados. É importante ressaltar que nesse tipo de teste, seja ele de stress, volume, carga, tempo de resposta etc., é necessário o envolvimento de várias áreas específicas de dentro da organização, como por exemplo, analistas de rede, analistas de banco de dados, desenvolvedores, administradores do sistema operacional. O profissional de testes deve ter todo esse apoio para que o trabalho seja realizado de forma satisfatória.

O *líder de testes* (ou coordenador ou gerente) é a pessoa responsável pela equipe. Além de conhecimentos específicos da área de testes, deverá possuir liderança e gestão.

Normalmente, uma pessoa experiente assume vários papéis dentro da equipe de testes. Mas a seqüência comum de atuação e aprendizagem seria está: Testador – Analista de Testes e Automatizador de Testes – Arquiteto de Testes – Líder de testes. O acúmulo de experiência proporciona essa "escalada" nas funções. Por exemplo, um bom arquiteto de testes deve possuir boa experiência como testador e analista de testes. Não faz sentido nomeá-lo arquiteto se nunca executou e nem elaborou casos de testes, ou se não possui experiência suficiente nessas atividades.

Na conclusão desse artigo, com base na experiência do autor, são apresentadas tabelas que sugerem o grau de aderência aos conhecimentos, habilidades e atitudes exigidas para cada uma das funções apresentadas.

Conclusão

As Tabelas **1**, **2** e **3** não representam as competências acumuladas, mas sim o necessário para a realização de uma determinada função isoladamente. Ou seja, mesmo que um bom arquiteto de testes deva ter sido um bom testador, os quadros sugerem para o arquiteto apenas os CHA necessários para essa função, esquecendo-se do tempo em que ele foi testador.

Legenda:

	Baixo
	Médio
	Alto

Papéis	Testador	Analista de Testes	Automatizador	Arquiteto de Testes	Líder de Testes
Conhecimentos					
Processo de *Software*					
Sistemas Operacionais					
Programação					
Banco de Dados					
Browser					
Língua Inglesa					

Tabela 1. Componente Conhecimento

Papéis	Testador	Analista de Testes	Automatizador	Arquiteto de Testes	Líder de Testes
Habilidades					
Comunicação escrita					
Comunicação oral					
Organização					
Capacidade analítica					
Facilidade de aprendizado					

Tabela 2. Componente Habilidade

Papéis	Testador	Analista de Testes	Automatizador	Arquiteto de Testes	Líder de Testes
Atitudes					
Pró-atividade					
Autoconfiança					
Paciência e Persistência					
Ceticismo e Investigação					
Perspectiva do Usuário					
Diplomacia					
Criatividade Destrutiva					
Entusiasmo com Novas Tecnologias					
Curiosidade					

Tabela 3. Componente Atitude

Referências

CHIAVENATO, Idalberto. Administração de Recursos Humanos – Fundamentos Básicos. São Paulo: Atlas, 1999.

DIMAGGIO, Len. A Roadmap for Software Test Engineering. Dr. Dobb´s, 2001. http://www.ddj.com/architect/184414700 - Visitado em 01/08/2009.

GRAMIGNA, Maria Rita. Modelo de Competências e Gestão dos Talentos. São Paulo: Pearson Education do Brasil, 2002.

LOVELAND, Scott; MILLER, Geoffrey; PREWITT, Richard; SHANNON, Michael; Software Testing Techniques – Finding the Defects that Matter. Charles River Media, INC. Hingham, Massachusets, 2005.

MYERS, Glenford J. The Art of Software Testing. John Wiley & Sons, Inc., New York, NY, 1979.

SPILLNER, Andreas; LINZ, Tilo; SCHAEFER, Hans; Software Testing Foundations – A Study Guide for the Certified Tester Exam. Rio de Janeiro: Fundação Getúlio Vargas, 2006.

CRAIG, R.D., JASKIEL, S. P., "Systematic Software Testing", Artech House Publishers, Boston, 2002.

PRESSMAN, R. S., "Software Engineering: A Practitioner's Approach", McGraw-Hill, 6th ed, Nova York, NY, 2005.

ROCHA, A. R. C., MALDONADO, J. C., WEBER, K. C. et al., "Qualidade de software – Teoria e prática", Prentice Hall, São Paulo, 2001.

Dê seu feedback sobre esta edição!

A Engenharia de Software Magazine tem que ser feita ao seu gosto.
Para isso, precisamos saber o que você, leitor, acha da revista!
Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/esmag/feedback

Chegou o Sistema de Créditos DevMedia

Agora o **conteúdo completo** do nosso site está **ao seu alcance!**

A partir de agora todas as **2000 vídeo-aulas** do site DevMedia podem ser compradas individualmente.

Economia - Você não precisa mais assinar oito produtos diferentes para ter acesso ao conteúdo completo do site!

Todas as vídeos que você sempre quis ver a partir **R$ 0,75!**

Saiba mais sobre o Sistema de Créditos!

DevMedia GROUP

www.devmedia.com.br/creditos

Fatores humanos

A influência na Qualidade de Software

De que se trata o artigo?

Quando se fala em qualidade de software, geralmente tem-se em mente as seguintes variáveis: pessoas, processos e tecnologia. Neste artigo pretende-se focar na variável pessoas, ou seja, nos fatores humanos que geram diretamente influência na qualidade do produto.

Para que serve?

Conhecer como as pessoas podem impactar na qualidade dos produtos de software é um importante fator a ser considerado pelas empresas ao definir seus processos e políticas de desenvolvimento.

Em que situação o tema é útil?

Melhoria de processos para desenvolvimento de software considerando o fator humano.

Kênia Reis

kenia_reis@hotmail.com

Especialista em Gestão de Software, atua no ramo de desenvolvimento de software há 8 anos. Atualmente trabalha como Analista de Requisitos na PC Sistemas, onde desempenha também atividades de Gerência de Projetos.

Conceituar a palavra qualidade pode ser uma tarefa subjetiva, pois gera percepções diferentes dependendo do ponto de vista. De uma maneira geral a qualidade pode ser conceituada como adequação a algum padrão pré-estabelecido. A qualidade de software reúne então a adequação das características explícitas, ou seja, aquelas citadas pelos usuários clientes de software e as características implícitas, ou seja, aquelas que não são citadas e que às vezes são consideradas "obvias" trazendo ainda mais subjetividade no desenvolvimento e manutenção do produto e/ou serviço de software.

As atividades do desenvolvimento do software não são como uma receita de bolo, não há um processo congelado, mestre e único que pode ser seguido por diversas organizações. Apesar de não existir esse processo congelado, há um conjunto de boas práticas vivenciadas por profissionais da área que utilizaram e constataram a efetividade e que podem ser melhoradas e/ou adaptadas continuamente.

Os problemas vividos há tempos atrás como projetos com prazos e custos extrapolados, projetos abortados, elevado número de erros, e dificuldades de manutenção são comuns ainda nos dias atuais em muitas organizações desenvolvedoras de software. Por esses e outros problemas, percebeu-se que o desenvolvimento de software não era uma arte, e que precisava de padrões e regras. Neste contexto destacam-se os modelos de maturidade como CMMI (Capability Maturity Model Integration) e MPS.BR (Melhoria de Processo de Software Brasileiro), que reúnem uma biblioteca de boas práticas dos resultados que podem ser alcançados buscando-se a maturidade organizacional (ler **Nota 1**).

Processos de software são extremamente importantes, pois a qualidade de um produto de software está intimamente ligada à qualidade do processo de software utilizado em seu desenvolvimento (FUGGETTA, 2000). Embora a importância dos processos de software seja destacada por diversos autores (ler **Nota 2**), neste artigo o foco será em demonstrar a importância das pessoas, ou seja, dos fatores humanos na Qualidade.

Pessoas são resistentes a processos, a mudanças. O ambiente das empresas deste ramo na sua grande maioria é desgastante. Lida-se diariamente com trabalhos sob pressão, prazos para "ontem", "atropelamento" de prioridades que podem culminar em uma cultura de horas extras excessivas, perda de produtividade, insatisfação do funcionário, perda da qualidade de vida afetando sem dúvida na qualidade desempenhado nas atividades em conseqüência no produto a ser desenvolvido.

Se não bastasse todo esse ambiente desgastante, os profissionais da área entram em conflito por conta dos diversos papéis desempenhados. Um exemplo clássico são as desavenças entre desenvolvedor e testador, o desenvolvedor se sente o "pai da criança" (em relação ao produto desenvolvido) e leva a atividade de detecção de erros como uma ofensa às vezes levando para o lado pessoal. Esquece-se que o importante é a harmonia do trabalho em equipe para a qualidade no produto final. É mais barato e produtivo corrigir um bug dentro do ciclo de desenvolvimento do que em produção.

Levando em consideração a subjetividade das atividades do desenvolvimento, que em cada novo projeto surgem novos requisitos cada vez mais complexos, com prazos mais curtos e tecnologias diferentes, medir o software não é uma tarefa trivial. E sem números em mãos se torna complicado para a alta gerência prover incentivos aos profissionais da área. Mais um motivo que traz insatisfação e que pode diminuir a produtividade do profissional.

Neste sentido, este artigo aborda fatores humanos que influenciam na qualidade do software e cita algumas práticas existentes para contornar alguns desses problemas que prejudicam a qualidade do produto de software.

Fatores humanos que influenciam na qualidade

Pressão no Desenvolvimento de Software

No ambiente de desenvolvimento de software, a pressão aumenta consideravelmente quando os cronogramas deixam

Nota do DevMan 1

Garantia de Qualidade

A Garantia da Qualidade visa avaliar a aderência das atividades executadas e dos produtos de trabalho gerados a padrões, processos, procedimentos e requisitos estabelecidos e aplicáveis, fornecendo uma visão objetiva e independente, tanto para atividades de processo quanto de produto, em relação a desvios e pontos de melhoria, de forma a assegurar que a qualidade planejada não será comprometida. Além de verificar se o processo está adequado, sendo seguido e trabalhando a favor da organização (evitando retrabalho, melhorando custos e prazos), busca-se identificar desvios o quanto antes e acompanhar a sua resolução até que seja concluído. A garantia da qualidade fornece suporte ao controle da qualidade por meio de evidência e confiança na habilidade do processo empregado em produzir um produto de software que atenda aos requisitos especificados. Ferramentas e técnicas utilizadas pela garantia da qualidade incluem auditorias (de produtos ou processos) e avaliações (appraisals ou assessments).

Desta forma, o processo de Garantia da Qualidade da organização também precisa ter sua aderência verificada de forma objetiva e independente. E, nesta situação, a equipe de garantia da qualidade não pode conduzir sua própria auditoria, sendo necessário que avaliadores externos (da própria organização ou contratados) realizem este avaliação de acordo com uma periodicidade previamente definida. Esta avaliação será tratada neste artigo por "Avaliação Independente de Garantia da Qualidade", porém na indústria de software também pode ser encontrada como "Auditoria Externa de GQA" e "QA do QA".

Nota do DevMan 2

Processo de Software

A partir da aplicação dos processos padrão da organização e outros ativos de processo organizacional na definição, planejamento e estimativa de processos para os projetos e da execução dos processos definidos, podem ser identificadas oportunidades de melhoria nos processos padrão para identificar pontos de ajustes nos processos de acordo com as necessidades de negócio da organização. A realização sistemática de revisões nos processos, planejamento e implementação de melhorias identificadas, a partir dessas revisões e da experiência em utilizar os processos padrão da organização, é o objetivo do processo Avaliação e Melhoria do Processo Organizacional.

A melhoria do processo de software pode ser considerada hoje uma das grandes prioridades para as organizações que trabalham com software. Isto se deve à exigência do mercado por produtos com maior qualidade, que sejam entregues mais rapidamente e com menor custo de desenvolvimento.

Estudos apontam que ao tentarem melhorar seus processos, as empresas estão em busca de:

- entender as características dos processos existentes e os fatores que afetam a sua capacidade;
- planejar, justificar e implementar ações que modificarão os processos, tornando-os mais coerentes com as necessidades de negócios e;
- avaliar os impactos e benefícios ganhos, comparando-os com os custos advindos das mudanças realizadas.

de funcionar e o número de defeitos encontrados no projeto torna-se incontrolável. Neste momento, em que ninguém mais sabe o que fazer para salvar o projeto, os indivíduos envolvidos sofrem com estresse, discussões podem surgir a cada minuto e a pressão acaba sendo combatida com mais pressão.

Segundo uma pesquisa do Standish Group em 1995, a pressão sobre os desenvolvedores pode quadruplicar o número de erros (KOSCIANSKI E SOARES, 2007).

Os gerentes devem estar atentos ao planejamento do projeto e ao seu acompanhamento. Problemas que podem determinar o aumento da pressão no trabalho como a corrida contra o tempo para cumprir o prazo inicialmente combinado podem ser evitadas se houver um planejamento adequado baseado em métricas anteriores e nos riscos reais envolvidos no projeto. A alocação dos recursos certos também determina o sucesso do planejamento. A ilusão de colocar mais recursos para cumprir a mesma atividade ou atividades correlatas pode aumentar o custo do projeto e a complexidade da interação entre os membros da equipe, e não aumentar a produtividade geral no projeto.

A área comercial, nas empresas de TI, também trabalha sob pressão para obter resultados. Além da concorrência, a margem de lucratividade é pequena e neste caso é necessário vender mais serviços para obter os resultados esperados. Nesta corrida sem freio, os vendedores às vezes são levados a realizar promessas irreais quanto a prazos ou especificações de produtos e serviços. Isto traz ainda mais pressão para a área técnica. Para cumprir os prazos idealizados pelos vendedores, muitas vezes a equipe técnica precisa simplificar o produto o máximo possível para reduzir a carga de trabalho o que acaba por comprometer a qualidade final do produto.

A cultura de horas extras

Os clientes das empresas de software estão cada vez mais exigentes. Uma das conseqüências é a redução dos prazos para a entrega dos diferentes produtos (some-se a isso o fato de termos uma complexidade crescente em termos de requisitos). Pela falta de processos definidos, na maioria das empresas existe muito re-trabalho. A triste realidade é que o sucesso de muitos projetos ainda depende do esforço heróico dos profissionais, soluções para "apagar incêndio" fazem parte do dia a dia dos profissionais de T.I. Com um mau planejamento, cumprir o cronograma se torna uma tarefa árdua, e para não decepcionar o cliente, os profissionais de T.I desdobram-se tentando realizar atividades às vezes complexas em que se envolve concentração e detalhe após o expediente. E as horas extras em muitas empresas acabam se tornando rotina.

Alguns destes poucos profissionais gostam do trabalho extra pela remuneração melhor no final do mês, além de terem em mente que serão mais reconhecidos por isto. Mas a realidade é que muitos perdem a produtividade, podem não ter a mesma eficácia de antes, sentem-se prejudicados na qualidade de vida e com isso as atividades executadas podem não ter o mesmo resultado final.

Normalmente, os desenvolvedores sentem-se desconfortáveis em fazer um trabalho sem qualidade. Eles também não querem fazer horas extras intermináveis e muito menos trabalhar sobre grande pressão de prazos. Horas extras levam à insatisfação da equipe, perda de produtividade e de qualidade (KOSCIANSKI e SOARES, 2007).

Aplicação de metodologias padronizadas: resistências

Algumas empresas, que ainda não possuem uma equipe ou uma maturidade suficiente, buscam consultorias para auxiliarem na definição de seus processos de software. Um ponto negativo nesta ótica é que a equipe de consultoria muitas vezes não envolve os profissionais da empresa cliente, e no momento de executar o processo percebe-se muita resistência, pois o processo definido pode não cobrir com harmonia a realidade da empresa cliente.

Mesmo quando não há uma equipe de consultoria auxiliando ou quando há e existe a presença também de representantes da equipe de projetos, ainda existe esta resistência porque o que ocorre na maioria das vezes durante a definição dos processos é que apenas alguns representantes participam do grupo de definição e se não há uma interação adequada e continuada com o restante da equipe, quando o processo for "rodar" haverá muitas discordâncias e muitas resistências da equipe.

Outro ponto que merece destaque é a quantidade de profissionais de nível técnico presentes nas equipes de software. Pela formação técnica, não entendem o porquê de todas as políticas e regras, não entendem e se dificultam a entender as vantagens de se utilizar os processos definidos. Pela formação técnica, **tendem** (já existem muitos cursos técnicos com este foco, mas essa ainda não é uma realidade comum) a não terem estudado sobre a Engenharia de Software, e acreditam que o desenvolvimento de software é uma arte que envolve forte uso da criatividade. Por conta deste pensamento, o uso de processos aparenta ser uma parte burocrática que o prejudica.

Conflitos interpessoais

A criatividade é um forte componente da atividade de desenvolvimento de software. Projetar e codificar não são tarefas repetitivas como erguer muro, o que determina uma importante influência do ego dos envolvidos durante o trabalho desenvolvido em equipe.

O desenvolvedor pode se sentir realmente responsável por sua criação a ponto de defendê-la (ou de defender seu trabalho) até as últimas instâncias. Por existir essa relação de posse entre o desenvolvedor e o produto, é possível entender o porquê os testes de sistema efetuados pelo próprio desenvolvedor se tornam tão ineficazes. A tendência (muitas vezes inconsciente) do desenvolvedor ao efetuar esses testes é a de promover situações assertivas em que o software apresenta bom desempenho e os erros reais não são percebidos. Daí a necessidade dos analistas de teste para executar as atividades de testes do sistema considerando os requisitos inicialmente levantados.

Mas essa relação de trabalho entre testador e/ou analista de testes e desenvolvedor em muitas empresas de TI têm sido a

principal origem de conflitos interpessoais, pois os defeitos encontrados no software pelo testador podem ser interpretados pelo desenvolvedor como uma crítica pessoal e a reação pode acontecer de forma inconsciente.

Os conflitos de interesses também acontecem entre as áreas comercial e técnica. As promessas irreais quanto a prazos feitas pelos vendedores em busca de resultados mais lucrativos muitas vezes é causa dos conflitos.

Dificuldades da Alta Gerência em prover incentivos

Para organizações não maduras, prover incentivos na área de TI não é uma atividade trivial. Se cada projeto tem um tamanho e uma complexidade, calcular a eficiência e produtividade do profissional de TI se não apoiado em números pode ser uma tarefa subjetiva. Dentre as dificuldades para se obter medidas de software, pode-se citar duas: a variedade de aspectos a considerar e a presença de muitos elementos intangíveis (KOSCIANSKI E SOARES, 2007).

Com isso, um profissional pode não ter o reconhecimento que deseja e trabalhar sem motivação. Entretanto, é importante destacar que o dinheiro nem sempre é alicerce de motivação. Principalmente para aquelas pessoas que já atingiram seus objetivos de vida, existem formas mais simples para gerir a motivação dentro do grupo e manter o time integrado (Portal Fator Brasil, 2006).

Boas práticas para as influências humanas

Institucionalização dos processos

Podemos considerar a institucionalização como uma atividade que dependerá exclusivamente de variáveis do ambiente para definir sua complexidade e sucesso. Inicialmente, os profissionais envolvidos não conseguem entender a sua importância, mas com o apoio dos superiores, esta dificuldade pode ser minimizada.

A Qualidade do produto está intimamente ligada aos processos utilizados no desenvolvimento. O principal objetivo da institucionalização do processo é geralmente maximizar a qualidade do produto, mas outros fatores positivos são adquiridos. Os profissionais necessitam entender a real necessidade da existência dos processos, precisam entender o porquê de cada atividade que executam para realizarem com maior qualidade.

Produtos de softwares criados sem processos definidos (ad-hoc) geralmente causam dependência do produto aos profissionais que o desenvolveram. Se o profissional sair da empresa o projeto passa por situações de aumento de prazos e custos. Esta situação traz desconforto para a organização e para minimizar este e outros problemas já conhecidos, é necessário que a organização tenha um processo institucionalizado.

Para alcançar este nível, a instituição necessita tomar algumas medidas que inicialmente exigirá investimento de recursos financeiros e tempo.

Os modelos MPS.BR e CMMI fornecem suporte para a institucionalização de processos. Geralmente as atividades do processo se apóiam na existência de Gerências e estas devem realizar e monitorar suas atividades para que o processo proposto seja institucionalizado na organização.

Planos realísticos

Planos realísticos e honestos devem fazer parte das organizações maduras. Uma estratégia muito adotada para ganhar a concorrência é propor custos ou prazos reduzidos. Essa prática é desleal e desastrosa. As principais consequências são projetos de baixa qualidade – perdendo-se a confiança do cliente, e também pressão e estresse no desenvolvimento, perdendo-se o espírito do trabalho em equipe (KOSCIANSKI E SOARES, 2007).

Os riscos devem ser analisados e gerenciados continuamente, é necessário buscar métricas mais precisas. A construção de uma base histórica de projetos anteriores é fundamental para servir de base a projetos futuros. A partir da base histórica pode ser possível definir parâmetros para os projetos atuais. Alem disso, é importante que esta base seja constantemente atualizada. Um dos segredos para o sucesso é aprender com os projetos anteriores reduzindo a possibilidade de fracasso (KOSCIANSKI E SOARES, 2007).

O planejamento para desenvolver um produto de software é um ponto crucial para o sucesso do projeto. Estipular prazos, alocar e coordenar recursos e gerenciar o escopo são atividades da gerência de projetos que determinam os caminhos de cada projeto.

É função do gerente de projeto fazer com que a equipe sinta-se bem no projeto. Por isso o planejamento correto se torna tão importante.

Como construir um planejamento correto?

É muito importante que o gerente de projetos exerça realmente a função de gerente e que não seja simplesmente um controlador de resultados e prazos. Ou seja, ele precisa "tirar as pedras" que aparecem nos caminhos de cada membro da equipe.

Além dos conceitos e técnicas fundamentais para gestão de projetos presentes no PMBOK, o gerente de projetos precisa também para gerenciar melhor os projetos segundo recomendação de FERRARI, 2009:

- Conhecer as pessoas da equipe de projeto e entender como atuam em conjunto. Caso esteja atuando com recursos até então desconhecidos, apenas a vivência e a habilidade pessoal lhe permitirão identificar estes aspectos rapidamente;
- Conhecer a organização (ou organizações, se estiver envolvido com clientes, fornecedores ou parceiros) - tanto a formal (publicada), quanto a informal (definida por influenciadores, elos pessoais, interesses individuais, valores, comprometimentos específicos, etc.). Apenas esse conhecimento lhe permitirá vislumbrar oportunidades não aparentes, como a de solicitar a um vendedor, que foi o idealizador de um produto há anos, que trabalhe num sábado para realizar, em poucas horas, o que um analista levaria dias;
- Conhecer o produto, a tecnologia e o negócio a que se aplicam. Similarmente, só isso lhe permitirá vislumbrar certas oportunidades e dificuldades, além de apoiar-lhe na adequada comunicação com a equipe técnica e usuários;

• Conhecer o contexto do negócio de sua própria empresa. A identificação de riscos, oportunidades e impactos no ciclo de vida do software na imagem da empresa e no contexto de negócios da organização somente será possível para quem teve a oportunidade de vivenciar e compreender a empresa, obtendo uma visão holística de seu funcionamento; e
• Mais objetivamente, conhecer a fundo o processo e as metodologias em si, segundo os quais o desenvolvimento ou a implantação do software será conduzido.

Com o entendimento do negócio e das tecnologias envolvidas pelo gerente de projetos, pode ser mais fácil conquistar a confiança da equipe técnica a ele subordinada. Além disso, "orquestrar" as tarefas do projeto não será "bicho de sete cabeças", pois ele terá o conhecimento necessário para isso. Esse gerente ainda terá argumentos convincentes para discutir prazos com a área comercial.

Os prazos impraticáveis de alguns projetos (apoiados em horas extras e nos finais de semana) comprometem a qualidade do produto final.

Todos esses fatores de planejamento podem prevenir a necessidade de "heróis" (funcionários ou gerentes extremamente competentes e que resolvem os problemas) nas organizações, pois as atividades estão organizadas e não vai ser necessário simplesmente reagir aos problemas que aparecerem.

Revisão por pares

Encontrar defeitos no resultado do próprio trabalho não é uma atividade fácil, por isso uma das técnicas recomendadas por modelos de maturidade é a técnica de revisão por pares (uma visão externa pode visualizar falhas que o criador não perceberia).

Qualquer artefato pode ser revisado por outra pessoa. A prática da revisão por pares é eficaz para encontrar erros e não-conformidades com os padrões estabelecidos. A condução adequada da revisão pode evitar conflitos entre as equipes e funcionários. Os gestores devem estar preocupados em evitar atritos, deve-se evidenciar que a intenção das revisões por pares é julgar os resultados do trabalho e não julgar o profissional em si (KOSCIANSKI e SOARES, 2007).

É interessante que as revisões em pares sejam executadas de acordo com um procedimento documentado. É recomendado também que sejam armazenados os dados da condução e resultados das revisões em pares.

O fator mais importante na qualidade de um software é o compromisso de cada desenvolvedor em trabalhar para esse objetivo de acordo com o PSP[1]. O PSP auxilia no aprendizado com os erros, permitindo entender o tipo, quantidade e freqüência de defeitos inseridos em um programa. Os autores KOSCIANSKI e SOARES apontam algumas orientações para a realização da tarefa de revisão:
• Usar listas de verificação (checklists) baseadas nos problemas encontrados em projetos anteriores;
• Seguir um processo estruturado de revisão;
• Criar produtos tendo em mente a futura revisão.

Investimentos no profissional de T.I

Atualizar, especializar, certificar são ações comuns e necessárias para profissionais de qualquer área. Para a tecnologia da Informação, estas ações ganham uma atração especial uma vez que esta área sofre constantes evoluções, o que requer acompanhamento por parte dos profissionais.

Evolução de hardware, software, técnicas, ferramentas, metodologias são frequentemente anunciadas e algumas destas trazem resultados expressivos em diversas circunstâncias a uma organização. Saber utilizar estes itens é uma das maiores tarefas a serem realizadas e para que isso ocorra, é necessário que o profissional de T.I obtenha conhecimentos através de bibliografia, cursos, palestras, workshops, congressos e qualquer outro meio de informação.

A capacitação e a oferta de condições de trabalho atraente, como salário e outras motivações que despertam os talentos dos profissionais, requer investimento da organização. Esta prática é motivo de sobrevivência para estas organizações uma vez que oferecer condições de trabalho favoráveis é garantir a permanência de bons profissionais de T.I.

Os funcionários mais antigos sentem-se valorizados quando há investimento em sua formação contínua. Novas necessidades de treinamento podem surgir no decorrer dos novos projetos como processos, linguagens ou ferramentas ainda não conhecidas. Uma estratégia que reduziria os custos é treinar o pessoal disponível ao invés de contratar novos (KOSCIANSKI e SOARES, 2007).

Avaliação da equipe de TI

A equipe de Tecnologia da Informação deve ser avaliada, como qualquer outra, pelos resultados produzidos. Para realizar uma análise consistente, é necessário obter informações sobre diferentes aspectos que podem influenciar a produção desta equipe. As condições de trabalho adequadas, metas definidas anteriormente e a estrutura de organização da equipe de T.I são fatores primordiais.

As metas definidas pela empresa têm que estar de acordo com a possibilidade da equipe de T.I. Não podemos determinar que uma equipe tenha uma produção indesejável se as metas que ela deve alcançar são irreais ou impossíveis em razão do conhecimento, prazo ou custo ou qualquer outro fator que impossibilite o cumprimento das metas.

A estrutura da equipe de T.I é um dos maiores problemas. Este fator também está relacionado às condições de trabalho. Um dos maiores problemas que podemos observar é na existência de líderes de equipe de T.I (presidente, gerentes e outros), que possui apenas características de gestor e/ou características técnicas.

---

1 - Personal Software Process (PSP) é um processo de desenvolvimento de software projetado para ser utilizado por engenheiros de software para a elaboração de projetos individuais. O PSP foi desenvolvido por Watts Humphrey e está descrito no seu livro "A Discipline for Software Engineering" (Uma disciplina para Engenharia de Software) de 1995. O PSP foi desenvolvido para orientar o planejamento e desenvolvimento de módulos de software ou pequenos programas, mas pode ser adaptado para outras tarefas pessoais. (Wikipédia)

Uma dica para as empresas que possuem conflitos entre a área de vendas e a área de desenvolvimento: criação de uma área intermediária entre as áreas de desenvolvimento e vendas. Os profissionais desta nova área deveriam ter conhecimentos técnicos suficientes para entender os problemas e propor soluções em prazos possíveis de serem cumpridos. É possível que esta área consiga harmonizar as expectativas do setor comercial com as possibilidades de realização do setor de desenvolvimento.

Considerações Finais

Quando falamos sobre melhoria no desenvolvimento de software, podemos citar vários fatores que contribuiriam. Esse artigo demonstrou a influência dos fatores humanos na qualidade do produto de software. Apesar dos inúmeros problemas enfrentados pelas empresas de TI, elas conseguem sobreviver. Isso graças à relativa tolerância em relação à má qualidade de produtos. Usuários frustrados, às vezes, atribuem os problemas a si próprios: "Não sei usar um computador". (KOSCIANSKI E SOARES, 2007).

No entanto, até quando o mercado vai ser tolerante?

As empresas devem estar atentas a todos os fatores que podem contribuir na qualidade do software. A começar pelos recursos humanos, responsáveis pela real produtividade e crescimento da empresa. É necessário entender que o desenvolvimento de software não pode ser encarado como uma atividade de manufatura industrial. Projetar arquiteturas e escrever código não são tarefas repetitivas e contam com a criatividade de um desenvolvedor.

Uma das formas de se tratar a maioria destes problemas é utilizar metodologias, métodos e técnicas que foram desenvolvidos para esse tipo de ambiente. Contudo, um método de trabalho não estará completo se as pessoas não tiverem uma compreensão do porquê de utilizar um método, como aplicá-lo, quais os problemas envolvidos e a função de cada um no processo. A boa comunicação desempenha papel essencial na correta realização de todas as tarefas (KOSCIANSKI E SOARES, 2007).

Não é intenção dos modelos de melhoria de qualidade tratar o fator humano como item relevante de qualidade de software, assim como não é de obrigação dos mesmos ditar as tecnologias que devem ser incorporadas. Mas fica aqui a mensagem de que os fatores humanos também precisam ser considerados uma vez que são fatores de sucesso do projeto. ●

Referência Bibliográfica

KOSCIANSKI, André, SOARES, Michel dos Santos. Qualidade de Software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. 2007.

FERRARI, Ernani. Gestão de Projetos de Software: além do PMBOK. Disponível em <http://www.ethesis.inf.br/index.php?option=com_content&task=view&id=4945&Itemid=53> Acesso em 11/04/2009.

Portal Fator Brasil. Dinheiro não é alicerce da motivação. Disponível em: <http://www.revistafator.com.br/ver_noticia.php?not=45497>. Acesso em 11/04/2009.

Wikipédia. Personal Software Process. Disponível em <http://pt.wikipedia.org/wiki/Personal_software_process>. Acesso em 14/04/2009.

SANDHOF, Karen. Fatores Humanos no processo de desenvolvimento de software: um estudo visando qualidade. São Paulo, 2004.

ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO (SOFTEX). MPS.BR – Guia de Implementação – Parte 3: Nível E (Versão 1.1), http://www.softex.br/mpsbr/_guias/MPS.BR_Guia_de_Implementacao_Parte_3_v1.1.pdf, 2007b.

ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO (SOFTEX). MPS.BR – Guia de Implementação – Parte 2: Nível F (Versão 1.1), http://www.softex.br/portal/mpsbr/_guias/MPS.BR_Guia_de_Implementacao_Parte_2_V1.1.pdf, 2007c.

Biondo, Aline. Uma Ferramenta para Garantia da Qualidade Aplicada na Implementação de Sistemas. Graduação em Sistemas de Informação, Universidade Luterana do Brasil (ULBRA), Canoas, 2007.

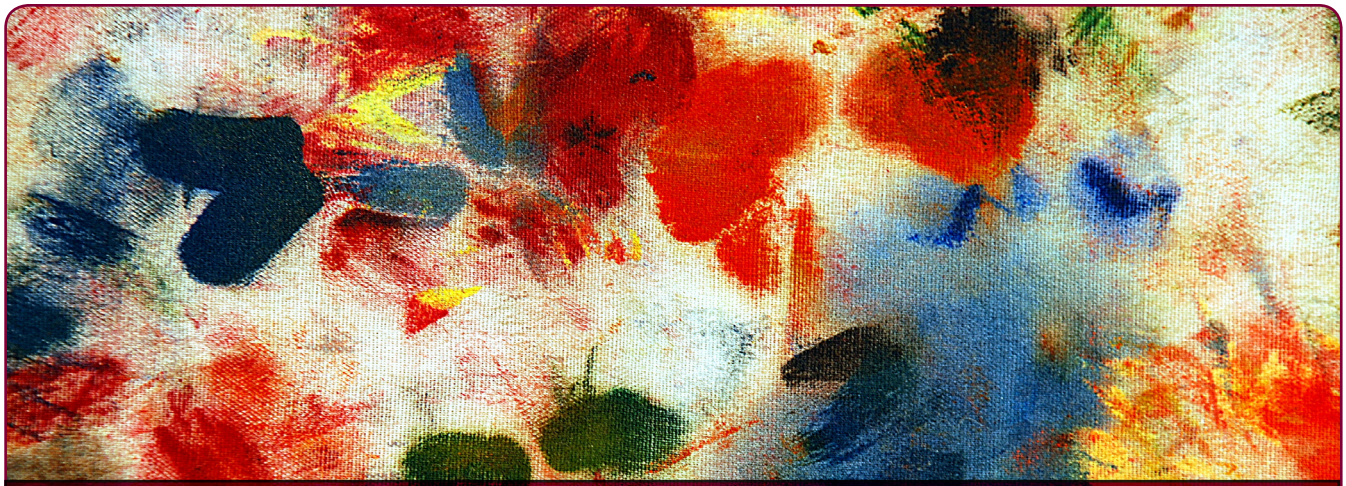
CMU/SEI, 2001, Standard CMMI Appraisal Method for Process Improvement (SCAMPI), Version1.1: Method Definition Document, CMU/SEI-2001-HB-001, Pittsburgh, Software Engineering Institute, Carnegie Mellon University. URL: http://www.sei.cmu.edu

CMU/SEI, 2002, Capability Maturity Model Integration (CMMI), Version 1.1 CMMI for Software Engineering (CMMI-SW, V1.1), Pittsburgh, Software Engineering Institute, Carnegie Mellon University. URL: http://www.sei.cmu.edu

Dê seu feedback sobre esta edição!

A Engenharia de Software Magazine tem que ser feita ao seu gosto.
Para isso, precisamos saber o que você, leitor, acha da revista!
Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/esmag/feedback



Modelo de Avaliação de Interfaces

Avaliação de interfaces através de checklist

Antonio Alberto Moreira
anjonyck@gmail.com
Bacharel em Sistemas de Informação pela Faculdade Metodista Granbery.

Eduardo Pagani Julio
epagani@gmail.com
Mestre em Computação pela Universidade Federal Fluminense e professor do Curso de Bacharelado em Sistemas de Informação da Faculdade Metodista Granbery.

Alessandreia Marta de Oliveira Julio
amojulio@granbery.edu.br
Mestre em Engenharia de Sistemas e Computação pela COPPE/UFRJ. Coordenadora e Professora do Curso de Bacharelado em Sistemas de Informação da Faculdade Metodista Granbery.

A qualidade em sistemas interativos está intimamente relacionada ao conceito de usabilidade. Alguns autores definem a usabilidade como um processo de elaboração e *design* de sistemas interativos que objetiva prover a facilidade de aprendizado e de uso, e proporcionar satisfação aos seus usuários.

Para que os usuários possam realizar suas atividades de forma satisfatória, é necessário que os projetistas se empenhem em aperfeiçoar a interação entre os usuários e os sistemas da melhor forma possível. Para isto, deve-se ter como diretriz o projeto centrado no usuário utilizando-se métodos específicos para alcançar estes objetivos.

Para analisar a usabilidade de um sistema, determinando e resolvendo os problemas de interação, é necessário estabelecer uma bateria de avaliações e testes a serem aplicadas como parte integrante do projeto.

De que se trata o artigo?

Avaliação de usabilidade em sistemas interativos. Este artigo descreve a avaliação de usabilidade, utilizando o modelo de checklist para efetuar este processo.

Para que serve?

A obtenção da qualidade de interação nos sistemas computacionais pode ser conseguida através de avaliações precisas e do uso de ferramentas adequadas. Neste contexto, observa-se a necessidade de estudar outros modelos de ferramentas que apóiam o projetista durante o processo de desenvolvimento e avaliação para que ele possa adequar as suas necessidades sem perda na qualidade do processo.

Em que situação o tema é útil?

Nos processos de desenvolvimento de sistemas interativos podem ser utilizados checklists como ferramentas que norteiam a concepção do sistema e alicerçam o processo avaliativo de forma econômica e eficaz para obter uma interface de qualidade.

As avaliações devem ter o seu planejamento bem estruturado de forma que, ao se iniciar o projeto, se inicie também a bateria de avaliações. Isto reflete na obtenção da qualidade do sistema e evita agregar maior custo ao projeto.

A necessidade de efetuar esta bateria de testes pode ser evidenciada pela quantidade de sistemas que apresentam um grau de interação deficiente. Isto ocorre devido a um projeto debilitado que não seguiu corretamente os métodos e as diretrizes que o estudo da Interação Humano Computador (IHC) propõe, como por exemplo, a falta do processo avaliativo durante o desenvolvimento do sistema ou uma aplicação deficiente destes métodos.

Avaliação de interfaces

A avaliação da interface tem como objetivos gerais validar a eficácia e a eficiência da usabilidade, conforme as tarefas que são realizadas pelos usuários e os recursos empregados tais como tempo, quantidade de erros, passos desnecessários, busca de ajuda, etc.

Dependendo da fase do projeto, a avaliação pode ser **formativa** (ou construtiva), que pode ser efetuada ao longo de todo o processo de *design* usando cenários, *storyboards* e protótipos do sistema, ou pode ser **somativa** (ou conclusiva), normalmente feita nas etapas finais de cada ciclo do desenvolvimento, usando o produto (ou protótipo intermediário ou final) funcionando.

Assim, o processo de avaliação não deve ser entendido como uma atividade isolada a ser efetuada somente ao final do projeto de interação. Ele deve ocorrer durante todo o ciclo de vida do desenvolvimento do sistema, avaliando os resultados e fornecendo sustentação à equipe, de forma a agregar melhorias gradativas à interface, como pode ser observado na **Figura 1**.

Figura 1. Processo de desenvolvimento e avaliação de interfaces

Esta atividade interfere diretamente na garantia da qualidade do software, entretanto, um projetista não deve pressupor que seguindo somente os métodos e princípios estabelecidos no projeto, obterá a qualidade do sistema. Deve-se considerar que cada usuário é uma pessoa ímpar e, neste caso, não é prudente aplicar a avaliação individualmente para atestar isto. É necessário avaliar também o contexto onde ele está inserido.

Independente da técnica de avaliação a ser utilizada, sua aplicação requer a efetivação de um planejamento como sugere a norma ISO 14598 (*Information technology-software*), que propõe que os processos de avaliação sigam a seguinte estrutura:

- análise: identificação dos requisitos da avaliação;
- projeto preliminar: seleção das técnicas aplicáveis;
- projeto detalhado: configuração das técnicas;
- implementação: realização da avaliação;
- documentação: elaboração dos relatórios;
- validação: confronto entre os resultados esperados e os obtidos com a avaliação.

É difícil prever uma perspectiva que garanta a perfeição do processo, por conseguinte, o planejamento deve pressupor métodos de avaliação contínua e correção dos eventuais problemas durante todo o ciclo de vida do projeto. Entretanto, deve-se considerar que o desenvolvimento da interface interativa recai sobre aprimoramentos sucessivos determinados pelo orçamento e motivação da equipe.

É imprescindível estruturar um planejamento do processo avaliativo para organizar as decisões a serem tomadas, colaborando com os avaliadores inexperientes na realização da avaliação. Os pontos relevantes que devem ser considerados são os seguintes:

- determinar os objetivos gerais que a avaliação deve tratar;
- explorar perguntas específicas a serem respondidas;
- escolher o paradigma e as técnicas de avaliação;
- identificar questões práticas que devem ser tratadas;
- decidir como lidar com questões éticas;
- avaliar, interpretar e apresentar os dados.

Neste contexto, a avaliação é utilizada para comparar a usabilidade atual dentro do processo com as especificações estipuladas anteriormente.

A efetivação de um teste pode ser realizada em dois tipos de ambientes: no local de trabalho e em um laboratório de usabilidade, conforme pode ser visualizado na **Figura 2**.

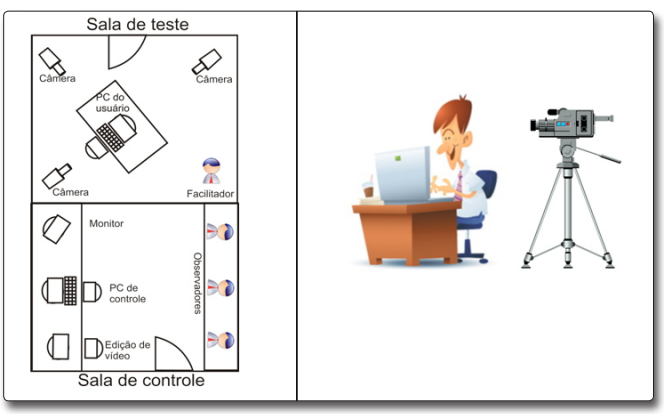


Figura 2. Teste feito no laboratório e no local de trabalho

Nos testes feitos em laboratórios, a interação dos usuários é filmada em vídeo, capturada em software, e opcionalmente observada através de uma janela espelhada. Este tipo de observação oferece uma grande riqueza de indicações (por gestos, posturas, falas dos usuários) da atitude e sensação que a tecnologia provoca nos usuários.

Apesar de ser fácil ver quando os registros indicam que a pessoa está pensando, é muito difícil saber em que estão pensando. Uma técnica muito utilizada é o *think aloud* (pensar

"em voz alta"), onde é solicitado ao participante do teste que vá falando em voz alta no que está pensando.

Embora esta técnica sirva para explicitar um raciocínio do usuário, ou suas dúvidas e expectativas, ele interfere na interação, pois as pessoas tendem a pensar antes de agir, o que não é sempre o caso (muitas vezes estamos agindo sem pensar na ação, concentrados apenas no objetivo que queremos atingir).

É importante nesse estilo de teste que os avaliadores preparem um termo de consentimento atestando de que a participação nos testes é voluntária e que a privacidade e anonimato dos participantes estão garantidos pela equipe de avaliação, resguardando assim todos os envolvidos no processo de eventuais problemas futuros.

Já nos testes realizados no local de trabalho, o ambiente é natural, em que o usuário interage com a tecnologia. O observador deve anotar cuidadosamente uma série de informações, como:

- Quem está presente no local da observação (incluindo o observador)? Quais as características destas pessoas?
- O que está acontecendo? O que estão fazendo? O que dizem? Qual a sua postura (estão sentados, de pé, agitados, confusos)? Que impressão passam (bom humor, nervosismo, contrariedade)? Qual o "clima" da cena (tenso, relaxado, animado)?
- Qual o local e o momento dos acontecimentos? O que desencadeou os acontecimentos? Qual a finalidade do que as pessoas estão fazendo? Como as atividades que se passam estão organizadas? Quem faz o quê? Em que ordem estão fazendo?

Esse tipo de teste, dependendo de como é conduzido, pode influenciar nos resultados da avaliação, visto que a presença de um superior no teste pode intimidar o funcionário, devendo então ser realizado com muito cuidado.

Outro fator que deve ser considerado nesse tipo de teste é a questão de nem sempre ser possível capturar o registro. Normalmente um observador fica junto ao usuário e apenas anota tudo que está acontecendo. Isso faz com que o observador tenha um papel fundamental na coleta dos registros de acontecimento. Além disso, o teste é mais realista que os testes realizados no laboratório.

Processos Metodológicos

Os métodos de avaliação de interface diferem entre si em vários aspectos. Conforme as necessidades do projeto, a equipe deve compreender as características de cada método e definir qual é o mais adequado para ser utilizado no sistema em desenvolvimento.

Porém, para se definir o método correto, deve-se considerar o prazo, o custo, o número mínimo de especialistas, o ambiente físico e os equipamentos envolvidos no desenvolvimento do sistema.

As técnicas de avaliação são diagnósticos baseados nas verificações e inspeções das interfaces que afetam a interação do usuário com o sistema. Sendo assim, é imprescindível manter o foco do desenvolvimento no usuário de forma a maximizar a sua utilização. Entretanto, pode-se observar que a escolha do método adequado está estritamente relacionada com as características do sistema a ser desenvolvido.

Os métodos de avaliação podem ser classificados sob o aspecto de duas dimensões: inspeção de usabilidade e testes de usabilidade.

A inspeção de usabilidade refere-se às avaliações efetuadas desde o início da fase de desenvolvimento do sistema e envolve necessariamente a participação de um avaliador experiente. Normalmente estes avaliadores são os projetistas que integram a equipe de desenvolvimento ou as pessoas envolvidas com o projeto.

Um avaliador deve possuir alguns conhecimentos básicos, como o conhecimento sobre o domínio da aplicação, sobre o projeto de interface com o usuário e também deve possuir experiência no método de avaliação. Assim, a avaliação se torna mais pontual, dando maior credibilidade ao processo.

Alguns modelos que são amplamente utilizados pela comunidade IHC para estes tipos de avaliações são: avaliação heurística, revisão de guias de referência, percurso cognitivo e *checklists*.

A avaliação heurística consiste em um estudo da interface seguindo uma lista de heurísticas de usabilidade e tem como característica o baixo custo, a rapidez e a facilidade de utilização. Deve ser realizada por especialistas experientes e competentes. Esta avaliação representa um exame detalhado sobre as qualidades da interface de usuário, objetivando diagnosticar os problemas e as dificuldades com que os usuários possam se deparar durante o processo de interação com o sistema. Para conseguir apontar todos os problemas da interface do sistema, faz-se necessário o envolvimento de mais de um avaliador, de três a cinco pessoas, sendo ideal para se obter um bom resultado.

A revisão de guias de referência verifica a conformidade da interface com atributos contidos em guias de referência publicadas, onde estão inseridas as descrições detalhadas dos elementos de interface de um sistema como menus, janelas, cores, etc. Essa não é uma tarefa simples, o que exige um conhecimento especializado. Entretanto é mais fácil efetuar a avaliação da usabilidade quando o contexto é apresentado, relacionando-o aos seus respectivos argumentos, como é apresentado na **Tabela 1**.

Guia de referência	Adotar uma organização consistente para as posições na tela, dos vários elementos do sistema
Exemplo	Posição para título Área de dados de saída Área de dados de entrada Área para opções de controle Área para instruções
Exceção	Pode ser desejável mudar formatos para distinguir entre tarefas diferentes
Comentário	Consistência ajuda na orientação do usuário

Tabela 1. Exemplo de proposta de guia de estilo

Muitas destas guias pertencem a grandes empresas como *Common User Access Guidelines* (IBM), *Web Design Guidelines* (IBM), *The Windows Interface Guidelines for Software Design (Microsoft), Macintosh Human Interface Guidelines (Apple)* e a *Osf Motif Style Guide (Open Software Foundation).*

O percurso cognitivo emprega uma lista de verificação orientada à tarefa interativa onde se estudam os processos cognitivos que ocorrem entre o usuário e o sistema. O avaliador navega nas funcionalidades da interface executando as tarefas que seriam realizadas pelo usuário, observando as ações disponíveis e comparando-as com o *feedback* da interface. É importante para uma boa análise que o avaliador conheça o percurso previsto no sistema na realização das tarefas e atente também para o conhecimento do usuário em relação a essas tarefas e em relação à operação do sistema.

O *checklist* é uma adaptação ou especialização das guias de referência contendo um conjunto de regras aplicáveis ao projeto, de forma a fornecer condições de detectar problemas inerentes à usabilidade do sistema.

Já os testes de usabilidade, são métodos de avaliação que envolvem os usuários do sistema durante o processo avaliativo. Esta atividade utiliza conceitos da IHC para aplicar testes abordando a usabilidade e a comunicabilidade do sistema. Neste caso, é necessário haver uma implementação parcial ou total do sistema para poder aplicar este método. Uma simulação da capacidade interativa do sistema, um protótipo, um cenário ou até mesmo a implementação completa do sistema, são bons exemplos de testes usabilidade.

Avaliação por *Checklist*

Este artigo aborda o modelo de avaliação *checklist* que consiste em um método de avaliação de sistemas interativos que contempla diversos critérios de usabilidade. Esta avaliação tem o intuito de determinar a qualidade da interação entre o usuário e o sistema, e a qualidade dos objetos utilizados para prover esta interação.

Checklists sugerem um conjunto de regras aplicáveis diretamente ao projeto. Essas regras permeiam uma das alternativas existentes para o processo avaliativo que não expressam necessidade de um grande esforço de interpretação, focalizando aspectos relevantes da interface de forma a fornecer condições de detectar problemas inerentes à usabilidade do sistema.

O *checklist* é um método de avaliação que está inserido na classe de inspeção de usabilidade e deve ser efetuado pela equipe de desenvolvimento. Este método contém regras e diretrizes que permeiam uma das alternativas existentes para efetuar o processo avaliativo e não expressa necessariamente um grande esforço de interpretação por parte do avaliador, focando em aspectos relevantes da interface, fornecendo condições de detectar problemas inerentes a usabilidade do sistema.

Alguns autores afirmam que *checklists* são como as guias de referência, entretanto, ele pode representar adaptações ou especializações destas guias, podendo mesmo ser elaboradas pela própria equipe de projetistas para atender as necessidades do projeto.

Neste contexto, deve-se considerar como resultado a obtenção de uma lista de requisitos mais restrita e direcionada para o sistema a ser avaliado do que as guias de referência genéricas.

A elaboração do *checklist* deve ser bem estruturada para alcançar resultados satisfatórios e abrangentes. Este modelo de avaliação objetiva a qualidade das ferramentas e não dos avaliadores. Os resultados obtidos com essa técnica dependem da qualidade da elaboração dos *checklists*, o que torna esta ferramenta responsável pela identificação dos problemas de usabilidade do sistema avaliado.

Os *checklists* servem como guia para a equipe, durante o processo de desenvolvimento do sistema, e também como ferramenta avaliativa para garantir a qualidade.

No processo de desenvolvimento, os projetistas consultam os *checklists* seguindo seus critérios com o intuito de antecipar e evitar os possíveis problemas de usabilidade. Por outro lado, a utilização no processo avaliativo se faz seguindo minuciosamente cada um dos critérios do *checklist* e verificando se todos os critérios estão em conformidade com os requisitos da interface do sistema.

A elaboração de um *checklist* criterioso acarreta em resultados uniformes e abrangentes, entretanto é essencial que seu conteúdo seja pertinente ao contexto e organizado de forma a apresentar não somente as próprias questões da avaliação em si, mas também um conjunto de elementos explicativos como notas, exemplos e glossários.

Nota-se desta forma que é absolutamente necessária a obtenção de um *checklist* de qualidade, pois esta condição influencia diretamente no resultado final da avaliação.

Modelos de *checklist*

Conforme já foi abordado, os modelos de *checklist* podem ser específicos para determinado projeto ou adaptações de guias de referência. Alguns modelos que são utilizados para se aplicar avaliações são encontrados na literatura ou estão disponibilizados na Internet.

A aplicação de um *checklist* consiste em um método simples e eficaz para aferir a usabilidade de um sistema, tais como:

- conformidade a padrões preestabelecidos (KDE, GNOME, *Windows*, etc.);
- certificação de consistência e padronização da interface;
- certificação de legibilidade dos itens da interface.

Um exemplo de *checklist* que se encontra disponível na Internet é o ErgoList, apresentado na **Figura 3**.

Segundo o LabIUtil (Laboratório de Utilizabilidade de Informática da UFSC), o ErgoList é uma ferramenta que tem como objetivo conceber, projetar, desenvolver e disponibilizar via Internet, condições para que os projetistas de sistema possam desenvolver dispositivos de software de forma correta, sob o ponto de vista da interação com o usuário, e efetuar avaliação através do modelo do *checklist* proposto.

O Ergolist está dividido em 18 critérios, onde cada critério está subdividido em vários itens, totalizando 194 questões, onde é possível verificar os índices de usabilidade do sistema.

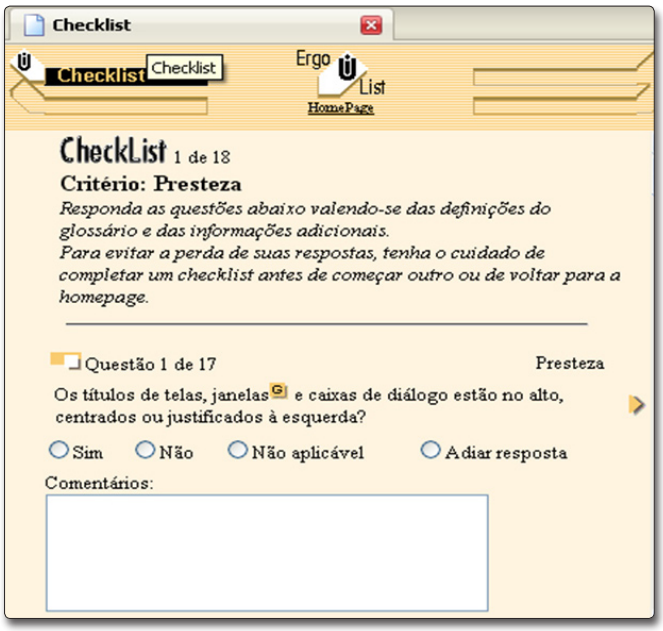


Figura 3. ErgoList do LablUtil (UFSC)

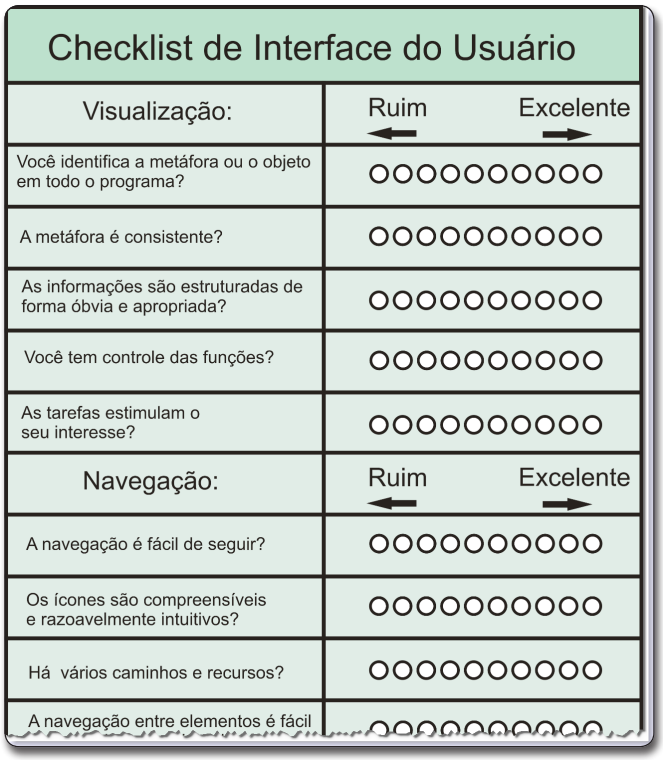


Figura 4. Modelo de checklist (Adaptado da Universidade de Auckland)

Como o ErgoList tem seus requisitos já pré-definidos, isto o torna uma ferramenta genérica o que muitas vezes foge ao contexto do projeto. Neste caso é necessário que haja um estudo de suas diretrizes para que se possa efetuar uma adaptação para o projeto que está sendo avaliado.

Outra opção é a equipe de desenvolvimento elaborar um modelo específico de *checklist* para satisfazer o contexto do projeto a ser implementado. Isto pode ser conseguido utilizando-se como matriz, as guias de referência, modelos prontos ou elaborar o *checklist* fundamentado no levantamento de requisitos do sistema. No entanto, para que resultados satisfatórios sejam obtidos, deve-se atentar para a organização, o conteúdo e a qualidade na elaboração dessas ferramentas.

Na **Figura 4** pode-se observar outro modelo de *checklist*, que enfoca uma avaliação para um sistema específico.

Para apoiar o processo de avaliação de forma eficaz, é relevante observar que o *checklist* deve ser construído de forma a atender os requisitos da avaliação dos elementos da aplicação.

Modelos específicos não necessitam seguir padrões pré-estabelecidos. Muitas vezes basta-se criar uma lista de critérios e segui-la de forma perspicaz.

É de responsabilidade da equipe de desenvolvimento definir os itens que devem constituir o *checklist* de forma a procurar satisfazer as necessidades do sistema. Entretanto, critérios como estrutura de menu, ícones, metáforas, navegação, entre outros, são sempre relevantes na hora de elaborar um *checklist* de qualidade.

Para se obter uma melhoria na qualidade do processo avaliativo, pode-se efetuar uma conjunção, utilizando-se dois processos simultâneos, a avaliação heurística e o *checklist*. Esta abordagem visa valer-se das vantagens que cada uma delas apresenta no seu contexto reforçando o processo. No entanto, é recomendável que primeiro se aplique a avaliação heurística, livre de qualquer influência, para depois utilizar o *checklist*. A adoção deste princípio é necessária para evitar que os avaliadores tendam a utilizar somente os itens contidos no *checklist*, já que este apresenta no seu contexto, um modelo mais simples de avaliação.

Conforme dito anteriormente, é importante frisar também que os avaliadores devem ser treinados para que o resultado da avaliação seja expressivo.

Vantagens do *checklist*

A coerência nos resultados é significativa, pois os avaliadores são conduzidos na inspeção, seguindo-se a série de quesitos estabelecidos pelo *checklist*.

Nesse aspecto, a avaliação por *checklist* apresenta algumas vantagens na sua utilização devido a algumas características fundamentais:

- a sistematização da avaliação certifica ao processo resultados coerentes e estáveis devido a verificações constantes nos critérios adotados;
- a facilidade na identificação dos problemas de usabilidade, devido à especificidade nos critérios do *checklist*;
- a melhoria na eficácia do processo avaliativo referente à restrição da subjetividade que normalmente estão agregadas aos processos avaliativos;
- diminuição do custo do processo, já que o método é simples e objetivo.

Análise de resultados

Ao final da avaliação é gerado um relatório onde são detalhados os problemas que foram encontrados durante o processo avaliativo, como mostra a **Figura 5**.

Legenda		
S = sim	N = não	NA = não aplicável
Presteza: verificação se o sistema informa e conduz o usuário durante a interação		
Critério	**Resposta**	Comentário
Os títulos de telas, janelas e caixas de diálogo estão no alto, centrados ou justificados à esquerda?	S	Os títulos de telas, janelas e caixas estão posicionados no alto mantendo um padrão constante.
Todos os campos e mostradores de dados possuem rótulos identificativos?	N	Alguns campos não apresentam informações dificultando o entendimento por parte do usuário.
Caso o dado a entrar possua um formato particular, esse formato encontra-se descrito na tela?	N	Foram observados alguns campos específicos sem descrição do formato.
As unidades para a entrada ou apresentação de dados métricos ou financeiros encontram-se descritas próximas	N	Foi observada a falta de identificador para a relação de custo na tela de inserção de projetos

Figura 5. Documento gerado após a avaliação (adaptação ErgoList)

Esse tipo de avaliação qualitativa é de suma importância e não deve ser tratado como um mero artefato burocrático. Fornece subsídios para se corrigir os problemas inerentes ao sistema e serve como base para se iniciar um novo ciclo no processo. Para isto, deve ser elaborado de forma a eliminar as redundâncias relatadas no processo. O importante é descrever cada um dos problemas e o impacto que podem causar, sugerindo soluções adequadas.

O relatório serve também como documento para análise e consultas futuras. Essencialmente, deve propor as devidas mudanças, orientar as ações a serem tomadas e educar, servindo como um veículo de comunicação entre a equipe, a fim de tornar o processo eficaz.

Conclusão

Através deste artigo pode-se observar que a avaliação por *checklist* é uma ferramenta eficaz que promove a obtenção da qualidade do sistema, além de prover uma redução no custo do processo, possibilitando uma adequação das necessidades do sistema em desenvolvimento.

Serve também como uma alternativa a outros modelos de avaliação, e quando utilizada em conjunto com outro modelo, contribui para maximizar a qualidade da avaliação.

Esta abordagem é o enfoque do próximo artigo, onde será apresentada a conjunção da avaliação heurística com um modelo de *checklist*, de modo a evidenciar que esta associação promove a melhoria da qualidade neste processo. ●

Dê seu feedback sobre esta edição!

A Engenharia de Software Magazine tem que ser feita ao seu gosto.
Para isso, precisamos saber o que você, leitor, acha da revista!
Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/esmag/feedback

Referências

CYBIS, Walter; BETIOL, Adriana Holtz; FAUST, Richard. Ergonomia e Usabilidade – Conhecimentos, Métodos e Aplicações. 1 ed
São Paulo: Novatec, 2007.

FERREIRA, Kátia Gomes. Teste de Usabilidade
UFMG – Belo Horizonte, 2002

Laboratório de Utilizabilidade da Informática – UFSC.
<http://www.labiutil.inf.ufsc.br>.

LIMA, Paulo Sérgio Rodrigues. Tópicos de Interfaces Humano-Computador: Conceitos, Usabilidade e Avaliação de Interfaces
UEPA – Belém: 2007.

ORTH, Afonso Inácio, Interface Homem-Máquina
Porto Alegre: AIO, 2005.

PÁDUA, Clarindo Isaías Pereira da Silva e. Apostila de Engenharia de Usabilidade
UFMG – Belo Horizonte, 2007.

ROCHA, Heloísa Vieira da; BARANAUSKAS, Maria Cecília Calani, Design e Avaliação de Interfaces Humano-Computador
Campinas: NIED/UNICAMP, 2003.

SILVA, Elton J. da. Sistemas Interativos
UFOP – Ouro Preto, 2006.

SIQUEIRA, Eunice Gomes de - Estratégias e Padrões Para a Modelagem da Interface Humano-Computador de Sistemas Baseados na Arquitetura Softboard
INPE – São José dos Campos: 2001.

Heuristics for User Interface Design
http://www.useit.com/papers/heuristic/heuristic_list.html

Design Guidelines for the Web
http://www.usabilitynet.org/tools/webdesign.htm

Modelos de Maturidade na Gestão de Projetos

Modelos Existentes e uma Análise Comparativa

Cleyverson Pereira Costa

cleyversoncosta@gmail.com

Possui graduação em Ciência da Computação pela Faculdades Integradas Espírito-Santenses. Concluiu em 2007 o Programa de Residência em Engenharia de Software, com especialização em testes, pelo Centro de Informática da universidade Federal de Pernambuco (CIn-UFPE). Mestrando de Pós-Graduação do CIn-UFPE na área de Engenharia de Software, com foco em Modelos de Maturidade em Gerenciamento de Projetos e Gestão do Conhecimento. Experiência na área de testes, atuou como Eng. de Testes pelo Motorola Brazil Test Center.

Hermano Perrelli de Moura

hermano@cin.ufpe.br

Possui graduação em Engenharia Eletrônica pela Universidade Federal de Pernambuco, Mestrado em Informática pela Universidade Federal de Pernambuco e PhD in Computing Science pela University of Glasgow. É certificado PMP pelo Project Management Institute. Atualmente é Professor Adjunto e Vice-Diretor do Centro de Informática da Universidade Federal de Pernambuco. Atua na área de Computação, com ênfase em Engenharia de Software e Sistemas de Informação, desenvolvendo pesquisas e projetos, principalmente, nas seguintes linhas: processo de desenvolvimento de software, gestão de projetos, gestão de projetos de software, gestão da TI e empreendedorismo em informática. Tendo iniciado suas atividades na área de informática, desenvolve atividades de ensino e pesquisa, a nível de graduação e pós-graduação, além de projetos de cooperação com a indústria e de gestão universitária.

Um dos assuntos que vem ganhando cada vez mais espaço na agenda de gestores e desenvolvedores de software é a maturidade organizacional, seja ela no contexto da gestão de projetos, gestão do conhecimento ou desenvolvimento de software.

Segundo visões de diversos autores, um modelo de maturidade é uma estrutura conceitual, composta por processos bem estabelecidos, através dos quais uma organização desenvolve-se de modo planejado e sistêmico a fim de atingir um estado futuro desejado. A cada degrau alcançado nessa evolução, um modelo de maturidade reconhece e sinaliza o amadurecimento progressivo da organização.

De que se trata o artigo?

Neste artigo veremos o funcionamento dos principais modelos de maturidade em gerenciamento de projetos, além de uma análise comparativa, onde serão abordados aspectos como semelhanças, pontos fortes e limitações.

Para que serve?

Um modelo de maturidade funciona como um guia para a organização, de tal maneira que ela possa localizar onde está e como está, "espelhando-se" nele para, em seguida, realizar um plano para que ela possa chegar a algum ponto melhor do que o atual, na busca da excelência.

Em que situação o tema é útil?

A exigência de melhoria contínua dos projetos faz com que o gerenciamento da maturidade dos projetos de uma empresa seja efetivamente implantado. Entretanto, tentar atropelar e forçar os limites da organização pode trazer grandes prejuízos. Assim, a melhor estratégia é adotar uma política passo a passo, descobrindo inicialmente o nível de maturidade da empresa, criando definições claras para os envolvidos no projeto e realizando um monitoramento do desempenho geral dos projetos, de modo a ter uma visão global de como se estava antes e a nova realidade.

O nível de maturidade é uma maneira de prever o futuro desempenho de uma organização dentro de cada disciplina ou conjunto de disciplinas. A experiência mostra que as organizações funcionam melhor quando concentram seus esforços de melhoria de processos em um número controlado de áreas que exigem um esforço cada vez mais sofisticado à medida que a organização melhora (Kerzner, 2003). Os modelos procuram unificar uma mesma visão, tratando a evolução da maturidade como estágios de crescimento nos quais as organizações vão evoluindo e conquistando um maior grau de maturidade.

Segundo Prado (2008), as organizações modernas, de pequeno, médio ou grande porte, devem utilizar os conceitos e modelos de maturidade para acelerar o crescimento no que diz respeito às práticas, processos e definição de responsabilidades.

Sendo a escolha do modelo de maturidade mais adequado uma tarefa crítica e trabalhosa, este artigo intenta por facilitar esta decisão através da apresentação das características de funcionamento e do resultado de uma análise comparativa realizada entre os principais modelos de maturidade em gerenciamento de projestos descritos na literatura.

Antes de iniciar a apresentação dos modelos de maturidade em gerenciamento de projetos, faz-se necessário descrever algumas características de um outro modelo, o Capability Maturity Model (CMM). Esta necessidade deve-se ao fato de alguns dos modelos que serão abordados, possuírem sua estrutura de níveis de maturidade ou estrutura de avaliação, baseada ou semelhante a ele. O CMM possui cinco níveis de maturidade (Inicial, Repetível, Definido, Gerenciado e Otimizado), onde cada nível, com exceção do primeiro, pode ser decomposto em áreas-chave de processo, objetivos e práticas. O CMM já possui seu sucessor, o Capability Maturity Model Integration (CMMI) que também foi desenvolvido pelo SEI. Enquanto o CMM possui seu foco na área de software, o CMMI, entre outras diferenças, possui maior abrangência (Sodré, Notari, 2003).

Modelos de Maturidade em Gestão de Projetos

Desenvolveram-se nos últimos anos diversos modelos de maturidade específicos para o gerenciamento de projetos, modelos estes que seguem em geral os princípios dos modelos de maturidade de qualidade, contudo tendo seu foco direcionado para as práticas gerenciais. Os modelos atualmente mais referenciados na literatura são: o Organizational Project Management Maturity Model (OPM3) (PMI, 2008); o Project Management Maturity Model (PMMM) (Crawford, 2006); o Portfolio, Programme and Project Management Maturity Model (P3M3) (OGC, 2008); o Kerzner Project Management Maturity Model (KPMMM) (Kerzner, 2005); e o Modelo de Maturidade em Gerenciamento de Projetos (MMGP) (Prado, 2008). Cada modelo citado é descrito nas proximas subseções.

Além dos cinco modelos que serão abordados em maior profundidade, existem outras iniciativas, como: o PRINCE2 Maturity Model (P2MM) (OGC, 2006) e os modelos definidos e apresentados nos trabalhos de Voivedich e Jones (2001), Branco Júnior e Belchior (2001), Kwak e Ibbs (2002) e Andersen e Jessen (2003).

Organizational Project Management Maturity Model (OPM3)

O Organizational Project Management Maturity Model (OPM3) foi oficialmente anunciado em 2003 e tem como principal objetivo desenvolver um padrão global para o gerenciamento organizacional de projetos. Sua principal fonte de referência é o livro Organizational Project Management Maturity Model - Knowledge Foundation, cuja segunda edição foi lança no final de 2008 (PMI, 2008).

O uso do OPM3 deve ocorrer através da execução sistemática de cinco passos: (1) Preparar-se para a avaliação, (2) Realizar a avaliação (este passo pode ser repetido várias vezes até que a equipe de avaliação esteja satisfeita com a precisão dos resultados), (3) Planejar as melhorias, (4) Implantar as melhorias e (5) Repetir o processo (todos os passos citados). O modelo agrupa e explica estes passos conforme três grupos: (1) Conhecimento (o passo 1), (2) Avaliação (o passo 2) e (3) Melhoria (os passos 3, 4 e 5). No primeiro grupo é apresentado o conhecimento associado ao OPM3. No segundo grupo é explicado como proceder com a avaliação. No último grupo, discute-se como planejar e implantar as melhorias. Veja na **Figura 1** uma representação gráfica destes passos e grupos.

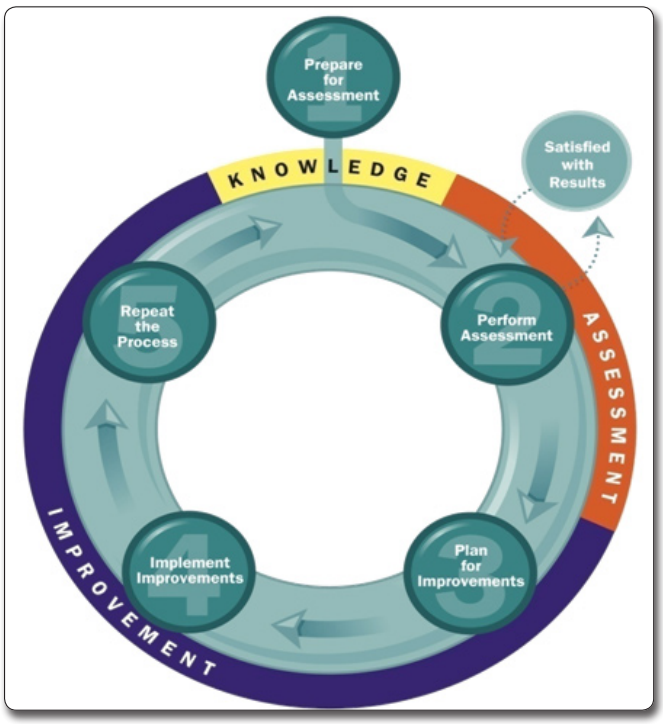


Figura 1. Passos para a Adoção do OPM3 (PMI, 2008)

Segundo o guia de referência do OPM3 (PMI, 2003), gerenciamento organizacional de projetos é a aplicação de conhecimento, habilidades, ferramentas e técnicas em atividades organizacionais e de projeto com o objetivo de atingir os objetivos da organização através de projetos, sendo a maturidade representada pelo nível de qualidade desta gerência nos domínios de Projeto, Programa e Portfólio.

Uma melhor prática para o OPM3 é a maneira, atualmente reconhecida pela indústria como ótima, para atingir um dado objetivo. Isto inclui a habilidade de entregar projetos de forma previsível, consistente e de implementar estratégias organizacionais com sucesso. Melhores práticas são atingidas desenvolvendo capacidades que podem ser observadas através de resultados mensuráveis.

Capacidade é uma competência específica que deve existir em uma organização de forma a executar processos de gerenciamento de projetos, ou seja, são passos incrementais que levam à realização de uma ou mais melhores práticas. No OPM3, cada melhor prática é composta por duas ou mais capacidades.

A existência de uma capacidade é demonstrada pela existência de um ou mais resultados correspondentes, podendo estes serem tangíveis ou intangíveis. Cada resultado deve estar associado a um indicador que permita à organização determinar, qualitativamente ou quantitativamente, se o resultado associado com certa capacidade, e em caso afirmativo, em qual nível se encontra. Veja na **Figura 2** a representação visual do relacionamento existente entre melhor prática, capacidade, resultado e indicador.

Figura 2. Melhor Prática X Capacidade X Resultado X Indicador (PMI, 2008)

O modelo ainda considera que pode haver dependências compartilhadas, como também dependências conjuntas. No primeiro caso, duas ou mais melhores práticas dependem da mesma capacidade. No segundo caso, uma melhor prática depende de uma capacidade que depende de uma capacidade de outra melhor prática, e esta última capacidade depende de uma capacidade da primeira melhor prática, como apresentado na **Figura 3**.

A progressão da maturidade no OPM3 é dada por uma escala de quatro níveis: padronizado, mensurável, controlado e melhoria contínua. Diferentemente de outros modelos de maturidade, há várias maneiras de observar a maturidade de uma organização; característica que faz o OPM3 ser muitas vezes referenciado como um modelo multidimensional.

Uma dimensão consiste em associar cada melhor prática com um nível de maturidade. Uma segunda dimensão consiste em associar cada domínio (Projeto, Programa e Portfólio) com um nível de maturidade. Outra dimensão consiste em associar cada melhor prática com os domínios considerados pelo OPM3. E uma quarta possibilidade é associar cada capacidade com os cinco grupos de processos definidos pelo PMBOK (PMI, 2004): Iniciação, Planejamento, Execução, Controle e Encerramento. A junção das três últimas possibilidades forma a visão tridimensional do OPM3 que pode ser vista na **Figura 4**.

Figura 3. Dependência entre Capacidades (PMI, 2008)

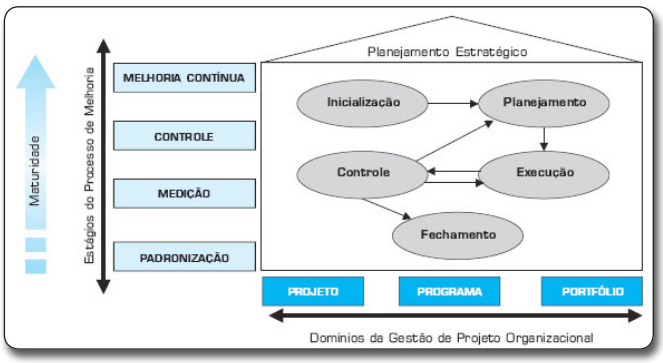


Figura 4. Visão Multidimensional do OPM3 (Zaguir, Martins, 2007)

Ainda é possível analisar a maturidade da organização de acordo com uma escala contínua que varia de 0% a 100% de maturidade. Claramente, o OPM3 possui uma estrutura complexa que é tida por alguns como necessária e flexível (Fahrenkrog, Abrams, *et al.*, 2003), enquanto que por outros é vista como desnecessariamente complexa e redundante (Hillson, 2003; Soler, 2005; Zaguir, Martins, 2007).

A avaliação do OPM3 pode ser realizada através de uma ferramenta on-line ou com um conjunto de ferramentas proprietárias do PMI, sendo as duas opções pagas. A primeira opção consiste em um questionário de 151 perguntas com respostas Sim ou Não que permitem identificar quais das mais de 600 melhores práticas a organização possui.

A segunda opção fornece mais detalhes, envolve a participação de um avaliador OPM3 certificado e requer um maior investimento por parte da organização. Esta opção oferece, inclusive, a concepção automática de um plano de melhoria formado pelas capacidades que a organização precisa desenvolver para alcançar as melhores práticas que esta considera prioritárias.

Project Management Maturity Model (PMMM)

O Project Management Maturity Model (PMMM) é um modelo de autoria da empresa PM Solutions, uma empresa de consultoria, treinamento e pesquisa em gerenciamento de

projetos fundada em 1996. A principal fonte de referência do modelo é o livro Project Management Maturity Model que está na sua segunda edição (2007).

O PMMM provê um arcabouço conceitual com o qual processos específicos de gerenciamento de projetos podem ser otimizados de forma a tornar mais eficiente a capacidade da organização gerir seus projetos (Crawford, 2006). O modelo é composto pelo relacionamento das nove áreas de conhecimento do PMBOK 2004 com os cinco níveis de maturidade do CMM (**Figura 5**). Por ser fortemente baseado no CMM, o modelo possui cinco níveis distintos de maturidade e para cada um examina as práticas organizacionais em função das nove áreas de conhecimento de gerenciamento de projetos.

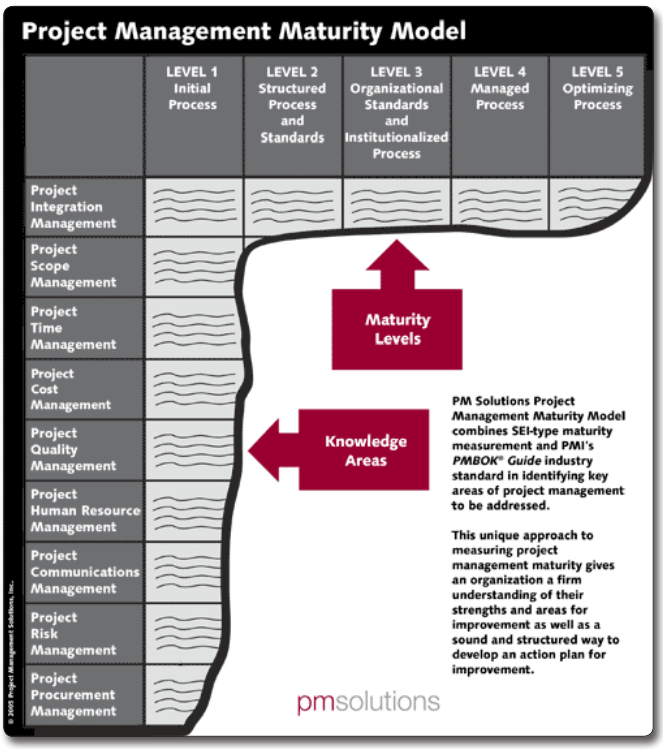


Figura 5. Visão Geral do PMMM (Crawford, 2006)

No Nível 1 (Processo Inicial), apesar de existirem processos de gerenciamento de projetos, não há práticas ou padrões estabelecidos. No Nível 2 (Processos Estruturados e Padrões), existem vários processos de gerenciamento de projetos dentro da organização, contudo eles não podem ser considerados um padrão organizacional. No Nível 3 (Padrões Organizacionais e Processo Institucionalizado), todos os processos de gerenciamento de projetos são efetivamente estabelecidos e utilizados como padrões organizacionais. No Nível 4 (Processo Gerenciado), projetos são gerenciados considerando o desempenho dos projetos passados, assim como tendo em vista o desempenho esperado para o futuro. No Nível 5 (Processo Otimizado), existe a preocupação com processos utilizados ativamente para melhorar as atividades de gerenciamento de projetos da organização.

Cada área de conhecimento do PMBOK é então detalhada para cada nível. Em particular, cada área é subdividida em componentes chaves. Estes componentes possuem certa correlação com os processos das respectivas áreas descritos no PMBOK. No PMMM são estes os componentes avaliados.

A empresa PM Solutions provê duas alternativas de avaliação que podem ser conduzidas para determinar a maturidade organizacional em gerenciamento de projetos. A primeira consiste de uma avaliação independente realizada por terceiros. A segunda, por sua vez, é uma avaliação realizada pela própria organização.

Após a avaliação, cada componente é situado em um dado nível. O nível da área de conhecimento é o menor nível comum a todos os seus componentes, como apresentado na **Figura 6**. O nível da organização é o menor nível comum a todas as áreas de conhecimento (veja um exemplo na **Figura 7**).

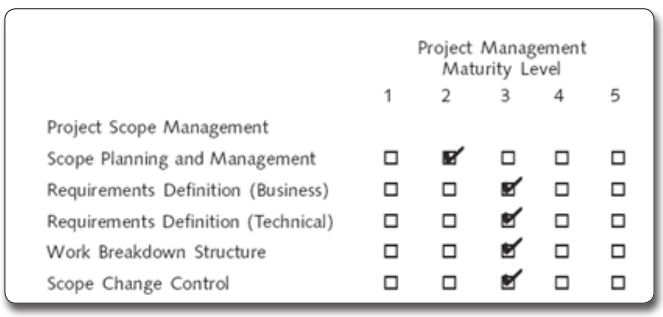


Figura 6. Nível 2 em Gerenciamento de Escopo (Crawford, 2006)

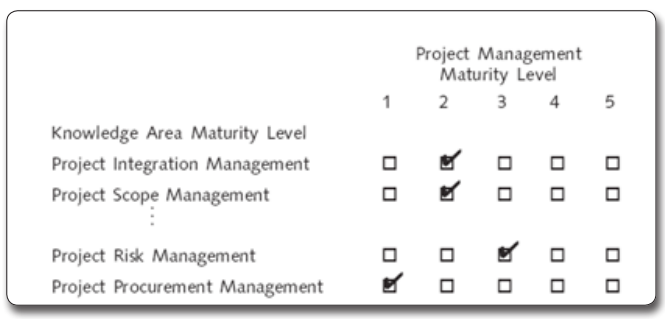


Figura 7. Organização Nível 1 (Crawford, 2006)

O modelo PMMM não incorpora diretamente elementos facilitadores para a construção de um plano de melhoria, porém, a empresa PM Solutions argumenta que este plano pode ser criado a partir da visão de especialistas no modelo e em gerenciamento de projetos. A empresa possui inclusive outra publicação que busca abordar o tema da melhoria (Appleby, Cabanis-Brewin, *et al.*, 2007).

Kerzner Project Management Maturity Model (KPMMM)

O Kerzner Project Management Maturity Model (KPMMM) é um modelo de autoria de Harold Kerzner. A principal fonte de referência do modelo é o livro Using the Project Management Maturity Model: Strategic Planning for Project Management que está na sua segunda edição (2005).

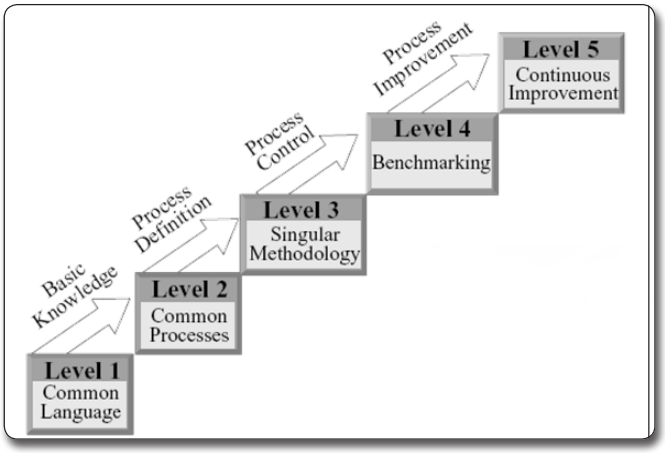


Figura 8. Níveis de Maturidade do KPMMM (Kerzner, 2005)

O objetivo do modelo é fornecer a base para a busca da excelência em gerenciamento de projetos. O KPMMM é composto por cinco níveis, onde cada um representa um nível diferente de maturidade em gerenciamento de projetos (**Figura 8**). Estes níveis são: (1) Linguagem comum, (2) Processos comuns, (3) Metodologia singular, (4) Benchmarking e (5) Melhoria contínua (Kerzner, 2005).

No Nível 1, é reconhecido a importância do gerenciamento de projetos e a necessidade de um conhecimento básico sobre a terminologia da área.

No Nível 2, a organização começa a definir e desenvolver processos comuns de gerenciamento de projetos.

No Nível 3, estes processos são combinados com outros processos organizacionais com o objetivo de criar uma metodologia singular centrada no gerenciamento de projetos.

No nível 4, a organização deve reconhecer que se ela deseja manter uma vantagem competitiva, a melhoria de processos através de benchmarking é essencial.

No Nível 5, os dados levantados pelo benchmarking são utilizados para decidir o que e como melhorar a metodologia singular definida no Nível 3. Ou seja, o último nível é responsável por fornecer feedback aos Níveis 3, aperfeiçoando a metodologia singular, e 4, direcionando as iniciativas de benchmaking.

Ao contrário de outros modelos de maturidade, o KPMMM assume a sobreposição de níveis de acordo com algumas regras. Normalmente, o nível de sobreposição baseia-se no nível de risco que a organização está disposta a aceitar. Por exemplo, é possível uma organização ainda não ter consolidado a terminologia em gerenciamento de projetos dentro da empresa, Nível 1, mas já ter iniciado a definição de processos de gerenciamento de projetos, Nível 2.

Outra diferença deste modelo em relação aos demais é que, apesar de ser orientado a projetos, admite a possibilidade de adaptação para avaliar o nível de maturidade do gerenciamento de operações. Contudo, o modelo não fornece guias de como esta adaptação pode ser feita.

Cada nível de maturidade possui um questionário associado capaz de conferir uma pontuação para a organização em relação ao nível em questão. Caso a empresa atinja o mínimo de pontuação especificado no modelo, conclui-se que o nível de sucesso foi atingido.

Portfolio, Programme & Project Management Maturity Model (P3M3)

O Portfolio, Programme & Project Management Maturity Model (P3M3) é um modelo de maturidade, originado como uma melhoria de um descontinuado modelo de maturidade em gerenciamento de projetos do OGC que por sua vez era baseado nos conceitos do CMM (SEI, 1991), modelo predecessor ao CMMI. O P3M3 também pode ser usado como meta-modelo para a construção sistemática de ferramentas, incluindo questionários (OGC, 2008).

O P3M3 descreve boas práticas nos domínios de portfólio, programa e projeto que devem ser seguidas. Cada boa prática está relacionada a uma área chave de processo. Os níveis do modelo indicam a maturidade da organização. Estes são estruturados hierarquicamente, permitindo que a organização melhore sua gestão de projetos de forma gradual. Os níveis propostos pelo modelo são (OGC, 2008):

- **Nível 1: Processos Iniciais (Initial Process):** a organização executa seus projetos informalmente, sem processos padronizados e sistemas para facilitar o acompanhamento;
- **Nível 2: Processos Repetíveis (Repeatable Process):** a organização assegura que cada projeto é executado com processos e procedimentos particulares, obedecendo a uma padronização mínima;
- **Nível 3: Processos Definidos (Defined Process):** a organização possui controle centralizado dos seus processos e consegue customizá-los de forma a atender uma característica específica de um projeto;
- **Nível 4: Processos Gerenciados (Managed Process):** a organização faz medições específicas nas características de gerenciamento de projetos e utiliza estes resultados para melhorar sua performance em gerenciamento, tornando resultados de projetos futuros mais previsíveis;
- **Nível 5: Processos Otimizados (Optimised Process):** a organização executa melhoria continua em seus processos de gerenciamento de projetos de forma pró-ativa, visando otimizá-los e gerar uma maior performance, diminuindo horas extras.

A avaliação do P3M3 pode ser conduzida de duas maneiras: a primeira consiste de uma auto-avaliação e a segunda de uma avaliação formal. A primeira pode ser realizada verificando quais atributos do modelo a organização satisfaz ou através de um questionário de auto-avaliação que pode ser baixado sem custos através do site do P3M3. A segunda avaliação é conduzida por empresas credenciadas ao APM Group (APMG) e confere um certificado.

O modelo P3M3 não auxilia explicitamente a criação de um plano de melhoria, contudo, pode ser desenvolvido a partir dos atributos genéricos e específicos que a organização precisa atender para evoluir do nível atual para o nível desejado de maturidade.

Modelo de Maturidade em Gerenciamento de Projetos (MMGP)

O Modelo de Maturidade em Gerenciamento de Projetos (MMGP) é de autoria de Darci Prado. O modelo é referenciado no livro Gerenciamento de Portfólios, Programas e Projetos nas Organizações, quarta edição, 2004. Entretanto, possui atualmente como fonte de referência atualizada o livro, do mesmo autor, Maturidade em Gerenciamento de Projetos (Prado, 2008).

O MMGP é baseado na experiência do autor e pode ser aplicado tanto a áreas isoladas da organização, como na organização como um todo. O modelo busca se diferenciar pela sua simplicidade. É composto por seis dimensões e cinco níveis de maturidade, como mostra a **Figura 9**.

Existe certo consenso entre os profissionais de GP (Prado, 2008; Kerzner, 2005) que um modelo de maturidade deve contemplar as seguintes áreas: Estratégia, Processos, Pessoas e Tecnologia. No caso do modelo MMGP, isso foi equacionado por meio de seis dimensões que se espalham pelos cinco níveis de maturidade em diversos momentos.

O modelo MMGP apresenta as seguintes dimensões, conforme **Figura 9**: (1) Conhecimentos de GP, (2) Metodologias, (3) Informatização, (4) Estrutura Organizacional, (5) Relacionamentos Humanos e (6) Alinhamento com Negócios da Organização.

A primeira dimensão compreende o conhecimento da terminologia e práticas de gerenciamento de projetos freqüentemente utilizadas por uma organização. A segunda dimensão considera que idealmente uma organização deve ter uma metodologia única de gerenciamento de projeto que possui pequenas variações de acordo com projetos específicos. A terceira dimensão preocupa-se com o apoio de ferramentas computadorizadas à metodologia única. A quarta dimensão pressupõe a criação de funções e papéis na estrutura organizacional da empresa com o objetivo de maximizar os resultados e minimizar os conflitos em projetos. Dentre estas funções e papéis destacam-se o escritório de projetos, o patrocinador dos projetos, entre outros. A quinta dimensão preocupa-se com a motivação das pessoas, com os relacionamentos entre os membros do projeto com o objetivo de evitar conflitos que afetam o projeto e a organização. A sexta e última dimensão tem o seu foco no alinhamento dos projetos com o negócio da organização.

Estas dimensões estão presentes nos cinco níveis de maturidade do modelo, porém, cada dimensão assume uma maior ou menor ênfase dependendo do nível. A primeira dimensão recebe uma maior ênfase no Nível 2 e cresce nos demais níveis. A segunda dimensão é mais intensa a partir do Nível 3 em diante. A terceira dimensão costuma receber mais foco a partir do Nível 3 também, assim como a quarta dimensão. A quinta e sexta dimensões recebem mais atenção a partir do Nível 4 em diante.

Já os cinco níveis de maturidade do modelo, de acordo com a **Figura 9**, são caracterizados conforme os seguintes aspectos:

- **Nível 1 – Inicial:** este nível apresenta um cenário que denota que a organização não possui esforços coordenados para realização de suas atividades. Assim, as atividades são geralmente executadas de maneira isolada, e na maioria das vezes, por meio de iniciativas individuais ou de poucos membros;
- **Nível 2 – Conhecido:** o segundo nível apresenta um cenário em que se realiza um esforço mais coordenado e começa a existir uma linguagem mais universalizada entre os membros, buscando realizarem atividades e ações coordenadas;
- **Nível 3 – Padronizado:** este nível é caracterizado por um cenário onde está implantado e se utiliza um modelo padronizado de gerenciamento. Há liderança e governança firmadas que desenvolvem a integração entre os membros. Existe uma base metodológica de trabalho e atividades conjuntas ocorrendo com freqüência e de forma harmoniosa e cooperativa entre os membros. É nesta fase que ocorrerá a ruptura que elevará a organização a um patamar diferenciado em

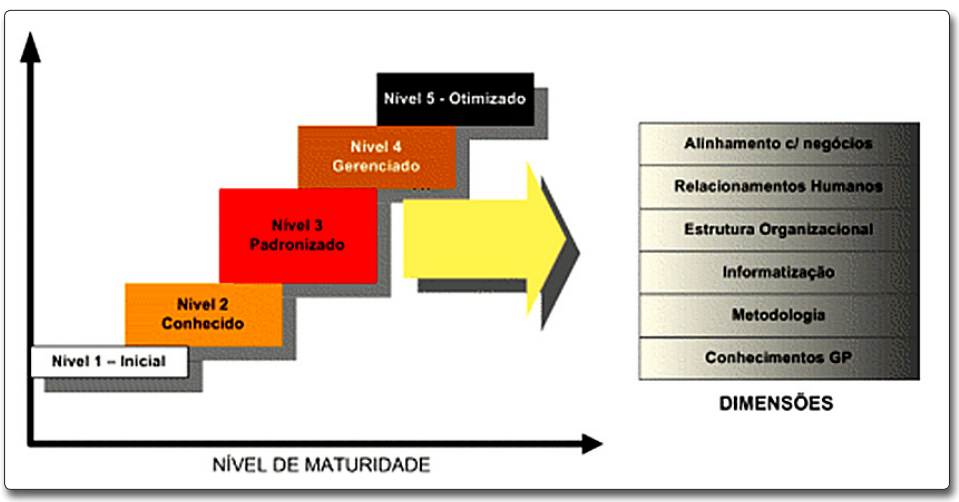


Figura 9. Dimensões e Níveis de Maturidade (Prado, 2008)

termos de maturidade (levando-o a ascender a níveis superiores de maturidade);

• **Nível 4 – Gerenciado:** neste momento existirão ações como o alinhamento das atividades com as estratégias da organização. Este nível preconiza um relacionamento humano sadio e austero entre os membros, abordando, principalmente o gerenciamento de pessoas e negociações (no estilo ganha-ganha);

• **Nível 5 – Otimizado:** este nível é caracterizado pela conquista da maturidade. Este momento apresenta um cenário em que se atinge a sabedoria. Todas as iniciativas, todo o trabalho e esforços ao longo dos níveis anteriores atingem o seu cume. Ao alcançar este patamar, pode-se inferir que a organização tornou-se uma referência em gerenciamento de projetos.

A avaliação da maturidade de acordo com o MMGP é feita a partir de um questionário com 40 perguntas e pode ser realizada considerando parte da organização (avaliação setorial), ou a organização como um todo. Um ponto interessante do modelo é ser utilizado anualmente em uma pesquisa feita com empresas brasileiras, sendo os resultados públicos e passíveis de serem utilizados para fins de benchmarking. Esta pesquisa por ser acessada em *http://www.maturityresearch.com*.

Análise Comparativa

Para decidir quais modelos de maturidade em gerenciamento de projetos são mais adequados para avaliar a maturidade em uma dada organização, é preciso entender as semelhanças e diferenças existente entre estes.

Portanto, esta seção compara os cinco modelos apresentados anteriormente (OPM3, PMMM, P3M3, KPMMM e MMGP) a partir de um conjunto de características identificadas pelos autores como relevantes, tendo sido extraídas por meio da revisão bibliográfica realizada. A consolidação dos resultados obtidos são apresentados nas **Tabelas 1**, **2** e **3**.

Do ponto de vista de custo de aquisição de material para estudo, o P3M3 se mostra a melhor escolha, uma vez que o guia de referência pode ser adiquirido gratuitamente no site da OGC. Quando se deseja executar uma avaliação de maturidade investigativa, este modelo é uma boa opção, entretanto, realizar uma avaliação baseada em questionário é penoso uma vez que todas as características do modelo terão de ser convertidas em perguntas, o que não é uma tarefa trivial, e possibilita deturpação da real intenção do modelo durante o desenvolvimento do questionário. Este modelo também destaca-se por possuir uma avaliação de certificação, no entanto não são achados muitos relatos de empresas certificadas.

O OMP3 se apresenta como sendo possívelmente o modelo de maior força no mercado mundial, uma vez que é mantido pela principal instituição voltada ao fomento da gestão de projetos no mundo, a PMI. Analisando suas características internas, o modelo pode ser considerado robusto, porém complexo, uma vez que conta com aproximadamente 600 melhores práticas e possui um programa interno para o auxílio no desenvolvimento do plano de melhoria. Este é também o modelo mais custoso do ponto de vista financeiro.

Ní-veis	CMM	Modelos de Maturidade de Gestão de Projetos Baseados no CMM				
		OPM3	PMMM	KPMMM	MMGP	P3M3
1	Inicial	Standardization	Initial Process	Common Language	Inicial	Initial Process
2	Repetível	Measurement	Structured Process and Standards	Common Processes	Conhecido	Repeatable Process
3	Definido	Control	Organizational Standard and Institutionalized	Singular Methodology	Padronizado	Defined Process
4	Gerenciado	Continuous Improvement	Managed Process	Benchmarking	Gerenciado	Managed Process
5	Otimizado	-	Optimizing Process	Continuous Improvement	Otimizado	Optimized Process

Tabela 1. Níveis de Maturidade dos Modelos de GP

	OPM3	PMMM	KPMMM	MMGP	P3M3
Domínios	1. Projeto 2. Programa 3. Portfólio	1. Projeto	1. Projeto	1. Projeto	1. Projeto 2. Programa 3. Portfólio
Dimensões	-	-	-	1. Conhecimento em Gestão de Projetos 2. Metodologia 3. Informatização 4. Estrutura Organizacional 5. Relacionamentos Humanos 6. Alinhamento com o Negócio da Organização	-
Grupos de Processo	1. Iniciação 2. Planejamento 3. Execução 4. Controle 5. Fechamento	-	-	-	-

Tabela 2. Domínios, Dimensões e Grupos de Processo

	OPM3	PMMM	KPMMM	MMGP	P3M3
Autoria	PMI	PM Solutions	Harold Kerzner	Darci Prado	OGC
Primeira Publicação	2003	2002	2001	2002	2003
Última Publicação	2008	2007	2005	2008	2008
Abrangência	Global	Global	Global	Nacional	Global
Guia Oficial (Livro)	Organizational Project Management Maturity Model (OPM3) – Knowledge Foundation	Project Management Maturity Model. 2nd Edition.	Using the Project Management Maturity Model. 2nd Edition.	Gerenciamento de Portfólios, Programas e Projetos nas Organizações	Portfolio, Programme & Project Management Maturity Model
Custo do Material de Apoio	Amazon: $37.98	Amazon: $56.66	Amazon: $68.00	Livraria Cultura: R$40.00	Site OGC: Gratuito
Sobreposição de Níveis	Não	Não	Sim	Não	Não
Instrumento de Avaliação	Software de Avaliação 1. OPM3 Versão para 1 usuário ($595.00 para membros PMI, $695.00 não membros) 2. OPM3 Versão multiusuários ($4,495.00 para até 15 usuários)	Software de Avaliação 1. Não disponível para avaliação individual, apenas para consultores oficiais da PM Solutions	Software de Avaliação 1. Versão online ou Questionário 1. Presente no livro	Software de Avaliação 1. Versão online (grátis) ou Questionário 1. Presente no livro	Software de Avaliação 1. Inexistente ou Guia de Práticas 1. Presente no livro
Plano de Melhoria Interno	Sim - Plano de melhoria é um dos elementos chave do modelo.	Não - Os gerentes e avaliadores podem trabalhar em conjunto no desenvolvimento do plano de melhoria.	Não - O modelo oferece algumas referências sobre como construir o plano de melhorias baseado nos resultados da avaliação.	Não - Os resultados da avaliação são base para criação do plano de melhoria, porém o modelo não indica como desenvolver.	Não - Os gerentes e avaliadores podem trabalhar em conjunto no desenvolvimento do plano de melhoria.

Tabela 3. Dados Gerais dos Modelos de Gestão de Projetos

Para uma organização fortemente projetizada, o modelo PMMM é uma alternativa interessante, pois é fortemente alinhado ao PMBOK e exige baixo investimento, entretanto, caso a organização não seja projetizada, este modelo não se apresenta como uma boa opção.

O KPMMM é dentre os modelos apresentados o mais antigo. O livro que é utilizado como referência do modelo tem um baixo custo de aquisição e possui um mecanismo de avaliação mais flexível, susceptível a adaptações. Porém, não há guias efetivos de como realizar estas adaptações e o processo de avaliação contém algumas lacunas como, por exemplo: como consolidar resultados de avaliações aplicadas a mais de um representante da mesma organização.

Por fim, o MMGP é, dentre os listados, o único modelo de maturidade em gerenciamento de projetos brasileiro. Possui um baixo custo de aquisição e o mecanismo de avaliação é consideravelmente simples. Há ainda dados sobre avaliações de empresas brasileiras a partir deste modelo o que facilita a realização de benchmarking.

Conclusões

Iniciamos este artigo apresentando a definição de maturidade e a necessidade e relevância da busca por esta característica por parte das organizacões que desejam possuir uma vantagem competitiva frente a seus concorrentes. Posteriormente foi discorrido sobre o funcionamento dos principais modelos de maturidade de gestão de projetos, apresentando ainda uma análise comparativa, abordando semelhanças, diferenças, pontos fortes e limitações. Ao observar as características dos modelos abordados, alguns aspectos são evidenciados, entre eles a inexistência de um modelo aberto, desenvolvido com a colaboração da comunidade, e que tenha o desenvolvimento do plano de melhoria integrado ao processo de avaliação. No Brasil, pode-se observar como fator agravante na dificuldade de aplicação de um modelo de maturidade, o fato dos principais guias de referência estarem em inglês. É bem verdade que existe uma opção brasileira, o MMGP, desenvolvido por Darci Prado, entretanto, alguns trabalhos colocam em cheque a efetividade deste modelo, uma vez que seu instrumento de avaliação está baseado em um conjunto de apenas 40 questões.

Diante das informações apresentadas, percebeu-se ainda que apesar de mais de uma década do início dos estudos sobre maturidade, especialmente na área de gerenciamento de projetos, pode-se considerar que os modelos existentes ainda estão em processo de aprimoramento, sofrendo revisões periódicas. Observou-se também que ainda não existe um modelo de maturidade de gerenciamento de projetos que seja referência mundial, como é o caso do CMMI para a área de software. ●

Referências

Andersen, E. S. and Jessen, S. A. "Project maturity in organisations," International Journal of Project Management (21), 2003, pp. 457 - 461.

Appleby, T., Cabanis-Brewin, J., Crawford, J. K., Cruz, F., Pennypacker, J. S., West, J. L. and White, K. R. J. Advancing Organizational Project Management Maturity, Center for Business Practices, Glen Mills, PA, USA, 2007.

Branco, Júnior, E. C. and Belchior, A. D. "Um Modelo para Avaliação da Qualidade da Gerência de Projetos de Software"'Proceedings do I Simpósio Brasileiro de Qualidade de Software', SBQS, Gramado, RS, Brasil, 2001, pp. 249-260.

Crawford, J. K. Project Management Maturity Model, Auerbach Publications, Pennsylvania, PA, USA, 2006.

Fahrenkrog, S., Abrams, F., Haeck, W. P. and Whelbourn, D. "Project Management Institute's Organizational Project Management Maturity Model (OPM3)"'PMI North American Congress', Project Management Institute, Maryland, MD, USA, 2003.

Hillson, D. "Assessing Organizational Project Management Capability," Journal of Facilities Management (2), 2003, pp. 298.

Kerzner, H. Using the Project Management Maturity Model: Strategic Planning for Project Management, Wiley, New Jersey, NJ, USA, 2005.

Kerzner, H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling, Wiley, New Jersey, NJ, USA, 2003.

Kwak, Y. and Ibbs, C. W. "Project Management Process Maturity Model (PMPM)," Journal of Management in Engineering (18), 2002, pp. 150-155.

OGC Portfolio, Programme and Project Management Maturity Model, Office of Government Commerce, UK, 2008.

OGC PRINCE2 Maturity Model, Office of Government Commerce, UK, 2006.

PMI Organizational Project Management Maturity Model (OPM3) - Knowledge Foundation (2nd Edition), Project Management Institute, Pennsylvania, PA, USA, 2008.

PMI Project Management Body of Knowledge (PMBOK Guide - 3rd Edition), Project Management Institute, Pennsylvania, PA, USA, 2004.

PMI Organizational Project Management Maturity Model (OPM3) - Knowledge Foundation (1st Edition), Project Management Institute, Pennsylvania, PA, USA, 2003.

Prado, D. Maturidade em Gerenciamento de Projetos, INDG Tecnologia e Serviços, Minas Gerais, MG, Brasil, 2008.

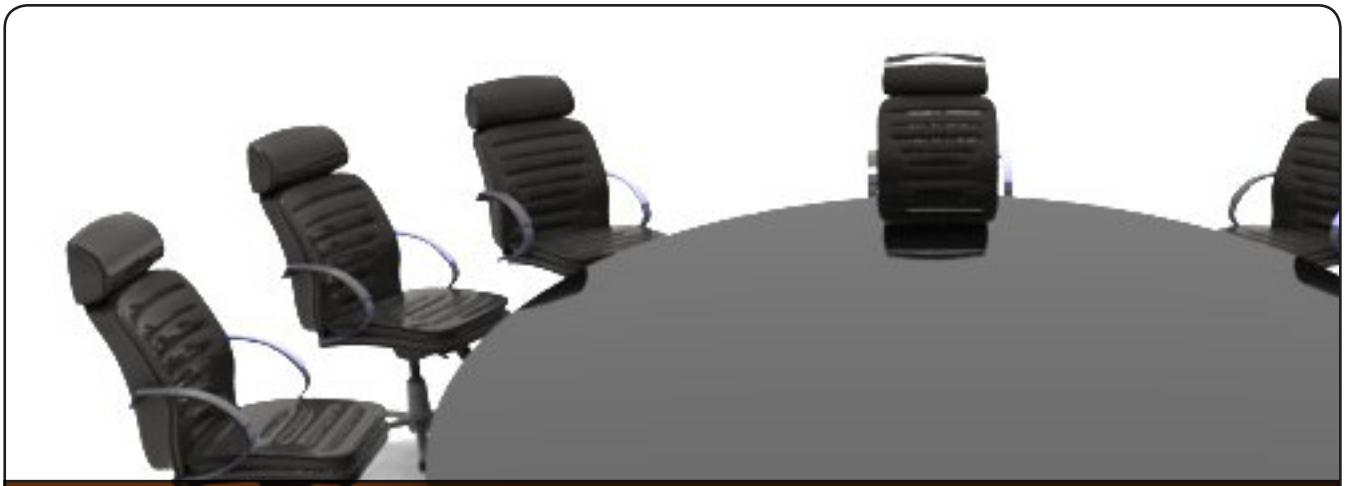
SEI Capability Maturity Model for Software, Version 1.0, Carnegie Mellon - Software Engineering Institute, Pittsburgh, PA, 1991.

Sodré, J. and Notari, D. L. "Proposta de Integração entre o PMBOK e o CMM para a Fase de Levantamento de Requisito", 2003.

Soler, A. M. "OPM3 - A contribuição do PMI para Maturidade em Gestão de Projetos"(2)'Revista Mundo PM', Revista Mundo PM, Curitiba, PR, Brasil, 2005.

Voivedich, B. and Jones, M. "Developing and Applying A Project Management Capability Maturity Model"'Proceedings of the Project Management Institute Annual Seminars & Symposium', Project Management Institute, Tennessee, TN, USA, 2001.

Zaguir, N. A. and Martins, M. R. "Revisão Crítica do OPM3: Um Estudo de Redundâncias," Revista Gestão Industrial (3), 2007, pp. 75-86.



Project Management Office

Conceito e evolução desta entidade tão atual

Antonio Luiz de O. Cavalcanti Júnior

antonio.cavalcanti@gmail.com

Especialista em Gestão de Projetos pela Universidade de Pernambuco e Bacharel em Ciências da Computação pela Universidade Católica de Pernambuco. Trabalha como Líder de Projetos no núcleo de P&D da Samsung no Centro de Informática da Universidade Federal de Pernambuco. Atua na área de TI há 10 anos e em Gestão de Projetos a 3 anos e atualmente foca seus estudos em desenvolvimento de um modelo de Project Management Office de baixo custo e sua adequação a dinâmica de projetos de TI.

Cristine Gusmão

cristine@dsc.upe.br

Professora do Departamento de Sistemas e Computação da Escola Politécnica da Universidade de Pernambuco (POLI - UPE), onde leciona várias disciplinas na graduação e pós-graduação (especialização e mestrado). Coordena o Mestrado em Engenharia da Computação e o PROMISE Project Management Improvements in Software Engineering - grupo de pesquisa na área de Engenharia de Software. È coordenadora do Curso de Bacharelado em Sistemas de Informação das Faculdades Integradas Barros Melo. Doutora e Mestre em Ciência da Computação pela Universidade Federal de Pernambuco. Graduada em Engenharia Elétrica - Eletrotécnica pela Universidade Federal de Pernambuco. Pesquisadora associada do Núcleo de Telesáude - NUTES - HC - UFPE.

De que se trata o artigo?

Estrutura organizacional de apoio ao gerenciamento de projetos e portfólio nas organizações – Escritório de Projetos.

Para que serve?

O escritório de projetos, a depender da maturidade organizacional, pode fornecer suporte metodológico, coaching aos gerentes da organização, além de sugerir uma administração centralizada do portfólio de projetos da organização permitindo um ponto de contato único e de apoio a decisão.

Em que situação o tema é útil?

Além de ser uma tendência na área de gerenciamento, a implantação de escritórios de projetos permite uma evolução, capacitação e maturidade da organização no alcance de sucesso em seus projetos.

Apesar da disciplina de Gerência de Projetos ser um tema atual e que mobiliza cada vez mais esforços para seu desenvolvimento, a humanidade realiza projetos desde seus primórdios. Frame [Frame 1995] relata que caçadas organizadas pelos nossos ancestrais pré-históricos foram projetos, bem como a construção das Pirâmides Egípcias, da Grande Muralha da China e mais recentemente o Projeto Manhattan, que construiu a primeira bomba atômica, ou o Projeto Apollo, que permitiu ao homem pisar em solo lunar. Ainda podemos considerar um Programa as conquistas de Alexandre "O Grande", onde cada cidade a ser conquistada era um projeto que contribuía para um único objetivo estratégico, expandir as fronteiras de Alexandria.

Ainda de acordo com Frame [Frame 1995], todos executamos projetos em nosso dia-a-dia, a maioria deles de maneira inconsciente e isso vem sendo feito rotineiramente há cerca de dez mil anos. Frame [Frame 1995] e Codas [Codas 1987] relatam também a história recente do Gerenciamento de Projetos, quando este passa a ser visto como uma disciplina, durante a Segunda Grande Guerra, e inclusive da primeira concepção de *Project Management Office* (PMO), fundado pela Marinha Estadunidense dirigido, primeiramente, para implantação de plantas industriais e posteriormente a projetos militares ligados à corrida espacial.

Hoje o gerenciamento de projetos é uma das disciplinas que mais crescem em todas as áreas da indústria mundial, segundo Rad & Raghavan [Rad & Raghavan 2000], e seu objetivo principal é manter uma relação lógica e eficiente entre o escopo, tempo, custo e qualidade do produto do projeto, ou seja seus objetivos. Apesar disso, os autores enfatizam um índice muito alto de falhas em projetos.

Na visão de Crawford [Crawford 2000] os motivos para os altos índices de falhas e aparente ineficácia de muitos projetos convergem para um ponto principal, a falta de processos adequados e padronizados de gerenciamento. Segundo o autor, a falta de procedimentos, metodologia e padrões é o fator principal para o fracasso de projetos, contribuindo para isso um baixo nível de treinamento específico dos gerentes de projetos.

Dai & Wells [Dai & Wells 2004] demonstram em seu estudo a correlação de algumas práticas de gerenciamento de projetos com resultados reais, complementando a visão de Crawford [Crawford 2000] que a gestão adequada do conhecimento gerado pelos projetos é também um importante fator de sucesso. A organização tem que conhecer bem seus projetos para garantir-lhes o acesso privilegiado a recursos de forma a produzir os melhores resultados possíveis.

No complexo contexto da atual indústria fica claro que produzir de forma mais eficiente, com maior capacidade em coordenar esforços e melhor controle sobre as atividades organizacionais, é um fator determinante para o sucesso do empreendimento. Permitindo, inclusive, respostas mais rápidas aos estímulos externos de mercado. Esse é o foco da disciplina de Gerência de Projetos e de seu desenvolvimento: eficiência produtiva com planejamento, coordenação, controle e gestão de mudanças.

Nesse cenário, o PMO surge como uma estrutura organizacional responsável por apoiar o gerenciamento de projetos nas organizações fornecendo a infra-estrutura necessária para um gerenciamento corporativo de projetos.

No decorrer desse artigo, mostraremos a evolução do PMO, os tipos já propostos e os aceitos na atualidade com suas principais características.

O Project Management Office

Ao longo dos anos diversas definições do que vem a ser um PMO foram propostas. Encontramos os mais variados entendimentos como, por exemplo, em Block [Block 1997] que afirma "PMO pode ser qualquer coisa que a organização queira que ele seja". Restringindo um pouco a amplitude da definição, Barcaui [Barcaui 2001] coloca que o principal ponto é que o PMO é constituído para servir as necessidades de gerenciamento de empreendimentos, o que é reafirmado por Duggall [Duggall 2001] onde define PMO como uma estrutura formalizada que direcionada ao suporte da comunidade de gerência de projetos da organização. Crawford [Crawford 2000] o coloca como um provedor de serviços e processos completos para gerenciamento de projetos, e Kate [Kate 2001] o vê como uma unidade de negócio focada na eficiência do gerenciamento de projetos.

Independente das diferenças e amplitude de responsabilidades que cada autor defende que o PMO deva ter, de forma geral, a academia e as empresas enxergam no PMO uma entidade que tem como principal objetivo alavancar a qualidade da gerência de projetos dentro das organizações. Nesse contexto, a definição do PMBoK [PMBoK 2009] (ler **Nota 1**) parece ser consoante com o entendimento atual do que vem a ser um Escritório de Gerência de Projetos:

> *Um corpo ou entidade organizacional à qual são atribuídas várias responsabilidades relacionadas ao gerenciamento centralizado e coordenado dos projetos sob seu domínio. As responsabilidades de um PMO podem variar desde o fornecimento de funções de suporte ao gerenciamento de projetos até o gerenciamento direto de um projeto.*

Nota do DevMan 1

PMBoK

O Project Management Body of Knowledge (PMBoK) é um padrão aprovado pela ANSI, e reconhecido pelo IEEE através do padrão 1490-1998. Ele descreve um conjunto de práticas consolidadas sobre gerenciamento de projetos que podem ser aplicadas em qualquer tipo de projetos (incluindo processos de desenvolvimento de software).

O PMBoK define projeto como "um empreendimento temporário cujo objetivo é criar um produto, serviço ou resultado distinto e único".

Um projeto é temporário, pois apresenta datas de início e término definidas. O término do projeto pode ser determinado por vários motivos, por exemplo: cumprimento dos objetivos determinados, abandono do mesmo. Cada projeto produz um produto único, mesmo havendo semelhanças entre diversos projetos, cada projeto possui metas, diretrizes, e envolvidos distintos. Além disso, um projeto tem como característica a elaboração progressiva. Na elaboração progressiva, a especificação do projeto vai sendo refinada ao longo do ciclo de vida do mesmo.

Para o PMBoK, o gerenciamento de projeto é "a aplicação de conhecimento, habilidades, ferramentas e técnicas atividades do projeto a fim de atender aos seus requisitos".

O ciclo de vida de um projeto é dividido em diversas fases para melhorar o controle e o gerenciamento do projeto. Cada fase é marcada pela conclusão de um ou mais produtos e cada uma delas possui um ponto de revisão. O ponto de revisão tem como objetivo determinar a continuidade ou não do projeto, e a detecção e correção de defeitos.

Tipicamente, um projeto tem uma fase inicial, onde é definido o escopo do projeto. Uma ou mais fases intermediárias, onde o alvo do projeto é planejado e realizado. E uma fase final, onde o produto é aceito e entregue para o cliente.

Contudo, é evidente que nenhuma definição pode cobrir toda gama de conceitos que uma entidade tão abrangente como o PMO pode ter. Elas servem apenas para nortear as atribuições básicas dessa entidade. Vemos muitos autores, como por exemplo, Dinsmore [Dinsmore 1999], Kate [Kate 2001], Crawford [Crawford 2001], Pellegrinelli & Garagna [Pellegrinelli & Garagna 2008], defendendo esse ponto de vista quando propõem seus tipos de PMO e deixam claro que normalmente o que se encontra no mundo real são modelos híbridos, derivados dos tipos básicos propostos e que agregam em maior ou menor grau as atribuições e responsabilidades dos outros tipos.

Voltamos então, de maneira menos radical, ao conceito dado por Block [Block 1997], o PMO pode ser o que a empresa que o implementa deseje ou necessite; que ele seja, desde que seja uma entidade focada na melhoria da gerência de seus projetos, sendo ela aplicada a qualquer um dos níveis organizacionais, desde o operacional ao estratégico.

Evolução da entidade PMO

As primeiras entidades reconhecidas como PMO tomaram forma entre a metade da década de 1950 e início da década de 1960. Esses escritórios eram dedicados a grandes e complexos empreendimentos, principalmente militares, aeroespaciais e de construção civil. Eram chamados, apesar de sua função ser de controle, de Project Support Office (Escritórios de Suporte a Projetos) e se desenvolveram com base em técnicas de controle e monitoramento de tempo.

No início da década de 1970, a popularização dos computadores e softwares de gerenciamento de projetos permitiu a disseminação dessa entidade nas organizações, adentrando cada vez mais em outras áreas de negócio. O desenvolvimento de softwares focados em gerência de tempo diminuiu a necessidade das organizações terem diversos especialistas para esse fim, ficando esses recursos humanos alocados como consultores nesses escritórios de suporte a projetos.

Na década de 1980 os softwares de gerenciamentos de projetos ficaram acessíveis a utilização de não especialistas. Segundo Murphy [Murphy 1997], *apud* Gonzalez & Rodrigues [Gonzalez & Rodrigues 2002], essa popularização permitiu também que pessoas não qualificadas passassem a usar essas ferramentas de forma cosmética, muitas vezes produzindo gráficos, de alta qualidade para época, que não refletiam a real situação dos projetos.

Nos anos 1990, o papel do PMO mudou drasticamente. O grande índice de fracassos em projetos na década anterior alertou a necessidade de desenvolvimento da disciplina de gerência de projetos e sua disseminação nos vários níveis das organizações. Com isso, o PMO ganhou importância e status nas organizações, principalmente como uma entidade concentradora de conhecimento de gerência de projetos e escalando os níveis hierárquicos das mesmas. Perto da virada do século, o PMO já era visto como órgãos de extrema importância estratégica que agregava cada dia mais funções, inclusive a de gerência do portfólio das organizações. Chegando nesse ponto, temos uma entidade que atravessa vertical e horizontalmente a organização, trabalhando desde o nível tático/operacional, dentro de projetos garantido o atendimento de seus objetivos, até o nível gerencial e estratégico controlando o andamento de múltiplos projetos e programas, bem como selecionando o portfólio que melhor atenda aos objetivos estratégicos da organização. A **Figura 1** resume essas três décadas de evolução.

Em meados do século XXI, começam a surgir propostas de modelos evolutivos de PMO. Os tipos definidos nessa época sugerem que o PMO dentro das organizações comece como uma pequena célula que tem como responsabilidade a disseminação da disciplina de gerência de projetos e definição de uma nomenclatura e metodologia padrões a serem utilizadas, e uma vez atingidos esses objetivos, evolua para

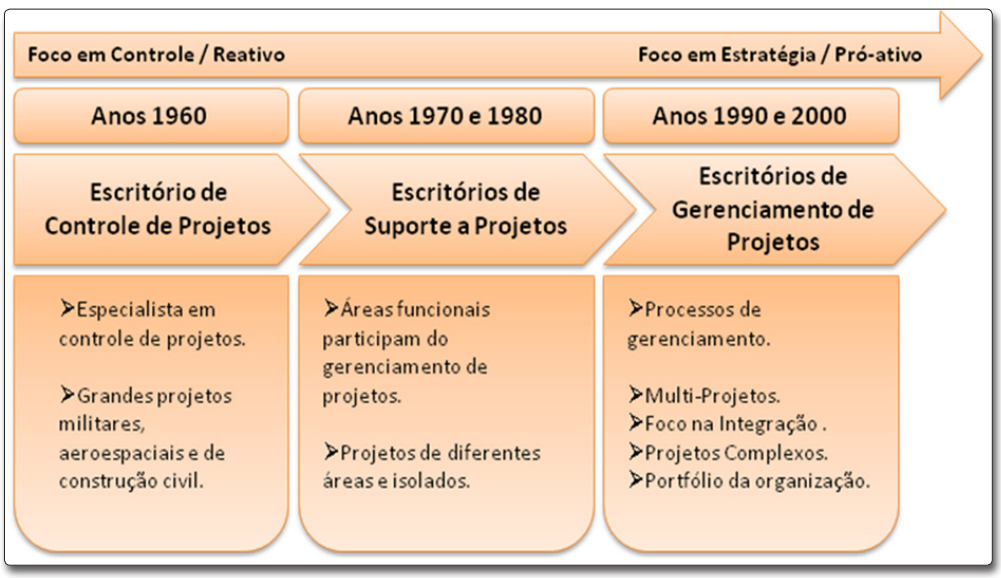


Figura 1. Evolução dos Escritórios de Projetos. Adaptado [Gonzalez & Rodrigues 2002]

uma entidade que passa a prover serviços mais abrangentes, como por exemplo, *coaching* e *mentoring* em projetos e programas. Por fim, sobe ao nível estratégico, primeiro definindo e reportando indicadores de desempenho dos projetos e posteriormente gerenciando o portfólio da organização. Essa abordagem pode ser claramente vista já em Dinsmore [Dinsmore 1999] e em Dugall [Dugall 2001], porém somente em Hill [Hill 2004] foi proposto um modelo que prega explicitamente essa evolução, o *Competency Continuum* onde o PMO, em seu gradual amadurecimento, atinge níveis de competência e responsabilidades. O trabalho de Hill [Hill 2004] resultou em um livro, última edição de 2008 [Hill 2008], que traz uma explanação detalhada sobre os níveis de competências atingidas pelo PMO e o papel, abrangência e responsabilidade do mesmo em cada um dos níveis de competência. Ainda se apresenta como um guia para organizações que desejam implantar um PMO e evoluí-lo seguindo o modelo do *Competency Continuum*.

Tipos de PMO já propostos

Como já foi dito, somente na década de 1990, o PMO começa a tomar a forma que conhecemos hoje, e um marco importante dessa conceituação aconteceu em 1997 no encontro do *Fortune 500 Project Management Forum*. O *Fortune 500 Project Management Forum* teve seu primeiro encontro em 1994 e é formado pelas 500 empresas - públicas e privadas - com maior receita, listadas pela publicação *Fortune Magazine* [Dinsmore 1999]. Dessas 500 empresas, algumas gigantes participam ativamente em todos os encontros, como por exemplo: Northwest Mutual Life, FedEx, Americam Airlines, Citybank, Sprint, AT&T, IBM, Allied Signal, Eli Lilly, Fujitsu, Intel, Bell Atlantic, Nynex, General Motors, dentre outras. O mapeamento dos tipos de PMO que se segue inicia exatamente baseado em um trabalho que foi fruto desse encontro de 1997 e em trabalhos paralelos que ocorreram após esse ano. A seguir trataremos dos modelos propostos nos trabalhos de Dinsmore [Dinsmore 1999], Kate [Kate 2001], Casey & Peck [Casey & Peck 2001], Crawford [Crawford 2001], Kerzner [Kerzner 2005] e Hill [Hill 2008], esse último sendo apontado como a tendência de adoção pelas organizações atuais. Apesar de existirem uma infinidade de variações, os trabalhos desses autores serviram como base para a maioria dessas variações.

DISMORE (1999)

Em 1997, durante o *Fortune 500 Project Management Forum*, houve um esforço para se chegar a um consenso de definição dos tipos de PMO existentes nas organizações da época, suas características e responsabilidades e unificação da nomenclatura a ser utilizada para cada tipo. Nesse evento foram definidos quatro tipos fundamentais de PMO [Dinsmore 1999]:

- ***Equipe autônoma de gerência de projetos***

Não podemos definir como um PMO propriamente dito. Na verdade, essa entidade nada mais é que a própria equipe de gerência do projeto. Todas as responsabilidades das atividades de gerência ficam a cargo do time de gerenciamento. Ainda, todo conhecimento prévio que será utilizado na realização do projeto é a experiência dos membros do time de gerenciamento e a responsabilidade sobre o sucesso do projeto também é do time de gerenciamento. O conhecimento e ativos gerados pelo projeto ficam internos aos projetos. Dinsmore [Dinsmore 1999] define esse tipo de PMO como o mais elementar e de onde os outros tipos direta, ou indiretamente, emergem. Esse tipo de escritório está ilustrado na **Figura 2**.

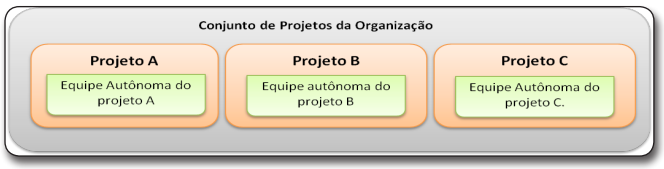


Figura 2. Equipe Autônoma de Gerência de Projetos

- ***Escritório de Suporte a Projetos***

Seu objetivo é fornecer suporte técnico e administrativo, ferramentas e serviços aos vários projetos da organização. Auxilia diretamente o Gerente de Projetos durante todo o ciclo de vida do projeto. Em alguns casos pode até alocar membros do PMO para trabalharem internamente nos projetos, de forma temporária ou não. Esse tipo de PMO já caracteriza um esforço para criar uma entidade independente que tem como prioridade garantir o sucesso individual de cada projeto da organização. O conhecimento e ativos gerados pelo projeto também ficam internos aos projetos. Este tipo de PMO está ilustrado na **Figura 3**.

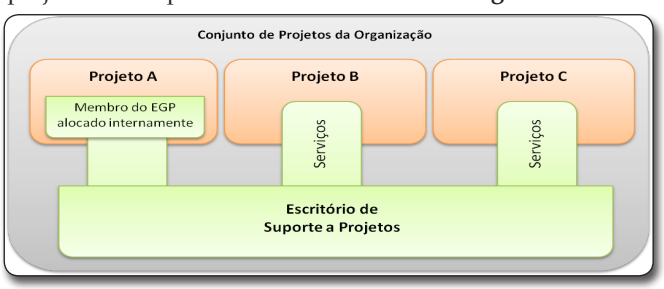


Figura 3. Escritório de Suporte a Projetos.

- ***Centro de Excelência em Gerenciamento de Projetos***

Esse tipo de PMO tem como responsabilidade a disseminação das práticas de gerência de Projetos na organização. Ele é visto como um centro concentrador de experiência e conhecimento sobre gerência de projetos e lições aprendidas. Sendo assim, é responsável pela devida organização e disponibilização de todo conhecimento gerado pelos projetos bem como seus ativos. Ainda, busca inovações para área de gerência de projetos no ambiente externo, academia, comunidade de gerência de projetos, etc., e as adéqua à realidade organizacional. Esse tipo de PMO, segundo Dinsmore [Dinsmore 1999] não se envolve em nenhuma atividade direta dos projetos, funcionando como uma entidade de influência indireta para o sucesso dos projetos, já que fornece informação e qualificação aos gerentes de cada projeto. Este tipo de PMO está ilustrado na **Figura 4**.

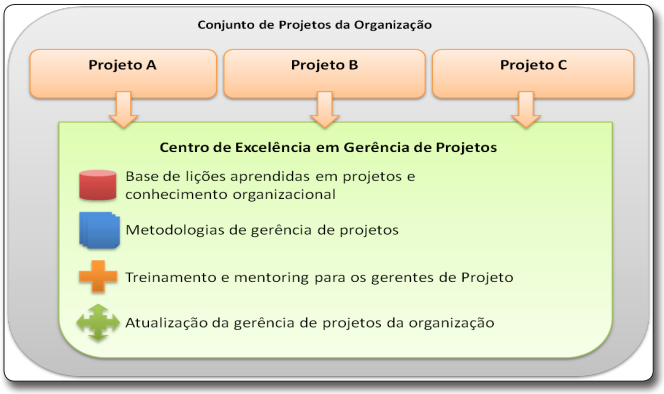


Figura 4. Centro de Excelência em Gerenciamento de Projetos.

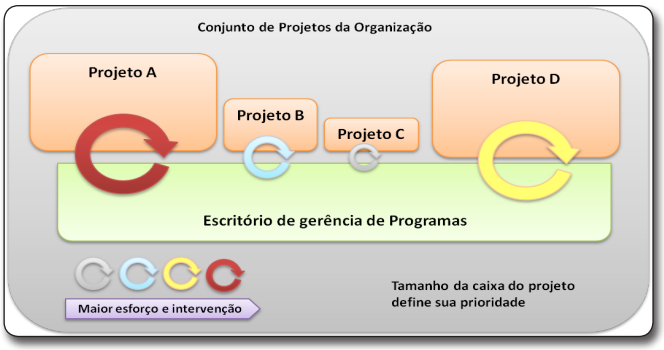


Figura 5. Escritório de Gerência de Programas.

• ***Escritório de Gerência de Programas***

Este tipo de PMO funciona como um centro gestor dos vários projetos da organização. É considerado como a gerência dos gerentes de projetos. Apesar de não participar da escolha do portfólio da organização é de sua responsabilidade concentrar esforços para garantir o sucesso dos projetos prioritários para estratégia da organização acompanhando seus desempenhos, ajustando possíveis desvios e garantindo que os recursos necessários serão priorizados para os projetos mais importantes. Esse PMO trabalha ativamente na gestão dos projetos. Este tipo de PMO está ilustrado na **Figura 5**.

• ***Escritório Chefe de Projetos***

Poderíamos, nos termos atuais, chamar essa definição de PMO feita por Dinsmore de *Escritório de Gerência de Portfólio*. A principal atribuição desse PMO é, junto com o alto poder executivo da organização, fazer a seleção e gerenciamento do conjunto de projetos que irão compor o Portfólio da organização. A seleção é feita com base na escolha de um conjunto de projetos que facilitará atingir os objetivos estratégicos da organização. A gerência desse conjunto não interfere diretamente no gerenciamento dos projetos individuais, mas sim em seus ciclos de vida. Acelerando, desacelerando, criando novos projetos ou até mesmo cancelando projetos existentes, sempre tendo como foco os objetivos da organização como um todo, colocando em segundo plano os objetivos individuais de cada projeto. Este tipo de PMO está ilustrado na **Figura 6**.

Dinsmore [Dinsmore 1999] deixa claro que os tipos de PMO definidos são apenas uma base para que as organizações escolham o que melhor atendem as suas necessidades, e que os tipos são evolutivos e normalmente híbridos, ou seja, tendem com sua maturação a agregar mais responsabilidades, tornando-se conseqüentemente outro mais abrangente, e que no mundo real misturam responsabilidades de cada um dos outros tipos com maior ou menor grau, se adequando aos objetivos da organização.

KATE (2001)

Quando define seus tipos de PMO - na verdade chama de estágios, Kate [Kate 2001] propõe um modelo evolutivo simples, ou seja, toda organização deve evoluir o seu PMO a partir do modelo mais simples, o *Escritório de Suporte a Projetos*, e uma vez que esse PMO esteja trabalhando com competência todas as suas responsabilidades é hora de evoluir para o modelo seguinte, e assim sucessivamente, até o PMO atingir o grau máximo de maturidade, onde passa a ser denominado *Escritório de Governança de Projetos e Programas*. Nesse estágio, Kate [Kate 2001] considera que a organização atingiu a competência de *Enterprise Project*

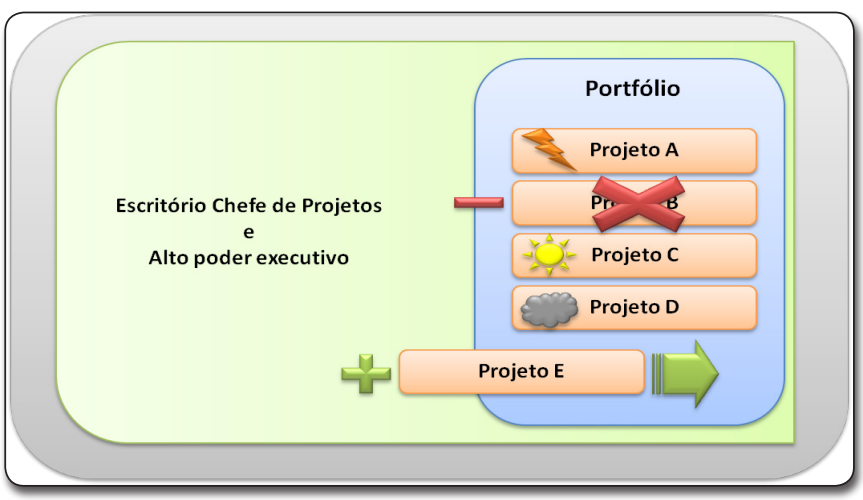


Figura 6. Escritório Chefe de Projetos.

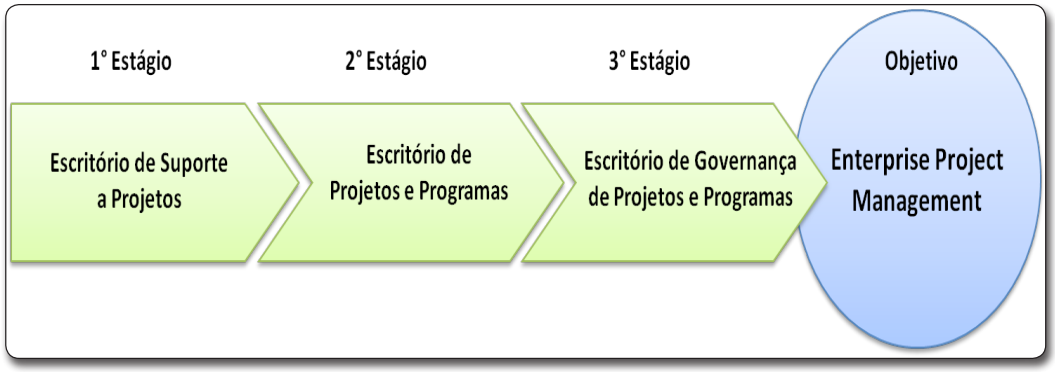


Figura 7. Modelo Evolutivo de Kate. Adaptado de [Kate 2001]

Management, ou seja, a gerência de projetos passa a ter uma abrangência em todos os níveis hierárquicos da organização e fundamental importância estratégica. A **Figura 7** ilustra o modelo proposto por Kate [Kate 2001].

Os modelos propostos são:

• ***Escritório de Suporte a Projetos***

Responsável por eliminar os pontos de falha de comunicação entre projetos e programas e agir como agente facilitador da comunicação entre os Gerentes de Projetos e de Programas com a organização. Ainda, identificar e eliminar a duplicação de esforço e conflitos nos projetos e programas. E, por fim, dar suporte aos gerentes de projetos e programas em relação às melhores práticas em gestão de projetos servindo como uma entidade guia e concentradora de *expertise*.

• ***Escritório de Projetos e Programas***

Além de assumir todas as responsabilidades de um *Escritório de Suporte a projetos*, deve coletar, analisar e reportar os problemas e status das atividades dos projetos e programas. Ainda provê um relatório sintético do status dos projetos aos clientes e poder executivo da organização e promover *coaching* e *mentoring* aos gerentes de projetos e programas baseado em uma metodologia mais estruturada e consolidada. Por fim, auditar os projetos e programas.

• ***Escritório de Governança de Projetos e Programas***

Acumula todas as responsabilidades do *Escritório de Suporte a Projetos* e do *Escritório de Projetos e Programas* aprimorando essas responsabilidades até ser capaz de governar todos os aspectos da gerência de projetos e programas em todos os projetos da organização. Nesse estágio, a metodologia precisa já ser um ativo organizacional e estar publicada em lugar acessível a todos os gerentes de projetos e programas. Esse tipo de PMO provê as ferramentas e técnicas a serem usadas pelos gerentes e constrói um *pool* de recursos em gerência de projetos, sendo a alocação de qualquer gerente em um projeto de responsabilidade do PMO. Ainda, deve garantir que todos os projetos estejam alinhados com os objetivos estratégicos da organização, porém Kate [Kate 2001] deixa claro que, em qualquer um dos níveis, o PMO só se envolve nos projetos a partir de sua iniciação, ou seja, a montagem do portfólio e todo o processo necessário para estudo de viabilidade dos projetos até sua aprovação não são de responsabilidade de nenhum de seus tipos de PMO.

CASEY & PECK (2001)

Casey & Peck [Casey & Peck 2001] partem do pressuposto de que não existe um único tipo de PMO que atenda a todas as necessidades e que se deve fugir de um modelo padrão que pode acabar operando como qualquer outro departamento funcional. Diferentes tipos de PMO resolvem diferentes problemas. A escolha do modelo deve levar em conta o estágio de maturidade do gerenciamento de projetos na organização. Os autores descrevem três tipos de PMO, ilustrados na **Figura 8**. Esses tipos são respostas a problemas que a organização enfrenta.

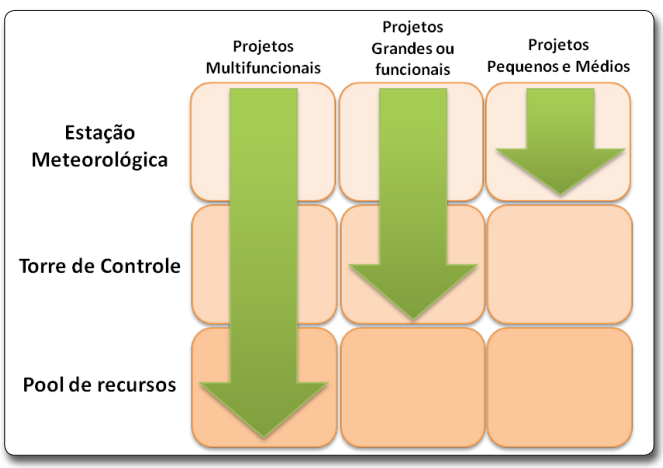


Figura 8. Tipos de PMO de Casey & Peck. Adaptado de [Casey & Peck 2001]

• ***PMO Nível 1 – Estação Meteorológica***

Quando o problema da empresa é a confusão causada por diferentes tipos de relatórios elaborados por diferentes gerentes de projetos, com jargões variados, a solução proposta é Estação Meteorológica. Este tipo de PMO apenas reporta o andamento dos projetos, mas não tenta influenciá-los. Responde questões tais como: como está o nosso progresso? Quanto já realizamos do nosso orçamento até aqui? Quais são os nossos riscos? Este tipo de PMO também pode ser responsável por manter uma

base de dados com documentos históricos de projetos e lições aprendidas.

• ***PMO Nível 2 – Torre de Controle***

Quando o problema é treinamento de pessoal, metodologias caras e pouco utilizadas, altos executivos com pouca compreensão ou visão equivocada sobre gerenciamento de projetos, lições aprendidas não utilizadas em novos projetos, trocas constantes de técnicas e ferramentas, esse tipo de PMO é proposto como solução. Sua responsabilidade é estabelecer uma metodologia de gerenciamento de projetos, gerenciar os riscos dos mesmos, aplicar um programa de *coaching* e *mentoring*, garantir a utilização da base de lições aprendidas e o devido uso e aprimoramento da metodologia definida. Esse tipo de PMO influencia as decisões e rumos do projeto, porém não tem poder para tomar as decisões ou mudar rumos por si só.

• ***PMO Nível 3 – Repositório de Recursos***

Organizações cujo negócio é fazer projetos necessitam estar permanentemente atentas à capacitação de seu pessoal em gerenciamento de projetos. Em geral, a pessoa que contrata e lida com os gerentes de projetos sabem muito pouco sobre a função. Entretanto, é fundamental para a empresa que elas sejam bem selecionadas, bem treinadas e que permaneçam na organização. A solução, neste caso, é o *Pool* de Recursos. Um *Pool* de Recursos pode oferecer um conjunto de gerentes de projetos com as habilidades necessárias para gerenciar os diferentes tipos de projetos para os quais forem designados, bem como uma supervisão para garantir que estas habilidades serão efetivamente aplicadas. O gerente do *pool* deve ser o responsável por designar os gerentes aos respectivos projetos e o *pool* é a única fonte disponível na empresa. Os executivos não podem contratar gerentes de projetos que não sejam do *pool* ou, pelo menos, sem consultar o gerente. O gerente do *pool* é a autoridade máxima no que diz respeito aos seus funcionários.

CRAWFORD (2001)

Os tipos de PMO propostos por Crawford [Crawford 2001] não trazem inovações substanciais dos vistos até o momento, porém seu trabalho foi que introduziu a idéia que os PMO devem iniciar em áreas de negócios bem consolidadas da organização e sua implantação nessa área de negócio deve servir como um projeto piloto, onde seus resultados serão utilizados para vender a idéia a outras áreas de negócio que sejam menos receptíveis a mudanças. Ainda, vemos em Dinsmore [Dinsmore *et al* 2003] uma reavaliação dos tipos definidos em Dinsmore [Dinsmore 1999] que convergem para a divisão de responsabilidades de Crawford [Crawford 2001], reforçando a solidez de sua proposta de divisão de responsabilidades.

• ***PMO Nível 1 – Controle***

O Escritório de Controle de Projetos possui como principais funções o desenvolvimento do planejamento do projeto e a emissão de relatórios de progresso. Normalmente esse escritório está dedicado a apenas um único projeto, porém é indicado apenas para projetos de grande porte e complexidade. Seu principal objetivo é garantir que o projeto esteja alinhado com os objetivos a serem atingidos, ou seja, com seu sucesso.

• PMO Nível 2 – Área de Negócio

O Escritório de Projetos Nível 2 oferece suporte aos projetos de uma área de negócio específica, de diferentes porte e complexidade. Crawford [Crawford 2001] destaca como principais funções do PMO a priorização entre os projetos e o gerenciamento de recursos e integração dos múltiplos projetos da área de negócio como um programa que foca o atendimento dos objetivos dessa área. Entretanto, a integração destes projetos ocorre ao nível da Unidade de Negócios, não atingindo o nível corporativo. Ainda, é de responsabilidade desse tipo de PMO garantir que a metodologia de gerenciamento de projetos seja unificada e reusada em cada novo projeto, internamente a unidade de negócio.

• PMO Nível 3 – Gestão estratégica do Portfólio

As principais atribuições do Escritório de Projetos Nível 3, segundo Crawford [Crawford 2001], são:

- Selecionar, priorizar e garantir a integração dos projetos que estejam alinhados à estratégia da organização, inclusive no que se refere ao uso de recursos;
- Desenvolver, atualizar e divulgar a metodologia de gerenciamento de projetos, bem como divulgar o conhecimento em gerenciamento de projetos;
- Tornar-se um centro de gestão do conhecimento, através do armazenamento de informações dos projetos na forma de lições aprendidas;
- Validar as estimativas de recursos feitas pelos projetos, baseado nas experiências de projetos anteriores.

Crawford [Crawford 2001], apesar de definir responsabilidades de níveis tático-operacionais, afirma que o foco desse escritório é a gestão de portfólio da organização, trabalhando alinhado com os altos executivos da mesma.

KERZNER (2005)

Em seu livro, Kerzner [Kerzner 2005] define três tipos de PMO básicos e afirma que mais de um desses tipos podem trabalhar em conjunto, ou seja, coexistir dentro da organização. Com isso, o autor define que os tipos possuem responsabilidades próprias, e não são escalas de maturidade em um processo evolutivo. Ainda, quando mais de um PMO existe na organização um novo tipo de PMO deve ser considerado; ele o chama de "*Networking Project Management Office*", que é responsável por reger o trabalho de todos os PMO existentes. Esse PMO passa a ser o responsável pelos padrões e conhecimentos sobre gerência de projetos na organização, porém isso não deve limitar as adaptações das especificidades de cada um dos PMO existentes.

- **PMO Funcional**

Deve ser criado para uma área funcional específica da organização, sua atuação e responsabilidade ficam restritas a essa área funcional. Sua principal responsabilidade é criar e gerenciar um pool de recursos em gerência de projetos para sua área funcional. Esse PMO pode, ou não, se envolver na gerência dos projetos dessa área funcional.

- **PMO Grupo de Clientes**

PMO focado na gerência de *stakeholders* do projeto, principalmente seu cliente. É uma entidade normalmente criada e atribuída para um único projeto e consequentemente temporária, porém, para projetos de um mesmo cliente, deve-se usar o mesmo PMO Grupo de Clientes para o trabalho. Sua principal responsabilidade é promover uma melhor integração entre o cliente e seu projeto. Esse tipo de PMO tem um gerente de projetos dedicado e que participa da gerência dos projetos.

- **PMO Corporativo ou Estratégico**

PMO de abrangência corporativa. Não está ligado a nenhuma área funcional específica. Sua principal responsabilidade é garantir que os projetos existentes na organização estejam alinhados com os objetivos estratégicos da mesma, independente de problemas nas áreas funcionais. Esse tipo de PMO só se envolve no gerenciamento dos projetos quando o objetivo é organizacional, por exemplo, redução de custos.

HILL (2008)

Em seu trabalho de 2004, Gerard Hill introduz um modelo de maturidade para PMO que chamou de *Competency Continuum*. A idéia de um modelo evolutivo não é nova, porém, até esse estudo, ninguém havia proposto um método para operacionalizar essa evolução, nem regras e fronteiras claras que definissem quando um PMO passa de um estágio para outro. Esse estudo de 2004 resultou na publicação de um livro em 2008, onde Hill [Hill 2004, 2008] define um conjunto essencial de 20 responsabilidades, listadas na **Tabela 1**, que um PMO deve ter. O autor estabelece um guia completo para implantação de um PMO, desde sua fundação como um *Project Office* até sua maturação completa atingindo o nível de *Center of Excelence*. Hill [Hill 2008] define claramente o que é preciso desenvolver para se chegar a cada estágio, qual o tempo estimado para realizar essas transições e alerta para os principais desafios a serem vencidos em cada estágio.

As 20 responsabilidades de um Escritório de Gerenciamento de Projetos	
Práticas de gerenciamento	Metodologia de Gerenciamento de Projetos
	Ferramentas de Gerenciamento de Projetos
	Padrões e métricas
	Gerenciamento de conhecimento de projetos
Infra-estrutura de gerenciamento	Governança de Projetos
	Avaliações
	Estrutura e organização
	Suporte de equipamentos e facilidades
Integração de recursos	Gerenciamento de recursos
	Treinamento e capacitação
	Desenvolvimento de carreiras
	Desenvolvimento do time de projeto
Suporte técnico	Mentoring
	Planejamento de suporte
	Auditoria de projetos
	Recuperação de projetos
Alinhamento de negócio	Gerência de portfólio de projetos
	Relacionamento com o cliente
	Relacionamento com o contratante
	Desempenho de negócio

Tabela 1. Responsabilidades de um PMO segundo Hill [Hill 2004]

- ***Project Office***

O primeiro estágio de maturidade, de acordo com o modelo de Hill [Hill 2004]. Esse tipo de PMO foca na aplicação das melhores práticas de gerência de projetos e em definição de uma metodologia padrão de gerência de projetos. Modelado para servir a um ou poucos projetos. Suas principais responsabilidades são:

- Garantir a aplicação das melhores práticas de gerenciamento de projetos desenvolvendo suas competências nos gerentes da organização.
- Concentrar esforços para garantir os entregáveis planejados e seu alinhamento com os objetivos do projeto.
- Controlar as principais dimensões do projeto, escopo, tempo, custo e qualidade, para que obedeçam ao planejado indicando as ações corretivas necessárias quando problemas forem identificados.
- Servir como interface direta entre o time do projeto e o desempenho do mesmo. Normalmente os times focam no desempenho técnico do projeto, e esse tipo de PMO procura aproximar o time técnico dos elementos de gerenciamento de projetos e alinhamento de negócio do projeto, mostrando as diferenças entre criação de produtos tecnicamente excelentes e de projetos gerenciados com excelência.
- Garantir a aplicação das regras organizacionais em todas as atividades do projeto, garantido a integração dos processos de gerência e dos processos de negócio.
- Servir como primeiro nível na hierarquia de decisão executiva e último nível na hierarquia de decisão técnica.

- ***Basic PMO***

Segundo estágio da evolução do PMO, de acordo com Hill [Hill 2004]. É concebido para múltiplos projetos e serve como uma gerência das gerências dos projetos. O autor afirma que, em outras definições, esse tipo de PMO se assemelha a um Escritório de Gerência de Programas. Nesse estágio, o PMO já deve contemplar todas as 20 responsabilidades definidas por Hill como essenciais, listadas na **Tabela 1**. Além de todas as funções do *Project Office* ainda é responsável por:

- Definir os padrões a serem utilizados nos diversos projetos da organização. Isso inclui a definição de uma metodologia com processos que possam ser repetidos e introdução de ferramentas mais comuns em gerenciamento de projetos.

• Prover meios de coletar, analisar e reportar o status do projeto e identificar e corrigir as possíveis variações ocorridas em relação ao desempenho do projeto e seu desvio do plano.
• Introduzir a gerência de projetos como uma disciplina profissional aos gerentes da organização capacitando-os na metodologia e padrões definidos, e especificar todos os papéis envolvidos no projeto garantindo seu entendimento pelo time e *stakeholders*.

• ***Standard PMO***

O estágio três de evolução é considerado por Hill como ponto central, a essência da competência da entidade PMO. Nesse estágio é introduzido um novo foco, alavancar o desempenho dos projetos individuais e melhorar o ambiente de gerência de projetos. Fortalece a gestão de múltiplos projetos centralizando ainda mais o controle e começa o esforço de alinhar os programas da organização aos seus objetivos estratégicos. Além de todas as funções do *Project Office* e do *Basic PMO*, ainda é responsável por:

• Servir como peça central do suporte ao gerenciamento de projetos na organização, realizando a concentração de atividades de gerenciamento de recursos em gerência de projetos para todas as unidades de negócio, na facilitação de acesso e entendimento para as práticas já definidas e difundidas e no fortalecimento do envolvimento do time de projeto, gerente de projeto, *stakeholders* e clientes nas atividades do projeto.
• Funcionar como uma interface entre o ambiente de projeto e o ambiente de negócios da organização, traduzindo adequadamente as decisões de negócios e mudanças de estratégias para ações a serem tomadas nos projetos.
• Iniciar o processo de amadurecimento da metodologia e padrões, adaptando-o cada vez mais às necessidades da organização.
• Servir como representante da gerência de projetos aos altos executivos da organização, seu *control board*, organizações parceiras e profissionais da própria organização.
• Fazer-se uma entidade organizacional reconhecida, deixando claro seu poder de alocação, capacitação e avaliação dos recursos dos projetos.

• ***Advanced PMO***

Segundo palavras de Hill [Hill 2004] "... O *Advanced PMO* funciona como um *Big Brother* do *Standard PMO*", enfatizando que a principal função desse PMO é integrar os interesses de negócio no ambiente de projeto. Isso implica em introduzir e integrar práticas comuns de processos de gerência de projetos e processos de negócio garantindo o total alinhamento dos projetos aos interesses e objetivos do negócio. Em termos comuns em gerência de projetos, é transformar o ambiente organizacional em uma estrutura projetizada. Além de todas as funções dos estágios anteriores, o *Advanced PMO* é responsável por:

• Se consolidar cada vez mais como uma estrutura de negócio independente. Se ainda não foi instituído um orçamento para o PMO em estágios anteriores, esse é o momento de fazê-lo.
• Colaborar com as unidades de negócio da organização para adaptação às suas necessidades da metodologia e práticas já instituídas.
• Promover *expertise* e buscar o estado da arte em gerenciamentos de projetos capacitando os gerentes nessas práticas.
• Alocação tempo integral de membros seniores, tanto da área de projetos quanto de negócio, dedicados às atividades do PMO.

• **Center of Excellence**

Estágio máximo de evolução de um PMO, segundo Hill [Hill 2004]. Nesse estágio o PMO tem como foco principal o alinhamento dos projetos aos objetivos estratégicos da corporação sob duas dimensões: a primeira é o alinhamento tático/operacional dos projetos em andamento e a segunda é a gestão completa do portfólio da organização, desde a seleção de projetos até seu encerramento. Hill [Hill 2004] explicita a necessidade de um alto executivo alocado em tempo integral no PMO e que tenha influência e acesso direto ao *Chief Executive Office* da organização. Esse tipo de PMO, segundo Hill [Hill 2004], é o único que pode ser desenvolvido sem a necessidade de instalação dos tipos anteriores, ou seja, ele não acumula todas as responsabilidades dos quatro primeiros tipos, porém alerta que essa evolução seria interessante do ponto de vista político dentro da organização. Suas responsabilidades são as seguintes:

•Prover direções e influência na gestão de projetos organizacionais.
• Consolidar o ambiente de gerência de projetos, gerência de *stakeholders*, gerência de clientes, gerência de fornecedores e gerência de contratantes, representando todos nas diversas áreas de negócio da organização.
• Patrocinar e conduzir estudos sobre efetividade da gerência de projetos e de negócios da organização.
• Representar os interesses da área de gerência de projetos na área de negócios e vice-versa, como também os interesses de ambas as áreas no conselho da organização.

O modelo evolutivo de Hill [Hill 2004, 2008] é consonante com a demanda atual das organizações, onde não se podem separar as áreas de produção e negócios como ambientes distintos. Mais ainda, Hill [Hill 2008] propõe que essa integração seja evolutiva, e indica os meios para que isso ocorra, o que explica seu reconhecimento cada vez maior no mercado corporativo.

Resumo das Abordagens

A **Tabela 2** apresenta um pequeno resumo das abordagens apresentadas.

Autor	Ano	Contribuição
Dinsmore	1999	Formalizou o conceito de tipos de PMO e documentou os tipos que eram consenso na sua época.
Tipos de Escritórios Propostos Equipe autônoma de gerência de projetos Escritório de Suporte a Projetos Centro de Excelência em Gerenciamento de Projetos Escritório de Gerência de Programas Escritório Chefe de Projetos		
Kate	2001	Primeira proposta de um escritório de projetos evolutivo com aumento incremental do nível de responsabilidade e abrangência.
Tipos de Escritórios Propostos Escritório de Suporte a Projetos Escritório de Projetos e Programas Escritório de Governança de Projetos e Programas		
Casey& Peck	2001	Defende que o PMO não é uma entidade única que abrange diversas áreas, mas sim se deve ter PMOs especializados em determinadas atividades.
Tipos de Escritórios Propostos PMO Nível 1 – Estação Meteorológica PMO Nível 2 – Torre de Controle PMO Nível 3 – Repositório de Recursos		
Crawford	2001	Seguindo a linha evolutiva de Kate, se diferencia por sugerir o início do PMO em uma área de negócio sendo implantado de forma incremental em toda organização.
Tipos de Escritórios Propostos PMO Nível 1 – Controle PMO Nível 2 – Área de Negócio PMO Nível 3 – Gestão estratégica do Portfólio		
Kerzner	2005	Kerzner sugere que o PMO tem como principal foco gerir os stakeholders dos projetos. Segue a linha de Casey & Peck sobre tipos bem definidos e co-existentes.
Tipos de Escritórios Propostos PMO Funcional PMO Grupo de Clientes PMO Corporativo ou Estratégico Networking Project Management Office		
Hill	2008	Um dos autores com mais contribuições da atualidade. Desenvolveu uma espécie de CookBook para implementação e desenvolvimento de PMO nas organizações. É leitura obrigatória para gerentes de PMO.
Tipos de Escritórios Propostos Project Office Basic PMO Standard PMO Advanced PMO Center of Excellence		

Tabela 2. Resumo das abordagens apresentadas

Considerações Finais

A disciplina de gerenciamento de projetos consolidou-se como diferencial competitivo nas organizações nesses últimos trinta anos. Seu amadurecimento foi de fundamental importância para a adequação às exigências de mercado. Esse amadurecimento se deve ao esforço de organizações como o IMPA e o PMI em busca da evolução de seus processos, métodos, práticas e ferramentas, comprovando a importância de entidades independentes dos ambientes internos organizacionais. Essa independência garante a essas entidades a liberdade necessária para inovar, o que muitas vezes não é possível ser feito nos ambientes internos das organizações por fatores políticos, financeiros, técnicos, etc.

Observando a necessidade por alavancar o nível da qualidade em gerência de projetos, as organizações aplicam cada vez mais esforços para se manterem competitivas, e atenderem seus clientes cada dia mais exigentes. Esse investimento, para o desenvolvimento da gerência de projetos interno, normalmente é feito com a implantação de um PMO, órgão que tem por objetivo principal garantir que a gerência de projetos organizacional seja evoluída se tornando cada vez mais eficiente e eficaz.

Durante esse cenário evolutivo, muitos pesquisadores propuseram diversas definições e formatos do que entendiam ser PMO e como eles poderiam atender as demandas sobre gerência de projetos das organizações em sua época. Apesar de ainda existirem muitas discordâncias entre conceitos,

Referências

[Barcaui 2001] BARCAUI, A.B. Project office: a better life for the organization (and for the project managers too!). In: PROJECT MANAGEMENT INSTITUTE SEMINARS & SYMPOSIUM, Tennesi, 2001. Proceedings.

[Block 1997] BLOCK, T.R. The project office – why more companies are adopting it to help manage it projects. In: PROJECT MANAGEMENT INSTITUTE SEMINARS & SYMPOSIUM, Chicago, 1997. Proceedings.

[Casey & Peck 2001] CASEY, W.; PECK, W. Choosing the right PMO setup. PM Network, Illions, v. 15 (2) p. 40-47, Feb. 2001.

[Codas 1987] CODAS, M.M.B. Gerência de Projetos - Uma reflexão histórica. Revista de Administração de Empresas, v. 27 (1). p. 33-37, jan/mar 1987.

[Crawford 2000] CRAWFORD, J.K. Improving organizational productivity with a project office. PM Solutions, Jun. 2000 (Expert Series).

[Crawford 2001] CRAWFORD, J.K. The strategic project office. PM Solutions, 2001 (White Paper).

[Dai & Weels 2004] DAI, C. X., WELLS, W. G. An exploration of project management office features and their relationship to project performance. International Journal of Project Management Volume 22, Issue 7, October 2004, Pages 523-532.

[Dinsmore 1999] DINSMORE, Paul. C. Winning in Business With Enterprise Project Management. AMACOM Div American Mgmt Assn, 1999.

[Dinsmore et al 2003] DINSMORE, Paul. C., GRAHAM, Robert J., ENGLUND, Randall L. Creating the Project Office: a Manager's Guide to Leading Organizational Change. San Francisco, Jossey – Bass, 2003.

[Duggall 2001] DUGGAL, J.S. Building a next generation PMO. In: PROJECT MANAGEMENT INSTITUTE SEMINARS & SYMPOSIUM, Tennesi, 2001. Proceedings.

[Frame 1995] FRAME, J.D. Managing projects in organizations. São Francisco, Jossey-Bass Inc., 1995.

[Gonzalez & Rodrigues 2002] GONZALEZ, F.; RODRIGUES, I. Implementação de escritórios de gerenciamento de projetos. 2002. 95p. Monografia (MBA) – Departamento de Administração da Faculdade de Economia, Administração e Ciências Contábeis, Universidade de São Paulo. São Paulo.

[Hill 2004] HILL, G. M. Envolving The Project Management Office: A Competency Continuum. Information Systems Management, vol. 21, no 4, p. 45-51, 2004.

[Hill 2008] HILL, G. M.. The complete project management office handbook. 2nd ed., ESI international project management series, 2008.

[Kate 2001] KATE, B. The Program Office: A Business Results Enabler. Disponível em http://www.pmforum.org/library/papers/, 2001.

[Murphy 1997] MURPHY, R. The role of Project Support Office. PM Network, p. 33, May, 1997.

[Pellegrinelli & Garagna 2008] PELLEGRINELLI, S., GARAGNA, L. Towards a conceptualization of PMOs as agents and subjects of change and renewal. International Journal of Project Management, accepted 8 December 2008, Waiting pressing.

[PMBoK 2009] PMBoK, A Guide to the Project Management Body of Knowledge (PMBOK guide), 4th ed. 2009, Project Management Institute, 2009.

[Rad & Raghavan 2000] RAD, P.F; RAGHAVAN, A. Establishing an organizational project office. In: AACE INTERNATIONAL TRANSACTIONS, 2000.

tipos de PMO, abrangência de sua atuação e responsabilidades, já é de comum acordo que essa entidade deve ser o guardião da disciplina de gerência de projetos. Responsável por sua evolução dentro das organizações, sua consolidação como prática difundida deve servir como ferramenta que possibilita o alinhamento dos objetivos de cada projeto e programa com os objetivos estratégicos da organização. ●

Personalidade X Papéis Funcionais

Adequação do Profissional ao Desenvolvimento de Software

César C. França

cesarfranca@gmail.com

É mestre em Ciência da Computação (UFPE), com foco em Engenharia de Software experimental. Autor do livro "UM ESTUDO SOBRE MOTIVAÇÃO EM INTEGRANTES DE EQUIPES DE DESENVOLVIMENTO DE SOFTWARE" (Ed. Universitária da UFPE, 2009). Escritor de artigos científicos e palestrante em congressos e conferências de Tecnologia. Consultor independente e sócio da Experience IT, atuando nas áreas de capital humano para TI, planejamento e gerenciamento de projetos e negócios online. Editor do blog QueroFicarRico.com, onde escreve sobre educação financeira, gerenciamento financeiro pessoal, mercado de capitais, investimento e negócios.

Desenvolvimento de software é uma atividade reconhecidamente complexa devido ao grande e diverso conjunto de fatores que podem afetar o desempenho e, consequentemente, o sucesso dos projetos. Entre estes fatores estão os aspectos essencialmente técnicos (linguagens, dispositivos, métodos de produção, processos de desenvolvimento, etc.), o conhecimento (ou não) do domínio do negócio, a qualificação técnica da equipe e a forma como as pessoas envolvidas interagem, se estruturam e se comportam diante das atividades do projeto de desenvolvimento. O último destes fatores está relacionado diretamente à composição do time (de desenvolvimento) de software, como discutido por Sawyer (2004).

De que se trata o artigo?

Neste artigo é apresentada uma comparação entre as habilidades necessárias de alguns papéis funcionais descritos no Rational Unified Process (RUP) com as descrições dos papéis de comportamento em equipe da Teoria de Papéis de Belbin.

Para que serve?

O objetivo é mostrar como a Teoria de Belbin pode ser utilizada para auxiliar na montagem de equipes de software. Desta forma, é possível montar equipes nas quais exista um melhor casamento entre o comportamento esperado no papel funcional e os comportamentos mais naturais do indivíduo alocado neste papel, impactando na produtividade e na motivação da equipe.

Em que situação o tema é útil?

Formar boas equipes está longe de ser uma tarefa fácil, pois não se trata apenas de colocar pessoas altamente qualificadas para trabalhar juntas. Mas, uma vez que se consegue organizar as pessoas de acordo com suas personalidades e potenciais habilidades, a qualidade e a produtividade do trabalho desenvolvido podem ser melhoradas.

A equipe de software tem sido estudada desde os primeiros dias da engenharia de software, em trabalhos seminais como Baker (1972) ou Brooks (1976). Estes enfoques essencialmente funcionais têm sido complementados, mais recentemente, com análises da influência do tipo de personalidade dos indivíduos no desempenho da equipe de software. Estes novos trabalhos têm buscado sua fundamentação conceitual em estudos e teorias sobre tipos de personalidade (por ex. Bradley & Herbert (1997)), teoria de papéis (por ex. Biddle (1979) e Belbin (1981)) ou estilos cognitivos (por ex. Kirton & Ciantis (1986)), entre outros.

Os resultados de vários autores, entre eles Gorla & Lam (2004), Rajendram (2005), Young et al. (2005) e Karn & Cowling (2006), apresentam evidências concretas de que certas personalidades têm melhor desempenho em determinados papéis do processo de desenvolvimento. Ou seja, conclui-se que um caminho para melhorar o desempenho de equipes de software é buscar o adequado casamento entre papel funcional ou técnico com a personalidade do indivíduo atribuído ao papel.

No entanto, todos conhecem histórias de equipes formadas apenas por estrelas (ótimos profissionais) e que não conseguiram um bom entrosamento, não renderam o esperado ou até mesmo falharam. Assim como o oposto, equipes desacreditadas, com participantes menos qualificados, obterem ótimos resultados. Isso acontece em diversas áreas, desde esportes e pesquisa até no mundo empresarial. Formar boas equipes está longe de ser uma tarefa fácil, pois não se trata apenas de colocar pessoas altamente qualificadas para trabalhar juntas. Mas, uma vez que se consegue organizar as pessoas de acordo com suas personalidades e potenciais habilidades, a qualidade e a produtividade do trabalho desenvolvido podem ser melhoradas (Ferreira, 2007).

Neste artigo é apresentada uma comparação entre as habilidades necessárias de alguns papéis funcionais descritos no Rational Unified Process (RUP) com as descrições dos papéis de time da Teoria de Papéis de Belbin (1981). O objetivo é identificar como a Teoria de Belbin pode ser utilizada para auxiliar na montagem de equipes de software nos quais exista um melhor casamento entre o comportamento esperado no papel funcional e os comportamentos mais naturais do indivíduo alocado neste papel.

Papéis de Time (Team Roles)

Segundo Belbin (1981), dentro de um contexto de trabalho em equipe todo indivíduo pode ser classificado de acordo com os seus conhecimentos e a sua função técnica e também com a forma como tende a se comportar, a contribuir e a se relacionar com outros integrantes. Para tanto, Belbin construiu um conjunto de Team Roles, ou papéis de time, que descrevem padrões que caracterizam o comportamento de um indivíduo em relação aos outros na facilitação do progresso de um time.

No estudo de Belbin, são identificados cinco princípios para o gerenciamento de times:

- Membros de um time podem contribuir de duas formas para alcançar os objetivos do time: eles podem desenvolver bem um papel funcional de conhecimento técnico, quando a situação requisita, e podem ter um perfil de time (team role) valioso, o qual descreve um padrão de características comportamentais de como o indivíduo interage com o restante do grupo facilitando o trabalho da equipe;
- Cada time precisa de um balanceamento ótimo nos dois aspectos: papéis funcionais e team roles. Esse nível dependerá dos objetivos e tarefas que o time está enfrentando;
- A eficácia do time se dá pela capacidade dos integrantes de reconhecerem corretamente e ajustarem suas forças e experiências ao restante do time, adotando um team role específico;
- As qualidades pessoais de cada um facilitam que o indivíduo se adeque a um team role, assim como limita as chances dele se adaptar a outro;
- Um time deve distribuir seus recursos técnicos apenas quando os team roles estão bem estruturados e conseguem garantir a eficiência no trabalho em equipe.

Criar um time forte e coeso é uma ciência inexata. Existem coisas que podemos fazer para aumentar as chances de serem bem sucedidos, mas é impossível dizer se o time realmente vai se sair bem. Trazer as pessoas certas para o projeto se torna crucial. Entretanto, não basta apenas entender como o indivíduo se comporta sozinho e sim, como ele interage em um determinado grupo. Não se trata de ter a pessoa certa para liderar, mas sim, pessoas capazes de trabalhar bem em conjunto e que possam se ajudar para alcançar um objetivo em comum.

Por tratar de aspectos pessoais não-estáticos, relativos ao comportamento situacional dos indivíduos, Belbin complementa que é possível um único indivíduo assumir mais de um papel na equipe, desde que não sejam papéis com características chocantes. Dessa forma, não é necessário que uma equipe tenha ao menos oito componentes para suprir todos os perfis.

Em seu trabalho original, Belbin (1981) definiu oito papéis de time:

- Co-ordinator: Papel de liderança. As pessoas que representam adequadamente este papel tendem a ser calmas, auto-confidentes e exercerem liderança controladamente. Eles servem para guiar e controlar os membros do time em uma dada situação. Este papel está presente normalmente em posições de liderança mais permanentes. O Co-ordinator tem um forte senso de objetividade e consegue extrair todo o potencial de contribuição de cada membro do grupo.
- Shaper: Segundo papel de Liderança. Líder autoritarista. É impaciente, provocativo e normalmente fica irritado nas reuniões do time. Melhor adequado a posições temporárias, como liderança de projetos. Apresentam uma direção forte, é dinâmico e tem coragem para levar o time a enfrentar os obstáculos.
- Plant: Papel de criatividade. Simpatia com a inovação e a resolução de problemas. Normalmente é individualista, sério e não-ortodoxo. Seu gênio, imaginação, intelecto e conhecimento são seus pontos fortes, enquanto sua principal fraqueza é

que despreza as implicações práticas dos problemas do time, além de criar uma forte relação pessoal com as suas idéias. Fonte de criatividade e originalidade do time. Tendência a ignorar o óbvio e optar pelo menos convencional.

• Resource-Ivestigator: Papel de criatividade. Provê idéias a partir do mundo exterior ao time. Grande capacidade de comunicação, entusiasmo, extroversão, interesse em explorar sempre novas alternativas e enfrentar novos desafios. Sua principal fraqueza é que é levado a perder a motivação quando passa a fascinação inicial pelo trabalho. É a pessoa de contato externo do grupo.

• Monitor Evaluator: Papel de equilíbrio. Tende a ser sóbrio e prudente, sem qualquer relação emocional com o trabalho exercido. Apresenta julgamento claro, discrição e pode parecer cínico e cético. Seu aspecto negativo é a sua falta de inspiração, bem como a sua incapacidade de motivar as outras pessoas. É a pessoa que constantemente analisa as soluções propostas e as decisões tomadas pelo time, evitando que o time cometa erros conceituais e aponta as falhas no plano antes de assumir compromissos.

• Implementer: Papel de execução. Transformam os planos e conceitos definidos pelo time, em processos e ações práticas de trabalho. Apresenta grande capacidade de organização, trabalha duro, auto-disciplinado e possui um senso prático destacável. Normalmente conversador, dá mais importância para o resultado geral da companhia do que aos seus interesses pessoais. Este membro tem uma grande habilidade de identificar as necessidades da companhia, bem como de assumir responsabilidades que os outros evitam, porém apresenta uma grande falta de flexibilidade e resposta lenta a mudanças de situações.

• Team Worker: Papel de execução. Mediador interno ao grupo, habilidade social e sensitiva para com os sentimentos dos outros integrantes. Promove o espírito de equipe e atua motivando a equipe para trabalharem todos juntos em prol do objetivo comum. Resolve conflitos e atua como uma espécie de pacificador. É responsável por manter a coesão da equipe.

• Completer Finisher: Papel de equilíbrio. É preciso, perfeccionista e busca sempre evitar erros, prestando muita atenção a todos os detalhes. É o papel que normalmente está disposto a levar o projeto até o fim.

A **Tabela 1** explora sucintamente as características típicas, as qualidades e as fraquezas de cada papel de time.

Os papéis de time podem ser agrupados de acordo com sua tendência em relação à liderança ou criatividade, da seguinte forma:

•Perfis de Liderança (Co-ordinator, Shaper): papéis de liderança de perfis opostos e, em certos contextos, antagônicos.

• Perfis de Criatividade (Plant, Resource Investigator): perfis associados a solução de problemas que buscam esta solução com recursos próprios (Plant) ou de terceiros (Resource Investigator).

• Perfis de Suporte (Specialist, Implementer, Teamworker, Monitor Evaluator): perfis que complementam o trabalho do time com competências e habilidades específicas.

A Teoria de Papéis é complementada por uma ferramenta de análise chamada Team Role Self-Perception Inventory (TRSPI, Belbin (1981), pg. 153), com o qual é possível identificar os papéis exercidos ou preferidos por um indivíduo em uma situação de trabalho em grupo. O TRSPI (versão original com 8 papéis) é composto de 7 questões, com 8 itens por questão. Cada item descreve um comportamento relativo a uma situação de trabalho em grupo e está relacionada a um papel de time. O respondente deve atribuir 10 pontos entre os 8 itens em cada questão, de forma a refletir a sua auto-percepção de como se comporta em cada situação descrita.

As pontuações não são utilizadas diretamente. Existe uma tabela de normas que classifica a tendência ao comportamento de acordo com o papel em uma escala de quatro valores: Low, Average, High e Very High. Com esta escala, indivíduos com pontuação nos níveis High e Very High tendem a exibir o comportamento descrito no papel, enquanto aqueles com pontuação Low ou Average terão deficiência ou dificuldades em assumir o comportamento descrito.

Adequação de perfis de personalidade e papéis funcionais do RUP

As recomendações abaixo foram desenvolvidas a partir de uma pesquisa de campo realizada através da aplicação de uma adaptação do questionário Self Perception Inventory - SPI em 40 profissionais distribuídos em três fábricas de software com atuação em Recife.

• O gerente de projeto deve apresentar um perfil de liderança e coordenação orientado a pessoas, como o perfil CO (Co-ordinator). De fato, a pesquisa demonstra que os gerentes de projetos entrevistados têm tendência a assumir tal papel. Porém, a pesquisa também demonstra uma aparição significativa do perfil IM (Implementer). O que se justifica por se tratar de um perfil focado para o resultado geral do processo da organização. O papel que é responsável direto pelo resultado geral do projeto deve exercer uma grande habilidade de identificar as necessidades da companhia. Os dados também indicam que os gerentes de projeto com maior nível de satisfação são aqueles que apresentam um grau extremamente alto de IM e CO em conjunto, ou que apresentam apenas um dos dois perfis, porém exerce exclusivamente esta função no projeto (a maioria dos gerentes de projeto entrevistados exerce a função em conjunto com algum papel de analista).

• O engenheiro de processos, ao contrário do gerente de projeto, deve assumir um papel de coordenação, porém mais orientado à atividade, a fim de estabelecer um processo objetivo e eficiente de desenvolvimento de software e de garantir que tal processo será respeitado e gerenciado. A pesquisa quantitativa indica que a incidência do perfil SH (Shaper) somado ao CO (Co-ordinator) garante a satisfação

Papel de Time (Título Original)	Sigla	Descritores	Pontos Fortes	Possíveis Fraquezas
Completer Finisher	CF	Ansioso, consciencioso, introvertido, tem autocontrole, tem autodisciplina, submisso e preocupado.	Meticuloso, consciencioso, procura por erros e omissões, entrega sem atraso.	Tendência a se preocupar demais. Relutante a delegar.
Implementer (Company Worker)	IMP	Conservador, controlado, disciplinado, eficiente, inflexível, metódico, sincero, estável e sistemático.	Disciplinado, confiável, conservador e eficiente, transforma idéias em ações práticas.	Um tanto inflexível. Lento para responder a novas possibilidades.
Team Worker	TW	Extrovertido, amigável, leal, estável, submisso, confortante, não assertivo e não competitivo.	Cooperativo, suave, boa percepção e diplomático, escuta, constrói, evita atritos, acalma o clima.	Indeciso em situações de conflito.
Specialist1	SP	Especialista, defensivo, não interessado nos outros, sério, tem autodisciplina, eficiente.	Focado, dedicado, auto-motivado, provê conhecimento e habilidades raros.	Contribui somente em um único tópico. Alonga-se em tecnicalidades.
Monitor Evaluator	ME	Seguro, fidedigno, justo, introvertido, de avanço lento, aberto a mudanças, sério, estável e sem ambições.	Sóbrio, estratégico e perspicaz, visualiza todas as opções, julga com precisão.	Não tem impulso e habilidade para inspirar outras pessoas.
Co-ordinator (Chairman)	CO	Dominante, confia nos demais, extrovertido, maduro, positivo, tem autocontrole, tem autodisciplina, estável.	Maduro, confiante, bom diretor, esclarece objetivos, promove a tomada de decisão, delega bem.	Pode ser visto como manipulador. Sobrecarregado com trabalho.
Plant	PL	Dominante, imaginativo, introvertido, original, pensamento radical, cheio de confiança, não se inibe.	Criativo, não ortodoxo, soluciona problemas difíceis.	Muito absorto em pensamentos; dificuldade para se comunicar efetivamente.
Shaper	SH	Abrasivo, ansioso, arrogante, competitivo, dominante, irritável, emocional, extrovertido, impaciente, impulsivo, autoconfiante.	Desafiador, dinâmico, prospera sob pressão, tem impulso e coragem para vencer obstáculos.	Suscetível a provocações. Ofende o sentimento das pessoas.
Resource Investigator	RI	Diplomático, dominante, entusiasta, extrovertido, flexível, inquisitivo, otimista, persuasivo, positivo, descontraído, social e estável.	Extrovertido, comunicativo, explora oportunidades, desenvolve contatos.	Excessivamente otimista. Perde interesse depois do entusiasmo inicial.

Tabela 1. Papéis de Time, Descritores, Pontos Fortes e Possíveis Fraquezas/ Fonte: Belbin (1993), pg. 22

do indivíduo, enquanto apenas a aparição do CO (Coordinator) causa sua insatisfação. Desse modo infere-se que o SH (Shaper) é o perfil mais indicado.

• O gerente de controle de mudanças, por sua vez, deve assumir uma posição de grande prudência e corretividade conceitual, adequando-se ao perfil ME (Monitor Evaluator), a fim de evitar que pequenas falhas possam se transformar em grandes problemas durante todo o projeto. É importante, porém não essencial, que o gerente de controle de mudanças apresente também um grau mais alto em um perfil de coordenação orientado à atividade, como o SH (Shaper).

• O gerente de configuração deve ser assertivo para assegurar que os desenvolvedores não ignorem a política e os procedimentos de configuração. Pela necessidade de ser atento aos detalhes, o gerente de configuração deve possuir um perfil analítico destacável, como ME (Monitor Evaluator). A pesquisa quantitativa mostra que em 50% dos casos este perfil se destaca entre os gerentes de configuração que estão satisfeitos. É comum a renegação da disciplina de gerência de configuração em muitos projetos, o que pode indicar que os 50% restante não tenha um bom controle de configuração de seus projetos.

• O papel de gerente de implantação deve assumir um perfil semelhante ao do gerente de projeto no que diz respeito ao trabalho pesado, capacidade de planejamento com foco na organização e orientação por resultados características referentes ao perfil IM (Implementer). Isso se deve pelo fato de ser uma disciplina crítica do projeto, da qual o resultado interfere completamente na satisfação do cliente. Inclusive, como sugestão do próprio RUP, ambos os papéis podem ser exercidos pelo mesmo indivíduo.

• O gerente de testes necessita de um alto desempenho em um perfil orientado à ação. É deste papel a responsabilidade pelo esforço geral dos testes. De fato, a pesquisa indica que o perfil mais comum entre os gerentes de testes satisfeitos é o IM (Implementer), aparecendo em 100% dos casos mesmo quando todos estes assumem simultaneamente papéis de Analista.

• O analista de negócios é o papel que lidera e coordena a modelagem dos casos de uso de negócios. Deve assumir um perfil que despende bastante atenção principalmente no tratamento com os clientes para captação das necessidades deste. É importante também que este papel tenha capacidade também de negociar com o cliente os aspectos de projeto que possam interferir nos interesses gerais da organização. Estas características são representadas pelos perfis CO (Co-ordinator) e IM (Implementer). Este analista pode ainda assumir um perfil mais cerebral que apoiará na modelagem da arquitetura de negócios. A pesquisa prática indica que, de fato, os dois perfis são os mais comumente destacados nos profissionais analistas de negócios. Mais dois detalhes que a pesquisa prática revela é que o nível do perfil cerebral é diretamente proporcional ao grau de

satisfação do indivíduo; e que, por apresentar características semelhantes ao gerente de projeto, em 80% dos casos o analista de negócio assume também papéis de gerencia no projeto.

• O analista de sistemas é responsável pela modelagem dos casos de uso de análise do sistema. Suas características comportamentais são semelhantes ao analista de negócios. Frequentemente o analista de negócios também assume o papel de analista de sistema. O principal diferencial aparece com o aumento da aparição de um perfil mais criativo. Por necessitar de um perfil essencialmente orientado a pessoas, a nova característica do analista de sistemas agora é o RI (Resource Ivestigator).

• O especificador de requisitos deve contemplar os perfis IM (Implementer) e CF (Completer Finisher), pois está sob sua responsabilidade um dos principais artefatos da fase de Análise, a especificação dos casos de uso. No contexto de fábricas de software, a qualidade desse artefato é definitiva para o desenvolvimento de um produto, visto que é o principal meio de comunicação entre o cliente final e o núcleo de desenvolvimento da fábrica. A pesquisa prática indica que o perfil IM (Implementer) está presente em 75% dos especificadores de requisitos. Porém exibe uma grande escassez de CF (Completer Finisher), o que pode ser uma das principais causas da excessiva quantidade de alterações comumente solicitadas nos sistemas produzidos. A pesquisa indica também que para este papel, a satisfação do usuário é inversamente proporcional ao grau de aparição do perfil CO (Co-ordinator).

• O analista de teste avalia a modelagem do sistema e identifica as necessidades de teste do sistema. Então, por necessitar de uma visão crítica e criteriosa, este analista deve assumir o perfil ME (Monitor Evaluator). Na prática, o que acontece é de indivíduos que exercem outros papéis de analista, bem como outros papéis de gerência, assumirem a responsabilidade pela análise de testes, o que acaba por prejudicar a produção de testes confiáveis por essas organizações, consequentemente aumentando as estatísticas sobre o insucesso de projetos de software.

• O papel de arquiteto de software é responsável pela definição de diretrizes para o desenvolvimento, e deve implementar a característica criativa da equipe. Na pesquisa prática, 50% dos arquitetos satisfeitos assumem um grande grau de CO (Co-ordinator), porém todos eles exercem simultaneamente papéis de analistas, o que descarta o perfil como essencial. O diferencial capaz de influenciar diretamente na satisfação do arquiteto é a aparição de um alto grau de PL (Plant) ou RI (Resource Investigator). Por outro lado, o nível de aparição do CO (Co-ordinator) é inversamente proporcional à satisfação do profissional. Outro fator ao qual o arquiteto de software não se adequa perfeitamente é ao acúmulo de papéis de gerente ou testador.

• O papel de implementador deve assumir um perfil completamente voltado à implementação de tudo que foi planejado e especificado até então. Este papel, por estar na base do processo produtivo, deve apresentar um interesse em produzir resultado mais do que em produzir planos. Essas características, típicas de um IM (Implementer), se confirmam na prática os implementadores satisfeitos apresentando um grau bastante significativo deste perfil. É interessante notar que dois terços dos implementadores entrevistados também exercem o papel de testador. Aqueles que atuam exclusivamente no papel de implementar possuem graus significativos do perfil TW (Team Worker).

• O integrador, por sua vez, deve assumir um perfil orientado à atividade, de persistência e detalhismo, como CF (Completer Finisher). De fato, o perfil mais evidente na pesquisa (100% dos casos) é orientado a atividade, porém é o IM (Implementer). É importante notar também que o integrador e por isso apresenta o perfil IM (Implementer) em evidência.

• O testador põe em prática os testes, sendo o papel central que cuida da condução e do registro dos seus resultados. Por apresentar características bastante pragmáticas, o perfil adequado para o testador é IM (Implementer). A pesquisa prática revelou que os testadores que destacam normalmente o perfil IM (Implementer) encontram-se satisfeitos. Isso também justifica o grande número de testadores, mais de 90%, que também exercem papéis de desenvolvedor.

A **Tabela 2** apresenta o mapeamento sumarizado entre todos os papéis do RUP e seus perfis adequados do Belbin.

Recomendações gerais para montagem de times

Com base na análise das adequações descritas anteriormente, é possível delinear as seguintes recomendações gerais para montagem de times:

• O Gerente de Projetos deve comunicar bem: não sendo possível alocar um Co-ordinator como gerente, os papéis Resource-Investigator e Team-Worker também relacionam positivamente com esta função, pois são papéis com forte perfil de comunicação e orientados a pessoas, característica universalmente aceitas como importante no gerente de projetos. Estes resultados também são consistentes com Fernandes & Silva (2007).

• Cuidado com Gerentes Shapers: indivíduos com preferência por um comportamento do tipo Shaper tendem a assumir posições de liderança em times, de acordo Belbin, mas suas características negativas dificultam o relacionamento interpessoal dentro e fora do time. Apesar de Shapers poderem ser importantes em projetos com necessidade de lideranças fortes e "enérgicas", é necessário ter cuidado com os potenciais conflitos que podem surgir, principalmente se outros membros do time tiverem tendências ao mesmo papel.

• Excesso de Criatividade pode Prejudicar a Implementação: papéis relacionados à criatividade, Plant e Resource-Investigator, se relacionam negativamente com as habilidades necessárias do papel Implementador. Indivíduos com estes papéis tendem a ser muito bons na solução de problemas complexos, mas perdem o interesse na atividade assim que

IM	CO	SH	PL	RI	ME	TW	CF	
X	X							Analista do Processo de Negócios
X	X			X				Analista de Sistemas
					X			Analista de Teste
			X	X				Arquiteto de Software
X	X							Designer
X	X							Designer de Banco de Dados
X	X							Designer de Cápsulas
X	X							Designer de Interface de Usuário
X	X							Designer de Negócios
X	X							Designer de Teste
		X						Engenheiro de Processo
X							X	Especificador de Requisitos
					X			Gerente de Configuração
		X						Gerente de Controle de Mudanças
	X							Gerente de Implantação
	X							Gerente de Projeto
X								Gerente de Testes
X						X		Implementador
							X	Integrador
					X			Revisor de Arquitetura
					X			Revisor de Código
					X			Revisor de Design
					X			Revisor do Modelo de Negócios
					X			Revisor do Projeto
					X			Revisor de Requisitos
X								Testador

CH – Chairman PL – Plant ME – Monitor Evaluator TW – Team Worker

SH – Shaper RI – Resource Investigator CW – Company Worker CF – Completer Finisher

Tabela 2. Mapeamento dos perfis de Belbin nos Papéis do RUP

a solução é encontrada. Atividades classicamente repetitivas ou rotineiras, como codificação e teste, não são adequadas para indivíduos com estes papéis.

• Criatividade é um Traço Importante para o Arquiteto de Software: a relação positiva do papel Arquiteto de Software com os papéis de time associados à criatividade, Plant e Resource-Investigator, mostra que este traço de personalidade é importante para as atividades de desenho da arquitetura de sistemas.

• O Analista de Sistema deve ter Afinidade com Pessoas: o modelo aponta para uma relação positiva do papel Analista de Sistemas com os papéis de Belbin que naturalmente tem facilidade no relacionamento interpessoal e boas habilidades de comunicação, como Resource-Investigator, Team-Worker e Co-ordinator.

• Completer-Finisher e Monitor-Evaluator são Importantes na Implementação: indivíduos que exibem comportamento de Completer-Finisher e Monitor-Evaluator são meticulosos, bons avaliadores e preocupados em entregar seus trabalhos no prazo. Apesar de serem apenas medianos em relação à criatividade, possuem características positivas para tarefas de codificação, teste e controle de qualidade.

• Lideranças devem ser estabelecidas dentro dos sub-times: O perfil TW (Team Worker) aparece em apenas uma sugestão de adequação para os papéis do RUP. Isso ocorre pelo fato do perfil TW ser um perfil de mediação interna do grupo. Num contexto de fabricação de software, podem ser identificados vários sub-times durante o processo produtivo sejam sub-times de analistas, sub-times de programadores, de testadores, etc. O perfil TW (Team Worker) é entendido que deve ser realçado por algum membro destas sub-equipes sempre que da existência destas.

Considerações Finais

Tanto a literatura recente de gerenciamento de projetos quanto a literatura da psicologia indicam que existe uma relação positiva entre a adequação do perfil pessoal ao papel funcional e o desempenho do indivíduo na função (Bradley & Herbert, 1997). Ou seja, pessoas tendem a ter melhor desempenho quando estão realizando funções que combinam com seu comportamento natural.

Sendo assim, para os gerentes de projetos, estas recomendações para a alocação de pessoas nos papéis do RUP servem como uma poderosa ferramenta para melhorar diversos aspectos da composição de times de software além de complementar os processos de seleção de pessoal e montagem de times. ●

Dê seu feedback sobre esta edição!

A Engenharia de Software Magazine tem que ser feita ao seu gosto.
Para isso, precisamos saber o que você, leitor, acha da revista!
Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/esmag/feedback

Referências

Andrade, P. C. (2000) "Guia de Profissões e Mercado de Trabalho", Ed. Oriente-se, 1ª edição.

Aritzeta, A., Swailes, S. and Senior, B., (2005) "Team Role Preference and Cognitive Styles". Small Group Research, Vol. 36, No. 4, 404-436.

Belbin, M.R. (1981) "Management Teams: Why they succeed or Fail", Butterworth-Heinemann Ltd.

Belbin, M.R. (1993) "Team Roles at Work", Elsevier Butterworth-Heinemann Ltd.

Biddle, B.J. (1979) "Role theory: Expectations, identities, and behaviors". New York: Academic Press.

Bradley, John H. and Herbert, Frederic J. (1997), "The effect of personality type on team performance", Journal of Management Development, Vol. 16, No. 5, pp. 337-353, MCB University Press.

Fernandes, Flávio L.M. e da Silva, Fabio Q.B. (2007) "Relações entre Competências Pessoais e Tipos de Personalidade do Gerente de Projetos de Software", II Congresso de Gerenciamento de Projetos, PMI.

Ferreira, Henrique S. (2007) "Um Estudo Da Adequação De Personalidades E Papéis Na Metodologia Scrum De Desenvolvimento De Software". Trabalho de graduação apresentado no Centro de Informática-CIn da UFPE.

França, A. César C. e da Silva, Fabio Q. B. (2007) "Um estudo sobre Relações entre Papéis Funcionais do RUP e o Comportamento Pessoal no Trabalho em Equipe em Fábricas de Software". III Workshop Um Olhar Sociotécnico sobre a Engenharia de Software – WOSES.

Gorla, N. and Lam, Y.W., (2004). "Who Should Work With Whom? Bulding Effective Software Project Teams". Communications of the ACM, Vol. 47, No. 6, pp. 79-82.

Karn, J. and Cowling, T. (2006) "A Follow up Study of the Effect of Personality on the Performance of Software Engineering Teams". Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering (ISESE'06), Rio de Janeiro, Brazil, pp. 232-241.

Kruchten, P. (2003) Introdução ao RUP – Rational Unified Process, Rio de Janeiro: Editora Ciência Moderna Ltda.

Rajendran, M. (2005) "Analysis of team effectiveness in software development teams working on hardware and software environments using Belbin Self-perception Inventory", Journal of Management Development, Vol. 24 No. 8, pp. 738-753, Emerald Group Publishing Limited, 0262-1711, DOI 10.1108/02621710510613753.

Sawyer, S. (2004) "Software Development Teams". Communications of the ACM, Vol. 47, No. 12, pp. 95-99.

Young, S.M. et al. (2005) "Personality Characteristics in an XP Team: A Repertory Grid Study". ACM Workshop on Human and Social Factors of Software Engineering (HSSE), ACM SIGSOFT Software Engineering Notes Vol. 30 , Issue 4.

Cursos Online

Assinatura

.net magazine PLUS

Mais conteúdo .NET por muito menos!

A revista **.net Magazine** oferece para seus assinantes uma série de **Cursos Online** de alto padrão de qualidade .
Conheça abaixo os cursos já disponíveis.

Curso em destaque

Crie uma loja virtual completa

Confira neste curso Online como construir uma loja virtual completa no Visual studio 2005. Aprenda como criar o banco de dados, carrinho de compras, profile, página de erros e muito mais .

Confira o plano de aula completo:
www.devmedia.com.br/lojavirtual

A sua melhor opção de aprendizagem!
Assine a **.net Magazine** e comece já seu treinamento!
www.devmedia.com.br/assine

Outros cursos disponíveis: **www.devmedia.com.br/curso**

- **Construindo relatórios com Crystal Reports e Visual Studio 2005**
- **Criando uma aplicação Web Completa**
- **Criando uma aplicação client/server no Visual Studio 2005**
- **Aprenda a criar um blog com ASP.NET**
- **Curso de C#**

* Curso em andamento

DevMedia GROUP **Mais Informações: www.devmedia.com.br/central - Tel.: 21 3382-5038/ 3382-5025**

Modéstia à parte, sua melhor opção para se destacar no mercado!

A Escola Superior da Tecnologia da Informação oferece as melhores opções em cursos, formações, graduações e pós-graduações para profissionais de desenvolvimento e programação.

São programas voltados para a formação de profissionais de elite, com **aulas 100% práticas, corpo docente atuante no mercado**, acesso à mais atualizada biblioteca de TI do Rio, laboratórios equipados com tecnologia de ponta, salas de estudo e exames.

PÓS-GRADUAÇÃO

- **Engenharia de Software: Desenvolvimento Java**

GRADUAÇÃO

- **Análise e Desenvolvimento de Sistemas**

FORMAÇÕES

- **Desenvolvedor Java**
- **Desenvolvedor Java: Sistemas Distribuídos**
- **Gestor de TI**
- **Desenvolvedor Web .NET 2008**
- **MCITP Server Administrator**
- **SQL Server 2008**

Acesse nosso site e conheça todos os nossos programas: **www.infnet.edu.br/esti**

TURMAS NO RIO DE JANEIRO

infnet INSTITUTO

ESCOLA SUPERIOR DA TECNOLOGIA DA INFORMAÇÃO

www.infnet.edu.br - cursos@infnet.edu.br - Central de Atendimento: (21) 2122-8800

EDUCAÇÃO SUPERIOR ORIENTADA AO MERCADO