9 Gerenciamento de Projeto de Software

Nos capítulos anteriores foi visto como planejar um projeto. Atenção especial foi dada aos processos de estimação de esforço e controle de riscos. Este capítulo vai tratar agora do momento em que o projeto efetivamente inicia. Alguns aspectos da gerência de um projeto já foram discutidos nos capítulos anteriores, mas aqui o processo de gerenciamento como um todo será aprofundado. Inicialmente será discutido o papel do *gerente de projeto* (Seção 9.1) e suas atribuições, além de recomendações práticas sobre como deve atuar. Em seguida serão apresentados rapidamente os conceitos de gerenciamento de projetos de uma fonte internacionalmente reconhecida, o *PMBOK* (Seção 9.2). Em seguida é apresentado um método de gerenciamento específico, o *PRINCE2* (Seção 9.3). São apresentadas recomendações gerais sobre como *conduzir* um projeto de software (Seção 9.4), *métricas* a serem usadas (Seção 9.5), como *revisar e avaliar* (Seção 9.6) o andamento de um projeto e como realizar o *fechamento* ou conclusão de um projeto (Seção 9.7).

A gerência de projeto pode ser entendida como uma disciplina dentro de um processo de engenharia de software (por exemplo, ver Seção 5.3.2.1), usualmente exercida por um único indivíduo (o gerente de projeto) que tem como objetivo levar o projeto a produzir os objetivos previamente planejados, dentro dos prazos, custos e qualidade previstos. Para ter sucesso nisto o gerente deve manter os riscos do projeto no nível mais baixo possível de probabilidade e impacto, avaliando continuamente o progresso e tomando medidas proativas para redução destes riscos, ou medidas de correção se apesar de tudo problemas ocorrerem.

Uma fonte de referência muito importante sobre gerência de projetos é o PMBOK (PMI, 2004). Porém, como ele é um referencial genérico para a condução de projetos, é necessário ter em mente que projetos na área de software têm as suas particularidades. E assim, um gerente de projeto de software precisa ter conhecimentos específicos sobre gerência de projetos *de software*.

O SWEBOK (IEEE Computer Society, 2004) indica, entre outras coisas, as seguintes particularidades para os projetos de software:

a) Os clientes dificilmente percebem as reais complexidades envolvidas com o processo de desenvolvimento de software, especialmente as dificuldades relacionadas com as mudanças de requisitos (ver mitos da Seção 1.2).
b) É praticamente inevitável que os próprios processos de engenharia de software acabem gerando a necessidade de introdução de novos requisitos ou de modificação dos requisitos existentes.
c) Como resultado disso, o software é frequentemente construído por um processo de refinamento interativo ao invés de uma sequência de atividades previamente bem definidas e programadas.
d) A engenharia de software necessariamente incorpora aspectos de criatividade e disciplina. Manter um balanceamento apropriado entre estes dois aspectos frequentemente é uma tarefa difícil.
e) O grau de novidade e de complexidade do software frequentemente é extremamente alto.

f) A tecnologia subjacente ao software muda com muita frequência.

O *SWEBOK* divide a área de gerenciamento de projeto em engenharia de software nas seguintes subáreas:

a) Inicialização e definição de escopo.
b) Planejamento de projeto de software.
c) Condução de projeto de software.
d) Revisão e avaliação.
e) Fechamento.
f) Medição em engenharia de software.

As primeiras duas subáreas são tratadas no Capítulo 6. O atual capítulo vai abordar as outras subáreas: condução de um projeto de software, bem como sua revisão, medição, a avaliação e fechamento.

9.1 O Gerente de Projeto

Segundo Levinson[111], um projeto bem sucedido depende muito de um bom gerente. Isso vale tanto em situações onde existe a figura do gerente como indivíduo, quanto nas situações onde a equipe se auto-gerencia. De uma forma ou de outra, o processo de planejamento, condução, avaliação e fechamento do projeto estará presente. Levinson indicou seis habilidades-chave observadas em gerentes de projeto de sucesso:

a) *Têm o dom de prever*. Bons gerentes têm a capacidade de visualizar e antecipar problemas antes que eles ocorram, e tomar ações preventivas.
b) *São organizados*. A organização tem vários aspectos, mas um que parece ser especialmente importante para o gerente de projeto é a capacidade de visualizar prioridades e passá-las para sua equipe. Assim, em meio à complexidade de um projeto o gerente será capaz de enxergar o que realmente é importante e concentrar os esforços da equipe nisso.
c) *Sabem liderar*. O gerente de projeto não é apenas um chefe. Além da equipe, ele precisa interagir com pessoas que não estão sob seu comando (clientes, usuários, especialistas de domínio, por exemplo). Assim, ele precisa ter o carisma de líder e ser capaz de motivar as pessoas a gastar seu tempo nas atividades que são necessárias para o projeto.
d) *São bons comunicadores*. Eles são capazes de utilizar múltiplos meios de comunicação, como email, telefone, reuniões, apresentações, etc., como forma de obter e transmitir as informações necessárias. Eles são efetivos, objetivos e pragmáticos quando se comunicam (não ficam fazendo rodeios). Além disso, são capazes de ouvir. Um gerente que não ouve sua equipe ou outros interessados poderá levar um projeto a fracassar, pois pequenos problemas não resolvidos de início muitas vezes acabam virando grandes problemas no final de um projeto de desenvolvimento de software.
e) *São pragmáticos*. Existem dois extremos quando se pensa na forma de tomada de decisão de um gerente: os que adiam a decisão até conhecer todas as implicações das

[111] cio.uol.com.br/gestao/2008/09/10/caracteristicas-dos-gerentes-de-projeto-bem-sucedidos/ (consultado em 10/01/2012)

opções, e os que tomam decisões por impulso, sem pensar. No equilíbrio entre estes dois extremos está o gerente pragmático: ele analisa os prós e contras da forma mais eficiente possível, e é capaz de avaliar rapidamente se está em condições de tomar uma decisão fundamentada ou não. Caso a análise dos pros e contras, porém, tomar mais tempo do que tentar uma das opções, ele vai perceber isso e, neste caso, decidir no sentido de tentar um caminho e, se não der certo, tentar o outro.

f) *São empáticos*. Um gerente não faz o trabalho da equipe, mas também não faz o seu trabalho sozinho. Ele precisa se apoiar em outras pessoas. Para obter essa colaboração ele precisa entender o que motiva as pessoas. Empatia é a capacidade de colocar-se no lugar do outro e entender suas necessidades. A partir disso, o gerente balizará suas ações de motivação.

Classicamente assumia-se que o gerente de projeto precisava equilibrar os três vértices do *triângulo de restrições*, que envolvia *tempo*, *custo* e *escopo*. Diminuir qualquer um destes ângulos provocaria aumento em pelo menos um dos outros dois.

Porém, este triângulo hoje é substituído por um conjunto de seis variáveis: escopo, qualidade, cronograma, orçamento, recursos e riscos. Mas assume-se ainda que qualquer mudança em um deles elementos afete pelo menos um dos outros.

9.2 Gerenciamento de Projetos segundo o PMBOK

Apesar de não tratar as particularidades dos projetos de software, o PMBOK (PMI, 2004) é uma excelente referência em termos de gerenciamento de projetos. Ele estrutura o corpo de conhecimentos em duas dimensões: *grupos de processo* e *áreas de conhecimento*.

Os grupos de processo PMBOK possuem suas equivalentes no SWEBOK, conforme mostrado na Tabela 9-1.

Tabela 9-1: Equivalência entre grupos de processo de gerência no PMBOK e no SWEBOK.

PMBOK	SWEBOK
Iniciação	Iniciação Definição de Escopo
Planejamento	Planejamento
Monitoramento e controle	Medição Revisão Avaliação
Encerramento	Fechamento

As áreas de conhecimento do PMBOK são descritas abaixo. As áreas de conhecimento não abrangem necessariamente todos os grupos de processo. Apenas o gerenciamento de integração abrange todos os grupos. As nove áreas de gerenciamento são, pois:

a) *Gerenciamento de integração*. São as atividades que o gerente de projetos executa de forma a garantir que todas as partes do projeto funcionem juntas.
b) *Gerenciamento do escopo*. São as atividades necessárias para que o projeto execute de fato todas as atividades necessárias para gerar o produto e *somente* as atividades necessárias.

c) *Gerenciamento de tempo*. São as atividades mais visíveis em gerência de projeto e consistem em garantir que as atividades do projeto ocorram dentro dos tempos previamente definidos.
d) *Gerenciamento de custos*. São as atividades que buscam garantir que o projeto ocorra dentro do orçamento definido.
e) *Gerenciamento da qualidade*. Do ponto de vista externo visa garantir que o produto atenda às expectativas do cliente, do pondo de vista interno visa garantir que o produto seja suficientemente maleável para não dificultar desnecessariamente o trabalho da equipe.
f) *Gerenciamento de recursos humanos*. São as atividades de aquisição, dispensa, formação e motivação da equipe, bem como de alocação de funções e relações hierárquicas.
g) *Gerenciamento das comunicações*. Trata-se de controlar as comunicações internas e externas ao projeto.
h) *Gerenciamento de riscos*. Uma das áreas mais importantes da gerência de projetos, que implica em acompanhar o nível de probabilidade e impacto dos riscos e tomar medidas para diminuí-los.
i) *Gerenciamento de aquisições*. São as atividades relacionadas à aquisição de produtos ou serviços necessários ao projeto que não sejam produzidos ou fornecidos pela equipe de desenvolvimento.

Ao invés de apresentar as combinações no formato de tabela[112] ou mesmo por área de conhecimento, as listas seguintes apresentam o que se espera do gerente de projeto nas várias áreas em cada um dos cinco momentos de um projeto.

No momento de *iniciação* de um projeto, apenas as áreas de integração e comunicação terão atividades relacionadas:

a) Atividade de Integração: desenvolver o termo de abertura do projeto
b) Atividade de comunicação: identificar as partes interessadas.

No momento de *planejamento* (Capítulo 6), todas as nove áreas terão atividades relacionadas:

a) *Atividade de gerenciamento de integração*: desenvolver o plano de gerenciamento do projeto (Seção 6.1).
b) *Atividades de gerenciamento de escopo*: coletar os requisitos do projeto (não do software), definir o escopo, criar a estrutura analítica do projeto (WBS)[113], conforme explicado na Seção 6.5.1. No planejamento por ciclos iterativos, a criação da WBS só ocorre no planejamento das iterações, não do projeto.
c) *Atividades de gerenciamento de tempo*: definir as atividades, sequenciar as atividades, estimar recursos das atividades, estimar a duração das atividades e desenvolver o cronograma (Seção 6.5). No planejamento por ciclos iterativos, essas atividades também são deixadas para o planejamento dos ciclos, embora existam atividades de

[112] pt.wikipedia.org/wiki/Project_Management_Body_of_Knowledge
[113] Em inglês: WBS – Work Breakdown Structure.

planejamento de tempo para o projeto como um todo (número e duração das iterações, por exemplo).

d) *Atividades de gerenciamento de custos*: estimar os custos e determinar o orçamento.
e) *Atividades de gerenciamento de qualidade*: planejar a qualidade.
f) *Atividades de gerenciamento de recursos humanos*: desenvolver o plano de recursos humanos.
g) *Atividades de gerenciamento de comunicação*: planejar as comunicações.
h) *Atividades de gerenciamento de riscos*: planejar o gerenciamento de riscos, identificar os riscos, realizar a análise qualitativa dos riscos, realizar a análise quantitativa dos riscos e planejar as respostas aos riscos (Capítulo 8).
i) *Atividades de gerenciamento de aquisição*: planejar as aquisições.

No momento da *execução* de um projeto, cinco das nove áreas terão atividades relacionadas:

a) *Atividades de gerenciamento de integração*: orientar e gerenciar a execução do projeto.
b) *Atividades de gerenciamento de qualidade*: realizar a garantia de qualidade.
c) *Atividades de gerenciamento de recursos humanos*: mobilizar a equipe do projeto, desenvolver a equipe do projeto e gerenciar a equipe do projeto.
d) *Atividades de gerenciamento de comunicação*: distribuir as informações, gerenciar as expectativas das partes interessadas.
e) *Atividades de gerenciamento de aquisição*: conduzir as aquisições.

O *monitoramento e controle* não é exatamente um momento distinto do momento de execução, mas ocorre simultaneamente a este. Em relação ao monitoramento e controle, as seguintes áreas terão atividades relacionadas:

a) *Atividades de gerenciamento de integração*: monitorar e controlar o trabalho do projeto e realizar o controle integrado de mudanças.
b) *Atividades de gerenciamento de escopo*: verificar o escopo e controlar o escopo.
c) *Atividades de gerenciamento de tempo*: controlar o cronograma.
d) *Atividades de gerenciamento de custos*: controlar os custos.
e) *Atividades de gerenciamento de qualidade*: realizar o controle de qualidade.
f) *Atividades de gerenciamento de riscos*: monitorar e controlar os riscos.
g) *Atividades de gerenciamento de aquisição*: administrar as aquisições.

O momento do *encerramento* de um projeto necessita de atividades gerenciais nas seguintes áreas:

a) *Atividades de gerenciamento de integração*: encerrar o projeto ou fase.
b) *Atividades de gerenciamento de comunicação*: reportar o desempenho.
c) *Atividades de gerenciamento de aquisição*: encerrar as aquisições.

Essa organização das atividades mostra como é complexo e multidimensional o trabalho do planejador e do gerente de software. Convém que mesmo que as atividades não sejam efetivamente detalhadas e executadas em momentos distintos, o gerente (especialmente o

iniciante) revise, a cada dia, suas realizações, para verificar se todas as dimensões de sua responsabilidade foram de alguma forma realizadas.

9.3 Prince2 – *Projects IN Controlled Environments* 2

Projects IN Controlled Environments[114] 2 ou, simplesmente, *Prince*2 é um método estruturado de gerência de projetos aceito como padrão de gerenciamento de projetos pelo governo do Reino Unido[115]. Prince2 foi lançado em 1996, como sucessor de outros métodos mais antigos. Desde 2006 o método tem sido revisado e atualizado, além de estar se tornando um pouco mais leve. A versão atual é conhecida como "Prince2:2009" (Turley, 2010)[116].

Prince2 é um método prescritivo, que não se aplica bem ao uso com modelos ágeis. Ele pode ser bem aplicado com modelos de processos mais prescritivos. Ele é baseado em sete princípios[117]:

a) *Justificação continuada de negócio*. Isto significa que existe uma justificativa para iniciar o projeto, e que essa justificativa continua válida ao longo do tempo de vida do projeto, tendo sido documentada e aprovada. Em Prince2, a justificativa é documentada no *caso de negócio*, que dirige o processo de tomada de decisão e garante que o projeto permaneça alinhado com os objetivos e benefícios de negócio.
b) *Aprender com a experiência*. Espera-se que as equipes e projeto aprendam com a experiência. Lições aprendidas são identificadas, registradas e praticadas ao longo do ciclo de vida do projeto. Em Prince2 as lições aprendidas são registradas no *lessons log*, ou *diário de lições* e se tornam parte do *relatório de lições aprendidas* preparado pelo gerente de projeto ao final de cada estágio e ao final do projeto.
c) *Papeis e responsabilidades definidos*. Todos os projetos tem papeis e responsabilidades definidos e acordados, engajando todos os aspectos das organizações envolvidas interna e externamente. Os papeis devem ser definidos de forma que cada participante saiba exatamente o que se espera dele.
d) *Gerenciar por estágios*. O gerenciamento de projetos em Prince2 é compatível com os modelos baseados em ciclos iterativos. Um ciclo é chamado de "*estágio*" em Prince2. Ao final de cada estágio o projeto é revisado para verificar se atingiu seus objetivos e vai produzir a entrega prevista no caso de negócio. Isso é feito pelo uso de dois níveis de planejamento: longo prazo e curto prazo, sendo o segundo bem mais detalhado do que o primeiro.
e) *Gerenciar por exceção*. Os projetos Prince2 definem limites de tolerância para tempo, custo, qualidade, escopo e risco. Esses limites são usados para definir os níveis de autoridade delegada. Isso permite que a gerência possa ser feita dentro de um processo de gerência por exceção, ou seja, se esses limites de tolerância forem excedidos, ou se for previsto que serão excedidos, então um nível superior de gerência deve ser acionado para decidir como proceder.
f) *Foco nos produtos*. Prince2 foca na definição e entrega de produtos que satisfazem os critérios estabelecidos de qualidade. Isso inclui o produto final de um projeto, bem

[114] Projetos em ambientes controlados.
[115] www.prince-officialsite.com/
[116] upload.wikimedia.org/wikipedia/commons/8/87/The_PRINCE2_Process_Model_Book.pdf
[117] project-management.learningtree.com/2010/04/03/the-7-prince2-principles-a-closer-look/

como outros subprodutos significativos gerados ao longo do ciclo de vida do projeto. Esta abordagem orientada ao produto resulta na definição de produtos sobre os quais se obtêm concordância, e depois planejar as atividades para obter esses produtos. Isso é obtido pelo uso de técnicas de planejamento baseadas em produto.

g) *Personalização para se ajustar ao ambiente de trabalho*. Prince2 deve ser personalizado para se adequar ao ambiente de trabalho, ou seja, o tamanho, complexidade, importância, capacidades e riscos do projeto. Porém, quando se está personalizando o método, é importante não omitir nenhuma parte, pois todas são interligadas.

Além disso, o método também possui sete temas[118], os quais descrevem aspectos críticos do gerenciamento de sistemas e que devem ser considerados pelo gerente ao longo de todo o ciclo de vida do projeto:

a) *Caso de negócio*. O caso de negócio dirige toda a tomada de decisões ao longo do projeto. Ele é criado no início do projeto e deve justificar o investimento inicial e continuado feito no projeto. Ele permite que o gerente de equipe defina a qualquer momento se o projeto é viável, desejável e factível. O caso de negócio é mantido atualizado ao longo do projeto com as atualizações referentes a riscos, benefícios e custos.
b) *Organização*. Este tema estabelece os papeis e responsabilidades ligados ao projeto. Os projetos Prince2 são organizados em quatro níveis de decisão: *corporação* ou *gerenciamento de programa*, responsável pela encomenda do projeto, a *diretoria* (*project board*), que fornece direção e gerenciamento geral (*governance*), a *gerência de projeto* que lida com as questões do dia-a-dia do projeto, e a *gerência de equipe* que se ocupa com a entrega dos produtos do projeto.
c) *Qualidade*. Esse tema procura garantir que o produto atenda ao seu propósito. A abordagem definida na estratégia de gerenciamento de qualidade requer que exista um entendimento explícito sobre o escopo do projeto, bem como critérios de qualidade contra os quais o produto será avaliado. Em outras palavras, o foco da qualidade está em o produto atender aos requisitos (especialmente os de qualidade). O método possui duas formas de aplicação de controle de qualidade: *in process*, que implica em construir os produtos com qualidade ao longo do projeto, e *appraisal*, usado para verificar se os produtos acabados satisfazem os critérios de qualidade.
d) *Planos*. Planos são usados para definir como, quando e por quem os produtos serão gerados. Planos são usados para permitir e facilitar a comunicação efetiva e o controle. Eles devem estar alinhados com o caso de negócio e precisam ser aprovados por todos os níveis de gerenciamento. Existem três níveis de planos do Prince2, *plano de projeto*, *plano de estágio* e, opcionalmente, *plano de equipe*. Adicionalmente, planos de exceção poderão substituir um dos outros planos em situações onde os níveis de tolerância de gerenciamento forem excedidos. Prince2 requer que o planejamento seja orientado ao produto, estabelecendo que primeiramente o produto do trabalho deve ser decidido; somente depois as atividades, dependências e recursos podem ser determinados, sempre com o objetivo de produzir o produto especificado.

[118] project-management.learningtree.com/2010/04/11/successfully-juggling-the-7-prince2-themes/

e) *Riscos*. Para Prince2 os riscos são incertezas que podem ser tanto positivas (oportunidades) quanto negativas (ameaças). O gerenciamento de risco trata da identificação proativa, avaliação e controle dos riscos do projeto, de forma a maximizar as chances de sucesso. A estratégia de gerenciamento de riscos, definida durante a iniciação do projeto possui cinco passos: *identificar* a causa, evento e efeitos do risco, *analisar* sua probabilidade e impacto, *planejar* as respostas necessárias, *implementar* as respostas como requerido e *comunicar* os riscos interna e externamente.
f) *Mudança*. Este tema envolve a gerência de configuração, problemas e solicitações de mudança. Mais detalhes sobre essas disciplinas no Capítulo 10 e na seção 9.4.2.
g) *Progresso*. Este tema se refere aos mecanismos usados para monitorar e comparar o estado atual do projeto em relação aos planos. Este mecanismo também permite que o gerente de projeto faça previsões de performance futura baseadas nas tendências atuais, e assim, tome ações proativas visando alternativas de projeto quando necessário.

Os sete princípios e sete temas aparecem juntos em sete processos[119]:

a) *Dar partida a um projeto*. Cujas atividades incluem: indicar um executivo e um gerente de projeto, planejar e indicar uma equipe de gerenciamento de projeto, preparar um resumo executivo do projeto, definir a abordagem do projeto e planejar para o próximo estágio.
b) *Iniciar um projeto*. Cujas atividades incluem: planejar qualidade, planejar o projeto, refinar o caso de negócios e riscos, definir os meios de controle do projeto, definir os arquivos do projeto e montar o documento de início de projeto.
c) *Dirigir um projeto*. Cujas atividades incluem: autorizar o início do projeto, autorizar um estágio ou plano de exceção, dar orientações *ad-hoc* e confirmar o fechamento do projeto.
d) *Controlar um estágio*. Cujas atividades incluem: autorizar um pacote de trabalho, avaliar o progresso, capturar e examinar assuntos do projeto, revisar o *status* do estágio, relatar destaques, tomar ação corretiva e receber um pacote de trabalho completo.
e) *Gerenciar limites de estágios*. Cujas atividades incluem: planejar um estágio, atualizar um plano de projeto, atualizar um caso de negócio, atualizar a lista de riscos, relatar o final de um estágio e produzir um plano de exceção.
f) *Gerenciar entrega de produto*. Cujas atividades incluem: aceitar um pacote de trabalho, executar o trabalho de um pacote e entregar um pacote de trabalho.
g) *Fechar um projeto*. Cujas atividades incluem: desmontar (*decomissioning*) o projeto, identificar ações posteriores e revisar a avaliação do projeto.

O método Prince2 estipula três níveis de certificação: *foundation*, para os que aprenderam os fundamentos do método, *practicioneer*, para aqueles que vão gerenciar projetos usando o

[119] www.prince2.com/prince2-process-model.asp

método e *certification*, para aqueles que buscam ainda mais competência em gerenciamento de projetos[120].

9.4 Condução de Projeto de Software

Após o planejamento do projeto de software, ele deve ser executado durante um período de tempo muitas vezes longo. É o papel do gerente garantir, então, que as variáveis de tempo, recursos, qualidade e escopo sejam mantidas nos valores esperados.

Após o planejamento, espera-se que riscos já tenham sido avaliados, e existam planos para mitigá-los, responsáveis nomeados, recursos alocados e que um processo de desenvolvimento já esteja perfeitamente definido. Só resta executar o projeto.

E pode ser que nesta hora tudo comece a dar errado.

Um projeto, mesmo bem planejado pode falhar por vários motivos: erros da equipe, erros no próprio projeto, erros na concepção do processo ou ainda fatores imprevistos. Isso decorre principalmente do fato de que os executores do projeto são pessoas, e não máquinas, e que um projeto não é um programa de computador que roda de forma previsível. Muitos fatores de incerteza estão envolvidos mesmo nos projetos mais bem planejados e gerenciados.

Staa (2003)[121] indica que um dos maiores desafios que o gerente de projeto enfrenta no momento do acompanhamento da execução de um projeto é a *indisciplina*. Membros da equipe poderão não seguir os padrões, prazos ou prioridades estabelecidos. O folclore da ciência da computação apresenta os "gênios" como super-desenvolvedores que não seguem regras, não tem horários preestabelecidos, que são desorganizados e muitas vezes com problemas de higiene pessoal. Se não houver disciplina no ambiente de trabalho, estes "heróis" poderão facilmente ressurgir e o projeto será refém deles.

Não se deve confundir os métodos ágeis, mesmo os mais radicais como XP, com indisciplina e caos. Para que um modelo como XP funcione na prática, a equipe deve ser muito bem disciplinada. No caso de *Scrum*, a equipe deve ser necessariamente auto-disciplinada, pois ela se auto-gerencia.

Então o uso de métodos ágeis não é desculpa para fazer as coisas de qualquer maneira. Mesmo que os desenvolvedores trabalhem com criatividade, buscando soluções inovadoras e às vezes ousadas para os problemas, ainda assim, eles devem entender que existem as prioridades de projeto que, ou são discutidas e mudadas, ou são mantidas e seguidas.

Além disso, existem padrões a serem seguidos. O desenvolvedor baixa uma versão do componente no sistema de controle de versões e tem total liberdade para trabalhar em sua cópia. Mas no momento de salvar uma nova versão no sistema, o que ele colocar ali vai afetar o restante da equipe. Então essa versão precisa estar de acordo com os padrões estabelecidos, testada e estabilizada.

É responsabilidade do gerente de projeto, então, identificar se há algum tipo de desvio nocivo e motivar os desenvolvedores, conscientizando-os da necessidade de trabalhar em harmonia,

[120] www.prince2.com/what-is-prince2.asp
[121] ftp://ftp.inf.puc-rio.br/pub/docs/techreports/03_13_staa.pdf

lembrando que *harmonia* não significa cada um fazer o que quer, significa todos seguirem as regras que o próprio grupo estabeleceu.

Staa identifica três perfis de gerência frente à equipe:

a) *Ditador*. É o mais danoso dos três tipos. Ele faz todo o planejamento e estimativas. Ele sozinho determina quem faz o que e quando. Ele gera belíssimos diagramas Gantt que ninguém leva a sério, e usa de força e ameaça para fazer projetos que não estão indo bem voltar para os trilhos, mas nem sempre consegue.
b) *Coordenador*. Ele ouve os outros e faz as previsões e planejamento em conjunto com a equipe. Ele faz revisões do planejamento periodicamente. Seus diagramas Gantt são levados mais a sério, pois representam um compromisso realista da equipe e não apenas uma determinação do gerente. Seus projetos têm maior chance de terminar no prazo. Este tipo de gestão usualmente funciona bem com métodos prescritivos.
c) *Facilitador*. Ele apenas agiliza e facilita do trabalho da equipe, mas não toma as decisões. A própria equipe define prazos e o planejamento. Este tipo de gerência funciona bem com os métodos ágeis.

Staa (2003) apresenta os três objetivos de acompanhamento de projetos, segundo CMMI, e acrescenta um quarto:

a) "Acompanhar os resultados e desempenhos reais confrontando-os com o plano de desenvolvimento de software."
b) "Tomar ações corretivas e gerenciá-las até sua conclusão, sempre que resultados ou desempenhos reais desviarem significativamente do que foi estabelecido (estimado) no plano de desenvolvimento de software."
c) "Assegurar que as alterações nos compromissos de software se dêem através de acordo entre as pessoas e os grupos envolvidos."
d) "Acompanhar processos e meta-processos obtendo indicadores quanto à sua eficácia em instanciar planos e processos."

Os dois primeiros objetivos mencionados acima estão relacionados com o dia-a-dia do gerente de projeto, e correspondem às atividades de verificação e controle por ações corretivas. O terceiro objetivo está relacionado às mudanças eventualmente necessárias no plano do projeto. O quarto objetivo relaciona-se com as necessárias mudanças no próprio processo, quando for detectado que este não é suficientemente adequado.

Finalmente convém mencionar que um plano mal feito será difícil de gerenciar. Assim, é fundamental que no momento do planejamento de um projeto as técnicas apresentadas no Capítulo 6 sejam observadas. Por exemplo, se o plano não prevê artefatos prontos em determinadas datas, fica difícil o gerente avaliar se as atividades previstas foram efetivamente realizadas. Como a verificação do gerente tende a ser pontual (ele avalia o *status* do projeto ou de cada desenvolvedor em determinados instantes do tempo), ele só será capaz de avaliar o estado e qualidade de artefatos produzidos, mas não diretamente das atividades em si.

Staa (2003) apresenta três formas de acompanhamento de projetos de software: folha de tempo, acompanhamento de problemas e registro de artefatos, as quais são resumidas nas seguintes subseções.

9.4.1 Folha de Tempo

A *folha de tempo* (*timesheet*) é uma forma de manter o registro das ações dos desenvolvedores ao longo do tempo, visando estudo para acompanhamento de desempenho e reavaliação de estimativas e do processo em si.

A folha de tempo deve conter as ações nas quais o desenvolvedor se envolveu ao longo de um dia, como, escrever um método, revisar um diagrama, entrevistar usuário, participar de reunião, resolver problemas pessoais, etc. Não se deve confundir aqui essas ações com a atividade estruturada de um processo, a qual tem artefatos de entrada e saída bem definidos.

As atividades de projeto usualmente serão realizadas ao longo de um ou mais dias. Então não são elas que são listadas na folha de tempo, mas as ações concretas que são de fato executadas pelo desenvolvedor, independentemente de gerarem ou não um artefato de saída.

A folha de tempo será uma excelente ferramenta não só para o gerente acompanhar o andamento do projeto, mas também para analisar possíveis erros de estimativa de esforço, o que poderá melhorar a capacidade de previsão da equipe em projetos futuros. Além disso, é uma ferramenta de auto-reflexão, onde o desenvolvedor efetivamente registra em quê gastou suas horas.

Além de registrar as ações desenvolvidas ao longo do dia, o desenvolvedor ainda poderá adicionar:

a) Um resumo das ações do dia e das dificuldades encontradas. Nos métodos ágeis essa informação normalmente é passada oralmente nas reuniões em pé.
b) Para cada ação, indicar sua natureza a partir de uma lista previamente selecionada (por exemplo, codificação, teste, depuração, planejamento, revisão, etc.). Essa informação poderá ser usada posteriormente para verificar quais os tipos de ações que mais tomam tempo dos desenvolvedores, o que permitirá tomar providências para a correção de gargalos, se for o caso.
c) Para cada ação, indicar o artefato alterado, se for o caso. Espera-se, a princípio que uma ação do desenvolvedor só vá alterar os artefatos que correspondem a saídas da atividade de projeto em que ele está trabalhando, mas essa ação poderá ter vários artefatos relacionados. Por outro lado, se cada atividade de projeto tem apenas um artefato de saída, essa anotação dos artefatos para cada ação poderá ser redundante.
d) Para cada ação, indicar o conjunto de artefatos consultados. Embora isso adicione ainda mais peso à folha de tempo, essa informação poderá ser útil no futuro para que se verifique quais foram efetivamente os artefatos de entrada para cada atividade de projeto. Se forem consultados artefatos que não estavam previstos como entradas da atividade de projeto, então possivelmente alguma coisa está errada na descrição da atividade ou nas ações do desenvolvedor.
e) Para cada ação, referenciar uma *ficha de acompanhamento de problema* (*FAP*), se necessário (ver seção seguinte).

Assim, uma folha de tempo poderá se parecer com a imagem da Figura 9-1.

Projeto: Atividade de projeto: Funcionário: Data:						
Resumo das ações do dia:						
Lista de problemas encontrados:						
Ações						
Inicio	Fim	Natureza	Descrição	Artefato alterado	Artefatos consultados	FAP

Figura 9-1: Uma folha de tempo (*timesheet*).

Embora a folha de tempo possa ser feita em papel, o ideal, para permitir uma gerência mais eficiente, é que ela seja automatizada por uma ferramenta. Embora seja um sistema tão simples que muitas empresas acabam implementando as suas próprias soluções, alguns produtos podem ser encontrados. Entre as ferramentas gratuitas pode-se citar: *Journix Timesheet*[122], *WR Time Tracker* [123], *HourGuard Time Sheet*[124] e *Timesheet4J*[125]. Muitas outras, inclusive comerciais podem ser encontradas na Internet.

9.4.2 Acompanhamento de Problemas

Uma das grandes diferenças entre o processo de desenvolvimento de software e outros processos de engenharia é que ele usualmente não pode ser totalmente definido a priori. Como novos requisitos são descobertos ou modificados ao longo do próprio projeto, pode-se considerar que o desenvolvimento de software é, antes de tudo, um processo de amadurecimento. Durante este processo, riscos e dúvidas surgirão e com eles, *problemas*, que precisam ser resolvidos e rastreados.

Além da garantia de solução dos problemas, um controle eficiente de problemas pode ser útil também para identificar a causa dos problemas, desta forma eles podem passar a ser evitados no futuro. Além disso, essa analise também poderá identificar problemas no próprio processo, que então poderá ser ajustado.

A identificação inicial de um problema normalmente tem duas fontes: os desenvolvedores e os usuários. Quando o problema é apontado pelo usuário, existem as seguintes possibilidades:

a) *Trata-se de um problema urgente que deve ser resolvido de forma emergencial*. Neste caso, ele é imediatamente repassado a uma equipe, que vai parar o que estiver fazendo para achar uma solução para o problema (esta não pode ser a regra para tratar todos os problemas, porém).

[122] www.journyx.com/
[123] www.anuko.com/content/time_tracker/
[124] hourguard-timesheet-software.soft32.com/old-version/142705/1.37
[125] code.google.com/p/timesheet4j/

b) *Trata-se de um problema real, mas não urgente*. Neste caso o problema vai gerar uma entrada na lista de problemas conhecidos da versão corrente, e também na lista de modificações solicitadas. Uma próxima versão do artefato vai possivelmente resolver o problema.
c) *Trata-se de um problema real, porém já resolvido numa versão mais atual do que a que o usuário possui*. Neste caso, o usuário deve ser orientado a atualizar sua versão do software. Uma ferramenta interessante para armazenar problemas e soluções, que permite a criação de um banco de dados de problemas e consulta inteligente é o *Responsive Knowledgebase*[126].
d) *Não se trata de um problema real*. Neste caso o usuário deve ser orientado sobre como proceder ou como compreender o sistema.

Uma *FAP* (*Folha de Acompanhamento de Problema*) é um documento ou entrada em um sistema que vai indicar um problema identificado, sua origem (cliente ou desenvolvedores), sua localização (quem ou qual setor da empresa está com o problema sob sua responsabilidade) e seu estado. Usualmente, os estados possíveis de um problema são:

a) Aguardando análise.
b) Em análise.
c) Aguardando solução.
d) Em solução.
e) Aguardando informação complementar.
f) Aguardando aprovação.
g) Aprovado/resolvido.

Aceita-se usualmente que o fato de que um problema tenha sido analisado e tratado não significa necessariamente que tenha sido resolvido. Assim, uma análise, com fins de aprovação da resolução do problema, deve ser feita ao final dos procedimentos de correção. Quando a solução final for aceita, necessariamente haverá a geração de uma nova versão do artefato no sistema de controle de versões.

Como o acompanhamento de problemas deve ser feito por diferentes interessados, é natural que acabe sendo feito por um sistema informatizado. Um exemplo de software nacional gratuito para controle de problemas (manutenção) é o *SIGMA*[127].

9.4.3 Registro de Artefatos

O processo de acompanhamento de projeto e controle de problemas depende de forma crucial do sistema gerenciador de configuração, ou controle de versões. O gerente deve se certificar de que todos os membros da equipe sigam corretamente a política de controle de versão estabelecida na empresa.

Uma política típica é que os desenvolvedores somente tenham acesso a um artefato quando liberado pelo seu proprietário. Eles podem trabalhar na cópia do artefato até realizar a tarefa a que se propuseram. Depois, submetem o artefato modificado ao controle de qualidade,

[126] www.responsivesoftware.com/rkb.htm
[127] www.redeindustrial.com.br/site/empresa.aspx

usualmente na forma de testes. Se aprovado, salvam uma nova versão do artefato no sistema de gerenciamento de configuração. Mais detalhes sobre este processo no Capítulo 10.

9.5 Medição em Engenharia de Software

Um processo de gerência, para ser mais eficaz, precisa se basear em medições. Como saber se a tarefa não está sendo feita se não se tem uma medida para chegar a essa conclusão? Como saber que a qualidade é inaceitável? Então, de uma maneira ou de outra, o gerente de projeto vai acabar se envolvendo com a atividade de medição de software, para a qual terá que aplicar uma ou mais métricas.

Inicialmente, alguns termos serão definidos, para evitar confusão:

a) *Medida*: é um valor obtido para alguma dimensão do software.
b) *Métrica*: é a escala na qual os valores de uma medida são tomados.
c) *Medição*: é o processo de obtenção de medidas.

O *Software Engineering Institute* (Mills E. E., 1988) indica que uma boa métrica precisa ter cinco qualidades:

a) Ser *simples*, ou seja, ter uma definição curta e fácil de ser compreendida.
b) Ser a mais *objetiva* possível, isto é, não deve depender de opiniões e, se duas pessoas avaliarem o mesmo produto usando essa métrica, devem obter o mesmo resultado.
c) Ser *facilmente obtida*, isto é, o custo para efetuar a medição deve ser razoavelmente baixo.
d) Ser *válida*, isto é, a métrica deve indicar um valor que seja útil e que seja efetivamente representativo da grandeza que se pretende medir.
e) Ser *robusta*, isto é, pequenas mudanças no produto devem produzir mudanças proporcionais na medida; apenas grandes mudanças no produto podem produzir grandes mudanças na medida.

Apesar disso, nem sempre as métricas apresentam todas essas qualidades. Um bom exemplo de métrica é a contagem de linhas de código (LOC - Seção 7.1). Desde que os padrões de contagem tenham sido estabelecidos (o que efetivamente conta e o que não conta), essa métrica deve produzir sempre o mesmo resultado. Já a contagem de pontos de função (Seção 7.3.3) a partir de um conjunto de requisitos poderá apresentar pequenas variações dependendo da interpretação do analista sobre o que efetivamente se constitui em uma função individual do software ou não. Mais ainda, a avaliação dos fatores técnicos e ambientais, por exemplo, em Pontos de Caso de Uso (Seção 7.5), é uma métrica bastante subjetiva. O método COCOMO II (Seção 7.3) tenta reduzir essa subjetividade nos multiplicadores de esforço e fatores de escala ao estabelecer uma série de padrões para atribuição de notas; mas ainda assim, a métrica tem uma significativa carga de subjetividade.

A Seção 11.5 apresenta mais alguns detalhes sobre medição e métricas referentes a qualidade de software.

9.5.1 Classificações de Métricas

Existem várias classificações para métricas. Pode-se falar inicialmente em métricas diretas, ou seja, aquelas que podem ser objetivamente definidas e contadas, sem necessidade de interpretação ou incerteza, como por exemplo:

a) Custo financeiro.
b) Esforço em desenvolvedor-hora.
c) Linhas de código (SLOC).
d) Velocidade de execução em segundos.
e) Memória em megabytes.
f) Número de defeitos localizados (total ou relativo ao número de KSLOC).
g) Complexidade ciclomática (ver Seção 13.4.1).

Outras métricas são indiretas e só podem ser definidas a partir de uma definição operacional (Wazlawick, 2009, p. 109), ou seja, um procedimento de medição, o qual, algumas vezes é *ad-hoc*, e nem sempre será consenso. Exemplos de métricas indiretas são:

a) Funcionalidade.
b) Qualidade.
c) Complexidade.
d) Eficácia.
e) Confiabilidade.
f) Manutenibilidade.
g) Usabilidade.

Nota-se que as métricas mais importantes estão fortemente ligadas aos aspectos de qualidade do software (Seção 11.1).

Um aspecto que sempre deve ser lembrado é que só vale a pena coletar medições quando se tem um propósito específico em mente, pois a coleta de dados poderá acarretar trabalho extra antes, durante e depois do projeto de desenvolvimento. Assim, a coleta de medições deve ser encarada como um investimento de tempo e esforço com o objetivo de melhorar algum aspecto do produto ou do processo de desenvolvimento que precisa ser melhorado.

Do ponto de vista dos processos de gerência, poderá ser interessante agrupar as métricas em termos de sua utilidade para o gerente. Assim, tem-se:

a) *Métricas de produtividade*, como custo, esforço (em AFP ou UCP) e KSLOC, que podem ser usadas para verificar o andamento do projeto e possíveis desvios.
b) *Métricas de qualidade*, como número de defeitos, qualidade, eficiência, confiabilidade e manutenibilidade. São usadas para avaliar se o produto satisfaz critérios de aceitação para uso por parte do cliente e também critérios internos, que afetam a eficiência da equipe.
c) *Métricas técnicas.* São outros aspectos ligados ao produto, não necessariamente ligadas nem a qualidade, nem a produtividade, mas a aspectos inerentes do sistema, como complexidade ciclomática, modularidade, paralelismo, distribuição, etc.

As métricas podem ser *absolutas* ou *relativas*. Por exemplo, um sistema com cinco erros não é necessariamente pior do que um sistema com dois erros. Se o primeiro sistema tiver um milhão de linhas e o segundo cinco mil linhas então o segundo terá mais erros por linha do que o primeiro.

A medida relativa, então, frequentemente é usada, especialmente quando se quer avaliar a qualidade do produto e do trabalho. Há pelo menos quatro formas relevantes para se relativizar uma métrica:

a) *Pelo tamanho*. Neste caso, divide-se o valor absoluto da métrica pelo número de linhas de código.
b) *Pela funcionalidade*. Neste caso, divide-se o valor absoluto da métrica pelo número de pontos de função, pontos de caso de uso ou pontos de história.
c) *Pelo tempo*. Neste caso, divide-se o valor absoluto pelo período de tempo. Por exemplo, número de defeitos detectados por mês.
d) *Por esforço*. Neste caso, divide-se o valor absoluto pelo esforço despendido, usualmente em desenvolvedor-mês ou desenvolvedor-hora. Por exemplo, número de linhas de código produzidas por desenvolvedor-mês.

Talvez o trabalho mais formal na área de métricas esteja relacionado à medição da complexidade do software, o que é útil para a área de testes (Capitulo 13). Outro tipo de métrica que recebeu muita atenção são as métricas de qualidade ligadas ao produto (Capitulo 11), embora, às vezes, elas sejam contraditórias entre si. Por exemplo, aumentar a flexibilidade (desejável), pode diminuir a eficiência de tempo (indesejável).

Porém, algumas métricas de qualidade poderão ser especialmente úteis ao gerente de projeto, pois melhorar suas medições quase sempre será muito salutar:

a) *Métricas de defeitos.* O número de defeitos em um produto de software deveria ser uma métrica objetivamente contável. Porém, encontrar defeitos não é uma tarefa trivial. Assim, normalmente a medição de defeitos é feita por uma das seguintes técnicas:
    a. *Número de alterações de design ou código.*
    b. *Número de erros detectados nas inspeções de código* (Seção 11.4.2).
    c. *Número de erros detectados nos testes de programa* (preferencialmente pela equipe de teste, mas também, possivelmente, pelos usuários).
b) *Métricas de confiabilidade.* Uma vez que se tenha uma medida nominal dos defeitos encontrados em um produto de software, sua confiabilidade pode ser calculada para um determinado período de tempo. Se um sistema complexo apresenta uma determinada taxa de falhas ao longo de um mês ou um ano, pelas Leis de Lehman (Seção 14.1), pode-se esperar que continue exibindo esse comportamento ao longo dos meses ou anos seguintes. Essas medidas podem ser tomadas tanto durante o processo de desenvolvimento quanto durante o período de operação pós-desenvolvimento do sistema.
c) *Métricas de manutenibilidade.* Embora a manutenibilidade seja, a princípio, uma métrica subjetiva, existem estudos que mostram que a complexidade do produto (complexidade ciclomática, por exemplo), afeta o esforço necessário para encontrar e

reparar defeitos no software. Assim, existe uma relação direta entre o número de defeitos encontrados e a complexidade do software com o esforço necessário para fazer manutenções. Novamente, essas atividades de manutenção podem ser tanto aquelas que ocorrem durante a operação do software, quando as modificações que ocorrem durante o desenvolvimento.

Deve-se ficar atento, porém, que essas métricas precisam ser sempre interpretadas em seu contexto. Por exemplo, ser capaz de detectar muitos defeitos no software pode significar tanto que a atividade de teste está sendo bem conduzida, quanto significar que as atividades de programação estão sendo mal conduzidas. Mas defeitos detectados por usuários durante o uso do sistema usualmente são péssimas notícias.

9.5.2 Planejamento de um Programa de Métricas

Para que um gerente possa se utilizar de métricas para dirigir suas atividades, ele deve primeiro implantar um *programa de métricas*. Segundo o SEI (Mills E. E., 1988), essa atividade deve ser muito bem planejada. Inicialmente, deve-se estabelecer os objetivos do programa de métricas: quais falhas ele vai corrigir, quais aspectos ele vai tentar melhorar?

Em seguida, devem ser estimados os custos do programa e ser obtido o apoio da gerência superior, pois este tipo de ação normalmente precisa de um suporte razoável para ocorrer. Em relação aos custos, estes podem ser divididos em custos iniciais de implantação e custos de manutenção do programa.

Em função dos objetivos e do aporte financeiro inicialmente comprometido, deve-se escolher o conjunto de métricas a serem implementadas e o modelo de avaliação (por exemplo, para estimar o tamanho de um projeto pode-se usar os modelos CII, Pontos de Função ou Pontos de Caso de Uso, cada um com suas peculiaridades e custos específicos). Durante o processo de seleção das métricas e modelos deve-se observar os seguintes pontos:

a) *Habilidade projetada para satisfazer objetivos*. As métricas e modelos disponíveis devem ser analisados e comparados em relação a sua capacidade de satisfazer os objetivos do programa de métricas.
b) *Dados e custos necessários estimados*. Os modelos que são capazes de satisfazer os objetivos devem ser comparados em função de seu custo de implantação e manutenção e da quantidade e variedade de dados necessários para funcionarem. Modelos mais econômicos normalmente são preferíveis.

Uma vez que o modelo tenha sido escolhido, deve-se identificar e refinar os dados que serão coletados. Apenas dados que possam satisfazer a algum objetivo imediato ou de longo prazo devem ser coletados. Deve-se evitar coletar quaisquer dados só porque estão disponíveis, pois o excesso de dados pode causar confusão quando se tenta fazer algum tipo de análise. Para executar esta atividade, deve-se observar o seguinte:

a) *Especificidade dos dados.* Os dados devem ser definidos e obtidos ao longo de todo o ciclo de desenvolvimento do software. Preferencialmente, deve-se identificar *quando* os dados foram obtidos. Isto permite analisar diferentes significados em diferentes fases ou para diferentes atividades do processo de desenvolvimento.

b) *Procedimentos de obtenção de dados*. Uma vez que os dados específicos tenham sido definidos, os procedimentos para sua coleta e as pessoas responsáveis devem ser identificadas.
c) *Manutenção do banco de dados*. Uma vez que o banco de dados de medições passará a ser um importante patrimônio da empresa, os recursos para sua perfeita manutenção devem ser definidos e destinados.
d) *Previsões refinadas de esforço e custo.* Com as informações obtidas nos itens anteriores deve ser agora possível obter uma estimativa bem mais realista de custo e esforço para implantação do programa de métricas.

Assumindo que todos os passos anteriores foram executados com sucesso e os custos são aceitáveis, o programa de métricas pode ser iniciado. Os seguintes itens ainda precisam ser enfatizados nesta fase:

a) *Esclarecimento de uso*. Os objetivos do programa de métricas devem ficar claros desde o início. Mas no momento de iniciar seu uso pode ser importante relembrá-los a toda a equipe. As pessoas devem ser informadas sobre as medidas que serão obtidas e o uso que será feito delas. Muito cuidado deve ser tomado especialmente se as medidas forem utilizadas para avaliação de membros da equipe, sem uma ampla discussão e aceitação das métricas, estas iniciativas poderão dar origem a *stress* e até sabotagens ao programa.
b) *Pessoal responsável*. Os responsáveis pela coleta, manutenção e interpretação dos dados devem ser definidos e informados. Deve-se lembrar que essa atividade deve ser executada continuamente ao longo de qualquer projeto, pois muitos dados não podem mais ser obtidos depois que o projeto termina.

Para que o programa de métrica tenha sucesso, ele deve ser continuamente usado, avaliado e reajustado. Os seguintes aspectos são relevantes:

a) *Avaliação de resultados*. Os resultados deveriam ser cuidadosamente resumidos e comparados com a realidade subjetivamente observada. Por vezes algum desvio em relação à percepção da equipe pode ser resultado de erros no processo de obtenção dos dados.
b) *Ajuste do modelo.* Muitos modelos de medição exigem que determinadas constantes sejam calibradas ao longo de seu uso (por exemplo, CII). Assim, esses procedimentos de calibração devem ser executados sempre que o modelo assim o exigir.

Finalmente, convém lembrar que, em termos de capacidade de processos e maturidade de empresas (CMMI) é absolutamente necessário que exista um plano de métrica e que ele seja efetivamente usado para melhorar aspectos dos processos da empresa. Os níveis 4 e 5 de maturidade do CMMI só são atingidos caso exista um plano de métricas (ver Seção 12.3).

9.6 Revisão e Avaliação

Nos métodos ágeis como *Scrum* e XP a revisão e avaliação de projeto é prevista e ocorre informalmente nas reuniões diárias em pé, e com mais formalidade nas reuniões de fechamento de iteração.

Outros métodos poderão definir reuniões de revisão de projeto, de forma periódica, seja ao final de uma iteração ou fase, ou a qualquer momento.

É importante que as reuniões, para que sejam efetivas, sejam planejadas com antecedência. As pessoas envolvidas devem ser comunicadas, local adequado reservado, e mais importante, os objetivos da reunião devem ser claramente comunicados. Quando se fala em objetivos, aqui, deve-se entender isso no sentido de definir quais serão os *artefatos de saída* da reunião. Pode-se ver então que a reunião de revisão e avaliação é também vista como uma atividade de projeto. Quando os participantes de uma reunião começam a divagar e discutir sem objetividade, cabe ao condutor da reunião (gerente) interromper a divagação e solicitar sugestões de encaminhamento para os assuntos pendentes, de forma que os objetivos da reunião possam ser atingidos o mais rapidamente possível.

Além disso, sempre que possível todo o material a ser usado na reunião deve ser distribuído previamente. A reunião não é o momento de apresentar novidades ou surpresas para a equipe. É recomendável que, sempre que possível, os assuntos já sejam adiantados antes, caso contrário a reunião poderá se estender demais.

Não é recomendado que na reunião de revisão e avaliação se faça com o grupo uma análise minuciosa, por exemplo, de todo o código produzido pela equipe. Isso é impossível por questões de tempo. As revisões e relatórios já deverão estar prontos para a reunião, tendo sido feitos por aqueles a quem a tarefa foi delegada. No momento da reunião, apenas uma visão geral deve ser apresentada e, se necessário, um aprofundamento em um ou outro pontos críticos.

Para a pauta da reunião sugere-se que o gerente de projeto procure apresentar:

a) Os principais marcos de projeto (*milestones*) alcançados.
b) Os desvios que eventualmente tenham ocorrido em relação ao plano de desenvolvimento do projeto.
c) Modificações significativas na alocação de esforço, ou seja, se mais (ou menos) tempo do que o previsto foi dedicado a alguma atividade.
d) Modificações significativas em termos de despesas, ou seja, se as atividades consumiram mais (ou menos) itens de orçamento do que o previsto.
e) Mudanças no escopo estimado de trabalho, ou seja, se houve alguma modificação em termos das funcionalidades ou outros aspectos que inicialmente consistiam em um objetivo da fase ou ciclo e que foram mudados, removidos ou acrescentados.
f) Mudanças na métrica de qualidade, ou seja, se por alguma razão a forma de medir a qualidade do trabalho foi mudada.
g) *Status* dos riscos de projeto, ou seja, quais os riscos que continuam sendo muito importantes, quais riscos tiveram sua importância alterada ao longo da fase ou ciclo (especialmente caso a alteração tenha sido no sentido de um risco ficar mais importante do que era anteriormente).
h) Novos riscos identificados.
i) Problemas relevantes que tenham surgido e ainda não tenham sido resolvidos.
j) Andamento de eventuais ações que tenham sido determinadas em reuniões anteriores.

k) Lições aprendidas para projetos futuros.

Durante e, especialmente ao final da reunião, devem ser tomadas decisões sobre ações a serem executadas, sempre que necessário. O gerente deve indicar responsáveis e prazos para as ações mais urgentes. Já as ações não urgentes devem ser colocadas na lista de modificações ou na lista de riscos de projeto para serem tratadas oportunamente, de acordo com a sua prioridade.

9.7 Fechamento

O fechamento ou encerramento de um projeto (ou de uma fase de um projeto) tem como objetivo formalizar a entrega do produto e a sua aprovação pelo cliente.

Será uma reunião ainda mais formal do que as reuniões periódicas de avaliação do projeto e, possivelmente, deverá envolver um número maior de interessados. Dentre os itens tratados nas reuniões de fechamento de projeto, pode-se imaginar que vários deles serão semelhantes aos itens abordados nas reuniões de avaliação. Porém, em uma avaliação final, será importante observar:

a) Se todos os objetivos iniciais do projeto foram alcançados.
b) Se houve algum desvio importante ao longo do projeto em relação ao seu plano.
c) Se houve alguma mudança significativa em relação ao esforço previsto e executado (o que poderá servir de entrada para calibragem de métricas de esforço).
d) Se houve alguma mudança significativa em relação aos recursos alocados e consumidos.
e) Se houve mudanças de escopo importantes.
f) Os resultados finais das métricas de qualidade e, eventualmente, sua evolução.
g) Riscos de operação, caso tenham sido identificados.
h) Problemas relevantes que tenham surgido e ainda não tenham sido resolvidos.
i) Lições aprendidas para projetos futuros.

Além de formalizar a entrega do produto, a reunião de encerramento do projeto também será uma oportunidade importante para que sejam produzidos documentos reportando o desempenho da empresa, bem como os problemas de percurso e soluções encontradas. Desta forma, a inteligência criada durante o projeto fica registrada para possível uso futuro.

A finalização do projeto também é importante para que fique claro para todos os envolvidos que as atividades de desenvolvimento terminaram e que quaisquer novas modificações serão alvo de projetos futuros (ou do processo contínuo de manutenção).

