

GERENCIAMENTO DE PROJETOS DE SOFTWARE

Nesta parte de *Engenharia de Software: Uma Abordagem Profissional* você aprenderá as técnicas de gerenciamento necessárias para planejar, organizar, monitorar e controlar projetos de software. Estas questões são tratadas nos capítulos que seguem:

- Como as pessoas, os processos e os problemas devem ser gerenciados durante um projeto de software?
- Como as métricas de software podem ser usadas para gerenciar um projeto de software e o processo de software?
- Como uma equipe de software pode gerar estimativas confiáveis de trabalho, custo e duração do projeto?
- Que técnicas podem ser usadas para avaliar os riscos que podem ter um impacto sobre o sucesso do projeto?
- Como um gerente de projeto de software seleciona um conjunto de tarefas de engenharia de software?
- Como é criado um cronograma de projeto?
- Por que a manutenção e a reengenharia são tão importantes para os gerentes de engenharia de software e os profissionais?

Uma vez respondidas essas questões, você estará mais bem preparado para gerenciar projetos de software de forma a obter um produto de alta qualidade entregue no prazo.

24

CONCEITOS DE GERENCIAMENTO DE PROJETO

CONCEITOS-CHAVE

aplicações críticas (práticas)	579
coordenação e comunicação	573
decomposição do problema	574
equipes ágeis	572
equipe de software	570
escopo de software	574
líderes de equipe	569
o princípio W³HH	578
partes interessadas	569
pessoas	568
produtos	574
projeto	577

No prefácio de seu livro sobre gerenciamento de projeto de softwares, Meller Page-Jones faz uma afirmação que pode ser divulgada por muitos consultores de engenharia de software:

Visitei dúzias de lojas, boas e ruins; observei dezenas de gerenciadores de processamento de dados e, novamente, bons e ruins. Com muita frequência, horrorizado enquanto esses gerentes dedicavam esforços em projetos que eram verdadeiros pesadelos, contorciam-se em prazos impossíveis ou entregavam sistemas que ultrajavam seus usuários e continuavam a devorar um bocadinho de tempo de manutenção.

O que Page-Jones descreve são sintomas que resultam em um leque de problemas técnicos e de gerenciamento. Entretanto, se um exame pós-morte fosse feito para todo projeto, iria se chegar a um tema constante: o gerenciamento do projeto estava fraco.

Neste e nos capítulos 25 a 29, serão apresentados conceitos que conduzem a um gerenciamento de projetos efetivo. Aqui, trataremos os princípios e os conceitos básicos do gerenciamento de projetos. No capítulo 25, abordaremos a medição de projeto e de processo, base para a tomada de decisões de um gerenciamento efetivo. Técnicas utilizadas para a estimativa de custos estão no Capítulo 26. O 27 auxiliará a definição de um cronograma realístico do projeto. As atividades de gerenciamento que levam a uma efetiva monitoração

PANORAMA

O que é? Embora muitos de nós (em momentos mais críticos) adotemos a visão de Dilbert, ainda resta uma atividade bastante útil quando sistemas e projetos computacionais são desenvolvidos. Gerenciamento de projeto envolve planejamento, monitoração e controle de pessoas, processos e eventos que ocorrem à medida que o software evolui desde os conceitos preliminares até sua disponibilização, operacional e completa.

Quem realiza? De certa forma, todas as pessoas gerenciam, mas o escopo das atividades de gerenciamento varia entre os envolvidos de um projeto de software para outro. Um engenheiro de software gerencia suas atividades diárias, planejando, monitorando e controlando as tarefas técnicas. Os gerenciadores de projetos planejam, monitoram e controlam o trabalho de uma equipe de engenheiros de software. Já um gerente sênior coordena a interface entre o "lado comercial" e os profissionais de software.

Por que é importante? Desenvolvimento de software computacional é uma tarefa complexa, principalmente se envolver muitas pessoas trabalhando por um tempo relativamente longo. Por isso os projetos de software precisam ser gerenciados.

Quais são as etapas envolvidas? Entenda os 4 Ps: Pessoas, Produto, Processo e Projeto. As

pessoas devem ser organizadas para o trabalho de desenvolvimento de forma efetiva. A comunicação com o cliente e com outros interessados deve ocorrer para que o escopo e os requisitos do produto sejam compreendidos. Deve ser selecionado um projeto adequado para as pessoas e para o produto. O projeto deve ser planejado com base na estimativa do esforço e do prazo para a realização das tarefas: definindo artefatos, estabelecendo pontos de verificação (checagem) de qualidade e identificando mecanismos para monitorar e controlar o trabalho no plano de projeto.

Qual é o artefato? Assim que as atividades de gerenciamento iniciam, faz-se um plano de projeto. Define-se o processo e as tarefas a ser conduzidas, as pessoas que realizarão o trabalho e os mecanismos de avaliação de riscos, do controle das alterações e de avaliação de qualidade.

Como garantir que o trabalho foi realizado corretamente? Nunca se está completamente seguro de que o plano de projeto está correto até que se entregue um produto de alta qualidade, no prazo e dentro do orçamento. Entretanto, um gerente de projeto age corretamente quando encoraja o pessoal de desenvolvimento a trabalhar em conjunto como uma verdadeira equipe, concentrando-se nas necessidades do cliente e na qualidade do produto.

e mitigação de riscos estão no Capítulo 28. Por fim, o 29 considera a manutenção, a reengenharia e discute elementos (itens) de gerenciamento encontrados quando se lida com sistemas legados.

24.1 O ESPECTRO DE GERENCIAMENTO

Gerenciamento efetivo de desenvolvimento de software tem um foco nos 4 Ps: Pessoas, produto, processo e projeto. Essa ordem não é arbitrária. O gerente que se esquecer de que o trabalho do engenheiro de software consiste em esforço humano nunca terá sucesso no gerenciamento de projeto. Da mesma forma, aquele que falha no encorajamento amplo para a comunicação entre os envolvidos, bem cedo, no início da elaboração de um produto, corre o risco de desenvolver uma solução elegante para o problema errado. Um gerente que preste pouca atenção ao processo, arrisca-se a inserir métodos e ferramentas técnicas competentes em um vácuo. Aquele que embarcar sem um plano de projeto sólido compromete o sucesso do projeto.

24.1.1 Pessoal

Desde os anos 1960, debate-se acerca da valorização da cultura de ter pessoal de desenvolvimento motivado e de alto nível. Realmente, recursos humanos é um fator de tal importância que o Software Engineering Institute (SEI) desenvolveu um modelo para a maturidade e capacidade dos recursos humanos – o People-CMM (*People Capability and Maturity Model* – Modelo de Capacidade e Maturidade para Recursos Humanos) – em reconhecimento ao fato de que: “Toda organização precisa aprimorar continuamente sua habilidade para atrair, desenvolver, motivar, organizar e reter a força de trabalho necessária para atingir os objetivos estratégicos de seus negócios” [Cur 01].

O People-CMM define as seguintes práticas-chave para o pessoal de software: formação de equipe, comunicação, ambiente de trabalho, gerenciamento do desempenho, treinamento, compensação, análise de competência e de desenvolvimento, desenvolvimento de carreira, do grupo de trabalho, da cultura de equipe e de outros mais.

Em organizações que conseguem altos níveis de maturidade e capacidade, o People-CMM tem maior probabilidade de implementar práticas de gerenciamento de software efetivos.

O People-CMM é um parceiro para o modelo de integração para maturidade e capacidade em software: o SW – CMMi (Capítulo 30) conduz as organizações para a criação de um processo de software maduro. Os itens relacionados ao gerenciamento de recursos humanos e à estruturação dos projetos de software serão discutidos posteriormente, neste capítulo.

24.1.2 O produto

Antes de traçarmos um plano de projeto, devemos estabelecer os objetivos do produto e seu escopo, considerar as soluções alternativas e identificar as restrições técnicas e de gerenciamento. Sem tais informações, é impossível definir de forma razoável (e precisa) a estimativa de custo, a avaliação efetiva dos riscos, a análise realística das tarefas do projeto ou um cronograma gerenciável do projeto que forneça a indicação significativa de progresso das atividades.

Como desenvolvedores, devemos nos reunir com os interessados no software para definir os objetivos e o escopo do produto. Em muitos casos, tal atividade se inicia como parte da engenharia do sistema ou de engenharia do processo de negócio e continua como a 1ª etapa da Engenharia de Requisitos do software (Capítulo 5).

Os objetivos identificam as metas gerais do produto (do ponto de vista dos interessados) sem considerar como tais metas serão alcançadas. O escopo identifica os principais dados, funções e comportamentos que caracterizam o produto e, mais importante, tenta mostrar as fronteiras e limitações dessas características de maneira quantitativa.

Uma vez entendidos os objetivos e o escopo, consideram-se soluções alternativas. Apesar de se discutir muito pouco os detalhes, as alternativas capacitam os gerentes e desenvolvedores

a selecionar a melhor abordagem, dadas as restrições impostas pelos prazos de entrega, restrições orçamentárias, disponibilidade de pessoal, interfaces técnicas e uma miríade de outros fatores.



Aqueles que aderem à filosofia do processo ágil (Capítulo 3) argumentam que ele é mais improdutivo que os outros. Isso pode ser verdade, mas ainda sim têm um processo, e um software de engenharia ágil ainda requer disciplina.

24.1.3 O processo

Um processo de software (Capítulos 2 e 3) fornece a metodologia por meio da qual um plano de projeto abrangente para o desenvolvimento de software pode ser estabelecido. Poucas atividades metodológicas são aplicáveis a todos os projetos de software, independentemente do seu tamanho e complexidade.

Uma quantidade de diferentes conjuntos de atividades-tarefas, pontos de controle, artefatos de software e pontos de garantia de qualidade possibilitam que as atividades metodológicas sejam adaptadas às características do projeto de software e aos requisitos de equipe. Por fim, as atividades de apoio, como Garantia de Qualidade de Software, Gerenciamento de Configuração e de Medições, sobrepõem-se ao modelo do processo. Estas são independentes de quaisquer das atividades metodológicas e ocorrem ao longo do processo.

24.1.4 O projeto

Empregam-se projetos com planejamento e com controle por uma única e principal razão: é a única maneira de administrar a complexidade. E, mesmo assim, as equipes de software têm de se esforçar. Em um estudo de 250 grandes projetos de software entre 1998 e 2004, Caper Jones [Jon 04] constatou que “cerca de 25 obtiveram sucesso em cumprir cronograma, custos e objetivos quanto à qualidade. Em torno de 50 apresentaram atrasos em, no mínimo, 35% retardamentos sérios, enquanto 175 projetos tiveram uma experiência de atrasos e retardamentos sérios, ou ainda não conseguiram terminá-lo”. Apesar de, atualmente, a taxa de sucesso nos projetos de software possa ter melhorado de algum modo, a taxa de falhas em projeto permanece mais alta do que deveria¹.

Para evitar falha de projeto, o gerente e engenheiros que desenvolvem o produto devem evitar uma série de sinais de alertas comuns, devem entender os fatores críticos de sucesso que conduzem ao bom gerenciamento e desenvolver uma abordagem de senso comum no que se referir a planejamento, monitoramento e controle de projeto. Cada um desses itens é discutido na Seção 24.5 e no capítulo seguinte.

24.2 As Pessoas

Em um estudo publicado pelo IEEE [Cur 88], os vice-presidentes de engenharia de três principais companhias de tecnologia foram questionados quanto ao fator mais importante que contribui para o sucesso de um projeto de software. Eles responderam da seguinte maneira:

Vp 1: Suponha-se que se tenha de selecionar um único item no ambiente que seja mais importante. Eu diria que não são as ferramentas usadas, são as pessoas.

Vp 2: O ingrediente mais importante neste projeto foi reunir pessoas espertas... Muito poucas coisas a mais importam na minha opinião... O mais importante para um projeto é selecionar a equipe... O sucesso da organização do desenvolvimento de software está bastante associado à habilidade de recrutar bom pessoal.

Vp 3: A única regra que tenho no gerenciamento consiste em assegurar que possa reunir bom pessoal – bem como cultivá-los – e propiciar ambiente no qual os bons profissionais possam produzir.

¹ Dadas essas estatísticas, é razoável questionar como os impactos computacionais evoluem exponencialmente. Parte da resposta, penso eu, é que um substancial número dos projetos que falham nas primeiras tentativas é preconcebido com falhas. Clientes perdem interesse muito rapidamente (porque o que eles pediram não é tão importante quanto acharam que era em princípio), e os projetos são cancelados.

Realmente, esses são testemunhos convincentes quanto à importância do pessoal no processo de engenharia de software. Ainda assim, dos vice-presidentes de engenharia ao desenvolvedor mais simples, frequentemente consideram pessoal como um privilégio. Os gerentes afirmam (como o fez o grupo anterior) que pessoal é prioritário, entretanto, suas ações rebaixam suas palavras. Na próxima seção examinam-se os interessados que participam do processo de software e a maneira pela qual são organizados para desempenhar ações efetivas de engenharia de software.

24.2.1 Os interessados (comprometidos)

O processo de software (e todo o projeto de software) é formado por interessados que podem ser categorizados em um dos cinco grupos:

1. *Gerentes seniores* – definem os itens de negócio e, com frequência, exercem influência significativa no projeto.
2. *Gerentes (técnicos) de projetos* devem planejar, motivar, organizar e controlar os programadores que executam o trabalho em software.
3. *Programadores* devem ter habilidades técnicas necessárias para desenvolver a engenharia de um produto ou aplicativo de software.
4. *Cientes* especificam os requisitos para o software a ser submetidos ao processo de engenharia e outros envolvidos que têm interesses periféricos no produto final.
5. *Usuários finais* interagem com o software uma vez liberado para uso operacional, em ambiente de produção.

Todo projeto de software é composto por pessoas que se enquadram nessa taxonomia². Para ser efetiva, a equipe do projeto deve estar organizada para maximizar cada capacidade e habilidade dos profissionais. E essa é a tarefa do líder da equipe.

24.2.2 Líderes de equipe

Gerenciamento de projeto é uma atividade intensiva de pessoal, e, por essa razão, programadores competentes em geral resultam em maus líderes de equipe. Apenas não possuem a mistura certa de habilidade com pessoas. Ainda assim, como Edgemon afirma: “infelizmente e, com demasiada frequência, parece que, os indivíduos só assumem o papel de gerente de projeto e se tornam gerentes de projetos por acidente”. [Ed 95].

Em um excelente livro sobre liderança técnica, Jerry Weinberg [Wei 86] sugere um modelo MOI de liderança:


Motivação: a habilidade para encorajar o pessoal técnico a produzir com o melhor da sua habilidade.

Organização: a habilidade para moldar os processos já existentes (ou inventar novos) que irão capacitar o conceito inicial para ser traduzido num produto final.

Ideias ou inovação: a habilidade de encorajar pessoas para criar e ser criativas mesmo quando estiverem trabalhando de acordo com padrões estabelecidos para um produto ou aplicativo de software específico.

Weinberg sugere que líderes de projeto bem-sucedidos aplicam um estilo de gerenciamento de solução de problemas. Isto é, um gerente de projeto de software deve se concentrar em entender o problema a ser resolvido, administrando o fluxo de ideias e, ao mesmo tempo, deixar claro para todos da equipe (por meio de palavras e, muito mais importante, de ações) que a qualidade conta muito e que não deve ser comprometida.

Uma outra visão [Edg 95] das características que definem um efetivo gerenciamento de projeto concentra-se em quatro aspectos-chave:

 O que buscamos ao selecionar alguém para liderar um projeto de software?

² Quando se desenvolvem aplicações para web (WebApps), outras pessoas não especialistas em software são envolvidas na criação de conteúdos.

"Resumindo, um líder é aquele que sabe para onde quer ir, e se levanta e vai."

John Erskine

Solução de problema. Um gerente de projeto eficaz sabe diagnosticar itens técnicos e organizacionais que são os mais relevantes, sistematicamente estrutura uma solução ou motiva apropriadamente outros desenvolvedores a buscar a solução, põe em prática as lições aprendidas de projetos anteriores para novas situações e permanece suficientemente flexível para mudar de direção, caso as tentativas iniciais para a solução de problemas tenham sido infrutíferas.

Identidade gerencial. Um bom gerente de projeto deve assumir a reponsabilidade do projeto. Deve ter a confiança para assumir o controle quando necessário e deve assegurar que permitirá ao pessoal técnico seguir os seus instintos.

Realizações. Um gerente competente deve recompensar iniciativas e realizações para otimizar a produtividade de uma equipe. Deve demonstrar por meio de seus próprios atos que decisões por riscos controlados não serão punidas.

Formação de equipe e de influência. Um gerente efetivo deve ser capaz de "ler" pessoas, deve ser capaz de compreender sinais verbais e não verbais e reagir às necessidades das pessoas que estão enviando esses sinais. O gerente deve permanecer sob controle em situações de alto estresse.

24.2.3 Equipe de software

Existem praticamente tantas estruturas organizacionais humanas para desenvolvimento de software quantas organizações que desenvolvem software. Para melhor ou pior, a estrutura organizacional não pode ser facilmente modificada. Preocupações com os efeitos práticos e políticos da mudança organizacional não fazem parte do escopo de responsabilidade do gerente de projeto de software. Entretanto, a organização do pessoal diretamente envolvido em um novo projeto está sob a perspectiva do gerente do projeto.

A melhor estrutura de equipe depende do estilo de gerenciamento das organizações, quantidade de pessoas na equipe, seus níveis de habilidade e do grau de dificuldade geral do problema. Mantei [man 81] descreve sete fatores que devem ser considerados ao planejarmos a estrutura da equipe de engenharia de software:

- Grau de dificuldade do problema a ser solucionado
- Extensão do(s) programa(s) resultante(s) em linhas de códigos ou pontos de função
- Tempo que uma equipe trabalhará em conjunto (tempo de vida da equipe)
- Grau de modularização do problema
- Qualidade e confiabilidade do sistema a ser construído
- Rigidez das datas de entregas
- Grau de sociabilidade (comunicação) requerido para o projeto

Constantine [Con 93] sugere quatro "paradigmas organizacionais" para equipes de engenharia de software:

1. O *paradigma fechado* estrutura a equipe em uma hierarquia de autoridade tradicional. Tais equipes podem trabalhar bem em produção de software bastante similar a esforços de épocas passadas, mas se mostrarão menos propícias a ser inovadoras trabalhando sob o paradigma fechado.
2. O *paradigma randômico* estrutura a equipe de forma livre e depende da iniciativa individual de seus membros. Quando for requerida inovação ou avanço tecnológico, as equipes que seguem esse paradigma irão se distinguir, mas poderão ter de se esforçar quando requisitada uma "performance ordenada".
3. O *paradigma aberto* atém-se a estruturar a equipe de maneira que consiga alguns dos controles associados com o paradigma fechado, mas também muito da inovação que ocorre ao

"Nem todo grupo é uma equipe, nem toda equipe é efetiva"

Glenn Parker

? Quais fatores devem ser considerados quando a estrutura de uma equipe de software já está estabelecida?

? Quais são as opções quando se tem de definir a estrutura de uma equipe de software?

usar o paradigma randômico. O trabalho é feito de forma colaborativa, com forte comunicação e tomada de decisão baseada no consenso, executando com as características marcantes das equipes de paradigma aberto. As estruturas das equipes de paradigmas abertos são bem adequadas para a solução de problemas complexos, mas não conseguem desempenhar tão bem quanto outras equipes.

4. O *paradigma sincronizado* baseia-se na compartimentalização natural de um problema e organiza os membros da equipe a trabalhar nas partes do problema com pouca comunicação entre si.

“Se deseja ser incrementalmente melhor, seja competitivo. Se quer ser exponencialmente melhor, seja cooperativo.”

Autor desconhecido

Como uma nota de rodapé histórica, uma das mais antigas organizações de equipe de software foi paradigma de uma estrutura fechada denominada originalmente de equipe com um programador-chefe (principal). Essa estrutura foi primeiramente proposta por Harlan Mills e descrita por Baker [Bak 72]. O núcleo da equipe era composto de um engenheiro sênior (o programador-chefe principal), que planeja, coordena e faz a revisão de todas as atividades técnicas da equipe, o pessoal técnico (normalmente de duas a cinco pessoas), que conduz as atividades de análise e de desenvolvimento, e um engenheiro reserva que dá suporte ao engenheiro sênior em suas atividades e pode substituí-lo com perdas mínimas de continuidade. O programador-chefe (principal) pode ter a seus serviços um ou mais especialistas (por exemplo, perito em telecomunicações, desenvolvedor de banco de dados), equipe de suporte (por exemplo, codificadores técnicos, pessoal de escritório) e um bibliotecário de software.

Como um contraponto para a estrutura da equipe de programadores principais (líderes), o paradigma randômico de Constantine [Con 93] sugere a primeira equipe criativa, cuja abordagem de trabalho pode ser mais bem denominada de “anarquia inovativa”. Embora a abordagem de espírito livre para o trabalho de software tenha apelo, a energia da criatividade direcionada para uma equipe de alta performance deve ter o objetivo central de uma organização de engenharia de software. Para obter uma equipe de alta performance:

- Os membros da equipe devem confiar uns nos outros.
- A distribuição de habilidades deve ser adequada ao problema.
- Estrelismos devem ser excluídos da equipe para manter a coesão do grupo.

? O que é equipe “consistente”?

Seja qual for a organização da equipe, o objetivo em todo o gerenciamento do projeto consiste em ajudar a montar uma equipe que apresente coesão. Em seu livro *Peopleware*, De Marco e Listes [DeM 98] discorrem sobre esse tema:

Há uma tendência em se utilizar a palavra equipe de forma constante e vaga na área de negócios, denominando qualquer grupo de profissionais designados a trabalhar juntos de “equipe”. Entretanto, muitos deles não se assemelham a equipes. Não há uma definição comum de sucesso nem um espírito de equipe identificável. O que falta é um fenômeno que se denomina *consistência*.

Uma equipe consistente é um grupo de pessoas tão fortemente unidas que o todo é maior do que a soma das partes. Uma vez que uma equipe comece a ser consistente, a probabilidade de sucesso segue em caminho ascendente. A equipe pode disparar para um caminho de sucesso...

Não é preciso gerenciá-la do modo tradicional e, com certeza, não precisará ser motivada. Ela adquire velocidade e ímpeto.

? Por que motivo equipes falham em ser consistentes?

De Marco e Lister sustentam que os membros de equipes consistentes são mais significativamente produtivos e mais motivados do que a média. Compartilham de um objetivo comum, de uma cultura comum e, em muitos casos de um senso de elitização que os torna únicos.

Mas nem todas as equipes tornam-se consistentes. De fato, muitas sofrem do que Jackman [Jac 98] denomina de “toxicidade de equipe”. Ela define cinco fatores que fomentam um ambiente potencialmente tóxico da equipe: uma atmosfera de trabalho frenética; alto grau de frustração que causa atrito entre os membros da equipe; um processo de software fragmentado ou pobremente coordenado; uma definição nebulosa dos papéis dentro da equipe de software; e contínua e repetida exposição a falhas.

"Ou faça ou não faça. Não existe o tentar".

Yoda,
personagem de
Star Wars

Para evitar um ambiente de trabalho frenético, o coordenador de projeto deve estar certo de que a equipe tem acesso a todas as informações necessárias para realizar o trabalho e de que as metas e objetivos prioritários (papéis), uma vez estabelecidos, não devem ser alterados a menos que absolutamente necessário. Uma equipe pode evitar frustrações se lhe for oferecida, tanto quanto possível, responsabilidade para tomada de decisão. Um processo inapropriado (por exemplo, tarefas pesadas ou desnecessárias ou artefatos mal selecionados) pode ser evitado por meio da compreensão do produto a ser desenvolvido, das pessoas que realizam o trabalho e pela permissão para que a equipe selecione o modelo do processo. A própria equipe deve estabelecer seus próprios mecanismos de responsabilidades (revisões técnicas³ são excelentes meios para conseguir isso) e deve definir uma série de abordagens para correções quando um membro falhar em suas atribuições. E, finalmente, a chave para evitar uma atmosfera de derrota consiste em estabelecer técnicas baseadas nas equipes voltadas para realimentação (*feedback*) e resolução de problemas.

Somando-se às cinco toxinas descritas por Jackman, uma equipe de software frequentemente depende esforços com as diferentes características de seus membros. Uns são extrovertidos, outros introvertidos. Uns coletam informações intuitivamente, destilando conceitos amplos de fatos disparatados. Outros processam informações linearmente, coletando e organizando detalhes mínimos dos dados fornecidos. Alguns se sentem confortáveis tomando decisões apenas quando um argumento lógico e ordenado for apresentado. Outros são intuitivos, acostumados a tomar decisões baseadas em percepções. Certos desenvolvedores querem um cronograma detalhado, preenchido por tarefas organizadas que os tornem aptos para atingir proximidade com elementos de projeto. Outros ainda preferem um ambiente mais espontâneo no qual resultados e questões abertas já estarão bons.

Alguns trabalham arduamente para conseguir que as etapas sejam concluídas bem antes da data estabelecida, evitando, portanto, estresse na medida em que se aproxima a data-limite, enquanto outros são energizados pela correria em fazer até a data e minutos-limite.

Uma discussão detalhada sobre a psicologia envolvida nessas características e formas pelas quais um hábil líder de equipe pode auxiliar pessoas com traços opostos a trabalhar juntas está além do escopo deste livro⁴. Entretanto, é importante notar que o reconhecimento das forças humanas é o primeiro passo em direção à criação de equipes consistentes.

24.2.4 Equipes ágeis

Ao longo da última década, o desenvolvimento de software ágil (Capítulo 13) tem sido indicado como o antídoto para muitos problemas que se alastraram nas atividades de projeto de software. Relembrando, a filosofia ágil enfatiza a satisfação do cliente e a entrega prévia incremental de software, pequenas equipes de projetos altamente motivadas, métodos informais, mínimos artefatos de engenharia de software e total simplicidade de desenvolvimento.

A pequena, altamente motivada equipe de projeto, também denominada de *equipe ágil*, adota muitas das características das equipes de software bem-sucedidas, discutidas na seção anterior e evita muito das toxinas geradoras de problemas. Entretanto, a filosofia ágil enfatiza competência individual (membro da equipe) casada com colaboração em grupo como fatores críticos de sucesso para a equipe. Cockburn e HighSmith [Coc01a] observam isso ao escreverem:

Se as pessoas do projeto forem boas o suficiente, podem usar praticamente qualquer processo e realizar a sua missão. Se elas não forem boas o suficiente, nenhum processo irá reparar a sua inadequação. "Pessoas são o trunfo do processo" é uma forma de dizer isso. Entretanto, falta de suporte ao desenvolvedor e ao usuário pode matar um projeto – "política é o trunfo de pessoas". Suporte inadequado pode fazer com que mesmo os bons fracassem na realização de seus trabalhos.

Para uso das competências de cada membro da equipe, para fomentar colaboração efetiva ao longo do projeto, equipes ágeis se auto-organizam. A equipe que se auto-organiza não mantém,

PONTO-CHAVE

Uma equipe ágil é aquela que se organiza, que tem autonomia para planejar e tomar decisões técnicas.

³ Revisões técnicas são tratadas em detalhes no Capítulo 15.

⁴ Uma excelente introdução a essas questões no que se refere a equipes de projeto de software pode ser encontrada em [Fer 98].

necessariamente, uma estrutura de equipe única, mas usa elementos da aleatoriedade de Constantine, paradigmas abertos e de sincronicidade discutidos na Seção 24.2.3.

Muitos modelos ágeis de processo (por exemplo, Scrum) dão à equipe ágil autonomia para fazer o gerenciamento do projeto e para decisões técnicas necessárias à conclusão do trabalho. O planejamento é mantido em um nível mínimo, e a equipe tem a permissão para selecionar sua própria abordagem (por exemplo, processo, método, ferramentas), limitada somente pelos requisitos de negócio e pelos padrões organizacionais. Conforme o projeto prossegue, a equipe se auto-organiza, concentrando-se em competências individuais para maior benefício do projeto em determinado ponto de cronograma. Para tanto, uma equipe ágil pode realizar reuniões de pessoal diariamente para coordenar e sincronizar as atividades que devem ser realizadas para aquele dia.

Com base na informação obtida durante essas reuniões, a equipe adapta sua abordagem para incrementar o trabalho. A cada dia, contínuas auto-organizações e colaboração conduzem a equipe em direção a um incremento de software completo.

24.2.5 Itens de comunicação e coordenação

Há muitas razões para que o projeto de software tenha problemas. A escala de muitos esforços em desenvolvimento é ampla, conduzindo a complexidade, confusão e dificuldades significativas na coordenação dos membros da equipe. Incertezas são comuns, resultando em contínua cadeia de alterações que racham a equipe de projeto. Interoperabilidade torna-se um aspecto-chave para muitos sistemas. Novo software deve se comunicar com os softwares existentes e se ajustar a restrições predefinidas impostas pelo sistema ou pelo produto.

Tais características do software moderno – escala, incerteza e interoperabilidade – são fatos da vida. Para lidar efetivamente com eles, devem-se estabelecer métodos efetivos para coordenar pessoas que realizam o trabalho. Para tanto, devem-se estabelecer mecanismos para comunicação formal e informal entre os membros de equipes. A comunicação formal é realizada por meio de “comunicação escrita, reuniões estruturadas e de outros canais de comunicação relativamente não interativos e impessoais” [Kra 95]. A comunicação informal é mais pessoal.

“Propriedades coletivas nada mais são do que um imediatismo da ideia de que os produtos deveriam ser atribuídos à equipe (ágil), não a indivíduos que compõem a equipe.”

Jim Highsmith

CASA SEGURA



Estrutura de time

Cena: Escritório de Doug Miller antes do início do projeto de software CasaSegura.

Atores: Doug Miller (coordenador/gerente da equipe de engenharia de software do CasaSegura), Vinod Ramann, Jamie Lazar e outros membros da equipe de engenharia de software do produto.

Conversa:

Doug: Vocês deram uma olhada no informativo preliminar do CasaSegura que o departamento de marketing preparou?

Vinod (balançando afirmativamente e olhando para seus companheiros de equipe): Sim, mas temos muitas dúvidas.

Doug: Vamos deixar isso de lado por um momento. Gostaria de conversar sobre como vamos estruturar a equipe, quem será responsável pelo quê?

Jamie: Estou realmente de acordo com a filosofia ágil, Doug. Acho que devemos ser uma equipe que se auto-organiza.

Vinod: Concorro. Devido ao cronograma apertado e o grau de incerteza e do fato de todos sermos realmente competentes (risos...), parece ser o caminho certo a tomar.

Doug: Tudo bem para mim, mas vocês conhecem o procedimento.

Jamie (solicitando e falando ao mesmo tempo): Tomamos decisões táticas acerca de quem faz o que e quando, mas é de nossa responsabilidade ter o produto pronto sem atraso.

Vinod: E com qualidade.

Doug: Exatamente. Mas lembrem-se de que há restrições. O marketing define os incrementos de software a ser desenvolvidos – consultando-nos, é claro.

Jamie: E?

Doug: E usaremos o UML como abordagem de modelagem.

Vinod: Mas mantenham documentações adicionais no mínimo (absoluto).

Doug: Quem manterá a ligação comigo?

Jamie: Decidimos que Vinod será o coordenador técnico – ele tem mais experiência, portanto, será o intermediário, mas sinta-se livre para conversar com qualquer um de nós.

Doug (rindo): Não se preocupe, farei isso.

Os membros de uma equipe de software compartilham ideias numa base *ad hoc*, solicitam ajuda conforme surgem os problemas e interagem uns com os outros diariamente.

24.3 O PRODUTO

Um gerente de projeto de software confronta-se com um dilema sempre que inicia um projeto. É preciso estimativas quantitativas e um plano organizado, entretanto informações sólidas não estão disponíveis ainda. Uma detalhada análise dos requisitos de software fornece as informações necessárias para as estimativas, mas as análises frequentemente levam semanas ou mesmo meses para estarem completas. Pior ainda, os requisitos podem ser fluidos, mudando regularmente conforme o projeto prossegue. Ainda assim, um planejamento é necessário “agora”. Gostando-se ou não, deve-se examinar o produto e o problema que se pretende solucionar logo no início do projeto. No mínimo, o escopo do produto deve ser estabelecido e delimitado.

24.3.1 Escopo de software

A primeira afinidade do gerenciamento do projeto de software consiste em determinar o escopo de software. Este é definido respondendo-se às seguintes questões:



Se puder estabelecer uma característica do software que pretende construir, liste-a como um risco de projeto (Capítulo 25).

Contexto. Como o software a ser desenvolvido se ajusta a um sistema maior, a um produto ou contexto de negócio e quais são as restrições impostas resultantes do contexto.

Objetivo da informação. Quais objetos de dados visíveis ao cliente são produzidos como saída de software? Quais objetos de dados são necessários como entrada?

Função e performance. Qual a função que o software desempenha para transformar os dados de entrada em dados de saída? Há quaisquer características especiais de desempenho a ser acessada?

O escopo do projeto de software não deve haver ambiguidades e deve ser compreensível tanto no nível gerencial quanto no nível técnico. Deve-se estabelecer o escopo de software. Isto é, dados quantitativos (por exemplo, número de usuários simultâneos, ambiente-alvo, tempo máximo de resposta) são estabelecidos explicitamente, restrições e/ou limitações (por exemplo, custo do produto restringe tamanho de memória) são determinadas e fatores mitigadores (por exemplo, algoritmos desejados são bem compreendidos e avaliados em Java) são descritos.

24.3.2 Decomposição do problema



Para elaborar um razoável planejamento de projeto, deve-se decompor o problema. Para tanto, utilize-se uma lista de funções ou empregam-se, como base, casos de uso.

A decomposição do problema, também chamada de *elaboração do problema* ou *particionamento*, consiste em atividade que ocupa o centro da análise de requisitos de software (Capítulos 6 e 7). Durante a atividade de escopo, não se busca decompor completamente o problema. De preferência, aplica-se a decomposição em duas áreas vitais: (1) na funcionalidade e no conteúdo (informação) que deve ser entregue; e (2) no processo que será utilizado para entregar o software.

As pessoas tendem a aplicar a estratégia de dividir para conquistar quando confrontadas por um problema complexo. Ao simplificarmos um problema complexo, este é particionado em pequenas questões mais gerenciáveis. Essa é a estratégia a aplicar no início do planejamento do projeto.

Funções de software, descritas no estabelecimento do escopo, são avaliadas e refinadas para proporcionar mais prioridades de detalhes logo no início das estimativas (Capítulo 26). Por serem estimativas, custos ou cronogramas são orientados pela funcionalidade, algum grau de decomposição é em geral útil. Do mesmo modo, conteúdo principal ou objetos de dados são decompostos em suas partes constituintes, propiciando compreensão razoável da informação a ser gerada pelo software.

Como exemplo, considere um projeto que irá desenvolver um produto de processador de texto. Entre os fatores importantes estão recursos de voz, bem como entrada por teclado virtual

através de uma tela de toques múltiplos, recursos de edição/correção com cópias automáticas extremamente sofisticadas, capacidade de formatação de layout de página, indexação automática e tabela de conteúdos, entre outros. O gerente de projeto deve primeiro estabelecer um escopo de declarações e procedimentos que abrange esses recursos (e outras funções mais comuns como correções, alterações, gerenciamento de arquivos e produção de documentos). Por exemplo, o recurso de entrada por voz irá requerer que o produto tenha o recurso de “aprendizado” e “treinamento” pelo usuário? Especificamente, quais capacidades serão fornecidas para as funções de edição das cópias? Qual o grau de sofisticação da função de formatação de página, considerando o uso da tela multitoque e suas capacidades implícitas?

À medida que os parâmetros do escopo evoluem, ocorre um particionamento natural em primeiro nível. A equipe é notificada de que o departamento de marketing, ao conversar com clientes potenciais, detectou que as seguintes funções deveriam fazer parte do recurso de edição automática das cópias: (1) checagem de separação silábica; (2) checagem de gramática; (3) checagem de referência para documentos externos (por exemplo, uma referência a uma entrada bibliográfica é encontrada na lista de entrada da bibliografia?); (4) implementação de recursos de folhas de estilos que imponha consistência ao longo do documento; e (5) validação da referência quanto à seção e capítulo para documentos externos.

Cada um desses recursos representa uma subfunção a ser implementada no software. Podem ser refinados caso a decomposição (subdivisão e o particionamento) facilite o planejamento.

24.4 O PROCESSO

As atividades de modelagem (Capítulo 2) que caracterizam o processo de software são aplicáveis a todos os projetos de software. A dificuldade está em selecionar o modelo de processo apropriado ao software a ser desenvolvido (pelo processo de engenharia) pela sua equipe.

A equipe deve decidir qual modelo de processo será mais apropriado: (1) aos clientes que solicitaram o produto e aos profissionais que realizarão o trabalho; (2) às próprias características de produto; e ao ambiente de projeto no qual a equipe trabalhará.

Quando um modelo de processo é selecionado, a equipe define o planejamento preliminar do projeto com base no conjunto de atividades estabelecidas no modelo do processo. Uma vez definido o planejamento preliminar, inicia-se o particionamento (decomposição) do projeto. Ou seja, um planejamento completo, que reflita as tarefas de trabalho necessárias para preencher as atividades de modelagem, deve ser criado. Essas atividades serão abordadas nas seções seguintes e uma visão mais detalhada, apresentada no Capítulo 26.

24.4.1 Combinando o produto e o processo

O projeto começa por meio da combinação do produto com o processo. Cada função a ser desenvolvida por engenharia deve passar pela estrutura de atividades definidas pela organização responsável pelo software.

Suponha que a organização tenha adotado a estrutura de atividades genéricas **comunicação, planejamento, modelagem, construção e empacotamento dos programas executáveis** tratados no Capítulo 2. Os membros da equipe que trabalha em uma funcionalidade do produto aplicarão cada uma das atividades estruturadas para a função. Em essência, uma matriz similar à Figura 24.1 será criada. Cada função principal do produto (a figura mostra a função do software do processador de texto discutido anteriormente) será listada na coluna da esquerda. As atividades estruturadas serão relacionadas (indicadas) na parte superior das colunas. As tarefas de trabalho de engenharia de software (para cada atividade estruturada) serão incluídas nas linhas seguintes⁵. O trabalho do gerente de projeto e de outros membros da equipe será estimar as necessidades de recursos para cada célula da matriz, datas de início e de fim para as tarefas

⁵ As tarefas devem ser adaptadas às necessidades específicas do projeto, baseando-se em critérios de adaptação.

Agora, considere um projeto mais complexo, com um escopo mais amplo e um impacto comercial mais significativo. Tal projeto pode vir a requerer as seguintes tarefas para **comunicação**:

1. Revisão da solicitação do cliente.
2. Planejamento e agendamento de reuniões viabilizadas e formais com todos os envolvidos.
3. Realização de uma pesquisa para especificar a solução proposta e as abordagens existentes.
4. Preparação de um documento de trabalho e de uma agenda para a reunião formal.
5. Realização de reunião.
6. Desenvolvimento conjuntamente de miniespecificações que reflitam os dados, a funcionalidade e os fatores comportamentais do software. De forma alternativa, desenvolvimento de casos de uso que descrevam o software sob o ponto de vista do usuário.
7. Revisão de cada miniespecificação ou caso de uso para realizar correções, consistências e eliminação de ambiguidades.
8. Reunião de miniespecificações em um documento de escopo (bases).
9. Revisão do documento de escopo (bases) ou a coletânea de casos de uso com todos os envolvidos.
10. Alteração do documento de escopo ou de casos de uso conforme o necessário.

Ambos os projetos constituem as atividades **comunicação**, entretanto, a primeira equipe executa metade das tarefas de trabalho de engenharia.

24.5 O PROJETO

Para gerenciar um projeto de software com sucesso, deve-se compreender o que pode sair errado, de modo que ações planejadas evitem tais problemas. Em excelente artigo sobre projetos de software, John Reel [Ree 99] indica 10 sinais indicadores de que um projeto de sistemas de informações está em perigo:

? Quais são os sinais de que um projeto de software está em perigo?

1. O pessoal de software não compreende as necessidades de seus clientes.
2. O escopo do produto está parcialmente definido.
3. As alterações são mal gerenciadas/administradas.
4. A tecnologia escolhida muda.
5. As necessidades de negócio mudam (ou são mal definidas).
6. Os prazos estão fora da realidade.
7. Os usuários mostram-se resistentes.
8. O patrocínio é perdido (ou nunca foi propriamente obtido).
9. Faltam profissionais à equipe ou esta não possui pessoal com habilidades adequadas.
10. Gerentes e desenvolvedores evitam práticas e lições aprimoradas e aprendidas.

Não há tempo para parar e reavaliar, já estamos atrasados.

M. Cieron

Com frequência profissionais da indústria já estressados referem-se à regra 90-90 ao debaterem sobre projetos particularmente difíceis. Os primeiros 90% de um sistema absorvem 90% dos esforços e tempos alocados. Os 10% restantes consomem outros 90% de esforço e tempo alocados [Zah 94]. As situações que conduzem à regra 90-90 foram incluídas nos indicadores da lista anterior.

Porém, chega de negatividade! Como um gerente age para evitar os problemas mencionados? Reel [Ree 99] sugere uma abordagem de cinco partes para os projetos de software:

1. *Comece com o pé-direito.* Trabalhando arduamente (muito arduamente), é possível compreender o problema a ser solucionado e, então, estabelecer expectativas e objetivos realísticos

para todos os envolvidos no projeto. Isso é reforçado por meio da formação da equipe correta (Seção 24.2.3) e concedendo-lhe autonomia, autoridade e tecnologia necessárias para realizar o trabalho.

2. *Mantenha a velocidade (ímpeto).* Muitos projetos começam bem e depois desintegram lentamente. Para manter velocidade (ímpeto), o coordenador deve fornecer incentivos para que a rotatividade de pessoal fixe-se em um nível absolutamente mínimo. A equipe deve dar ênfase à qualidade em todas as tarefas que realiza e o coordenador sênior deve fazer todo o possível para posicionar-se fora do caminho da equipe⁷.
3. *Rastreie o andamento.* Em um projeto de software, o andamento é rastreado e mapeado como os artefatos (códigos-fonte, conjunto dos pacotes de testes) são produzidos e aprovados (usando-se as revisões técnicas) como parte de uma atividade de garantia de qualidade. Como acréscimo, o processo de software e as medições do projeto (Capítulo 25) podem ser coletados e utilizados para avaliar o progresso (andamento) em relação às médias desenvolvidas para a organização de desenvolvimento de software.
4. *Tome decisões com astúcia (rapidez).* Em essência, a decisão do gerente de projeto e da equipe de software deve ser a de "manter a simplicidade". Sempre que possível, decida-se por utilizar software comercial ou por componentes e modelos de software existentes. Evite interfaces customizadas quando houver disponibilidade de abordagens-padrão. Fique atento aos riscos óbvios para evitá-los e decida-se por alocar maior prazo do que o necessário para tarefas arriscadas ou complexas (serão necessários todos os minutos). Somente as práticas vitais associadas à "integridade do projeto" são registradas aqui.
5. *Faça uma análise post-mortem.* Estabeleça um mecanismo consistente para extrair aprendizados de cada projeto. Avalie os cronogramas planejados e os realizados, as métricas de projetos de software coletadas e analisadas, obtenha feedback dos membros da equipe e dos clientes e registre por escrito.

"Um projeto é como uma rodovia. Alguns são simples e rotineiros, como dirigir até uma loja em dia ensolarado. Entretanto, a maioria dos que valem ser realizados parece mais com dirigir um caminhão à noite em uma serra cheia de curvas."

Com Kaner, James Bach e Bret Pettichord

24.6 O PRINCÍPIO W⁵HH

Em excelente artigo sobre projeto e processo de software, Barry Boehm [Boe 96] afirma: "É necessário um princípio organizacional que facilite a obtenção de planejamentos simples para projetos simples". Boehm propõe uma abordagem voltada para os objetivos do projeto, marcos (pontos de referência) e cronogramas (agendas), responsabilidades, gerenciamento, abordagens técnicas e recursos necessários. Ele a denominou de *Princípio W⁵HH*, após uma série de perguntas que conduzem a uma definição das características-chave do projeto e do planejamento do projeto resultante:

? Como definir as características-chave do projeto?

Por que o sistema está sendo desenvolvido? Todos os interessados devem avaliar a validade das razões comerciais para o trabalho de software. O propósito justifica os gastos referentes a pessoal, tempo e dinheiro?

O que será feito? Define-se o conjunto de tarefas necessárias para o projeto.

Quando será feito? A equipe definirá o cronograma de projeto, identificando quando serão realizadas as tarefas e quando os pontos de referências (marcos) serão atingidos.

Quem será o responsável por uma função? Os papéis e responsabilidades de cada membro serão definidos.

Onde se posicionam organizacionalmente? Nem todas as situações e responsabilidades estão a cargo dos desenvolvedores de software. O cliente, os usuários e outros envolvidos também têm as suas responsabilidades.

⁷ A implicação dessa orientação é que se reduz a burocracia a um mínimo, reuniões extras são eliminadas e radicalismos ao processo e às regras do projeto são atenuados. A equipe deve ser auto-organizada e autônoma.

Como será realizado o trabalho técnico e gerencialmente? Uma vez estabelecido o escopo, deve-se definir uma estratégia técnica e gerencial.

Quanto cada recurso será necessário? A resposta a essa pergunta irá derivar-se de estimativas desenvolvidas (Capítulo 26), baseando-se nas respostas a questões anteriores.

O princípio W⁵HH de Boehm é aplicável não importando o tamanho ou complexidade do projeto. As questões apontadas fornecem excelente esquema de planejamento.

24.7 PRÁTICAS VITAIS

O Conselho de Airlie⁸ desenvolveu uma lista de “práticas de software vitais para gerenciamento baseado no desempenho”. Esses procedimentos são usados de forma consistente e considerados críticos para projetos de software altamente bem-sucedidos e por organizações cujo desempenho mínimo é consistentemente melhor que as mídias da indústria [Air 99].

Práticas vitais⁹ incluem: gerenciamento de projeto baseado em métricas (Capítulo 25), custos empíricos e estimativas de cronogramas (Capítulos 26 e 27), acompanhamento de valorização (Capítulo 27), acompanhamento de defeitos em contrapartida com os objetivos de qualidade (Capítulos 14 a 16) e gerenciamento consciente de pessoal (Seção 24.2). Cada uma dessas práticas é mencionada ao longo das Partes 3 e 4 deste livro.

FERRAMENTAS DO SOFTWARE



Ferramentas de software para gerentes de projeto

As ferramentas listadas aqui são genéricas e se aplicam a uma larga escala de atividades realizadas por gerentes de projetos. Ferramentas específicas para o gerenciamento de projeto (por exemplo, ferramentas para elaboração de cronogramas, para estimativas e para análise de riscos) serão consideradas em capítulos posteriores.

Ferramentas representativas¹⁰:

O Software Program Manager's Network (www.spmn.com) desenvolveu uma ferramenta simples denominada *Project Control Panel*

(Painel de Controle de Projeto), que oferece aos gerentes de projeto uma indicação direta do status do projeto. A ferramenta tem indicadores (medidas/padrões) semelhantes a um painel (quadro) e é implementada com o Excel da Microsoft. Está disponível para download no site www.spmn.com/products_software.html.

Ganthead.com (www.ganthead.com/) desenvolveu um conjunto de listas úteis de verificações para gerentes de projetos.

Ittoolkit.com (www.ittoolkit.com) fornece uma coletânea de guias de planejamento de processo e modelos de processo e folhas de trabalho inteligentes, disponíveis no CD-ROM.

24.8 RESUMO

O gerenciamento de projeto de software é uma atividade de apoio da engenharia de software. Inicia-se antes de qualquer atividade técnica e prossegue ao longo da modelagem, construção e utilização do software.

Os quatro Ps (4Ps) têm influência substancial no gerenciamento do projeto de software – pessoas, produto, processo e projeto. As pessoas devem ser organizadas em equipes efetivas, motivadas para realizar um trabalho de software de alta qualidade e coordenada para uma comunicação efetiva. Requisitos de produto devem ser comunicados do cliente ao desenvolvedor, decompostos em partes constituintes e posicionados para a execução pela equipe de software. O processo deve ser adaptado às pessoas e aos produtos. Uma estrutura comum de processo é

⁸ O conselho Airlie foi formado por uma equipe de especialistas em engenharia de software e registrado pelo Departamento de Defesa dos Estados Unidos para ajudar a desenvolver um guia de melhores práticas para o gerenciamento de projeto de software e para engenharia de software. Mais informações encontram-se em www.swqual.com/newsletter/vol1/no3/vol1no3.html.

⁹ Práticas vitais adotadas aqui referem-se à integridade do projeto.

¹⁰ As ferramentas aqui apresentadas não significam um aval, mas sim uma amostra dessa categoria. Na maioria dos casos, seus nomes são marcas registradas pelos respectivos desenvolvedores.

selecionada, é aplicado um paradigma de engenharia de software apropriado e um conjunto de tarefas é escolhido para que o trabalho se realize. Por fim, deve-se organizar de uma forma que capacite a equipe de software a um trabalho bem-sucedido.

O elemento-chave de todos os projetos de software são os profissionais. Engenheiros de software podem ser organizados em uma série de diferentes estruturas de equipes que abarcam desde hierarquias de controle tradicional a equipes de "paradigma aberto". Uma variedade de técnicas de comunicação e coordenação pode ser aplicada para dar suporte ao trabalho da equipe. Em geral, revisões técnicas e comunicação informal de pessoa a pessoa têm o maior valor para os desenvolvedores.

As atividades de gerenciamento de projeto englobam medições e métricas, estimativas e agendamento, análise de riscos, acompanhamento e controle. Cada um desses tópicos será considerado nos capítulos seguintes.

PROBLEMAS E PONTOS A PONDERAR

Baseando-se nas informações deste capítulo e na sua experiência, desenvolva os "dez mandamentos" para fortalecer o engenheiro de software. Ou seja, faça uma lista de dez princípios que guiarão os desenvolvedores a trabalhar com o máximo de potencial.

O People-CMM do Instituto de Engenharia de Software (SEI) adota uma visão organizada para as "áreas de processo-chave" (KPA) cultivando o bom pessoal de software. O seu instrutor irá lhe indicar uma KPA para análise e resumo.

Descreva três situações cotidianas nas quais o cliente e o usuário final são os mesmos. Descreva três situações nas quais eles são distintos.

As decisões tomadas por gerenciamento sênior podem ter impacto significativo na eficiência da equipe de engenharia de software. Forneça cinco exemplos ilustrativos em que isso seja verdadeiro.

Faça uma análise do livro do Weinberg [Wei 86] e um resumo de duas ou três páginas sobre os itens a considerar ao aplicarmos o modelo MOI.

Você foi nomeado gerente de projeto em uma organização de sistemas de informações. Sua tarefa é construir uma aplicação bastante similar a outras que sua equipe desenvolveu, embora esta seja maior e mais complexa. Requisitos foram completamente documentados pelo cliente. Qual estrutura de equipe você escolheria e por quê? Qual modelo de processo de software escolheria e por quê?

Você foi nomeado gerente de projeto em uma companhia de software. Sua tarefa é desenvolver algo inovador que combine hardware de realidade virtual com software "estado da arte". Pelo fato de a competitividade pelo mercado de entretenimento doméstico ser intensa, há pressão significativa quanto à conclusão do trabalho. Qual estrutura de equipe escolheria e por quê? Qual modelo de processo de software escolheria e por quê?

Você foi nomeado para gerente de projeto em uma companhia de software. Seu trabalho é gerenciar o desenvolvimento da versão da próxima geração do seu amplamente utilizado processador de texto. Por haver competição intensa, prazos de entrega curtos foram estabelecidos e anunciados. Qual estrutura de equipe você escolhe e por quê? Qual modelo de processo de software você escolhe e por quê?

Você foi nomeado gerente de projeto de software em uma companhia que presta serviços para o setor de engenharia genética. Seu trabalho é administrar o desenvolvimento de um novo software que acelerará o ritmo de classificação de tipos de genes. O trabalho é orientado por Pesquisa e Desenvolvimento (P&D), mas o objetivo é produzir um produto para o próximo ano. Qual estrutura de equipe você escolhe? Qual estrutura de processo você escolhe? E por quê?

Foi-lhe solicitado desenvolver uma pequena aplicação que analise cada curso oferecido por uma universidade e emita relatórios sobre a média obtida no curso (para determinada turma). Faça uma declaração de escopo que englobe esse problema.

Faça uma decomposição funcional de nível 1 da função de formatação de página discutida rapidamente na Seção 24.3.2.

LEITURAS E FONTES DE INFORMAÇÃO COMPLEMENTARES

O Project Management Institute (*Guide to the Project Management Body of Knowledge*, PMI, 2001) aborda todos os aspectos importantes do gerenciamento de projeto. Bechtold (*Essentials of Software Project Management*, 2d ed., Management Concepts, 2007), Wysocki (*Effective Software Project Management*, Wiley, 2006), Stellman e Greene (*Applied Software Project Management*, O'Reilly, 2005), e Berkun (*The Art of Project Management*, O'Reilly, 2005) ensinam práticas básicas e fornecem orientações detalhadas para todas as tarefas de gerenciamento de projeto de software. McConnell (*Professional Software Development*, Addison-Wesley, 2004) oferece informações pragmáticas para conseguir "cronogramas mais breves, produtos de melhor qualidade e projetos de melhor êxito". Henry (*Software Project Management*, Addison-Wesley, 2003) oferece conselhos práticos que serão úteis para todos os gerentes de projeto.

Tom DeMarco e seus colegas (*Adrenaline Junkies and Template Zombies*, Dorset House, 2008) escreveram um tratado bastante esclarecedor sobre os padrões humanos encontrados em todos os projetos de software. Uma excelente série de quatro volumes escrita por Weinberg (*Quality Software Management*, Dorset House, 1992, 1993, 1994, 1996) introduz conceitos de pensamento e gerenciamento para sistemas básicos, explica como usar as medições eficazmente, e trata da "ação congruente", a habilidade de estabelecer "a adaptação" entre as necessidades do gerente, as necessidades do pessoal técnico e as necessidades dos negócios. Ele apresenta informações úteis para gerentes iniciantes ou experientes. Futrell e seus colegas (*Quality Software Project Management*, Prentice-Hall, 2002) apresenta um volumoso tratado de gerenciamento de projeto. Brown e seus colegas (*Antipatterns in Project Management*, Wiley, 2000) discutem o que não fazer durante o gerenciamento de um projeto de software.

Brooks (*The Mythical Man-Month*, Anniversary Edition, Addison-Wesley, 1995) atualizou seu livro clássico para proporcionar nova visão sobre os aspectos de projeto e gerenciamento de software. McConnell (*Software Project Survival Guide*, Microsoft Press, 1997) apresenta excelente guia pragmático para aqueles que devem gerenciar projetos de software. Purba e Shah (*How to Manage a Successful Software Project*, 2d ed., Wiley, 2000) fornecem vários estudos de caso que indicam por que alguns projetos têm sucesso e outros não. Bennatan (*On Time Within Budget*, 3d ed., Wiley, 2000) dá dicas úteis e diretrizes para gerentes de projeto de software. Weigers (*Practical Project Initiation*, Microsoft Press, 2007) fornece orientações práticas para realizar de forma bem-sucedida um projeto de software desde o início.

Pode-se argumentar que o aspecto mais importante do gerenciamento de projeto de software seja o gerenciamento das pessoas. Cockburn (*Agile Software Development*, Addison-Wesley, 2002) apresenta uma das melhores discussões sobre software desenvolvido atualmente. DeMarco e Lister [DeM98] produziram o livro definitivo sobre profissionais e projetos de software. Além disso, nos últimos anos foram publicadas também as seguintes obras que devem ser examinadas:

- Cantor, M., *Software Leadership: A Guide to Successful Software Development*, Addison-Wesley, 2001.
- Carmel, E., *Global Software Teams: Collaborating Across Borders and Time Zones*, Prentice Hall, 1999.
- Constantine, L., *Peopleware Papers: Notes on the Human Side of Software*, Prentice Hall, 2001.
- Garton, C., e K. Wegryn, *Managing Without Walls*, McPress, 2006.
- Humphrey, W. S., *Managing Technical People: Innovation, Teamwork, and the Software Process*, Addison-Wesley, 1997.
- Humphrey, W. S., *TSP Coaching Development Teams*, Addison-Wesley, 2006.
- Jones, P. H., *Handbook of Team Design: A Practitioner's Guide to Team Systems Development*, McGraw-Hill, 1997.

Karolak, D. S., *Global Software Development: Managing Virtual Teams and Environments*, IEEE Computer Society, 1998.

Peters, L., *Getting Results from Software Development Teams*, Microsoft Press, 2008.

Whitehead, R., *Leading a Software Development Team*, Addison-Wesley, 2001.

Apesar de não se relacionarem especificamente com o mundo do software e muitas vezes apresentarem uma simplificação demasiada e uma ampla generalização, os *best-sellers* sobre "gerenciamento" de Kanter (*Confidence*, Three Rivers Press, 2006), Covey (*The 8th Habit*, Free Press, 2004), Bossidy (*Execution: The Discipline of Getting Things Done*, Crown Publishing, 2002), Drucker (*Management Challenges for the 21st Century*, Harper Business, 1999), Buckingham e Coffman (*First, Break All the Rules: What the World's Greatest Managers Do Differently*, Simon e Schuster, 1999), e Christensen (*The Innovator's Dilemma*, Harvard Business School Press, 1997) enfatizam "novas regras" definidas por uma economia em rápida evolução. Títulos mais antigos como *Who Moved My Cheese?*, *The One-Minute Manager* e *In Search of Excellence* continuam a proporcionar visões valiosas que podem ajudá-lo a gerenciar pessoas e projetos mais eficazmente.

Uma ampla variedade de fontes de informação sobre gerenciamento de projeto de software pode ser encontrada na Internet. Uma lista atualizada das referências na Web, relevantes ao gerenciamento e projeto de software, pode ser obtida no site www.mhhe.com/engcs/compsci/pressman/professional/olc/ser.htm.