CAPÍTULO

27 CRONOGRAMA DE PROJETO

CONCEITOS-CHAVE

No fim da década de 1960, um jovem e brilhante engenheiro foi selecionado para "escrever" um programa para uma aplicação de fábrica automatizada. O motivo de ter sido escolhido era simples. Ele era a única pessoa em seu grupo que havia participado de um seminário sobre programação de computadores. Conhecia detalhes da linguagem assembly e FORTRAN, mas não sabia nada sobre engenharia de software e menos ainda sobre cronograma e acompanhamento de projeto.

Seu chefe lhe deu os manuais apropriados e uma descrição do que precisava ser feito, além de informá-lo de que o projeto tinha de ficar pronto em dois meses.

Ele leu os manuais, escolheu sua abordagem e começou a escrever o código. Duas semanas depois, o chefe o chamou ao escritório e perguntou como as coisas estavam andando.

"Excelente", disse o jovem engenheiro com um entusiasmo juvenil. "Era mais simples do que eu pensava. Já estou com quase 75% do trabalho pronto."

O chefe sorriu e encorajou o jovem engenheiro a continuar seu excelente trabalho. Marcaram outra reunião para uma semana depois.

Uma semana depois, o chefe chamou o engenheiro ao escritório e perguntou, "Como estamos?".

"Tudo está indo muito bem", disse o jovem, "mas estou enfrentando alguns obstáculos. Vou resolver tudo isso e voltar logo à rotina normal".

"Como está o prazo de entrega?", perguntou o chefe.

"Sem problemas", disse o engenheiro. "Estou quase com 90% do trabalho pronto".

PANORAMA

O que é? Você selecionou um modelo de processo apropriado, já identificou as tarefas de engenharia de software que precisam ser executadas, estimou a mão de obra e o número de pessoas, conhece os prazos e até já considerou os riscos. Agora chegou a hora de ligar os pontos. Isto é, tem de criar uma rede de tarefas de engenharia de software que lhe permitirá terminar o trabalho no prazo. Uma vez criada a rede, precisa designar um responsável para cada tarefa, verificar se ela é feita e adaptá-la à medida que os riscos se tornam realidade. Resumindo, isso se chama cronograma e acompanhamento de projeto de software.

Quem realiza? Em nível de projeto, os gerentes de projeto de software usando informações solicitadas dos engenheiros de software. Em nível individual, os próprios engenheiros de software.

Por que é importante? Para criar um sistema complexo, muitas tarefas de engenharia de software ocorrem em paralelo, e o resultado do trabalho executado durante uma tarefa pode ter um profundo efeito sobre o trabalho a ser executado em outra tarefa. Essas interdependências são muito difíceis de entender sem um cronograma. É também praticamente impossível avaliar o progresso em um projeto de software de tamanho moderado ou grande sem um cronograma detalhado.

Quais são as etapas envolvidas? As tarefas de engenharia de software ditadas pelo modelo de processo de software são refinadas para a funcionalidade a ser criada. Duração e esforço são alocados para cada tarefa e é criada uma rede de tarefas (também chamada de "rede de atividades") para possibilitar que a equipe de software cumpra os prazos de entrega estabelecidos.

Qual é o artefato? É produzido o cronograma de projeto e informações relacionadas.

Como garantir que o trabalho foi realizado corretamente? Um cronograma apropriado requer que: (1) todos os riscos apareçam na rede, (2) esforço e duração sejam alocados inteligentemente para cada tarefa, (3) as interdependências entre as tarefas sejam indicadas corretamente, (4) sejam alocados recursos para o trabalho a ser feito, e (5) sejam alocados pontos de verificação bem próximos uns dos outros para que o progresso possa ser acompanhado.

Se você já trabalhou no mundo do software por alguns anos, pode imaginar como acaba essa história. Não ficará surpreso em saber que o jovem engenheiro[1] manteve-se nos 90% do trabalho completo durante toda a duração do projeto e terminou (com a ajuda de outros) com um mês de atraso.

Essa história tem se repetido dezenas de milhares de vezes pelos desenvolvedores de software durante as últimas cinco décadas. A grande pergunta é: por quê?

27.1 Conceitos Básicos

Embora haja muitas razões para atrasos na entrega do software, muitas delas podem ser atribuídas a uma ou mais das seguintes causas básicas:

- Um prazo de entrega não realístico estabelecido por alguém de fora da equipe de software e imposto sobre os gerentes e profissionais.
- Alterações nos requisitos do cliente não refletidas em alterações no cronograma.
- Uma subestimativa honesta do esforço e/ou quantidade de recursos que serão necessários para executar o serviço.
- Riscos previsíveis e/ou não previsíveis não considerados quando o projeto foi iniciado.
- Dificuldades técnicas que não puderam ser previstas com antecedência.
- Dificuldades humanas que não puderam ser previstas com antecedência.
- Falha de comunicação entre o pessoal de projeto que resulta em atrasos.
- Falha do gerente de projeto em não perceber o atraso do cronograma do projeto e, desse modo, não realizar nenhuma ação para corrigir o problema.

"Cronogramas apertados ou irracionais é provavelmente a influência individual mais destrutiva em todo o software."

Capers Jones

Prazos de entrega agressivos (leia-se "não realísticos") são fato comum nos negócios de software. Às vezes são exigidos por motivos até legítimos, do ponto de vista daquele que define esses prazos. Mas o bom senso indica que a legitimidade deve ser entendida também pelas pessoas que estão executando o trabalho.

Napoleão disse certa vez: "Qualquer comandante-chefe que se propõe a executar um plano que considera falho está cometendo um erro; ele deve apresentar suas razões, insistir para que o plano seja alterado e, por fim, solicitar seu afastamento em vez de ser o instrumento do fracasso de seu exército". Essas são palavras fortes que muitos gerentes de projeto de software deveriam considerar.

As atividades de estimativa discutidas no Capítulo 26 e as técnicas de cronograma discutidas neste capítulo são muitas vezes implementadas sob a pressão de um prazo de entrega definido. Se as melhores estimativas indicam que o prazo de entrega não é realístico, um gerente de projeto competente deverá "proteger sua equipe contra pressões indevidas sobre o cronograma... E enviar a pressão de volta para aqueles que a originaram" [Pag85].

"Adoro prazos de entrega. Gosto do som sibilante que eles fazem quando passam voando."

Douglas Adams

Para ilustrar, suponha que a sua equipe de software tenha sido encarregada de desenvolver um controlador em tempo real para um instrumento de diagnóstico médico que deve ser lançado no mercado em nove meses. Após cuidadosa estimativa e análise de riscos (Capítulo 28), você chega à conclusão de que o software, da maneira como foi solicitado, levará 14 meses corridos para ser criado com o pessoal disponível. Como deve proceder?

Não há sentido ir ao escritório do cliente (nesse caso, o provável cliente é o departamento de marketing/vendas) e pedir que a data de entrega seja alterada. As pressões de marketing externo ditaram a data, e o produto tem de ser entregue. É igualmente uma tolice recusar o trabalho (do ponto de vista profissional). Então, o que fazer? Recomendo tomar as seguintes providências nessa situação:

---

1 Caso esteja imaginando, essa história é autobiográfica.

1. Faça uma estimativa detalhada usando dados históricos de projetos anteriores. Determine o esforço estimado e a duração do projeto.
2. Usando um modelo incremental de processo (Capítulo 2), desenvolva uma estratégia de engenharia de software que fornecerá a funcionalidade crítica no prazo de entrega imposto, mas deixe outras funcionalidades para depois. Documente o plano.

O que devemos fazer quando a gerência nos exige um prazo de entrega impossível?

3. Reúna-se com o cliente e (usando a estimativa detalhada), explique por que o prazo de entrega imposto não é praticável. Não deixe de destacar que todas as estimativas são baseadas no desempenho de projetos anteriores. Não deixe também de indicar a porcentagem de melhora necessária para cumprir o prazo de entrega da forma como ele está definido.[2] É apropriado fazer o seguinte comentário:

   > Acredito que temos um problema com o prazo de entrega do software controlador XYZ. Já encaminhei a cada um de vocês um resumo das taxas de produtividade de desenvolvimento de projetos realizados e uma estimativa que já fizemos de muitas maneiras diferentes. Vocês devem observar que considerei uma melhora de 20% nas taxas de produtividade anteriores, mas ainda temos um prazo de entrega de 14 meses, em vez de 9 meses.

4. Ofereça a estratégia de desenvolvimento incremental como uma alternativa:

   > Tenho algumas opções a oferecer e gostaria que vocês tomassem a decisão com base nelas. Primeiro, podemos aumentar o orçamento e buscar recursos adicionais para termos certeza de completar o trabalho em nove meses. Mas entendo que isso aumentará o risco de má qualidade devido ao prazo apertado.[3] Segundo, podemos remover uma série de funções e recursos do software que vocês estão solicitando. Isso tornará a versão preliminar do produto um pouco menos funcional, mas podemos anunciar uma funcionalidade completa e fornecê-la após 14 meses. Terceiro, podemos ignorar toda a realidade e esperar completar o projeto em nove meses. Vamos acabar não tendo nada a entregar ao cliente. A terceira opção, espero que concordem comigo, é inaceitável. Dados anteriores e nossas melhores estimativas dizem que isso não é realístico: é uma receita para o desastre.

Haverá protestos, mas se for apresentada uma estimativa sólida baseada em bons dados históricos, é provável que sejam escolhidas versões negociadas das opções 1 ou 2. O prazo de entrega não realístico é eliminado.

27.2 Cronograma de Projeto

Perguntaram certa vez a Fred Brooks como é que os projetos de software acabam se atrasando. Sua resposta foi tão simples quanto profunda: "Um dia de cada vez".

A realidade de um projeto técnico (pode envolver a construção de uma usina hidrelétrica ou o desenvolvimento de um sistema operacional) é que centenas de pequenas tarefas devem ocorrer para se atingir o objetivo maior. Algumas dessas tarefas estão fora da rotina principal e podem ser completadas sem muita preocupação quanto ao impacto na data de entrega do projeto. Outras tarefas estão no "caminho crítico". Se essas tarefas "críticas" atrasarem-se, o prazo de entrega do projeto inteiro é ameaçado.

AVISO

As tarefas necessárias para que o gerente de projetos atinja seus objetivos não devem ser executadas manualmente. Há muitas ferramentas excelentes para cronogramas. Use-as.

Como gerente de projetos, o seu objetivo é definir todas as tarefas, criar uma rede que mostre suas interdependências, identificar as tarefas críticas dentro da rede e acompanhar o progresso para garantir que os atrasos sejam detectados "um dia de cada vez". Para tanto, deve ter um cronograma definido com um grau de resolução que permita monitorar o progresso e controlar o projeto.

Cronograma de projeto de software é uma atividade que distribui o esforço estimado por toda a duração planejada do projeto alocando esse esforço para tarefas específicas de engenharia de software. No entanto, é importante notar que o cronograma evolui com o tempo. Durante os

2 Se a melhora exigida for de 10 a 25%, pode ser possível fazer o trabalho no prazo. Mas é mais provável que a melhora de desempenho da equipe precisará ser maior do que 50%. Essa é uma expectativa não realística.

3 Você pode argumentar também que o aumento do número de pessoas não reduz proporcionalmente o prazo.

primeiros estágios do planejamento do projeto, desenvolve-se um cronograma macroscópico. Este identifica as principais atividades do processo e as funções do produto para as quais se aplicam. Conforme o projeto caminha, cada item é refinado em um cronograma detalhado. Nesse momento, ações e tarefas de software específicas (necessárias para realizar uma atividade) são identificadas e dispostas em um cronograma.

"A sobreposição de cronogramas otimistas não resulta em cronogramas reais mais breves, resulta em mais longos."

Steve McConnell

O cronograma para projetos de engenharia de software pode ser visto sob duas perspectivas bem diferentes. Na primeira, uma data final para entrega de um sistema computacional já foi definida (de maneira irreversível). A organização de software é forçada a distribuir o esforço no prazo prescrito. A segunda visão considera que os limites cronológicos aproximados foram discutidos, mas a data final é definida pela organização de engenharia de software. O esforço é distribuído para fazer o melhor uso dos recursos, e uma data final é refinada após cuidadosa análise do software. Infelizmente, a primeira situação ocorre com muito maior frequência do que a segunda.

27.2.1 Princípios básicos

Assim como em todas as outras áreas da engenharia de software, há uma série de princípios básicos que guiam os cronogramas de projeto de software:

PONTO-CHAVE

Ao desenvolver um cronograma, divida o trabalho, anote as dependências entre as tarefas, atribua esforço e tempo para cada tarefa e defina responsabilidades, resultados e pontos de controle.

Divisão do trabalho. O projeto deve ser dividido em uma série de atividades e tarefas gerenciáveis. Para tanto, o produto e o processo são refinados.

Interdependência. Deve ser determinada a interdependência de cada atividade ou tarefa resultante da divisão. Algumas tarefas devem ocorrer em sequência, enquanto outras podem acontecer em paralelo. Certas atividades não podem começar enquanto o resultado de uma outra não esteja disponível. Outras atividades podem ocorrer de forma independente.

Alocação do tempo. Para cada tarefa a ser programada deve ser alocado certo número de unidades de trabalho (por exemplo, pessoas-dias de esforço). Além disso, para cada tarefa deve ser definida uma data de início e uma data de término, que são uma função das interdependências e se o trabalho será realizado em tempo integral ou parcial.

Validação do esforço. Cada projeto tem um número definido de pessoas na equipe de software. Na medida em que ocorre a alocação do tempo, você deve assegurar que, em determinado momento, não seja programado mais do que o número alocado de profissionais. Por exemplo, considere um projeto para o qual são designados três engenheiros de software (três pessoas-dia estão disponíveis por dia de esforço atribuído[4]). Em determinado dia, sete tarefas concorrentes devem ser executadas. Cada tarefa requer 0,50 pessoas-dia de esforço. Foi alocado mais esforço do que pessoas disponíveis para fazer o trabalho.

Definição de responsabilidades. Cada tarefa que é disposta em cronograma deverá ser atribuída a um membro específico da equipe.

Definição dos resultados. Cada tarefa disposta em cronograma deve ter um resultado definido. Para projetos de software, o resultado normalmente é um artefato (por exemplo, o projeto de um componente) ou uma parte de um artefato. Artefatos muitas vezes são combinados em entregáveis.

Definição dos pontos de controle (milestones). Cada tarefa ou grupo de tarefas deve estar associada a um ponto de controle no projeto. Um ponto de controle é atingido quando um ou mais artefatos teve sua qualidade examinada (Capítulo 15) e foi aprovado.

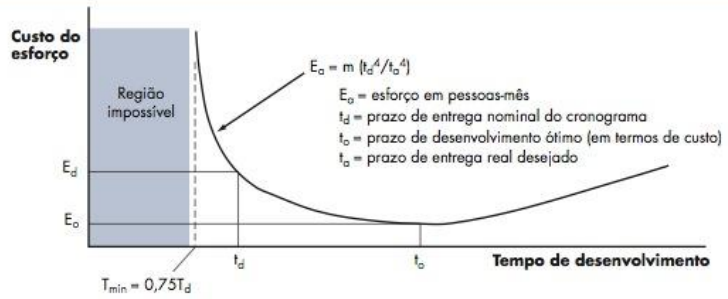
Cada um desses princípios é aplicado à medida que o projeto evolui.

27.2.2 Relação entre pessoas e esforço

Em um pequeno projeto de desenvolvimento de software uma única pessoa pode analisar os requisitos, fazer o projeto, gerar o código e realizar os testes. Conforme o tamanho do projeto aumenta, mais profissionais devem ser envolvidos. (Raramente podemos nos dar ao luxo de considerar o esforço de dez pessoas-ano sendo executado por um só indivíduo trabalhando por dez anos!)

4 Na realidade há menos de 3 pessoas-dia disponíveis devido a reuniões, ausência por doença, férias e uma variedade de outras razões. Para nossos propósitos, no entanto, vamos considerar uma disponibilidade de 100%.

FIGURA 27.1
A relação entre esforço e prazo de entrega



AVISO

Se você tiver de acrescentar pessoas a um projeto atrasado, não se esqueça de atribuir-lhes trabalho que já esteja bastante dividido.

Há um mito comum no qual ainda acreditam muitos gerentes responsáveis por projetos de desenvolvimento de software: "Se nos atrasarmos, podemos sempre acrescentar mais programadores e recuperar o tempo perdido mais tarde". Infelizmente, acrescentar pessoas nas últimas fases de um projeto muitas vezes tem um efeito prejudicial, fazendo o cronograma se arrastar ainda mais. Os profissionais incluídos precisam aprender sobre o sistema, e os encarregados de ensiná-los são os mesmos que estavam fazendo o trabalho. Enquanto explicam, nada é feito, e o projeto torna-se ainda mais atrasado.

Além do tempo necessário para conhecer o sistema, mais pessoas aumentam o número de caminhos de comunicação e a complexidade das comunicações em todo o projeto. Embora a comunicação seja absolutamente essencial para o desenvolvimento bem-sucedido do software, cada novo caminho requer esforço adicional e, portanto, tempo adicional.

Durante anos, dados empíricos e análises teóricas têm demonstrado que os cronogramas de projeto são elásticos. É possível abreviar uma data de conclusão desejada para um projeto (acrescentando recursos) até certo ponto. É possível também estender uma data de conclusão (reduzindo o número de recursos).

PONTO-CHAVE

Se a entrega pode ser atrasada, a curva PNR indica que os custos do projeto podem ser reduzidos substancialmente.

A *curva Putnam-Norden-Rayleigh* (*PNR*)[5] fornece uma indicação da relação entre esforço aplicado e prazo de entrega para um projeto de software. Uma versão da curva, representando esforço de projeto em função do prazo de entrega, é apresentada na Figura 27.1. A curva indica um valor mínimo t_o, que representa o custo mínimo para a entrega (o prazo de entrega que resultará no trabalho mínimo despendido). Quando nos movemos para a esquerda de t_o (quando tentamos acelerar a entrega), a curva sobe não linearmente.

Como exemplo, suponhamos que uma equipe de projeto tenha estimado que um nível de esforço E_d será necessário para conseguir um prazo de entrega nominal t_d que é ótimo em termos de cronograma e recursos disponíveis. Embora seja possível acelerar a entrega, a curva sobe de forma muito aguda para a esquerda de t_d. De fato, a curva PNR indica que o prazo de entrega do projeto não pode ser comprimido além de $0{,}75t_d$. Se tentarmos uma compressão maior, o projeto passará para a "região impossível", e o risco de fracasso se torna muito alto. A curva PNR indica também que para a opção de prazo de entrega de custo mais baixo, $t_o = 2t_d$. A implicação aqui é que o atraso na entrega pode reduzir os significativamente custos. Naturalmente, isso deve ser ponderado, levando-se em conta as consequências comerciais associadas com o atraso.

A equação do software [Put92] introduzida no Capítulo 26 é derivada da curva PNR e demonstra a relação altamente não linear entre o tempo cronológico para completar um projeto e o esforço humano aplicado ao projeto. O número de linhas de código produzidas, *L*, está relacionado a esforço e tempo de desenvolvimento pela equação:

$$L = P \times E^{1/3} t^{4/3}$$

5 As pesquisas originais podem ser encontradas em [Nor70] e [Put78].

em que E é o esforço de desenvolvimento em pessoas-mês, P é um parâmetro de produtividade que reflete uma variedade de fatores que levam a uma alta qualidade do trabalho de engenharia de software (valores típicos para P variam entre 2.000 e 12.000) e t é a duração do projeto em meses corridos.

AVISO

Conforme o prazo de entrega se torna cada vez mais apertado, você chega a um ponto em que o trabalho não pode ser terminado no prazo, independentemente do número de pessoas que estejam trabalhando. Enfrente a realidade e defina uma nova data de entrega.

Rearrajando essa equação de software, podemos chegar a uma expressão para trabalho de desenvolvimento E:

$$E = \frac{L^3}{P^3 t^4} \tag{27.1}$$

em que E é o esforço despendido (em pessoas-ano) durante todo o ciclo de vida do desenvolvimento de software e manutenção e t é o tempo de desenvolvimento em anos. A equação para o esforço de desenvolvimento pode ser relacionada com o custo do desenvolvimento pela inclusão de um fator inflacionado da taxa de trabalho ($/pessoa-ano).

Isso leva a alguns resultados interessantes. Considere um projeto de software complexo, de tempo real, estimado em 33.000 LOC, 12 pessoas-ano de esforço. Se forem atribuídas 8 pessoas para a equipe de projeto, o projeto pode ser feito em aproximadamente 1,3 ano. No entanto, se estendermos a data final para 1,75 ano, a natureza altamente não linear do modelo descrito na Equação (27.1) nos dará:

$$E = \frac{L^3}{P^3 t^4} \sim 3{,}8 \text{ pessoas-ano}$$

Isso implica que, estendendo em seis meses a data de entrega, podemos reduzir o número de pessoas de oito para quatro! A validade desses resultados está aberta ao debate, mas a consequência é clara: podem ser obtidos benefícios usando menos pessoas durante um período de tempo um pouco mais longo para atingir o mesmo objetivo.

27.2.3 Distribuição de esforço

Cada uma das técnicas de estimativa de projeto de software discutidas no Capítulo 26 leva a estimativas de unidades de trabalho (por exemplo, pessoa-mês) necessárias para completar o desenvolvimento do software. Uma distribuição recomendada do trabalho durante o processo de software muitas vezes é conhecida como *regra 40–20–40*. Quarenta por cento de todo o esforço é alocado na análise preliminar e projeto. Uma porcentagem similar é aplicada ao teste posterior. Você pode inferir corretamente que a codificação (20% do trabalho) está em segundo plano.

? Como o trabalho deve ser distribuído no fluxo do processo de software?

Essa distribuição do esforço é usada apenas como guia.[6] As características de cada projeto ditam a distribuição de esforços. O trabalho despendido em planejamento de projeto raramente passa de 2 a 3% do trabalho, a menos que o plano induza uma organização a grandes despesas com alto risco. A comunicação com o cliente e as análises de requisitos podem chegar de 10 a 25% do trabalho do projeto. O esforço gasto em análise e protótipo deve aumentar em proporção direta com o tamanho e complexidade do projeto. Uma faixa de 20 a 25% do esforço é aplicada normalmente ao projeto de software. Deve ser considerado também o tempo gasto para revisão de projeto e subsequente iteração.

Devido ao esforço aplicado no projeto do software, a codificação deve seguir com relativamente pouca dificuldade. Pode-se aceitar um intervalo de 15 a 20 por cento do esforço total. O teste e o debugging subsequente podem totalizar 30 a 40 por cento do esforço de desenvolvimento do software. A criticidade do software muitas vezes determina o volume de teste necessário. Se o software estiver relacionado com vidas humanas (isto é, a falha do software pode resultar em perda de vidas humanas), as porcentagens podem ser tipicamente mais altas.

6 Hoje, a regra 40-20-40 é criticada. Alguns acreditam que se deveria despender mais de 40% do esforço total durante a análise e o projeto. Por outro lado, alguns defensores do desenvolvimento ágil (Capítulo 3) argumentam que deveria ser gasto menos tempo "no início" e que uma equipe deveria passar rapidamente para a construção.

27.3 Definindo um Conjunto de Tarefas para o Projeto de Software

Independentemente do modelo de processo escolhido, o trabalho que uma equipe executa é obtido por meio de uma série de tarefas que permitem definir, desenvolver e, por fim, suportar software de computador. Não há um conjunto único de tarefas que seja apropriado para todos os projetos. O conjunto de tarefas que seria apropriado para um sistema grande e complexo, provavelmente seria considerado exagerado para um software pequeno e razoavelmente simples. Portanto, um processo de software eficaz definiria uma coleção de conjuntos de tarefas, cada uma delas projetada para atender às necessidades de diferentes tipos de projetos.

Conforme mencionamos no Capítulo 2, um conjunto de tarefas é uma coleção de tarefas de engenharia de software, pontos de controle, artefatos e filtros de garantia de qualidade que precisam ser obtidos para completar um projeto em particular. O conjunto de tarefas deve proporcionar disciplina o bastante para obter alta qualidade do software. Mas, ao mesmo tempo, não deve sobrecarregar a equipe com trabalho desnecessário.

Para desenvolver um cronograma de projeto, um conjunto de tarefas deve ser distribuído ao longo da duração do projeto. O conjunto irá variar dependendo do tipo de projeto e do grau de rigor com que a equipe decide fazer seu trabalho. Embora seja difícil desenvolver uma classificação abrangente de tipos de projeto, muitas organizações de software encontram os seguintes projetos:

WebRef
Um modelo de processo adaptável (*Adaptable Process Model* – APM) foi desenvolvido para ajudar na definição de conjuntos de tarefas para vários projetos de software. Uma descrição completa do APM pode ser encontrada no site **www.rspa.com/apm**.

1. *Projetos de desenvolvimento de conceito* iniciados para explorar algum conceito novo de negócio ou aplicação de uma nova tecnologia.
2. *Projetos de desenvolvimento de novas aplicações* feitos em consequência de uma solicitação de um cliente específico.
3. *Projetos de aperfeiçoamento de aplicação* ocorrem quando um software existente passa por grandes modificações em sua função, desempenho ou interfaces observáveis pelo usuário final.
4. *Projetos de manutenção de aplicação* corrigem, adaptam ou ampliam software existente de maneira não muito óbvia ao usuário final.
5. *Projetos de reengenharia* empreendidos com a intenção de recriar um sistema existente (legado) no todo ou em parte.

Mesmo em um único tipo de projeto, muitos fatores influenciam o conjunto de tarefas a ser selecionado. Esses fatores incluem [Pre05]: tamanho do projeto, número de usuários potenciais, importância da missão, longevidade da aplicação, estabilidade dos requisitos, facilidade de comunicação cliente/desenvolvedor, maturidade da tecnologia aplicável, restrições de desempenho, características internas e não internas, pessoal de projeto e fatores de reengenharia. Quando tomados de forma combinada, esses fatores fornecem uma indicação do *grau de rigor* com que o processo de software deve ser aplicado.

27.3.1 Exemplo de conjunto de tarefas

Projetos de desenvolvimento de conceito ocorrem quando deve ser explorado o potencial para alguma nova tecnologia. Não há certeza de que a tecnologia será aplicável, mas um cliente (por exemplo, marketing) acredita que existem benefícios potenciais. Projetos de desenvolvimento de conceito são abordados aplicando as seguintes ações:

1.1 Definição do escopo do conceito determina o escopo geral do projeto.

1.2 Planejamento preliminar do conceito estabelece a capacidade da organização em assumir o trabalho originado pelo escopo do projeto.

1.3 Avaliação do risco da tecnologia avalia o risco associado à tecnologia a ser implementada como parte do escopo de projeto.

1.4 Prova de conceito demonstra a viabilidade de uma nova tecnologia no contexto de software.

1.5 Implementação do conceito implementa a representação do conceito de maneira que possa ser examinada por um cliente e usada para finalidades de "marketing" quando um conceito deve ser vendido para outros clientes ou gerentes.

1.6 Reação do cliente ao conceito que solicita informações sobre um novo conceito de tecnologia e focaliza aplicações especiais do cliente.

Um rápido exame dessas ações resultará em algumas surpresas. Na verdade, o fluxo da engenharia de software para projetos de desenvolvimento do conceito (e para todos os outros tipos de projetos também) não é nada mais do que bom senso.

27.3.2 Refinamento das ações de engenharia de software

As ações de engenharia de software descritas na seção anterior podem ser usadas para definir um cronograma macroscópico para um projeto. No entanto, este deve ser refinado para criar um cronograma de projeto detalhado. O refinamento começa tomando cada ação e decompondo-a em uma série de tarefas (com os artefatos correspondentes e seus pontos de controle).

Como exemplo de decomposição de tarefa, considere a Ação 1.1, Escopo do conceito. O refinamento da tarefa pode ser conseguido usando-se o formato de esboço, mas neste livro empregamos a abordagem de uma linguagem de projeto de processo para ilustrar o fluxo das ações de escopo de conceito:

```
Definição de tarefa: Ação 1.1 Definição do Escopo do Conceito
1.1.1   Identifique as necessidades, benefícios e clientes potenciais;
1.1.2   Defina saída/controle desejados e eventos de entrada que orientam a aplicação;
        Início da Tarefa 1.1.2
        1.1.2.1 RT: Reveja a descrição da necessidade[7]
        1.1.2.2 Faça uma lista de saídas/entradas visíveis ao cliente
        1.1.2.3 RT: Examine saídas/entradas com o cliente e revise conforme necessário;
        Fim da tarefa 1.1.2
1.1.3   Defina a funcionalidade/comportamento para cada função principal;
        Início da Tarefa 1.1.3
        1.1.3.1 RT: Examine os objetos saída e entrada de dados produzidos na tarefa 1.1.2;
        1.1.3.2 Crie um modelo de funções/comportamentos;
        1.1.3.3 RT: Examine as funções/comportamentos com o cliente e revise conforme necessário;
        Fim da tarefa 1.1.3
1.1.4   Isole os elementos da tecnologia a ser implementados em software;
1.1.5   Pesquise a disponibilidade de software existente;
1.1.6   Defina a viabilidade técnica;
1.1.7   Faça uma estimativa rápida do tamanho;
1.1.8   Crie uma definição do escopo;
        Fim da definição: Ação 1.1
```

As tarefas e subtarefas mencionadas no refinamento da linguagem de projeto de processo formam a base de um cronograma detalhado para a ação de escopo de conceito.

27.4 Definindo uma Rede de Tarefas

PONTO-CHAVE
A rede de tarefas é um mecanismo útil para mostrar as dependências entre elas e determinar o caminho crítico.

As tarefas e subtarefas individuais têm interdependências baseadas em sua sequência. Além disso, quando há mais de uma pessoa envolvida em um projeto de engenharia de software, é provável que as atividades e tarefas de desenvolvimento sejam executadas em paralelo. Quando isso ocorre, tarefas concorrentes devem ser coordenadas para que estejam prontas quando outras mais adiante necessitarem de seus artefatos.

Uma *rede de tarefas*, também chamada de *rede de atividades*, é uma representação gráfica do fluxo de tarefas de um projeto. Às vezes é usada como um mecanismo por meio do qual a sequência

7 RT indica que deve ser feita uma revisão técnica (Capítulo 15).

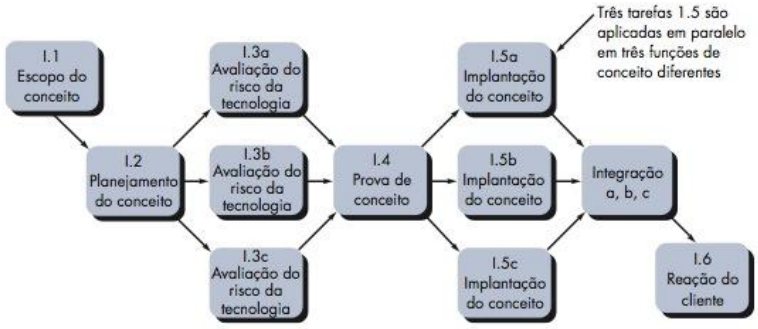
e dependências de tarefa são colocadas em uma ferramenta automática de cronograma de projeto. Em sua forma mais simples (usada ao criar um cronograma macroscópico), a rede de tarefas representa as principais ações da engenharia de software. A Figura 27.2 mostra uma rede de tarefas esquemática para um projeto de desenvolvimento de conceito.

A natureza concorrente das ações de engenharia de software leva a uma série de requisitos importantes para a elaboração de cronograma. Como tarefas paralelas ocorrem de forma assíncrona, você deverá determinar as dependências entre elas para garantir o progresso contínuo até o término do trabalho. Além disso, atenção àquelas tarefas que ficam no *caminho crítico*. Isto é, tarefas que devem ser terminadas no prazo para que o projeto como um todo possa encerrar no prazo. Esses problemas serão discutidos em mais detalhes ainda neste capítulo.

É importante notar que a rede de tarefas da Figura 27.2 é macroscópica. Em uma rede de tarefas detalhada (o precursor de um cronograma detalhado), cada ação na figura seria expandida. Por exemplo, a Tarefa 1.1 seria expandida para mostrar todas as tarefas detalhadas no refinamento das Ações 1.1 mostradas na Seção 27.3.2.

FIGURA 27.2

Uma rede de tarefas para desenvolvimento de conceito



27.5 CRONOGRAMA

> "Tudo o que temos de decidir é o que fazer com o tempo que nos é concedido."
>
> ***Gandalf, em O senhor dos anéis: a sociedade do anel***

O cronograma de um projeto de software não difere muito do cronograma de qualquer esforço de engenharia multitarefa. Portanto, ferramentas e técnicas generalizadas de cronogramas podem ser aplicadas com poucas modificações aos projetos de software.

A *técnica de avaliação e revisão de programa* (*Program Evaluation and Review Technique* - PERT) e o *método do caminho crítico* (*Critical Path Method* – CPM) são dois métodos de cronograma de projetos que podem ser aplicados ao desenvolvimento de software. Ambas as técnicas são controladas por informações já desenvolvidas em atividades anteriores de planejamento de projeto: estimativas de esforço, uma decomposição da função do produto, a escolha do modelo de processo apropriado e o conjunto de tarefas e decomposição das tarefas que estão selecionadas.

As interdependências entre as tarefas podem ser definidas por meio de uma rede de tarefas. Tarefas, também chamadas de *estrutura de subdivisão do trabalho* (*Work Breakdown Structure* – WBS) do projeto, são definidas para o produto como um todo ou para funções individuais.

Tanto PERT quanto CPM fornecem ferramentas quantitativas que lhe permitem (1) determinar o caminho crítico – a cadeia de tarefas que determinam a duração do projeto, (2) estabelecer estimativas de tempo "mais prováveis" para tarefas individuais aplicando modelos estatísticos, e (3) calcular "tempos-limite" que definem uma "janela" de tempo para uma tarefa em particular.

FERRAMENTAS DO SOFTWARE

Cronograma de projeto

Objetivo: o objetivo das ferramentas de cronograma de projeto é permitir que o gerente defina as tarefas; estabeleça suas dependências; atribua recursos humanos às tarefas e desenvolva uma variedade de cartas, diagramas e tabelas que ajudem a acompanhar e controlar o projeto de software.

Mecânica: em geral, as ferramentas de cronograma de projeto requerem a especificação de uma estrutura de subdivisão do trabalho das tarefas ou a geração de uma rede de tarefas. Uma vez definido o desmembramento (um esboço) ou rede de tarefas, datas de início e fim, recursos humanos, prazos de entrega e outros dados são anexados a cada uma delas. A ferramenta gera então uma variedade de cartas de tempos e outras tabelas que permitem ao gerente avaliar o fluxo de tarefas. Esses dados podem ser atualizados continuamente no decorrer do projeto.

Ferramentas representativas:[8]

AMS Realtime, desenvolvida pela Advanced Management Systems (**www.amsusa.com**), tem recursos de cronograma para projetos de todos os tamanhos e tipos.

Microsoft Project, desenvolvida pela Microsoft (**www.microsoft.com**), é a ferramenta de cronograma de projeto baseada em PC mais amplamente utilizada.

4C, desenvolvida pela *4C Systems* (**www.4csys.com**), suporta todos os aspectos do planejamento de projeto incluindo cronograma.

Uma lista abrangente de fornecedores e artefatos de software de gerenciamento de projeto pode ser encontrada no site **www.infogoal.com/pmc/pmcswr.htm**.

27.5.1 Gráfico de Gantt

Ao criar o cronograma de um projeto de software, você começa com um conjunto de tarefas (a estrutura de subdivisão do trabalho). Se forem usadas ferramentas automáticas, a subdivisão de trabalho entra como uma rede de tarefas ou resumo de tarefas. Dados de esforço, duração e data de início são então definidos para cada tarefa. Além disso, as tarefas podem ser atribuídas a indivíduos específicos.

PONTO-CHAVE

Um gráfico de Gantt permite determinar que tarefas serão executadas em determinado ponto no tempo.

Como resultado dessas informações, é gerado um *gráfico de Gantt* (*Gantt chart*). Um gráfico de Gantt pode ser desenvolvido para o projeto inteiro, ou podem ser desenvolvidos gráficos separados para cada função do projeto ou para cada indivíduo que trabalha no projeto.

A Figura 27.3 ilustra o formato de um gráfico de Gantt. Ela mostra uma parte de um cronograma de projeto de software que destaca a tarefa de escopo do conceito para um software processador de texto (*Word Processor* – WP). Todas as tarefas do projeto (para escopo do conceito) são listadas na coluna da esquerda. As barras horizontais indicam a duração de cada tarefa. Quando ocorrem múltiplas barras ao mesmo tempo no calendário, é sinal de que há concorrência de tarefas. Os losangos indicam pontos de controle.

Uma vez introduzidas as informações necessárias para a geração de uma carta de tempo, a maioria das ferramentas de cronograma de projeto de software produz *tabelas de projeto* – uma listagem tabular de todas as tarefas de projeto, suas datas de início e fim planejada e atual, e uma variedade de informações relacionadas (Figura 27.4). Usadas em conjunto com a carta de tempo, as tabelas de projeto permitem acompanhar o progresso.

27.5.2 Acompanhando o cronograma

"A regra básica do relatório de status de software pode ser resumida em uma única frase: 'Sem surpresas'."

Capers Jones

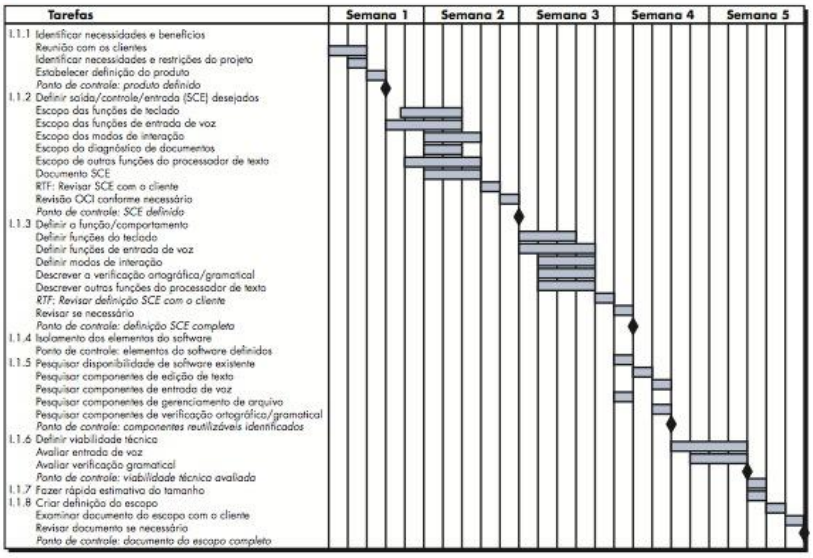
Se tiver sido bem desenvolvido, o cronograma de projeto torna-se um roteiro que define as tarefas e pontos de controle a ser acompanhados e controlados à medida que o projeto avança. O acompanhamento pode ser feito de várias maneiras diferentes:

- Promovendo reuniões periódicas sobre o estado do projeto nas quais cada membro da equipe relata o progresso e os problemas

8 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.

FIGURA 27.3

Exemplo de gráfico de Gantt



- Avaliando os resultados de todas as revisões feitas durante o processo de engenharia de software
- Determinando se os pontos de controle formais do projeto (os losangos da Figura 27.3) foram atingidos na data programada
- Comparando a data de início real com a data de início programada para cada tarefa de projeto listada na tabela de recursos (Figura 27.4)

FIGURA 27.4

Exemplo de tabela de projeto

Tarefas	Início planejado	Início real	Término planejado	Término real	Pessoa designada	Esforço alocado	Notas
I.1.1 Identificar necessidades e benefícios							Escopo
Reunião com clientes	sem1, d1	sem1, d1	sem1, d2	sem1, d2	BLS	2 p-d	necessitará
Identificar necessidades e restrições do projeto	sem1, d2	sem1, d2	sem1, d2	sem1, d2	JPP	1 p-d	mais
Estabelecer definição do produto	sem1, d3	sem1, d3	sem1, d3	sem1, d3	BLS/JPP	1 p-d	esforço/tempo
Ponto de controle: produto definido	sem1, d3	sem1, d3	sem1, d3	sem1, d3			
I.1.2 Definir saída/controle/entrada (SCE) desejada							
Escopo das funções de teclado	sem1, d4	sem1, d4	sem2, d2		BLS	1,5 p-d	
Escopo das funções de entrada de voz	sem1, d3	sem1, d3	sem2, d2		JPP	2 p-d	
Escopo dos modos de interação	sem2, d1		sem2, d3		MLL	1 p-d	
Escopo do diagnóstico de documentos	sem2, d1		sem2, d2		BLS	1,5 p-d	
Escopo de outras funções de processamento de texto	sem1, d4	sem1, d4	sem2, d3		JPP	2 p-d	
Documento SCE	sem2, d1		sem2, d3		MLL	3 p-d	
RTF: Revisar SCE com o cliente	sem2, d3		sem2, d3		all	3 p-d	
Revisar SCE se necessário	sem2, d4		sem2, d4		all	3 p-d	
Ponto de controle: SCE definido	sem2, d5		sem2, d5				
I.1.3 Definir a função/comportamento							

- Reunindo-se informalmente com os profissionais para obter sua avaliação subjetiva do progresso até o momento e os problemas previstos
- Usando análise de valor agregado (Seção 27.6) para avaliar o progresso quantitativamente

A melhor indicação do progresso é a conclusão e revisão bem-sucedida de um artefato de software definido.

Na realidade, todas essas técnicas de acompanhamento são empregadas por gerentes de projeto experientes.

O controle é usado por um gerente de projeto de software para administrar os recursos do projeto, enfrentar os problemas e dirigir a equipe. Se tudo estiver bem (isto é, o projeto dentro do prazo e do orçamento, as revisões indicando que há progresso real e os pontos de controle estão sendo alcançados), o controle é fácil. Mas quando ocorrem problemas, você deve exercer o seu controle para conciliar todos os itens o mais rápido possível. Depois que um problema foi diagnosticado, recursos adicionais podem ser focalizados na área problemática: o pessoal pode ser realocado ou o cronograma do projeto redefinido.

Quando enfrentam pressões severas de prazo de entrega, os gerentes de projeto experientes às vezes usam uma técnica de cronograma e controle de projeto chamada de *time-boxing* (caixa de tempo) [Jal04]. A estratégia caixa de tempo reconhece que o produto completo pode não estar pronto no prazo de entrega predefinido. Assim, é escolhido um paradigma de software incremental (Capítulo 2) e gerado um cronograma para cada entrega incremental.

PONTO-CHAVE

Quando é atingida a data de término de uma tarefa de tempo limitado, o trabalho daquela tarefa é interrompido e a próxima tarefa começa.

As tarefas associadas a cada incremento são então limitadas em tempo. Isso significa que o cronograma para cada tarefa é ajustado retroativamente a partir da data de entrega para o incremento. Uma "caixa" é traçada ao redor de cada tarefa. Quando uma tarefa chega ao limite de sua caixa de tempo (mais ou menos 10%), o trabalho é interrompido e inicia-se a próxima tarefa.

A reação inicial à abordagem caixa de tempo quase sempre é negativa: "Se o trabalho não está pronto, como podemos prosseguir?". A resposta está na maneira como o trabalho é feito. Quando se atinge o limite da caixa de tempo, é provável que 90% da tarefa já tenha sido feita.[9] Os restantes 10%, embora importantes, podem (1) ser adiados até o próximo incremento ou (2) serem concluídos mais tarde se for necessário. Em vez de ficar "preso" em uma tarefa, o projeto prossegue em direção à data de entrega.

27.5.3 Acompanhando o progresso de um projeto orientado a objeto

Embora um modelo iterativo seja a melhor estrutura para um projeto orientado a objeto, o paralelismo de tarefas torna difícil o acompanhamento do projeto. Você pode ter dificuldades para estabelecer pontos de controle significativos para um projeto orientado a objeto porque uma série de coisas diferentes está acontecendo ao mesmo tempo. Em geral, os seguintes pontos de controle principais podem ser considerados "completos" quando os seguintes critérios forem atingidos.

Ponto de controle técnico: Análise orientada a objeto completa

- Todas as classes e a hierarquia de classes foram definidas e revisadas.
- Os atributos de classe e operações associados a uma classe foram definidos e revisados.
- Os relacionamentos entre classes (Capítulo 6) foram estabelecidos e revisados.
- Um modelo comportamental (Capítulo 7) foi criado e revisado.
- As classes reutilizáveis foram identificadas.

Ponto de controle técnico: projeto orientado a objeto completo

- O conjunto de subsistemas foi definido e revisado.
- Classes foram alocadas a subsistemas e revisadas.
- A alocação de tarefas foi estabelecida e revisada.
- As responsabilidades e colaborações foram identificadas.

9 Um cínico pode se recordar do ditado: "Os primeiros 90% do sistema tomam 90% do tempo; os restantes 10% tomam 90% do tempo."

- Os atributos e operações foram atribuídos e revisados.
- O modelo de comunicação foi criado e revisado.

Ponto de controle técnico: a programação orientada a objeto está completa

- Cada nova classe foi implementada em código por meio do modelo de projeto.
- As classes extraídas (de uma biblioteca de reutilização) foram implementadas.
- Foi criado o protótipo ou incremento.

Ponto de controle técnico: teste orientado a objeto

Depuração e teste ocorrem em harmonia. O status da depuração muitas vezes é avaliado considerando-se o tipo e número de erros "pendentes" (bugs).

- Foi examinada a exatidão e totalidade dos modelos de análise e projeto orientado a objeto.
- Foi desenvolvida e revisada a rede responsabilidade-colaboração de classe (Capítulo 6).
- Foram criados os casos de teste e feitos os testes em nível de classe (Capítulo 19) para cada classe.
- Os casos de teste foram criados e o teste de conjunto (Capítulo 19) está completo e as classes integradas.
- Foram finalizados os testes de sistema.

Lembrando que o modelo de processo orientado a objeto é iterativo, cada um desses pontos de controle deve ser revisitado conforme diferentes incrementos são entregues ao cliente.

27.5.4 Cronograma para projetos para WebApp

O *cronograma para projeto para WebApp* distribui o esforço estimado ao longo do tempo planejado (duração) para criar cada incremento de WebApp. Isso é conseguido alocando-se o esforço para tarefas específicas. É importante notar, no entanto, que o cronograma para WebApp geral evolui com o tempo. Durante a primeira iteração, é desenvolvido um cronograma macroscópico. Este identifica todos os incrementos de WebApp e projeta as datas nas quais cada um será entregue. À medida que o desenvolvimento de um incremento progride, o item para o incremento no cronograma macroscópico é refinado em um cronograma detalhado. Nesse momento, são identificadas e agendadas tarefas específicas de desenvolvimento (necessárias para executar uma atividade).

Como exemplo de cronograma macroscópico, considere a WebApp **CasaSeguraGarantida.com**. Recordando discussões anteriores do **CasaSeguraGarantida.com**, podem ser identificados sete incrementos para o componente baseado na Web do projeto:

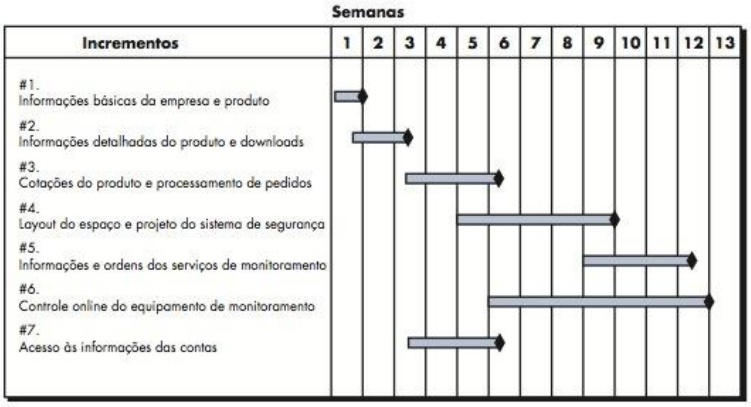
Incremento 1: Informações básicas da empresa e do produto
Incremento 2: Informações detalhadas do produto e downloads
Incremento 3: Cotações do produto e processamento de pedidos do produto
Incremento 4: Layout do espaço e projeto do sistema de segurança
Incremento 5: Informações e ordem dos serviços de monitoramento
Incremento 6: Controle on-line do equipamento de monitoramento
Incremento 7: Acesso às informações das contas

A equipe consulta e negocia com os interessados e desenvolve um cronograma *preliminar* de entrega para os sete incrementos. Um gráfico de Gantt para esse cronograma está na Figura 27.5.

É importante notar que as datas de entrega (representadas por losangos na carta de tempo) são preliminares e podem mudar conforme detalham-se os cronogramas dos incrementos. No entanto, esse cronograma macroscópico fornece ao gerente uma indicação de quando o conteúdo e funcionalidade estarão disponíveis e quando o projeto inteiro estará completo. Como estimativa preliminar, a equipe trabalhará para entregar todos os incrementos em um prazo de 12 semanas. Deve-se notar também que alguns dos incrementos serão desenvolvidos em paralelo (por exemplo, incrementos 3, 4, 6 e 7). Isso implica que a equipe terá pessoas em número suficiente para fazer todo o trabalho em paralelo.

FIGURA 27.5

Gráfico de Gantt para cronograma macroscópico de projeto



Uma vez desenvolvido o cronograma macroscópico, a equipe está pronta para agendar as tarefas para um incremento específico. Para tanto, pode-se usar a estrutura genérica de processo aplicável a todos os incrementos de WebApp. É criada uma *lista de tarefas* usando as tarefas genéricas derivadas como parte da estrutura como ponto inicial e depois adaptando-as considerando conteúdo e funções a ser derivadas para um incremento de WebApp específico.

Cada ação da estrutura (e suas tarefas relacionadas) pode ser adaptada por uma dentre quatro maneiras: (1) uma tarefa é aplicada como ela está, (2) uma tarefa é eliminada porque não é necessária para o incremento, (3) uma nova tarefa (personalizada) é acrescentada, e (4) uma tarefa é refinada (elaborada) em uma série de subtarefas e cada uma delas se torna parte do cronograma.

Para ilustrar, considere uma ação genérica de *modelagem de projeto* para WebApps que pode ser executada aplicando-se uma ou todas as tarefas a seguir:

- Projeto da estética da WebApp.
- Projeto da interface.
- Projeto do esquema de navegação.
- Projeto da arquitetura da WebApp.
- Projeto do conteúdo e da estrutura que a suporta.
- Projeto dos componentes funcionais.
- Projeto dos mecanismos apropriados de segurança e privacidade.
- Revisão do projeto.

Como exemplo, considere a tarefa genérica *Projeto da Interface* da forma como é aplicada ao quarto incremento de **CasaSeguraGarantida.com**. Lembre-se de que o quarto incremento implementa o conteúdo e função para descrever o espaço residencial ou espaço comercial a ser protegido pelo sistema de segurança *CasaSegura*. De acordo com a Figura 27.5, o quarto incremento começa no início da quinta semana e termina no fim da nona semana.

Não há dúvida de que a tarefa *Projeto da Interface* deve ser executada. A equipe reconhece que o projeto da interface é fundamental para o sucesso do incremento e decide refinar (elaborar) a tarefa. As subtarefas a seguir são derivadas para a tarefa *Projeto da Interface* para o quarto incremento:

- Desenvolver um esboço do layout de página para a página de projeto do espaço.
- Rever o layout com os interessados.
- Projeto dos mecanismos de navegação do layout do espaço.
- Projeto do layout "prancheta de desenho".[10]
- Desenvolver detalhes procedurais para a função de layout da parede gráfica.
- Desenvolver detalhes procedurais para cálculo do comprimento da parede e função de display.
- Desenvolver detalhes procedurais para a função de layout da janela gráfica.
- Desenvolver detalhes procedurais para a função de layout da porta gráfica.
- Projetar mecanismos para selecionar componentes do sistema de segurança (sensores, câmeras, microfones etc.).
- Desenvolver detalhes procedurais para o layout gráfico dos componentes do sistema de segurança.
- Conduzir revisões em pares se necessário.

Essas tarefas se tornam parte do cronograma de incremento para o quarto incremento da WebApp e são alocadas no cronograma de desenvolvimento do incremento. Elas podem ser colocadas no software de cronograma e usadas para acompanhamento e controle.

CASASEGURA

Acompanhando o cronograma

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

Atores: Doug Miller (gerente da equipe de engenharia de software *do CasaSegura*) e Vinod Raman, Jamie Lazar e outros membros da equipe.

Conversa:

Doug (observando um slide PowerPoint): O cronograma para o primeiro incremento *do CasaSegura* parece razoável, mas vamos ter problemas para acompanhar o andamento.

Vinod (com cara de preocupado): Por quê? Temos as tarefas dispostas em cronograma diariamente, muitos artefatos, e temos certeza de que não estamos alocando os recursos em excesso.

Doug: Tudo bem, mas como saberemos quando o modelo de requisitos para o primeiro incremento estará completo?

Jamie: As coisas são iterativas, por isso, difíceis.

Doug: Compreendo que, mas... Bem, por exemplo, considere "classes de análise definidas". Você indicou isso como um ponto de controle.

Vinod: Sim.

Doug: Quem fez essa determinação?

Jamie (sério): Elas estão prontas quando estiverem prontas.

Doug: Isso não é suficiente, Jamie. Temos que agendar RTs [revisões técnicas, Capítulo 15], e você não fez isso. Uma revisão bem-sucedida do modelo de análise, por exemplo, é um ponto de controle razoável. Entendeu?

Jamie (contrariado): Ok, voltemos para a prancheta.

Doug: Não vai levar mais de uma hora para fazer as correções... Todos os demais podem começar já.

27.6 ANÁLISE DE VALOR AGREGADO

PONTO-CHAVE

O valor de retorno proporciona uma indicação quantitativa do progresso.

Na Seção 27.5, discutimos uma série de abordagens quantitativas para o acompanhamento de projeto. Cada uma proporciona ao gerente uma indicação do progresso, mas uma avaliação das informações fornecidas é um tanto subjetiva. É razoável questionar se há uma técnica quantitativa para avaliar o progresso conforme a equipe de software avança nas tarefas alocadas para o cronograma do projeto. Na verdade, existe uma técnica para executar análise quantitativa do progresso. Ela é chamada de *análise de valor agregado* (*Earned Value Analysis* – EVA). Humphrey [Hum95] discute o valor de retorno da seguinte maneira:

10 Neste estágio, a equipe imagina criar o espaço literalmente desenhando as paredes, janelas e portas por meio das funções gráficas. As linhas das paredes vão se "encaixar" nos pontos de fixação. As dimensões da parede serão mostradas automaticamente. Janelas e portas serão posicionadas graficamente. O usuário pode também selecionar sensores, câmeras etc. específicos e posicioná-los quando o espaço estiver definido.

> O sistema de valor agregado proporciona uma escala comum de valor para todas as tarefas [de projeto de software], independentemente do tipo de trabalho que está sendo executado. É estimado o total de horas para completar o projeto, e a cada tarefa é dado um valor agregado com base em sua porcentagem estimada do total.

Em outras palavras, o valor agregado é uma medida do progresso. Permite que você avalie a "porcentagem de conclusão" de um projeto usando análise quantitativa em vez de depender de suposições. De fato, Fleming e Koppleman [Fle98] argumentam que a análise do valor agregado "proporciona leituras precisas e confiáveis do desempenho desde os 15% do projeto". Para determinar o valor de retorno, executam-se os seguintes passos:

? Como calcular o valor agregado e utilizá-lo para avaliar o progresso?

1. O *custo orçado do trabalho programado* (*Budgeted Cost of Work Scheduled* – BCWS) é determinado para cada tarefa representada no cronograma. Durante a estimativa, é planejado o trabalho (em pessoas-horas ou pessoas-dias) para cada tarefa de engenharia de software. Desse modo, o $BCWS_i$ é o trabalho planejado para a tarefa i. Para determinar o progresso em um dado ponto ao longo do cronograma de projeto, o valor de BCWS é a soma dos valores de $BCWS_i$ para todas as tarefas que deveriam ter sido completadas naquele ponto no tempo no cronograma do projeto.
2. Os valores de BCWS para todas as tarefas são somados para derivar o *orçamento no final* (*Budget At Completion* – BAC). Assim,

$$BAC = \sum (BCWS_k) \text{ para todas as tarefas } k$$

3. Em seguida, é computado o valor para *custo orçado do trabalho executado* (*Budgeted Cost of Work Performed* – BCWP). O valor de BCWP é a soma dos valores de BCWS para todas as tarefas que foram realmente completadas em certo momento no cronograma de projeto.

Wilkens [Wil99] observa que "a distinção entre BCWS e BCWP é que o primeiro representa o orçamento das atividades planejadas para ser completadas, e o último representa o orçamento das atividades que realmente foram completadas". Ao darmos valores para BCWS, BAC e BCWP, podemos computar importantes indicadores de progresso:

Índice de desempenho do cronograma (*schedule performance index*), $SPI = \frac{BCWP}{BCWS}$

Variância do cronograma (*schedule variance*), $SV = BCWP - BCWS$

WebRef

Uma grande variedade de recursos de análise de valor agregado pode ser encontrada no site **www.acq .osd. mil/pm/**.

O SPI é uma indicação da eficiência com a qual o projeto está utilizando os recursos programados. Um valor SPI próximo de 1,0 indica execução eficiente do cronograma de projeto. O SV é apenas uma indicação absoluta da variância em relação ao cronograma planejado.

$$\text{Porcentagem programada para conclusão} = \frac{BCWS}{BAC}$$

fornece uma indicação da porcentagem do trabalho que deveria ter sido completada no tempo t.

$$\text{Porcentagem completada} = \frac{BCWP}{BAC}$$

fornece uma indicação quantitativa da porcentagem completada de um projeto em dado instante t.

É possível também calcular o *custo real do trabalho executado* (*Actual Cost of Work Performed* – ACWP). O valor de ACWP é a soma do trabalho realmente dispendido em tarefas completadas até determinado instante no cronograma de projeto. É possível então calcular

Índice de desempenho de custo (*cost performance index*), $CPI = \frac{BCWP}{ACWP}$

Variância do custo (*cost variance*), $CV = BCWP - ACWP$

Um valor de CPI próximo de 1,0 fornece uma forte indicação de que o projeto está de acordo com o seu orçamento. CV é uma indicação absoluta de economia de custos (em relação aos custos planejados) ou déficit em determinado estágio de um projeto.

Assim como o radar que vigia o horizonte, a análise de valor agregado esclarece as dificuldades do cronograma antes que possam se tornar aparentes. Isso lhe permite tomar as ações corretivas antes que se desenvolva uma crise no projeto.

27.7 Resumo

O cronograma é o resultado da atividade de planejamento que é um componente primário do gerenciamento de projeto de software. Quando combinado com métodos de estimativa e análise de riscos, o cronograma estabelece um mapa para o gerente de projeto.

O cronograma começa com a decomposição do processo. As características do projeto são empregadas para adaptar um conjunto de tarefas apropriado para o trabalho a ser feito. Uma rede de tarefas mostra cada tarefa de engenharia, sua dependência de outras tarefas e duração projetada. A rede de tarefas é utilizada para calcular o caminho crítico, uma carta de tempo e uma variedade de informações de projeto. Usando o cronograma como guia, você pode acompanhar e controlar cada etapa no processo de software.

Problemas e Pontos a Ponderar

27.1. Prazos de entrega "não razoáveis" são um fato real no negócio de software. Como você procederá se tiver de enfrentar uma situação dessas?

27.2. Qual a diferença entre cronograma macroscópico e cronograma detalhado? É possível gerenciar um projeto se houver apenas o cronograma macroscópico? Por quê?

27.3. Pode haver um caso em que o ponto de controle de um projeto de software não esteja vinculado a uma revisão? Em caso afirmativo, forneça um ou mais exemplos.

27.4. "Sobrecarga de comunicação" pode ocorrer quando múltiplas pessoas trabalham em um projeto de software. O tempo gasto em comunicação reduz a produtividade individual (LOC/mês), e o resultado pode ser menor produtividade para a equipe. Ilustre (quantitativamente) como os engenheiros que são bem versados em boas práticas de engenharia de software e usam revisões técnicas podem aumentar a taxa de produção de uma equipe (quando comparado com a soma das taxas de produção individuais). Dica: Você pode considerar que as revisões reduzem o retrabalho e que o retrabalho pode ser responsável por 20 a 40% do tempo de um profissional.

27.5. Embora acrescentar pessoas a um projeto de software em atraso possa retardá-lo ainda mais, há circunstâncias em que isso não é verdade. Descreva-as.

27.6. A relação entre pessoas e tempo é altamente não linear. Usando a equação do software de Putnam (descrita na Seção 27.2.2), desenvolva uma tabela que relaciona número de pessoas com duração de projeto para um projeto de software que requer 50.000 LOC e 15 pessoas-ano de esforço (o parâmetro produtividade é 5.000 e $B = 0{,}37$). Considere que o software deve ser entregue em mais de 24 meses ou menos de 12 meses.

27.7. Suponha que você foi contratado por uma universidade para desenvolver um sistema de registro de curso on-line (*online course registration system* - OLCRS). Primeiro, aja como o cliente (se você é um estudante, isso é fácil!) e especifique as características de um bom sistema. (Como alternativa, o seu professor lhe fornecerá uma série de requisitos preliminares para o sistema.) Usando os métodos de estimativa discutidos no Capítulo 26, desenvolva uma estimativa de esforço e duração para OLCRS. Sugira como você faria para:

a. Definir atividades paralelas durante o projeto OLCRS.
b. Distribuir o esforço através do projeto.
c. Estabelecer pontos de controle para o projeto.

27.8. Selecione uma série de tarefas apropriadas para o projeto OLCRS.

27.9. Defina uma rede de tarefas para OLCRS descrita no Problema 27.7 ou, como alternativa, para outro projeto de software que lhe interesse. Não deixe de mostrar as tarefas e os pontos de controle e faça estimativas de trabalho e duração para cada tarefa. Se possível, utilize uma ferramenta de cronograma automática para executar esse trabalho.

27.10. Se há uma ferramenta de cronograma automática, determine o caminho crítico para a rede definida no Problema 27.9.

27.11. Usando uma ferramenta de cronograma (se estiver disponível) ou papel e lápis (se necessário), desenvolva uma carta de tempo para o projeto OLCRS.

27.12. Suponha que você seja o gerente de projeto de software e lhe pediram para computar estatísticas de valor agregado para um pequeno projeto de software. O projeto tem 56 tarefas planejadas que, segundo as estimativas, requerem 582 pessoas-dia para se completar. No instante em que lhe pediram para fazer a análise de valor agregado, 12 tarefas já estavam concluídas. No entanto, o cronograma do projeto indica que 15 tarefas já terão sido completadas. Estão disponíveis os seguintes dados de cronograma (em pessoas-dia):

Tarefa	Esforço planejado	Esforço real
1	12,0	12,5
2	15,0	11,0
3	13,0	17,0
4	8,0	9,5
5	9.5	9,0
6	18,0	19,0
7	10,0	10,0
8	4,0	4,5
9	12,0	10,0
10	6,0	6,5
11	5,0	4,0
12	14,0	14,5
13	16,0	—
14	6,0	—
15	8,0	—

Calcule SPI, variância de cronograma, porcentagem programada para completar, porcentagem completa, CPI e variância de custo para o projeto.

Leituras e Fontes de Informação Complementares

Quase todos os livros escritos sobre gerenciamento de projeto de software contêm uma discussão sobre cronograma. Wysoki (*Effective Project Management,* Wiley, 2006), Lewis (*Project Planning Scheduling and Control,* 4th ed., McGraw-Hill, 2006), Luckey e Phillips (*Software Project Management for Dummies,* For Dummies, 2006), Kerzner (*Project Management: A Systems Approach to Planning, Scheduling, and Controlling,* 9th ed., Wiley, 2005), Hughes (*Software Project Management,* McGraw-Hill, 2005), The Project Management Institute (*PMBOK Guide,* 3d ed., PMI, 2004), Lewin (*Better Software Project Management,* Wiley, 2001), e Bennatan (*On Time, Within Budget: Software Project Management Practices and Techniques,* 3d ed., Wiley, 2000) apresentam discussões úteis sobre o assunto. Embora específico de aplicação, Harris (*Planning and Scheduling Using Microsoft Office Project 2007,* Eastwood Harris Pty Ltd., 2007) fornece uma discussão útil sobre como as ferramentas de cronograma podem ser usadas para acompanhar e controlar de forma bem-sucedida um projeto de software.

Fleming e Koppelman (*Earned Value Project Management*, 3d ed., Project Management Institute Publications, 2006), Budd (*A Practical Guide to Earned Value Project Management*, Management Concepts, 2005), e Webb e Wake (*Using Earned Value: A Project Manager's Guide*, Ashgate Publishing, 2003) discutem o uso das técnicas de valor agregado para planejamento de projeto, acompanhamento e controle com considerável detalhe.

Uma ampla variedade de fontes de informação sobre cronograma de projeto de software está disponível na Internet. Uma lista atualizada das referências da Web relevantes para cronogramas de projetos de software pode ser encontrada no site **www.mhhe.com/engcs/compsci/pressman/professional/olc/ser.htm**.