PARTE 2

PLANEJAMENTO E GERÊNCIA DE PROJETOS

Esta parte do livro aborda os seguintes tópicos do SWEBOK: *Gerenciamento de Engenharia de Software*, *Gerenciamento de Configuração de Software* e outros aspectos de *Prática Profissional da Engenharia de Software* que não são abordados na Parte 1.

A área de gerenciamento, neste livro, é subdividida em duas grandes áreas: *planejamento* e *gerenciamento* propriamente dito. Assim, o Capítulo 6 inicia esta parte apresentando os conceitos relacionados ao *planejamento* de um projeto de software.

O planejamento, em software, necessita de *estimativas de esforço* que são muito particulares dessa área. O Capítulo 7, portanto, apresenta técnicas para que um planejador consiga calcular, antes de iniciar um projeto, quanto tempo vai levar para ele ser concluído e quanto vai custar. Dessa forma, minimizará um dos maiores problemas nessa indústria, que é a imprevisibilidade, já que em geral os projetos atrasam e custam mais do que o previsto.

O Capítulo 8 apresenta outro subtema da área de planejamento e gerenciamento, que é o *tratamento dos riscos* de projeto. É sabido que um projeto falha em atingir seus objetivos por conta de riscos que não são devidamente tratados. Assim, esse capítulo vai mostrar como o assunto pode ser abordado de forma organizada para que os riscos sejam mantidos sob controle e os projetos possam ser bem-sucedidos.

O Capítulo 9 vai apresentar os aspectos de condução de um projeto de software, ou seja, de seu *gerenciamento*. Entre outras coisas, esse capítulo vai apresentar técnicas para que um gerente de projeto consiga manter seu projeto nos trilhos e possa se recuperar de desastres, caso eles ocorram, da forma mais organizada possível.

O Capítulo 10 trata de um aspecto da gerência de projeto de software que merece atenção especial: o *gerenciamento de configuração e mudança*. Essa prática deve ser fortemente incentivada nas empresas de software, porque permite aumento de produtividade e segurança no processo de desenvolvimento, reduzindo riscos inerentes importantes do projeto.

Capítulo 6

Planejamento

Este capítulo apresenta os principais conceitos de planejamento de projeto de software, iniciando com algumas reflexões sobre *seleção de projetos* (Seção 6.1), um passo importante a ser tomado antes de se iniciar qualquer ação. Depois será conceituado o *termo de abertura* de um projeto (Seção 6.2). Em seguida (Seção 6.3) é apresentada uma ferramenta para concepção de negócio, o *Business Model Canvas*. A Seção 6.4 apresenta conceitos relacionados à *declaração de escopo* de um projeto. Na sequência, é mostrado como *planejar um projeto* que adota um modelo de processo iterativo (Seção 6.5), ou seja, o planejamento de longo prazo, e também como *planejar uma iteração* (Seção 6.6), ou planejamento detalhado de curto prazo.

Já foi visto que o desenvolvimento de software e as atividades relacionadas estruturam-se a

partir de um modelo de processo, escolhido pelo engenheiro de software para servir à organização. A partir desse modelo de processo, a empresa usualmente vai instanciar um processo próprio de desenvolvimento a ser seguido e constantemente aprimorado pela equipe de desenvolvimento, sob supervisão ou orientação do engenheiro de software.

Cabe agora discutir a prática de um processo para produzir um produto, ou seja, como planejar e executar um *projeto* de desenvolvimento de software.

Neste capítulo será considerado que o modelo de processo utilizado é iterativo, o que inclui o Processo Unificado, a grande maioria dos modelos ágeis e mesmo algumas variações iterativas do Modelo Cascata, como Entrega em Estágios. Assim, dois níveis de planejamento serão abordados:

- *Planejamento de fase ou projeto*: de longa duração e mais genérico.
- *Planejamento de iteração*: de curta duração e mais detalhado.

Boa parte da literatura de planejamento de projetos em geral não considera esses dois níveis de planejamento, que, embora típicos de projetos de desenvolvimento de software, não são ainda tão comuns em outras áreas.

6.1 Seleção de Projetos

Uma empresa de desenvolvimento de software vai executar um projeto que normalmente servirá a outra organização ou grupo de usuários. Em geral, existe mais de uma possibilidade de projeto, e nem sempre todos eles podem ser desenvolvidos. Assim, a empresa desenvolvedora deverá pesar alguns pontos antes de decidir iniciar um projeto:

- A empresa tem competência para desenvolver esse tipo de produto?
- A empresa está dando conta dos projetos atuais, ou seja, tem folga operacional para assumir um novo projeto?
- O cliente é conhecido e confiável?
- O produto dará um bom retorno financeiro?

Essas e outras perguntas normalmente são avaliadas pela gerência superior da empresa antes de ela assumir um compromisso para o desenvolvimento de um projeto.

De outro lado, a empresa cliente não tem recursos ilimitados, e os projetos de desenvolvimento de sistemas poderão competir entre si ou com outros projetos que necessitem de investimento. Assim, o compromisso da empresa cliente com o projeto de desenvolvimento de software possivelmente será afetado pelos seguintes fatores (Xavier, 2011):

- Retorno financeiro em relação ao investimento.
- Grau de incremento da participação da empresa no mercado.
- Melhoria da imagem da empresa.
- Utilização de capacidade ociosa.
- Aquisição de novas tecnologias.

As empresas clientes tenderão a pontuar essas e outras questões antes de se comprometerem com o desenvolvimento de um projeto.

No caso de desenvolvimento de software para o mercado em geral (COTS – *Commercial off the Shelf*), as mesmas questões consideradas pela empresa cliente deverão ser consideradas pela empresa desenvolvedora, já que ela é que vai investir seus recursos para gerar um produto que poderá ter ou não sucesso.

6.2 Termo de Abertura

Havendo comprometimento entre as duas organizações, ou a decisão da empresa desenvolvedora

de COTS de que o projeto será iniciado, isso deve ser oficializado em um termo de abertura (*project charter*).

O PMBOK, ou *Project Management Book of Knowledge* é uma das principais referências hoje em termos de planejamento e gerenciamento de projetos. Segundo esta publicação (PMI, 2017), o termo de abertura deverá conter ou referenciar documentos externos com as seguintes informações:

- *Objetivo e justificativa do projeto*. Por que vamos investir neste projeto?
- *Descrição em alto nível do projeto*. O que vamos fazer?
- *Requisitos de alto nível que satisfazem os principais interessados*. A quem vamos atender e que valor vamos entregar?
- *Nomeação do gerente de projeto e definição do nível de autoridade conferida*. Por exemplo, ele pode usar os recursos sem aprovação superior? Pode contratar pessoal?
- *Cronograma de marcos* (milestones) *resumido*. Quando vamos entregar o que?
- *Definição dos papéis e responsabilidades das partes interessadas*. Quem faz o que?
- *Organização funcional do projeto*. Quem responde a quem?
- *Premissas ou hipóteses*. São perguntas para as quais ainda não se tem resposta, mas que são aceitas, a princípio, para iniciar o projeto. Por exemplo, haverá um especialista disponível na tecnologia X?.
- *Restrições*. Quais limites existem para nossos objetivos? O que não pode ou não deve ser feito?
- *Estudo de viabilidade* (business case) indicando o retorno previsto, seja ele financeiro ou não.
- *Orçamento* previsto em linhas gerais.

O termo de abertura deverá ser aprovado e assinado por um gerente de nível superior ao gerente de projeto, pois isso é o que lhe dará autoridade para iniciar o projeto.

6.3 *Business Model Canvas*

Uma ferramenta que tem se tornado bastante popular para a construção do *business case* é o *business model canvas* (Osterwalder & Pigneur, 2011). Com ele, o projeto é organizado em uma estrutura de quadro com divisões na qual vão ser identificados os segmentos de clientes, a proposta e valor para cada um deles, os meios de comunicação e entrega desta proposta de valor bem como a estrutura de custos e lucro (Figura 6.1).

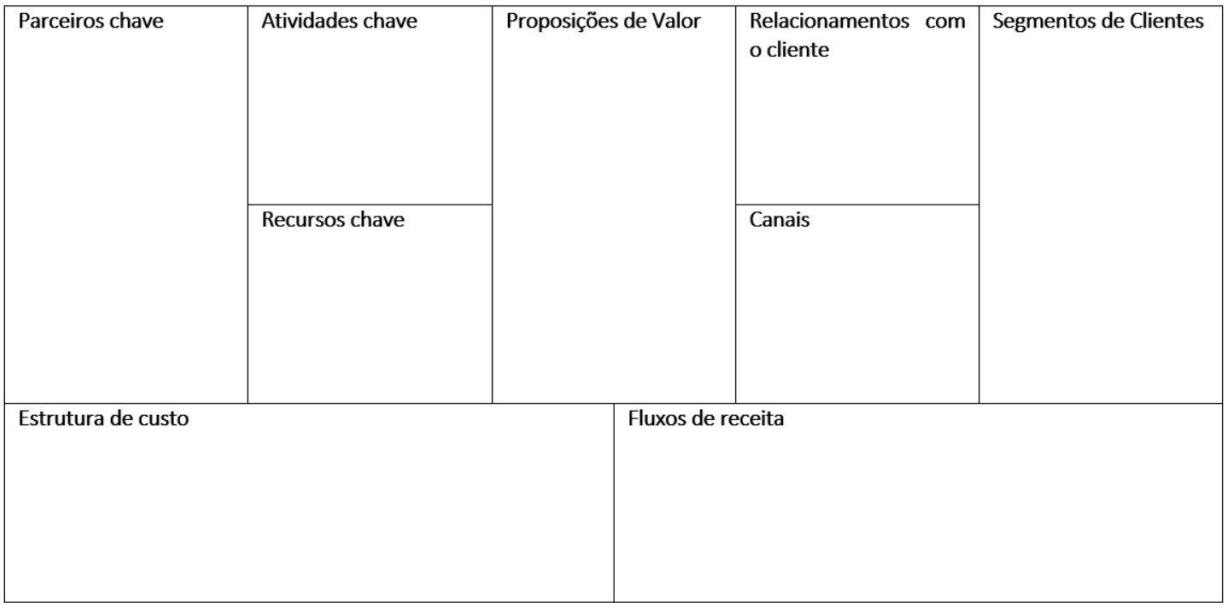

Figura 6.1 *Business Model Canvas*

O *canvas* é basicamente uma ferramenta de pensamento e colaboração. A ideia é que os participantes do projeto procurem preencher suas áreas com *post-its* nos quais palavras ou expressões curtas são usadas.

O lado direito do canvas indica qual o produto ou serviço que está sendo proposto; já o lado esquerdo indica a estrutura necessária para que esta proposição se torne realidade.

O campo *segmentos de clientes* deve conter um ou mais tipos de clientes para os quais o negócio vai gerar valor. Justifica-se identificar diferentes grupos de clientes se:

- Suas necessidades em termos de valor são diferentes.
- São alcançados por canais de distribuição diferentes.
- Exigem tipos diferentes de relacionamento.
- Tem lucratividade substancialmente diferente, como por exemplo, clientes que usam o produto de forma gratuita e clientes que pagam pelo uso.
- Estão dispostos a pagar por aspectos diferentes da proposta de valor.

Outro campo muito importante é a *proposta de valor* em si, ou seja, o que esse projeto entrega de valor para seus segmentos de clientes. O projeto ajuda a resolver algum problema? Atende a alguma necessidade?

Nem todo segmento de cliente estará interessado exclusivamente em propostas de valor financeiras. Há vários tipos de valor que podem ser gerados por um produto ou serviço, como, por exemplo, novidade, desempenho, personalização, fazer o que deve ser feito, design, marca ou status, preço, redução de custo, mitigação de risco, acessibilidade, conveniência ou usabilidade.

Em relação aos canais de atendimento, é necessário observar quais são as formas de atendimento aos clientes. Isso pode implicar na existência ou não de equipes de venda, lojas parceiras, plataformas Web, etc. Em geral as fases de canal identificam cinco momentos distintos:

- *Conhecimento*. Como potenciais clientes chegam a conhecer nosso produto ou serviço?
- *Avaliação*. De que forma ajudamos o potencial cliente a reconhecer nossa proposta de valor?
- *Compra*. De que forma permitimos que nosso cliente compre nosso produto ou serviço?

- *Entrega*. Como nosso produto ou serviço alcançará o cliente?
- *Pós-venda*. Como fornecemos apoio ao nosso cliente após a entrega do produto ou serviço.

No campo de *relacionamento com clientes* a ideia é descrever as formas como vamos nos relacionar com cada segmento de cliente. Estas formas podem variar desde assistência pessoal personalizada, até comunidades de práticas e serviços automatizados.

No campo *fontes de receita* deve-se indicar como cada segmento de cliente vai colaborar para a geração de receita. É possível gerar receita a partir de taxas de uso ou assinatura, venda de recursos, empréstimos, licenciamentos, comissões e até anúncios.

Os *recursos principais* indicam as principais necessidades físicas, humanas, intelectuais e financeiras para a realização do projeto.

As atividades-chave estão relacionadas às principais iniciativas necessárias para que o produto seja construído e/ou que os serviços relacionados sejam disponibilizados.

As *parceiras principais* apontam empresas ou organizações que podem apoiar as atividades de produção, divulgação ou distribuição do produto ou serviço. Elas não são necessariamente clientes ou fornecedores, mas organizações cujos objetivos estejam alinhados ao do negócio que se está modelando. Os parceiros tanto podem beneficiar o novo negócio quanto se beneficiar pelo fato de estarem alinhados a ele.

A *estrutura de custos* indica quais os principais custos fixos e variáveis que a proposta de valor exige para que seja disponibilizada aos segmentos de clientes.

Assim, a ideia é que a equipe que vai elaborar o modelo de negócio para o novo produto ou serviço se reúna preferencialmente em frente a um quadro canvas e utilizando *post-its* discuta e organize as informações nos diferentes campos do canvas.

6.4 Declaração de Escopo

Inicialmente, o planejador de um projeto deve estabelecer quais são seus objetivos. O produto nem sempre é apenas o software funcionando; outros elementos costumam ser necessários e desejáveis.

Sem definir claramente onde o projeto vai chegar, é muito difícil estabelecer um bom plano. Como escolher o caminho, se não se sabe aonde se quer chegar? Infelizmente, muitos planejadores de projetos se esquecem dessa importante etapa. Por exemplo, o projeto termina com a entrega do software ou com a confirmação de sua plena utilização pelo cliente?

O objetivo de um projeto (e também das iterações) deve ser sempre um conjunto de artefatos, ou seja, coisas palpáveis. Um objetivo não pode ser descrito como "executar tal ação", porque isso não define um artefato palpável. "Gerar tal diagrama ou tal relatório" seria muito mais adequado nesse sentido, ou ainda "implementar este e aquele requisitos".

Segundo Xavier (2011), a declaração de escopo do projeto deve conter as seguintes informações:

- *Descrição do produto do projeto*: embora o termo de abertura já contenha uma definição do produto em alto nível, a declaração de escopo deverá refinar essa descrição. É importante mencionar que, normalmente, a declaração não pode relacionar características novas em relação ao termo de abertura. Se for necessária uma alteração de escopo em relação ao inicialmente previsto, isso deverá ser negociado entre as partes.
- *Principais entregas do projeto*: devem ser definidas as principais entregas do projeto, ou seja, os momentos em que o cliente estará recebendo algum tipo de entrega dos desenvolvedores. Normalmente, trata-se de versões implementadas do sistema, mas essa lista poderá incluir outros itens, como projeto, manuais, software de instalação, treinamento etc.

- *Objetivos do projeto*: itens quantificáveis que serão usados para determinar se o projeto foi um sucesso ou não. Os objetivos do projeto devem incluir pelo menos métricas relacionadas a prazo, custo e qualidade do produto. Objetivos não quantificáveis (por exemplo, "cliente satisfeito" ou "sistema fácil de usar") representam um fator de alto risco para a determinação do sucesso do projeto. Os objetivos devem ser claramente avaliáveis a partir de uma métrica definida. Devem ser evitados a todo custo objetivos vagos e de avaliação subjetiva, como "desenvolver tecnologia de última geração".
- *Critérios de aceitação do produto*: é preciso *definir* o processo e os critérios para que o produto, como um todo, seja aceito, e o projeto, finalizado.

Outras informações poderão ser adicionadas à declaração de escopo, se houver necessidade (por exemplo, principais riscos, tecnologias a serem usadas etc.). A declaração de escopo é o documento-base em que deve haver concordância entre o cliente e o gerente de projeto para que, a partir dele, o projeto como um todo possa ser planejado.

É importante mencionar que nesse momento, normalmente, ainda não foi feita uma análise de requisitos, portanto as informações aqui contidas são fruto de entendimentos prévios. Entende-se que a análise de requisitos que virá depois deverá aprofundar o escopo, mas não o aumentar em abrangência. Por exemplo, na análise de requisitos pode-se detalhar como será feito o processo de venda, mas, se não estava prevista a implementação de uma folha de pagamento na declaração de escopo, então, a necessidade de inclusão desse item tornará necessária a renegociação do escopo com o cliente.

6.5 Planejamento de Projeto com Iterações

O objetivo do planejamento de projeto é criar um *plano* para o projeto como um todo. Entre outras coisas, é importante que o responsável por esse planejamento utilize as melhores ferramentas possíveis para avaliar a quantidade de esforço a ser despendido no projeto. Tal estimativa poderá dar origem tanto ao cronograma geral do projeto quanto à estimativa de seu custo total.

Considera-se que a declaração de escopo já definiu os objetivos do projeto e os critérios de aceitação do produto. Assim, as atividades necessárias ao planejamento de um projeto são as seguintes:

- Estimar o *esforço total* para realizar o projeto.
- Em função do esforço total, calcular o *tempo linear* ideal e o *tamanho médio da equipe* para o projeto como um todo ou por fase.
- Estimar a *duração* e o *esforço* empregado nas diferentes fases do projeto.
- Estimar a *duração* e o *número de iterações*.

No Capítulo 7 são apresentadas algumas técnicas para estimar o esforço total necessário para desenvolver um projeto de software, bem como para estimar seu tempo linear e o tamanho médio da equipe. As subseções seguintes apresentam um detalhamento das outras duas atividades mencionadas.

6.5.1 Estimação da Duração e do Esforço nas Diferentes Fases do Projeto

Se o modelo de processo utilizado for iterativo, após estimar o esforço total do projeto, sua duração linear ideal e o tamanho médio da equipe (Capítulo 7), pode-se tentar refinar um pouco mais essa estimativa. O tamanho médio da equipe, por exemplo, não significa que sempre o mesmo número de desenvolvedores estará trabalhando no projeto. Em geral, há mais pessoas trabalhando nas fases de elaboração e construção do UP do que nas fases de concepção e transição.

A Figura 6.2 indica um perfil aproximado de tempo e esforço despendido em cada uma das fases do UP. Evidentemente, esse perfil pode ser alterado de acordo com as características de cada projeto ou das ferramentas de automatização de projeto, geração de código e teste que se utilize.

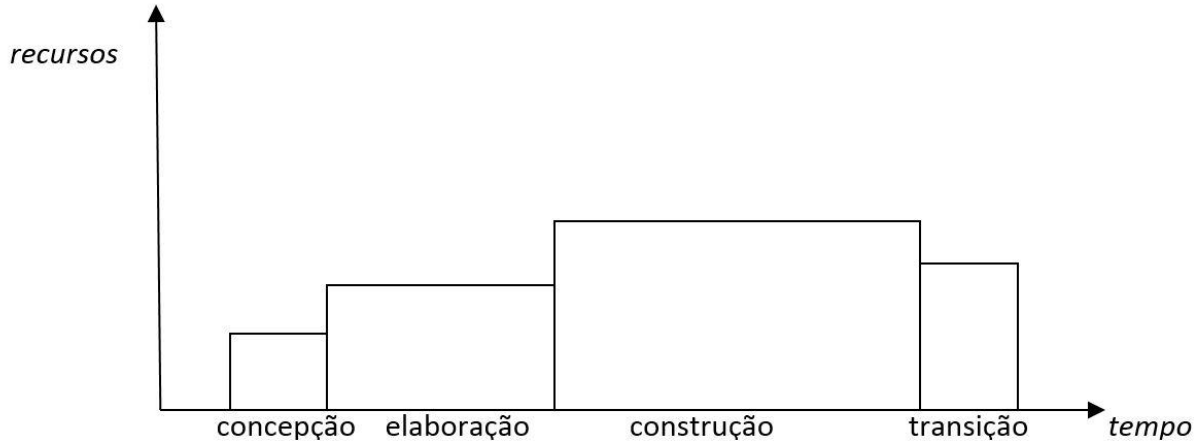

Figura 6.2 Perfil de duração e esforço típicos para um projeto usando UP

Essa figura considera que um projeto típico, de tamanho e esforço moderados, sem arquitetura predefinida e com poucos riscos críticos pode ser desenvolvido aproximadamente com as seguintes estimativas de tempo e esforço:

- *Concepção*: 10% do tempo e 5% do esforço.
- *Elaboração*: 30% do tempo e 20% do esforço.
- *Construção*: 50% do tempo e 65% do esforço.
- *Transição*: 10% do tempo e 10% do esforço.

Existe uma equação simples para encontrar o tempo linear ideal para um projeto baseado no esforço total. Se E é o esforço total em desenvolvedores-mês, então o tempo linear ideal T é dado por $T = 2{,}5 \times \sqrt[3]{E}$. Assim, aplicando-se essa equação, um projeto típico de desenvolvimento cujo esforço foi estimado em 40 desenvolvedores-mês deverá ter uma duração linear ideal de cerca de 8,5 meses.

A duração das fases calculada como a porcentagem de tempo definida acima, aplicada à duração linear do projeto em meses, ficará assim:

- *Concepção*: 10% de 8,5, ou seja, cerca de 0,85 meses.
- *Elaboração*: 30% de 8,5, ou seja, cerca de 2,55 meses.
- *Construção*: 50% de 8,5, ou seja, cerca de 4,25 meses.
- *Transição*: 10% de 8,5, ou seja, cerca de 0,85 meses.

Já o cálculo do tamanho médio da equipe para cada fase deve ser feito da seguinte forma: toma-se o valor do esforço total estimado (40 desenvolvedores-mês) e aplica-se a porcentagem de esforço da fase, conforme definido acima. Depois, divide-se o resultado pela duração linear da fase, conforme obtido acima. No exemplo, fica-se com:

- Concepção: 5% de 40, ou seja, 2 desenvolvedores-mês, o que, dividido por 0,85 meses, dá cerca de 2,35 desenvolvedores em média na fase.
- Elaboração: 20% de 40, ou seja, 8 desenvolvedores-mês, o que, dividido por 2,55 meses, dá cerca de 3,13 desenvolvedores em média na fase.

- Construção: 65% de 40, ou seja, 26 desenvolvedores-mês, o que, dividido por 4,25 meses, dá cerca de 6,11 desenvolvedores em média na fase.
- Transição: 10% de 40, ou seja, 4 desenvolvedores-mês, o que, dividido por 0,85 meses, dá cerca de 4,7 desenvolvedores em média na fase.

A Tabela 6.1 resume estes cálculos e resultados. Na tabela, E é o esforço total, T a duração linear total, $\%T_{fase}$ e $\%E_{fase}$ são respectivamente as porcentagens de tempo e esforço de cada fase e T_{fase}, E_{fase} e P_{fase} são respectivamente o tempo linear, esforço e tamanho médio da equipe por fase.

TABELA 6.1 Cálculo do tempo, esforço e tamanho de equipe para as fases de um projeto

$E = 40$, $T = 8{,}5$	$\%T_{fase}$	$T_{fase} = T * \%T_{fase}$	$\%E_{fase}$	$E_{fase} = E * \%E_{fase}$	$P_{fase} = E_{fase} / T_{fase}$
Concepção	10%	0,85	5%	2	2,35
Elaboração	30%	2,55	20%	8	3,13
Construção	50%	4,25	65%	26	6,11
Transição	10%	0,85	10%	4	4,7

Os valores em meses podem ser convertidos em semanas, bastando multiplicá-los por 4 (ou por 3,9, segundo alguns autores). Assim, a concepção teria cerca de 3,5 semanas, a elaboração, 10,2 semanas, a construção, dezessete semanas e a transição, 3,5 semanas.

Um valor fracionado de desenvolvedor como 2,35 indica uma média, ou seja, espera-se que uma parte da fase necessite de dois desenvolvedores e uma parte menor de três desenvolvedores. Isso também pode significar que, se apenas dois desenvolvedores estiverem disponíveis para a fase, possivelmente será necessário esticar mais o tempo, pois eles não darão conta do trabalho, ou ainda que, se três desenvolvedores estiverem disponíveis, talvez seja possível diminuir um pouco o tempo, pois haverá certa folga.

Além disso, algumas observações podem alterar esse perfil típico (Kruchten, 2003):

- Se for necessário mais tempo para estabelecer o projeto, achar financiadores, fazer pesquisa de mercado ou construir provas de conceito, a fase de concepção deve ser prolongada.
- Se houver altos riscos técnicos ou de pessoal, ou se houver restrições de desempenho importantes e nenhuma arquitetura prévia definida, então a fase de elaboração deve ser prolongada, porque serão necessários mais ciclos de elaboração para definir a arquitetura e/ou mitigar os riscos conhecidos.
- Se essa não for a primeira geração do produto (pode ser um ciclo de evolução) e se não forem feitas maiores alterações na arquitetura, as fases de concepção e elaboração poderão ser encolhidas.
- Se o objetivo for atingir o mercado rapidamente por causa de concorrentes ou porque se está criando esse mercado, a fase de construção pode ser encolhida, e a fase de transição, aumentada. Assim, versões executáveis serão liberadas mais cedo e gradativamente no mercado.
- Se houver necessidade de uma transição complicada, como substituir um sistema em funcionamento sem interromper os serviços, ou no caso de domínios que exigem certificações ou regulamentos a serem avaliados (medicina, aeronáutica etc.), a fase de transição deve ser aumentada.

Assim, essas e outras questões devem ser avaliadas pelo planejador de projetos, que, a partir da previsão de esforço nominal, vai prever esforços específicos para as diferentes fases em seu projeto específico.

Entretanto, a principal fonte de informação para esse tipo de previsão deve ser sempre o

histórico de medições da empresa desenvolvedora, pois, como cada empresa tem seu próprio estilo de trabalho, ferramentas e competências, diferentes valores de esforço nas diferentes fases poderão ser obtidos. Por exemplo, empresas que usam intensivamente modelos baseados em ferramentas e geração automática de código em geral terão uma fase de construção relativamente bem menor do que a fase de elaboração.

6.5.2 Estimação da Duração das Iterações

Uma iteração se inicia com planejamento e termina com uma nova versão do sistema disponibilizada internamente ou até mesmo uma *release* ao cliente. A duração estimada de uma iteração no Processo Unificado ou métodos ágeis costuma variar de uma a oito semanas e depende basicamente da complexidade do projeto e da equipe.

Equipes pequenas com até cinco pessoas poderão fazer o planejamento juntas numa manhã de segunda-feira, executar o trabalho ao longo da semana e gerar uma *release* na sexta-feira. Equipes um pouco maiores, entre seis e vinte pessoas poderão precisar de mais tempo, com iterações de pelo menos duas semanas

Equipes com mais de vinte pessoas precisarão de mais tempo para distribuir e sincronizar as atividades, até porque a carga de trabalho será naturalmente bem maior. Além disso, a geração da *release* tomará mais tempo, pois haverá um volume maior de partes a serem integradas e testadas. Assim, nesse caso, uma iteração de três a quatro semanas seria mais recomendável.

Equipes com mais de quarenta pessoas precisarão trabalhar em um ambiente muito mais formal e com mais documentação intermediária, de forma que o fluxo de informação será naturalmente mais lento. Dessa forma, um ciclo de cinco a oito semanas seria recomendável nesse caso.

Outros fatores que podem afetar a duração de uma iteração são os seguintes:

- Quanto mais automatização no processo de geração de código e no ambiente de desenvolvimento em geral, mais curtas poderão ser as iterações.
- Quanto mais familiaridade a equipe tiver com o modelo de desenvolvimento e com as técnicas de análise e *design*, mais curtas poderão ser as iterações.
- Quanto mais crítico for o fator "qualidade" no desenvolvimento e quanto mais críticas forem as revisões e testes que precisarem ser feitos, mais longas deverão ser as iterações.

Via de regra, porém, as iterações devem ser as mais curtas possíveis. Assim, se uma equipe puder fazer iterações de uma ou duas semanas em função de seu tamanho e características, ela não deve optar por iterações mais longas, sob pena de perder a agilidade e *feedback* que são possíveis com iterações mais curtas.

6.5.3 Número de Iterações

O número de iterações de um projeto dependerá do tempo linear a ser despendido, especialmente nas fases de elaboração e construção, dividido pelo tamanho previsto das iterações. Por exemplo, um projeto com iterações de duas semanas, cujas fases de elaboração e construção devem durar seis meses no total (24 semanas), terá doze ciclos de elaboração e construção.

A quantidade proporcional de ciclos de elaboração e de construção dependerá da necessidade de tratar assuntos ligados à estabilização da arquitetura. Assim, uma boa indicação de que muito trabalho na arquitetura será necessário é a proporção de casos de uso complexos em relação ao número total de casos de uso. Um sistema com poucos casos de uso complexos e muitos CRUDs e relatórios, possivelmente terá proporcionalmente menos atividades de arquitetura do que um sistema com muitos casos de uso complexos e poucos relatórios e CRUDs.

A Seção 6.5.1 indica que a fase de elaboração ocupará aproximadamente 30% do tempo linear

total e a fase de construção 50% do tempo linear total para projetos de complexidade média sem características especiais. Assim, para cada três iterações de elaboração haverá cinco iterações de construção.

Em geral, a fase de concepção não é organizada em iterações, a não ser que mais de um protótipo seja necessário ou que um número significativo de riscos muito importantes deva ser tratado antes de se iniciar a fase de elaboração. Deve-se lembrar, porém, que o objetivo da disciplina de implementação na fase de concepção não é produzir código funcionando, mas gerar rapidamente protótipos (se necessário) que ajudem a compreender melhor os verdadeiros requisitos do sistema.

A fase de transição também não costuma ser organizada em mais de uma ou duas iterações. Apenas transições muito complexas deverão ser organizadas em mais de uma iteração, com diferentes objetivos definidos para cada uma delas.

6.5.4 Definição dos Marcos ou Entregas

Uma vez definido o tamanho das iterações, o tamanho da equipe em cada fase e a duração de cada fase (em número de iterações), o planejador deverá retomar a declaração de escopo para definir os marcos de projeto e as datas de entregas. O Processo Unificado já estabelece marcos-padrão ao final de cada fase, mas convém que no plano de projeto esses marcos, bem como outros momentos importantes do projeto, sejam claramente identificados. Será considerado novamente o projeto do exemplo das seções anteriores, resumido na Tabela 6.2.

TABELA 6.2 Esforço e duração de um projeto típico por fase do UP

Fase	Duração (semanas)	Duração arredondada	Número médio de desenvolvedores	Número de desenvolvedores arredondado
Concepção	3,5	3	2,35	3
Elaboração	10,2	10	3,13	3
Construção	17	18	6,11	6
Transição	3,5	3	4,7	5
Total	34,2	34		

Note que, na Tabela 6.2, os arredondamentos procuraram fazer que as fases de elaboração e construção ficassem com um número par de semanas, em razão do fato de as iterações terem sido definidas com duas semanas. O arredondamento do número de desenvolvedores foi feito para cima nas fases de concepção e transição, porque o arredondamento da duração dessas fases foi feito para baixo. De outro lado, o arredondamento do número de desenvolvedores da fase de construção foi feito para baixo porque sua duração foi aumentada em uma semana. Apenas a fase de elaboração teve o número de desenvolvedores e o de duração arredondados para baixo, indicando que poderá haver algum aperto nessa fase ou que o planejador do projeto estima que a elaboração será um pouco mais simples do que em um projeto típico.

Um plano simplificado possível para esse projeto seria parecido com o da Tabela 6.3. Note que foram definidas iterações de duas semanas, exceto para as fases de concepção e transição. Não foram definidos ainda os objetivos específicos de cada uma das fases não-finais de elaboração e construção porque se admite que o planejamento possa ser dinâmico, ou seja, no inicio de cada iteração a equipe vai verificar quais os casos de uso, riscos ou solicitações de mudança de maior prioridade e estabelecê-los como objetivos da iteração. Porém, se alguma entrega intermediária se fizer necessária, como por exemplo, ter um subsistema pronto antes do final da construção ou

algum aspecto da arquitetura resolvido antes do final de elaboração, então objetivos prefixados ou marcos, podem ser explicitamente estabelecidos.

TABELA 6.3 Exemplo de plano de projeto simplificado com definição de entregas

Fase	**Prazo (semana)**	**Desenvolvedores**	**Entregas**
Concepção	3	3	Modelo de casos de uso preliminar para revisão.
Elaboração	5	3	Resolução de riscos e questões arquiteturais. (20%)
	7	3	Resolução de riscos e questões arquiteturais. (40%)
	9	3	Resolução de riscos e questões arquiteturais. (60%)
	11	3	Resolução de riscos e questões arquiteturais. (80%)
	13	3	Arquitetura estabilizada. (100%)
Construção	15	6	Incorporação de código final. (20%)
	17	6	Incorporação de código final. (30%)
	19	6	Incorporação de código final. (40%)
	21	6	Incorporação de código final. (50%)
	23	6	Incorporação de código final. (60%)
	25	6	Incorporação de código final. (70%)
	27	6	Incorporação de código final. (80%)
	29	6	Incorporação de código final. (90%)
	31	6	Todo código finalizado. (100%)
Transição	34	5	Sistema instalado. Migração de dados concluída.

Considera-se que a primeira iteração da fase de construção já irá herdar algum código construído durante a fase de elaboração. Assim, esse ciclo já prevê a conclusão de 20% do código ao seu final, enquanto que os demais ciclos de construção adicionam somente 10% cada um a esse código.

Essas porcentagens são apenas estimativas aproximadas. Em um projeto real, elas poderiam ser substituídas por objetivos reais relacionados a casos de uso e riscos específicos a serem desenvolvidos e mitigados respectivamente.

6.6 Planejamento de Iteração

Concluído o planejamento do projeto, se este for feito com iterações, apenas a primeira iteração será planejada detalhadamente de início. Apenas quando essa iteração estiver em andamento deve-se iniciar o planejamento da segunda iteração, e assim por diante. Esta seção tratará do planejamento detalhado, ou seja, do planejamento das iterações.

Caso se esteja trabalhando com um método ágil, os objetivos da iteração serão definidos pelas histórias de usuário ou requisitos a serem implementados, conforme explicado no Capítulo 4. Já no caso do UP, os objetivos de uma iteração poderão ser de três tipos:

- Implementar total ou parcialmente um ou mais *casos de uso* de maior prioridade.

- Mitigar um *risco* conhecido de alta exposição (ou seja, um risco com alta probabilidade de ocorrer e alto impacto), gerando ou executando um plano de redução de probabilidade, redução de impacto ou ainda de recuperação de desastre caso o risco já tenha se tornado um problema concreto.
- Implementar total ou parcialmente uma ou mais *modificações* solicitadas. À medida que a arquitetura do sistema evoluir nas iterações, modificações poderão ser solicitadas em função da não adequação aos requisitos ou, ainda, à sua mudança. Incorporar essas solicitações de mudança ao software pode ser um dos objetivos de uma iteração.

Para cada caso de uso, risco ou modificação deve haver uma estimativa total de esforço de desenvolvimento. Os elementos serão selecionados considerando-se em primeiro lugar sua prioridade. A maior prioridade deve ser dada aos elementos mais complexos, de maior risco ou com os quais mais se possa aprender em relação à arquitetura do sistema. A sugestão é escolher em primeiro lugar:

- Casos de uso que representem os processos de negócio mais críticos para a organização, por exemplo, aqueles através dos quais a organização realiza seus objetivos, como obtenção de lucros ou atendimento aos seus clientes ou associados.
- Riscos de alta exposição, ou seja, com alto impacto e alta probabilidade de ocorrer.
- Modificações urgentes, como refatorações da arquitetura.

Considerados os elementos de maior prioridade, outros elementos de prioridade não tão alta, mas com certa afinidade, poderão ser colocados na mesma iteração por conveniência. O importante é que o esforço total estimado não ultrapasse a quantidade de desenvolvedores-mês que se pode alocar dentro da duração prevista da iteração.

Selecionados os elementos a serem tratados na iteração, deve-se estabelecer claramente qual o objetivo da iteração, ou seja, até que ponto os elementos selecionados deverão ser desenvolvidos. Um caso de uso, por exemplo, poderá ser desenvolvido até que todos os detalhes de seus fluxos sejam conhecidos apenas, ou até que o código final esteja totalmente desenvolvido. Um risco poderá ter uma de suas componentes (probabilidade ou impacto) mitigada, ou ambas até chegar a um grau de exposição média ou baixa. Uma solicitação de modificação poderá ser totalmente ou parcialmente atendida. Enfim, é importante que os objetivos da iteração sejam detalhados a ponto de poderem ser identificados os artefatos que serão produzidos ao final da iteração.

Na sequência, deve-se estabelecer a *WBS da iteração* (Seção 6.6.1), ou seja, o conjunto de atividades que devem ser executadas para obter os artefatos que constituem o objetivo da iteração. Se o método de desenvolvimento for ágil, esse inventário de atividades será feito pela própria equipe de maneira mais informal.

Contudo, no caso de processos prescritivos, o inventário deve ser obtido a partir da instanciação dos *workflows* das disciplinas necessárias para a iteração. Cada atividade prevista no *workflow* deverá ser atribuída a uma pessoa com capacidade de exercer o papel previsto. As atividades deverão ter sua duração estimada e, em função de suas dependências, um diagrama PERT e/ou Gantt deverá ser construído (Seções 6.6.4 e 6.6.5).

6.6.1 WBS – Work Breakdown Structure

A EAP, Estrutura Analítica do Projeto, ou WBS, *Work Breakdown Structure*, em inglês (Tausworthe, 1980) apresenta as atividades que devem ser executadas para se atingir os objetivos determinados para o projeto (quando se vai planejar o projeto como um todo) ou iteração (no caso do planejamento por iterações).

Além da lista de atividades, é importante utilizar um método de estimação de esforço para prever a duração de cada atividade. Equipes ágeis farão a estimação possivelmente usando pontos de história enquanto que equipes baseadas em outros modelos poderão usar, por exemplo, um conjunto de estimativas individualizadas por fase e ou por disciplina, de acordo com o método utilizado.

Se for usado um ciclo de vida prescritivo, os *workflows* podem indicar quais são as atividades a serem executadas e quais as dependências entre elas. Dependendo do processo adotado, o *workflow* poderá até indicar formas de estimativa de esforço para cada atividade individual.

Se for usado um método ágil, recomenda-se que a equipe decida quais atividades serão desenvolvidas. Isso não impede que equipes usando métodos ágeis se baseiem em *workflows* existentes, se o grupo entender que isso poderá ser útil ao projeto.

Seja qual for o modelo de processo adotado, a WBS pode ser definida em uma reunião de planejamento com toda a equipe para que as várias visões do projeto sejam pesadas no momento de se estabelecerem atividades e estimar esforço. No caso de Scrum, essa reunião seria a *sprint planning meeting*. O ideal, porém, é que uma WBS não seja iniciada do zero. Possivelmente existem projetos anteriores semelhantes ou *template*s do próprio processo para que a WBS seja elaborada a partir de experiências passadas.

É importante que cada atividade caracterize muito bem o produto de trabalho ou artefato de saída a ser entregue ao final. Se um processo for usado, ele próprio vai estabelecer esses artefatos.

A WBS é uma estrutura exaustiva, ou seja, ela deve incluir todas as atividades necessárias para a execução do projeto ou iteração. A WBS poderá ser estruturada como uma árvore, isto é, as atividades podem ser aglutinadas ou detalhadas estabelecendo-se uma árvore de decomposição entre elas. As atividades nas folhas dessa árvore são as atividades *terminais* e devem seguir a regra 8-80 especificada na Subseção 6.6.1.1.

É muito importante que o planejador do projeto determine artefatos de saída, e não meramente ações. Atividades devem necessariamente produzir algo palpável, e não apenas a execução de ações do responsável ou dos participantes. Tentar modelar um projeto baseado em ações pode levar ao detalhamento excessivo das atividades e, consequentemente, à impossibilidade de gerenciar de modo adequado o trabalho. Então, a regra de ouro do planejamento é: *cada atividade deve gerar pelo menos um novo artefato palpável ou uma alteração consistente e objetivamente verificável em algum artefato existente*.

Não devem ser usados nomes vagos que deixem dúvida sobre o subproduto a ser gerado pela atividade. Devem-se usar substantivos para definir o subproduto, e não verbos. Por exemplo, deve-se usar "relatório de teste do módulo" em vez de "testar o módulo".

Estilos de WBS que preveem diferentes estágios de um artefato (por exemplo, *versão inicial*, *versão intermediária* e *versão final*) devem caracterizar exatamente o que esperam de cada uma dessas versões. Por exemplo, a versão inicial de um documento de requisitos poderia ter apenas uma lista de funções identificadas. Uma versão intermediária desse documento poderia exigir que as funções fossem agrupadas por similaridade e que requisitos não funcionais tivessem sido adicionados. Uma versão final do mesmo documento poderia exigir que ele estivesse organizado e revisado dentro de determinado padrão e que os requisitos tivessem sido verificados em relação a sua completeza e consistência por algum processo padrão definido.

6.6.1.1 Regra 8-80

Tarefas que, de acordo com as estimativas, levarão muito tempo para ser completadas devem ser subdivididas em tarefas mais curtas. Tarefas estimadas para levar pouco tempo devem ser

aglutinadas com outras. A regra 8-80 estabelece que nenhuma atividade terminal deve durar nem mais de oitenta horas (duas semanas ou dez dias de trabalho ideais), nem menos de oito horas (um dia de trabalho ideal).

Não se deve ter tarefas com duração muito longa, porque fica muito difícil gerenciá-las e acompanhar seu andamento. Também não se deve ter tarefas muito curtas, porque micro gerenciá-las pode provocar um *overhead* de gerenciamento que, em vez de ajudar, vai atrapalhar o projeto.

Métodos como XP são ainda mais restritivos em relação ao tamanho das tarefas, pois exigem que sua duração seja de um a três dias ideais de trabalho, ou seja, de oito a 24 horas. Mas, embora isso seja mais restritivo, não contradiz a regra 8-80.

A WBS deve ser organizada, precisa e pequena o suficiente para que possa servir de base para a gerência do projeto durante a iteração sem ser um estorvo.

6.6.1.2 Regra dos Níveis e do Número Total de Atividades

Além de respeitar a regra 8-80, a estruturação de uma boa WBS não deve ter mais de três ou quatro níveis de decomposição de atividades. Os elementos terminais, ou seja, os elementos não decompostos (no nível mais baixo) são também chamados de *pacotes de trabalho.*

O número total de pacotes de trabalho em uma WBS não deve ultrapassar o limite de duzentos elementos, embora cem já seja considerado um número muito alto.

Considerando-se que cada atividade terminal poderá ter no máximo oitenta horas, essa regra estabelece que, no pior dos casos, uma iteração ou projeto gerenciável deverá ter $80 \times 200 = 16.000$ horas de trabalho (mas o típico são iterações bem abaixo desse limite). Qualquer projeto ou iteração com carga horária maior do que essa deve necessariamente ser subdividido em projetos ou iterações menores. Nesse sentido, as iterações de duas semanas dos métodos ágeis e do UP garantem que o número de horas total nunca seja muito grande. Com apenas oitenta horas de atividade por desenvolvedor em duas semanas, seriam necessários duzentos desenvolvedores trabalhando numa iteração para atingir tal limite. Acima disso (por exemplo, sessenta pessoas trabalhando em ciclos de oito semanas, ou seja, uma carga de 19.200 horas por ciclo) seria altamente recomendável a subdivisão do projeto e/ou da equipe com a aplicação, por exemplo, de modelos de grande escala como *Huge LeSS*, *Crystal Diamond* ou RUP-SE.

6.6.1.3 Regra dos 100%

A regra dos 100% estabelece que uma WBS deve incluir 100% de todo o trabalho que deve ser feito na iteração. Nenhum artefato será produzido se não estiver definido como saída de alguma das atividades da WBS e nenhuma atividade deixará de produzir algum artefato de saída.

A regra dos 100% vale em todos os níveis da hierarquia de decomposição da WBS. Além disso, quando uma atividade se decompõe em subatividades, o trabalho definido pela atividade será exatamente igual a 100% do trabalho definido nas subatividades (sempre em termos de artefatos de saída).

O subproduto que não estiver na WBS não será desenvolvido. Então, nenhum artefato pode ficar de fora da WBS quando for o momento de construí-lo ou revisá-lo. Se em algum momento um desenvolvedor estiver trabalhando em algo que não contribui para nenhum subproduto da WBS, ele estará trabalhando fora do escopo e será preciso decidir se o subproduto deverá ser incluído na WBS ou a atividade do desenvolvedor deverá ser revista.

6.6.2 Identificação dos Responsáveis por Atividade

Um *workflow* costuma definir que o responsável por uma atividade é um papel, ou seja, um tipo

de pessoa com uma ou mais habilidades desejáveis. Quando uma iteração for planejada a partir desse *workflow*, deve-se atribuir as tarefas a pessoas reais que atendam ao perfil desejado.

Cada tarefa da WBS deverá ser atribuída a um responsável e possivelmente outros participantes podem ser indicados. Essas atribuições poderão ter efeito sobre o cronograma de projeto, pois, embora certas tarefas possam ser executadas em paralelo, não é possível fazê-lo assim caso estejam atribuídas ao mesmo responsável.

6.6.3 Identificação dos Recursos Necessários e Custo

É possível que a maioria das tarefas a serem executadas, além de recursos humanos (responsáveis e participantes), também tenha recursos físicos consumíveis ou não consumíveis a serem alocados.

No momento do planejamento da iteração é necessário prever e alocar o uso desses recursos. O custo de uma tarefa individual será, portanto, o custo das pessoas que se dedicam a ela somado ao custo dos recursos alocados.

6.6.4 Identificação das Dependências entre Tarefas

As dependências entre tarefas são dadas em função do *workflow* ou identificadas caso a caso pela equipe de planejamento. Em geral, essas dependências existem porque as entradas de uma tarefa são as saídas de outra. Não havendo essa condição, as tarefas podem ser executadas potencialmente em paralelo.

A partir da estruturação das tarefas, o planejador do projeto deverá estimar os tempos necessários para a execução de cada uma. Costuma ser difícil estimar tempo com grande precisão. Trabalha-se, então, com o *timeboxing* da iteração. O esforço total é o número de dias multiplicado pelo número de desenvolvedores. Devem-se determinar as sequências de tarefas mais difíceis primeiramente e alocar desenvolvedores a elas.

É necessário também verificar se as dependências entre as tarefas criam um caminho crítico (Seção 6.6.4.2) cujo comprimento seja maior que a duração da iteração. Nesse caso, talvez seja necessário replanejar as tarefas de forma que o caminho crítico e quaisquer outros caminhos caibam no *timeboxing* da iteração. Depois, distribui-se o tempo restante para as outras tarefas. Eventuais erros para mais ou para menos nas estimativas podem compensar-se mutuamente.

6.6.4.1 Rede PERT

O grafo de dependências entre atividades com a duração prevista para cada tarefa constitui-se na *rede PERT* do projeto ou iteração.

Há várias ferramentas que permitem a elaboração quase automática de uma rede PERT, como Redmine (ver primeiro *QR code*) [QRC 6.1] e OpenProject (ver segundo *QR code*) [QRC 6.2]. Com o uso destas ferramentas, em geral, basta que se definam o conjunto das atividades, suas dependências e sua duração, e a ferramenta calcula as datas de início e de finalização prováveis de cada atividade, conforme mostrado na Tabela 6.4.

TABELA 6.4 Exemplo de conjunto de atividades com suas durações e dependências

#	Tarefa	Duração (dias)	Predecessoras	Início	Término
1	Desenvolver visão geral do sistema	4		08/10/18	11/10/18

2	Eliciar necessidades dos interessados	5	1	15/10/18	19/10/18
3	Gerenciar dependências	2	1	15/10/18	16/10/18
4	Capturar vocabulário comum	1	3	17/10/18	17/10/18
5	Encontrar atores e casos de uso	1	2; 4	22/10/18	22/10/18
6	Estruturar o modelo de casos de uso	2	5	23/10/18	24/10/18
7	Priorizar os casos de uso	1	5	23/10/18	23/10/18
8	Detalhar os casos de uso	3	7	24/10/18	26/10/18
9	Modelar interface com usuário	3	8	29/10/18	31/10/18
10	Prototipar interface com usuário	6	9	01/11/18	09/11/18
11	Revisar requisitos	2	6; 10	12/11/18	13/11/18

Nessa tabela, o planejador preencheu as colunas “Nome”, “Duração” e “Predecessoras” e a data de início da primeira atividade. Assim, a ferramenta será capaz de automaticamente preencher as colunas “Início” e “Término” para todas as demais atividades.

Um exemplo de rede PERT gerado a partir da WBS da Tabela 6.4 é apresentado na Figura 6.3. Essa rede considera apenas dias úteis de trabalho, ignorando sábados, domingos e feriados.

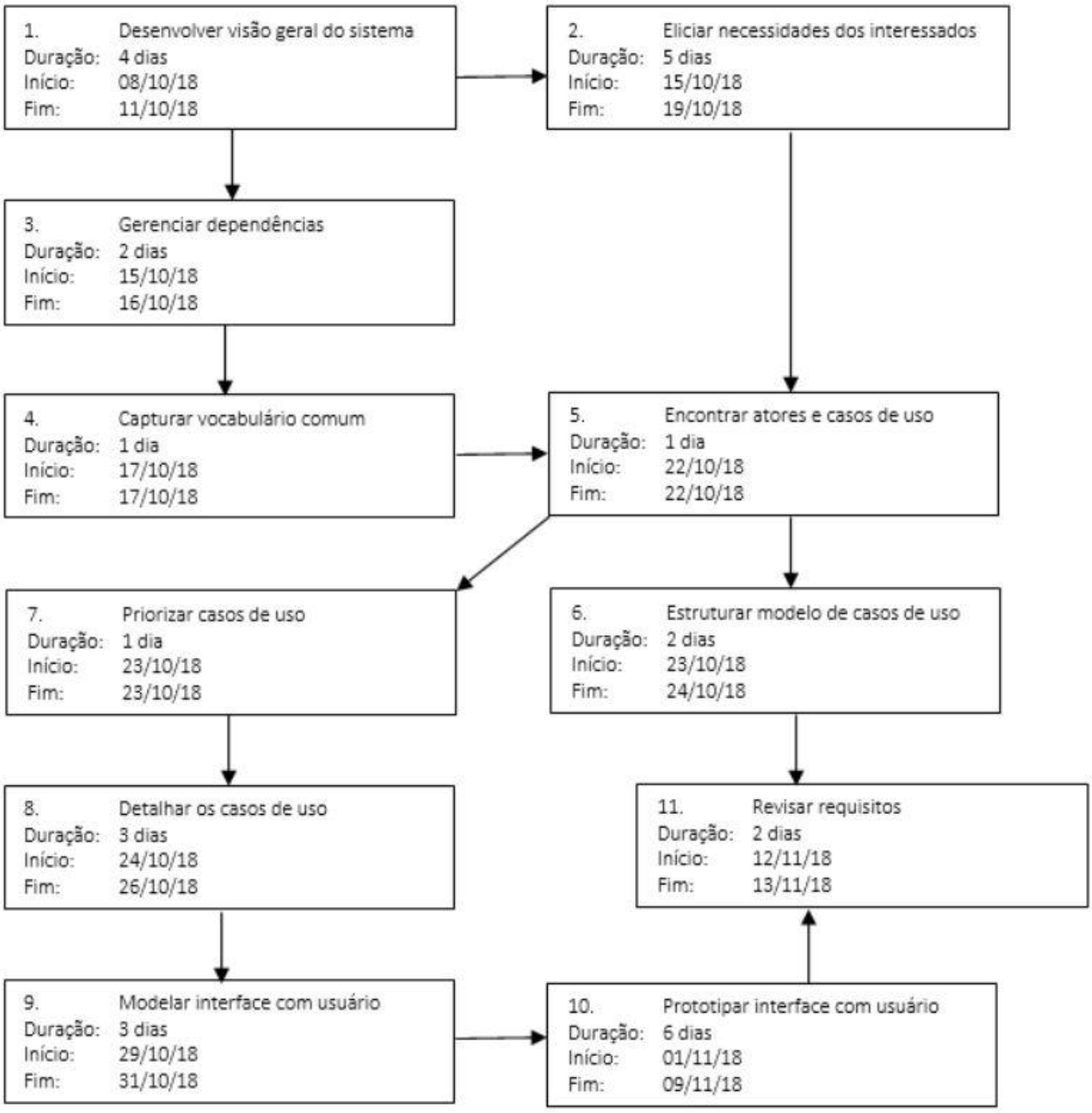

Figura 6.3 Uma rede PERT para as atividades da WBS

6.6.4.2 Caminho Crítico

Um conceito importante no diagrama PERT é o *caminho crítico*, que consiste no mais longo caminho que leva do início ao fim do projeto ou iteração. Esse caminho crítico é importante porque, se qualquer atividade prevista nele atrasar por algum motivo, todo o projeto vai atrasar. Esse é um caminho sem folga.

Entretanto, as atividades que não pertencem ao caminho crítico podem ser adiadas até certo limite sem prejuízo ao projeto como um todo. Na Figura 6.3, as atividades do caminho crítico são as atividades numeradas como: 1, 2, 5, 7, 8, 9, 10 e 11. As atividades que não estão no caminho crítico podem atrasar sem prejudicar o projeto como um todo até um certo limite. Por exemplo, a atividade 6 tem término previsto para o dia 24/10, mas devido ao caminho crítico não passar por

ela, seus resultados só serão necessários em 12/11, quando a atividade 11 iniciar. Assim, ela pode ser postergada desde que termine antes do dia 12/11.

O caminho crítico pode não ser simplesmente uma única via, mas um caminho composto, ou seja, atividades paralelas podem estar no caminho crítico.

Quando uma atividade do caminho crítico atrasa, pode ser necessário acelerar alguma atividade posterior no caminho crítico para manter a iteração dentro do cronograma. A forma de obter essa aceleração será definida a critério do gerente de projeto ou da equipe ágil. Existem três opções usuais:

- *Aumentar a jornada da equipe*, o que não pode se transformar em rotina.
- *Aumentar o tamanho da equipe*, o que pode causar transtornos de gerência em função da colocação de pessoas novas no projeto, possivelmente com menos experiência. Essa abordagem nem sempre dá o resultado esperado.
- *Eliminar alguns objetivos* (*artefatos*) *ou características de artefatos da iteração*. Por exemplo, em vez de implementar três casos de uso, caso haja atrasos, implementam-se apenas dois, deixando para outra iteração a implementação do terceiro.

O aumento da jornada ou intensificação do foco pode ajudar a recolocar nos trilhos um projeto ou iteração atrasados, mas, se isso ocorrer com muita frequência, o moral da equipe vai baixar e, possivelmente, atrasos serão cada vez mais frequentes.

Já o aumento do tamanho da equipe costuma produzir apenas resultados positivos em médio prazo, ou seja, duas ou três iterações depois daquela em que um ou mais novos membros foram adicionados. Por isso, inicialmente, essa solução pode atrasar ainda mais o projeto.

A eliminação de artefatos ou características de artefatos, que são remanejados para a lista de mudanças solicitadas a fim de serem resolvidos oportunamente em uma iteração futura, costuma ser a recomendação mais acertada nesses casos. Assim, a equipe se concentra em terminar algumas funcionalidades, obtém uma vitória relativa de curto prazo e consegue se reorganizar para retomar as funcionalidades faltantes em um momento de maior folga, de forma organizada.

6.6.5 CRONOGRAMA

Em geral, o cronograma do projeto é mostrado em um diagrama Gantt, que consiste em uma visualização do tempo linear transcorrido e da ocorrência das diferentes atividades ao longo desse tempo. Em relação ao diagrama PERT, o diagrama Gantt apresenta o andamento das atividades ao longo de uma linha de tempo, permitindo visualizar claramente as atividades que devem ser executadas a cada dia.

A Figura 6.4 mostra um diagrama Gantt para a rede PERT da Figura 6.3. Para que as atividades em paralelo como 2 e 3-4 possam, de fato, ser executadas ao mesmo tempo, elas devem ser alocadas a responsáveis diferentes. As atividades que pertencem ao caminho crítico estão em cor mais escura e as demais em cor mais clara.

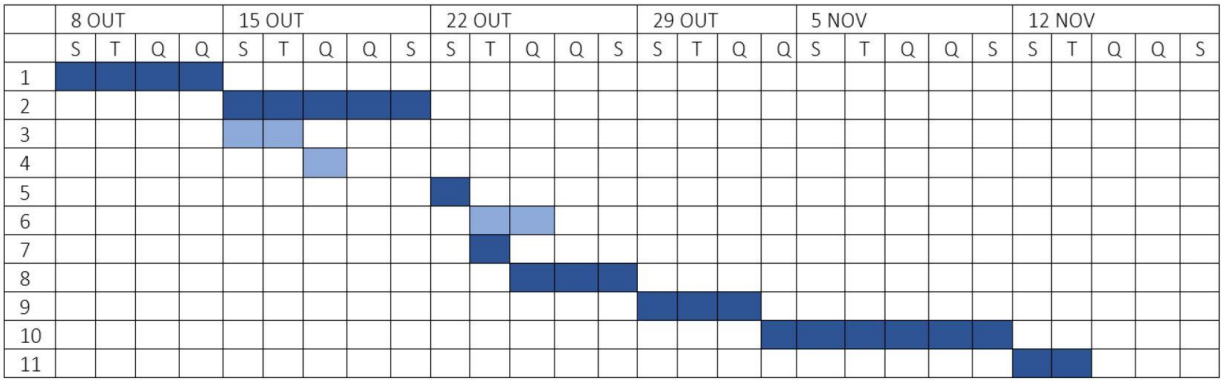

Figura 6.4 Diagrama Gantt para as atividades da WBS

Essas atividades de planejamento, porém, de nada adiantarão se não forem levadas a sério pelos desenvolvedores e pelo próprio gerente. Os capítulos seguintes indicam como fazer para que as estimativas de tempo sejam efetivamente realistas e como se preparar para possíveis problemas ao longo do projeto. Além disso, é mostrado mais adiante como o gerente deve fazer para bem conduzir um projeto durante seu desenvolvimento.

REMISSIVO DO CAPÍTULO