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 iniciar qualquer ação. Depois será conceituado o *termo de abertura* (Seção 6.2) de um projeto e sua *declaração de escopo* (Seção 6.3). Na sequência será mostrado como *planejar um projeto* que adota um modelo de processo iterativo (Seção 6.4), ou seja, o planejamento de longo prazo, e também como *planejar um ciclo iterativo* (Seção 6.5), ou planejamento detalhado de curto prazo.

Já foi visto até aqui que o desenvolvimento de software e atividades relacionadas estrutura-se a partir de um modelo de ciclo de vida, escolhido pelo engenheiro de software para servir à empresa. A partir deste modelo de ciclo de vida, a empresa deve 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 colocação em 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 ciclo de vida utilizado é iterativo. Assim, dois níveis de planejamento serão abordados:

a) O *planejamento de projeto*, de longa duração.
b) O *planejamento de iteração*, de curta duração e mais detalhado.

A literatura de planejamento de projetos em geral não considera estes dois níveis de planejamento, que são típicos de projetos de desenvolvimento de software, mas não são comuns em outras áreas.

6.1 Seleção de Projetos

Uma empresa de desenvolvimento de software vai executar um projeto que normalmente servirá a outra empresa ou grupo de usuários. Normalmente existe mais de uma possibilidade de projeto e nem sempre todos os projetos podem ser desenvolvidos. A empresa desenvolvedora, assim, deverá pesar alguns pontos antes de decidir iniciar um projeto, por exemplo:

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

Estas e outras perguntas normalmente são avaliadas pela gerencia superior da empresa, antes de assumir um compromisso para o desenvolvimento de um projeto.

Por outro lado, a empresa cliente também não tem recursos ilimitados e os projetos de desenvolvimento de sistemas poderão estar competindo entre si, ou ainda com outros projetos, que necessitam investimento. Assim, o compromisso da empresa cliente com o

projeto de desenvolvimento de software possivelmente será afetado pelos seguintes fatores (Xavier, 2011)[82]:

a) Retorno financeiro em relação ao investimento.
b) Grau de incremento da participação da empresa no mercado.
c) Melhoria da imagem da empresa.
d) Utilização de capacidade ociosa.
e) Aquisição de novas tecnologias.

As empresas cliente tenderão a pontuar essas e outras questões antes de se comprometer 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 empresas, ou a decisão da empresa desenvolvedora de *COTS* de que haverá o início do projeto, isto deve ser oficializado em um termo de abertura (*project charter*).

Segundo o PMBOK (PMI, 2004), o termo de abertura deverá conter ou referenciar documentos externos com pelo menos as seguintes informações:

a) Objetivo e justificativa do projeto.
b) Descrição em alto nível do projeto.
c) Requisitos de alto nível que satisfazem os principais interessados.
d) Nomeação do gerente de projeto e definição do nível de autoridade conferida (Pode usar os recursos sem aprovação superior? Pode contratar pessoal?).
e) Cronograma de marcos (*milestones*) resumido.
f) Definição dos papeis e responsabilidades das partes interessadas.
g) Organização funcional do projeto.
h) 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?).
i) Restrições.
j) Estudo de viabilidade (*business case*) indicando o retorno financeiro e não financeiro previsto.
k) 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 isto é o que lhe dará autoridade para iniciar o projeto.

[82] www.administracaovirtual.com/administracao/downloads/apostilas/Apostila-Gerencia-Escopo-Magno-FGV.pdf

6.3 Declaração de Escopo

Inicialmente o planejador de um projeto deve estabelecer quais são os objetivos finais do projeto. O produto nem sempre é apenas o software funcionando. Outros elementos são usualmente 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 onde se quer estar? Infelizmente, muitos planejadores de projetos esquecem-se desta importante etapa. Por exemplo, o projeto termina com a entrega do software, ou com a confirmação de sua correta utilização pelo cliente?

O objetivo de um projeto (e também das iterações) deve ser sempre um conjunto de artefatos, ou seja, coisas palpáveis, que se possa dizer "é isto aqui". 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 neste sentido, ou ainda "implementar este e aquele requisitos".

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

a) *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 esta 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.
b) *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 *deliverable* dos desenvolvedores. Normalmente trata-se de versões implementadas do sistema, mas essa lista poderá incluir também outros itens como projeto, manuais, discos de instalação, treinamento, etc.
c) *Objetivos do projeto*. São os 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 ao prazo, custo e qualidade do produto. O estabelecimento de objetivos não quantificáveis (por exemplo, "cliente satisfeito", ou "sistema fácil de usar") representa um fator de alto risco para a determinação do sucesso do projeto. Objetivos devem ser claramente avaliáveis a partir de uma métrica definida. Devem ser evitados a todo custo objetivos vagos e de avaliação subjetiva, como, por exemplo, "desenvolver tecnologia de última geração".
d) *Critérios de aceitação do produto*. Define-se o processo e 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 sobre o qual 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 neste momento, normalmente, ainda não foi feita uma análise de requisitos. Então, 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 aumentá-lo 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 deste item tornará necessária a renegociação do escopo com o cliente.

6.4 Planejamento de Projeto com Ciclos Iterativos

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 este 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 custo total do projeto.

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:

a) Estimar o *esforço total* para realizar o projeto.
b) Em função do esforço total, calcular o *tempo linear* necessário e *tamanho médio da equipe*.
c) Estimar a *duração e esforço nas diferentes fases do projeto*.
d) Estimar a *duração e número dos ciclos iterativos*.

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

6.4.1 Estimação da Duração e Esforço nas Diferentes Fases do Projeto

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

A Figura 6-1 indica um perfil aproximado de tempo e esforço despendido em cada uma das fases do UP. Evidentemente, esse perfil pode ser alterado dependendo das características individuais de cada projeto ou das ferramentas de automatização de projeto, geração de código e teste que se utilize.

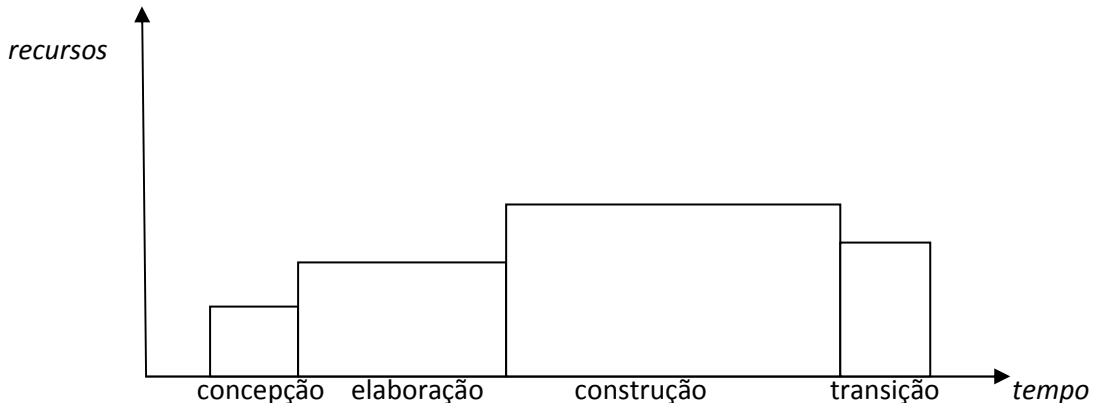

Figura 6-1: Perfil de duração e esforço típicos para um projeto usando UP.

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

a) Concepção: 10% do tempo e 5% do esforço.
b) Elaboração: 30% do tempo e 20% do esforço.
c) Construção: 50% do tempo e 65% do esforço.
d) Transição: 10% do tempo e 10% do esforço.

Assim, por exemplo, um projeto típico de desenvolvimento cujo esforço foi estimado em 40 desenvolvedor-mês, aplicando-se a equação de cálculo de tempo linear da Seção 7.4.4, deverá ter uma duração linear ideal de 8,5 meses.

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

a) Concepção: 10% de 8,5, ou seja, cerca de 0,85 meses.
b) Elaboração: 30% de 8,5, ou seja, cerca de 2,55 meses.
c) Construção: 50% de 8,5, ou seja, cerca de 4,25 meses.
d) 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 estimado (40 desenvolvedor-mês) e aplica-se a percentagem de esforço da fase, conforme definido acima. Depois, divide-se o resultado pela duração linear da fase, conforme obtido logo acima. No exemplo, fica-se com:

a) Concepção: 5% de 40, ou seja, 2 desenvolvedor-mês, o que dividido por 0,85 meses dá cerca de 2,35 desenvolvedores em média na fase.
b) Elaboração: 20% de 40, ou seja, 8 desenvolvedor-mês, o que dividido por 2,55 meses dá cerca de 3,13 desenvolvedores em média na fase.
c) Construção: 65% de 40, ou seja, 26 desenvolvedor-mês, o que dividido por 4,25 meses dá cerca de 6,11 desenvolvedores em média na fase.
d) Transição: 10% de 40, ou seja, 4 desenvolvedor-mês, o que dividido por 0,85 meses dá cerca de 4,7 desenvolvedores em média na fase.

Os valores em meses acima podem ser convertidos em semanas, bastando multiplicar por 4 (ou 3,9 segundo alguns autores). Assim, a concepção teria cerca de 3,5 semanas, a elaboração 10,2 semanas, a construção 17 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 2 desenvolvedores e uma parte menor 3 desenvolvedores. Isso também pode significar que se apenas 2 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.

Porém, algumas observações podem alterar este perfil típico (Kruchten, 2003):

a) Se for necessário um tempo mais longo para estabelecer o projeto, achar financiadores, fazer pesquisa de mercado ou construir provas de conceito, a fase de concepção deve ser tornada mais longa.
b) 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 alongada, porque serão necessários mais ciclos de elaboração para definir a arquitetura e/ou mitigar os riscos conhecidos.
c) Se essa não for a primeira geração do produto (pode ser um ciclo de evolução) e se não serão feitas maiores alterações na arquitetura então as fases de concepção e elaboração podem ser encolhidas.
d) Se o objetivo for atingir o mercado rapidamente por causa de concorrentes ou porque se está criando o mercado, então a fase de construção pode ser encolhida e a fase de transição aumentada. Assim, versões executáveis serão liberadas mais cedo e gradativamente no mercado.
e) Se houver necessidade de uma transição complicada, como por exemplo, 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, estas 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.

Porém, a principal fonte de informação para esse tipo de previsão deve ser sempre o histórico de medições da emrpesa 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 terão a fase de construção usualmente bem menor do que a fase de elaboração.

6.4.2 Estimação da Duração e Número dos Ciclos Iterativos

Uma iteração inicia com planejamento e termina com uma nova versão do sistema disponibilizada internamente ou mesmo uma *release* ao cliente. A duração estimada de um

ciclo no Processo Unificado ou métodos ágeis usualmente varia de 2 a 8 semanas e depende basicamente da complexidade do projeto e da equipe.

Equipes pequenas com até 5 pessoas poderão fazer o planejamento juntos numa manhã de segunda-feira, executar o trabalho ao longo da semana e gerar uma *release* na sexta.

Equipes com mais de 20 pessoas precisarão de mais tempo para distribuir atividades e sincronizar as atividades, até porque a carga de trabalho naturalmente será bem maior. Além disso, a geração da *release* tomará mais tempo, pois haverá um volume maior de partes a serem integradas. Assim, neste caso, uma iteração de 3 a 4 semanas seria mais recomendável.

Equipes com mais de 40 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. Desta forma, um ciclo de 6 a 8 semanas seria recomendável neste caso.

Outros fatores que podem afetar a duração de um ciclo são:

a) Quanto mais automatização no processo e geração de código e no ambiente de desenvolvimento em geral, mais curtos poderão ser os ciclos.
b) Quanto mais familiaridade a equipe tiver com o Processo Unificado e com as técnicas de análise e *design*, mais curtos poderão ser os ciclos.
c) Quanto mais crítico for o fator "qualidade" no desenvolvimento, quanto mais críticas as revisões e testes que precisam ser feitas, mais longos deverão ser os ciclos.

6.4.3 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 planejado para as iterações. Por exemplo, um projeto com iterações de duas semanas, cujas fases de elaboração e construção devem durar 6 meses no total (24 semanas) terá 12 ciclos de elaboração e construção.

A quantidade de ciclos de elaboração e de construção dependerá da quantidade de casos de uso complexos a serem abordados, lembrando que os ciclos de elaboração terminam quando a arquitetura estabiliza, ou seja, quando os últimos casos de uso considerados mais críticos e os principais riscos foram devidamente tratados e incorporados à arquitetura do sistema. A Seção 6.4.1 indica que haverá cinco ciclos de construção (50% do tempo) para cada três ciclos de elaboração (30% do tempo), em média, para projetos nominais, sem características especiais.

A fase de concepção usualmente não é organizada em ciclos, a não ser que mais de um protótipo seja necessário ou que um número significativo de riscos muito importantes deva ser tratado antes de iniciar o projeto propriamente dito. Deve-se lembrar, porém, que o objetivo da 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, usualmente não é organizada em mais do que um ou dois ciclos. Apenas transições mais complexas deverão ser organizadas em mais de um ciclo, com diferentes objetivos definidos para cada um deles.

6.4.4 Definição dos Marcos ou Entregas

Uma vez definido o tamanho das iterações, tamanho da equipe em cada fase e a duração de cada fase (em número de ciclos), 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 6-1.

Tabela 6-1: 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-se que na tabela acima os arredondamentos procuraram fazer com que as fases de elaboração e construção ficassem com um número par de semanas devido aos ciclos iterativos terem sido definidos com duas semanas. O arredondamento do número de desenvolvedores foi feito para cima nas fases de concepção e transição porque o arredondamento da duração destas fases foi feito para baixo. Por 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 duração arredondados para baixo, o que poderia significar que poderá haver algum aperto nesta 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 este projeto seria parecido com o da Tabela 6-2. Note-se que foram definidos ciclos iterativos de 2 semanas exceto para as fases de concepção e transição.

Tabela 6-2: Um 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.
	5	3	20% dos casos de uso de maior risco incorporados na arquitetura.
	7	3	40% dos casos de uso de maior risco incorporados na arquitetura.
	9	3	60% dos casos de uso de maior risco incorporados na arquitetura.
	11	3	80% dos casos de uso de maior risco incorporados na arquitetura.
	13	3	100% dos casos de uso de maior risco incorporados na arquitetura.
Construção	15	6	11% dos demais casos de uso incorporados na arquitetura.
	17	6	22% dos demais casos de uso incorporados na arquitetura.
	19	6	33% dos demais casos de uso incorporados na arquitetura.
	21	6	44% dos demais casos de uso incorporados na arquitetura.
	23	6	55% dos demais casos de uso incorporados na arquitetura.
	25	6	66% dos demais casos de uso incorporados na arquitetura.
	27	6	77% dos demais casos de uso incorporados na arquitetura.
	29	6	88% dos demais casos de uso incorporados na arquitetura.
	31	6	100% dos demais casos de uso incorporados na arquitetura.
Transição	34	5	Sistema instalado. Migração de dados concluída.

A primeira vista pode parecer que a fase de elaboração será mais difícil porque haverá menos desenvolvedores e estes deverão desenvolver uma percentagem grande de casos de uso mais complexos a cada iteração, mas usualmente a quantidade de casos de uso mais críticos em relação ao total de casos de uso de um sistema é menor. Assim, possivelmente haverá menos casos de uso a serem desenvolvidos em cada ciclo da fase de elaboração do que nos ciclos da fase de construção.

Além disso, as porcentagens linearmente crescentes de casos de uso são um tanto simplistas, pois existem casos de uso de complexidade distinta. Um plano de projeto mais realista consideraria porcentagens de *pontos de caso de uso* (Seção 7.5) ou ainda, preferivelmente, os nomes dos casos de uso que deveriam ser desenvolvidos em cada iteração, se estes já forem conhecidos (possivelmente isso só será possível no final da fase de concepção), ordenados pela sua prioridade.

6.5 Planejamento de Iteração

Concluído o planejamento do projeto, se este for feito com ciclos iterativos, apenas o primeiro ciclo será planejado detalhadamente inicialmente. Apenas quando este ciclo estiver em andamento inicia-se o planejamento do segundo ciclo, e assim por diante. Esta seção tratará então do planejamento detalhado, ou seja, do planejamento dos ciclos iterativos.

Se se estiver trabalhando com um método ágil, os objetivos da iteração serão definidos pelas histórias de usuário ou requisitos a serem implementados. No caso do UP, os objetivos de uma iteração poderão ser de três tipos:

a) Implementar total ou parcialmente um ou mais *casos de uso*.

b) Mitigar um *risco* conhecido, gerando ou executando um plano de redução de probabilidade, redução de impacto ou ainda de recuperação de desastre.
c) Implementar total ou parcialmente uma ou mais *modificações* solicitadas. À medida que a arquitetura do sistema evolui nas iterações, modificações poderão ser solicitadas em função da não adequação aos requisitos ou ainda à mudança destes. Incorporar ao software estas solicitações de mudança pode ser então 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 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:

a) Casos de uso que representam os processos de negócio mais críticos para a organização, por exemplo, aqueles através dos quais a organização realiza seus objetivos, como por exemplo, obtenção de lucros.
b) Riscos de alta importância, ou seja, com alto impacto e alta probabilidade de ocorrer.
c) 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 também colocados no mesmo ciclo por conveniência. O importante é que o esforço total estimado não ultrapasse a quantidade de desenvolvedor-hora que se pode alocar dentro da duração prevista do ciclo.

Selecionados os elementos a serem tratados no ciclo, deve-se estabelecer claramente qual o objetivo da iteração, ou seja, até que ponto os elementos selecionados deverão ser desenvolvidos. É importante que o objetivo da iteração também seja detalhado a ponto de que possam ser identificados os artefatos que serão produzidos ao final da iteração.

O objetivo da iteração deve ser buscado inicialmente no plano de projeto, mas este poderá ser alterado em função do estado do projeto (atrasado, por exemplo) e das listas de priorização de casos de uso, riscos e modificações.

Na sequência, deve-se estabelecer a estrutura analítica da iteração (Seção 6.5.1), ou seja, o conjunto de atividades que devem ser executadas para obter os artefatos que são objetivo da iteração. Se o método de desenvolvimento for ágil, esse inventário de atividades será feito pela própria equipe de maneira mais informal.

Porém, 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 (Seção 6.5.5 e 6.5.6).

6.5.1 WBS - Estrutura Analítica da Iteração

A estrutura analítica da iteração (WBS ou *Work Breakdown Structure*) (Tausworthe, 1980) é uma estrutura que apresenta as atividades que devem ser executadas para atingir os objetivos

determinados para o projeto (quando se vai planejar o projeto como um todo, atividade por atividade) ou iteração (no caso de planejamento por ciclos). Várias resoluções de trabalho do Governo Norte Americano exigem a apresentação de uma WBS para a execução de um projeto.

Além da lista de atividades, é importante utilizar um método de estimação de esforço para prever a duração de cada atividade individual. A Seção 7.3.3 apresenta um conjunto de estimativas individualizadas por fase para as disciplinas do RUP.

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

Se for usado um método ágil, recomenda-se que a equipe decida sobre as atividades a serem 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 estabelecer atividades e estimar esforço.

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

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 com uma árvore, sendo que atividades podem ser aglutinadas ou detalhadas e estabelecendo uma árvore de decomposição entre elas. As atividades nas folhas desta árvore são as atividades *terminais* e estas devem seguir a regra 8-80 especificada na subseção seguinte.

É muito importante que o planejador do projeto planeje artefatos de saída e não meramente ações. Atividades devem necessariamente produzir algo físico e não apenas a execução de ações por parte do responsável ou participantes. Tentar modelar um projeto baseado em ações vai necessariamente levar a um detalhamento excessivo das atividades e consequentemente à impossibilidade de gerenciar adequadamente o trabalho. Então, a regra de ouro do planejamento é: “cada atividade deve gerar pelo menos um produto palpável”.

Estilos de WBS que prevêem diferentes estágios de um artefato (por exemplo, versão inicial, versão intermediária e versão final), devem caracterizar exatamente o que esperam de cada uma destas versões.

Por exemplo, a versão inicial de um documento de requisitos poderia ter apenas uma lista de funções identificadas. Uma versão intermediária deste 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 estivesse organizado e revisado dentro de um determinado padrão e que os requisitos tivessem sido verificados em relação a sua completeza e consistência por algum processo padrão definido.

6.5.1.1 Regra 8-80

Atividades que se estima, levarão muito tempo para serem completadas, devem ser subdivididas em atividades mais curtas. Atividades que se estima levarão pouco tempo, devem ser aglutinadas com outras. A Regra 8-80 estabelece que nenhuma atividade terminal deve durar mais de 80 horas (duas semanas, ou dez dias de trabalho ideais), nem menos de 8 horas (um dia de trabalho ideal).

Não se deve ter atividades com duração muito longa porque fica muito difícil gerenciá-las e acompanhar seu andamento.

Também não se deve ter atividades muito curtas porque microgerenciar essas atividades minúsculas pode provocar um *overhead* de gerenciamento que, ao invés de ajudar, vai atrapalhar o projeto.

Métodos como XP são ainda mais restritivos em relação ao tamanho das atividades, pois exigem que sua duração seja de 1 a 3 dias ideais de trabalho, ou seja, 8 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 ela possa servir de base para a gerência do projeto durante a iteração sem ser um estorvo.

6.5.1.2 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 3 ou 4 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.*

Ainda mais, o número total de pacotes de trabalho em uma WBS não deve ultrapassar o limite de 200 elementos, embora 100 já seja considerado um número muito alto.

Considerando-se que cada atividade terminal poderá ter no máximo 80 horas, essa regra estabelece que uma iteração ou projeto gerenciável deverá ter no pior dos casos 80x200 = 16000 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. Neste sentido, as iterações de duas semanas dos métodos ágeis e do UP garantem que o número de horas total nunca seja muito grande. Sendo apenas 80 horas de atividade por desenvolvedor em duas semanas, seriam necessários 200 desenvolvedores trabalhando numa iteração para se atingir tal limite. Acima disso (por exemplo 60 pessoas trabalhando em ciclos de 8 semanas, ou seja, uma carga de 19200 horas por ciclo) seria altamente recomendável a subdivisão do projeto e/ou da equipe.

6.5.1.3 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).

6.5.2 Os Dez Mandamentos da WBS

Xavier (2011) apresenta uma lista com dez recomendações, ou dez mandamentos para a elaboração de WBS:

a) *Cobiçarás a WBS do próximo*. Idealmente uma WBS não deve ser iniciada do zero. Possivelmente existem projetos anteriores semelhantes ou *template*s do próprio processo para que a WBS já seja elaborada a partir de experiências passadas.
b) *Explicitarás todos os subprodutos, inclusive os necessários ao gerenciamento do projeto*. 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 algum desenvolvedor estiver trabalhando em algo que não contribui para nenhum subproduto da WBS, então ele estará trabalhando fora do escopo, e deve-se decidir se o subproduto deve ser incluído na WBS ou a atividade do desenvolvedor revista.
c) *Não usarás os nomes em vão*. Não devem ser usados nomes vagos que deixem dúvida sobre o subproduto a ser gerado pela atividade. Deve-se usar substantivos para definir o subproduto e não verbos. Por exemplo, usa-se "relatório de teste do módulo" ou invés de "testar o módulo".
d) *Guardarás a descrição dos pacotes de trabalho no dicionário da WBS.* O dicionário da WBS é o documento que define cada um dos pacotes de trabalho. Na WBS aparece apenas o seu nome curto, mas o dicionário deve descrever em maior detalhe o que é esse subproduto.
e) *Decomporás as atividades até o nível de detalhe que permita o planejamento e controle do trabalho necessários para a entrega do produto.* Respeitadas as regras 8-80, dos 100% e dos níveis, vistas anteriormente, o planejamento deve criar atividades que efetivamente possam ser verificadas.
f) *Não decomporás em demasia, de forma que o custo/tempo de planejamento e controle não traga o benefício correspondente.* Embora possa haver variações, em geral a regra 8-80 é uma boa medida para evitar que se detalhe demais as atividades da WBS, evitando assim a sobrecarga de trabalho de gerência.
g) *Honrarás o pai.* Cada atividade da WBS decomposta em subatividades funciona como um todo com partes. Deve-se verificar que as subatividades realmente sejam um detalhamento do pacote de trabalho da atividade pai.
h) *Decomporás de forma que a soma dos subprodutos das atividades filho seja igual a 100% do subproduto da atividade pai*. Aplica-se a regra dos 100%.
i) *Não decomporás somente um subproduto*. Nenhum elemento da WBS deve ter um único filho. Pela regra dos 100%, neste caso, ele seria igual ao pai. Então porque representar duas vezes o mesmo elemento? Caso isso aconteça, ou a atividade deve ficar sem ser decomposta, ou deve-se verificar se não estão faltando outras subatividades que foram esquecidas.
j) *Não repetirás o mesmo elemento como componente de mais de um subproduto*. Nenhuma atividade pode ter dois pais. Podem existir atividades com o mesmo nome, mas cada vez que ela aparecer como componente de outra atividade, será uma

atividade diferente, executada em outro período de tempo e/ou por outro desenvolvedor.

6.5.3 Identificação dos Responsáveis por cada Atividade

Um *workflow* usualmente define que o responsável por uma atividade é um papel, ou seja, uma pessoa com uma ou mais habilidades desejáveis. Quando uma iteração for planejada a partir deste *workflow* deve-se então atribuir as atividades a pessoas reais que atendam a este papel desejado sempre que possível.

Cada atividade da WBS deverá ser atribuída a um ou mais responsáveis. Essas atribuições poderão ter efeito sobre o cronograma de projeto, pois embora certas atividades possam ser executadas em paralelo, não é possível fazê-lo assim caso estejam atribuídas ao mesmo responsável.

6.5.4 Identificação dos Recursos Necessários e Custo

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

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

6.5.5 Identificação das Dependências entre Atividades

As dependências entre atividades são dadas em função do *workflow* ou identificadas caso a caso pela equipe de planejamento. Usualmente estas dependências existem porque as entradas de uma atividade são as saídas de outra atividade. Não havendo essa condição as atividades podem ser executadas virtualmente em paralelo.

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

É necessário também verificar se as dependências entre as atividades criam um caminho crítico (Seção 6.5.5.2) cujo comprimento seja maior que a duração da iteração. Neste caso, possivelmente será necessário replanejar as atividades de forma a que o caminho crítico e quaisquer outros caminhos caibam no *timeboxing* da iteração.

Depois se distribui o tempo restante para as outras atividades. Eventuais erros para mais ou para menos nas estimativas podem compensar-se mutuamente.

6.5.5.1 Rede PERT

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

Há várias ferramentas que permitem a elaboração quase automática de um diagrama PERT. Nos exemplos abaixo os gráficos foram gerados com a ferramenta gratuita OpenProj[83].

Usualmente basta que se defina o conjunto das atividades, suas dependências e sua duração e a ferramenta vai calcular as datas de início e de finalização prováveis de cada atividade, conforme mostrado na Figura 6-2.

		Nome	Duração	Início	Término	Predecessoras
1		Desenvolver visão geral do sistema	4 dias	08/09/10 08:00	13/09/10 17:00	
2		Eliciar necessidades dos interessados	5 dias	14/09/10 08:00	20/09/10 17:00	1
3		Gerenciar dependências	2 dias	14/09/10 08:00	15/09/10 17:00	1
4		Capturar vocabulário comum	1 dia	16/09/10 08:00	16/09/10 17:00	3
5		Encontrar atores e casos de uso	1 dia	21/09/10 08:00	21/09/10 17:00	4;2
6		Estruturar o modelo de casos de uso	2 dias	22/09/10 08:00	23/09/10 17:00	5
7		Priorizar os casos de uso	1 dia	22/09/10 08:00	22/09/10 17:00	5
8		Detalhar os casos de uso	3 dias	23/09/10 08:00	27/09/10 17:00	7
9		Modelar interface com usuário	3 dias	28/09/10 08:00	30/09/10 17:00	8
10		Prototipar interface com usuário	6 dias	01/10/10 08:00	08/10/10 17:00	9
11		Revisar requisitos	2 dias	11/10/10 08:00	12/10/10 17:00	6;10

Figura 6-2: WBS em uma ferramenta de gerência de projetos, com duração prevista e dependências.

Na Figura 6-2, o planejador preencheu as colunas “nome”, “duração” e “predecessoras”, sendo que as colunas “início” e “término” foram calculadas automaticamente pela ferramenta. Este WBS foi gerado a partir do *workflow* da Figura 5-4.

Um exemplo de rede PERT gerado a partir da WBS da Figura 6-2 é apresentado na Figura 6-3.

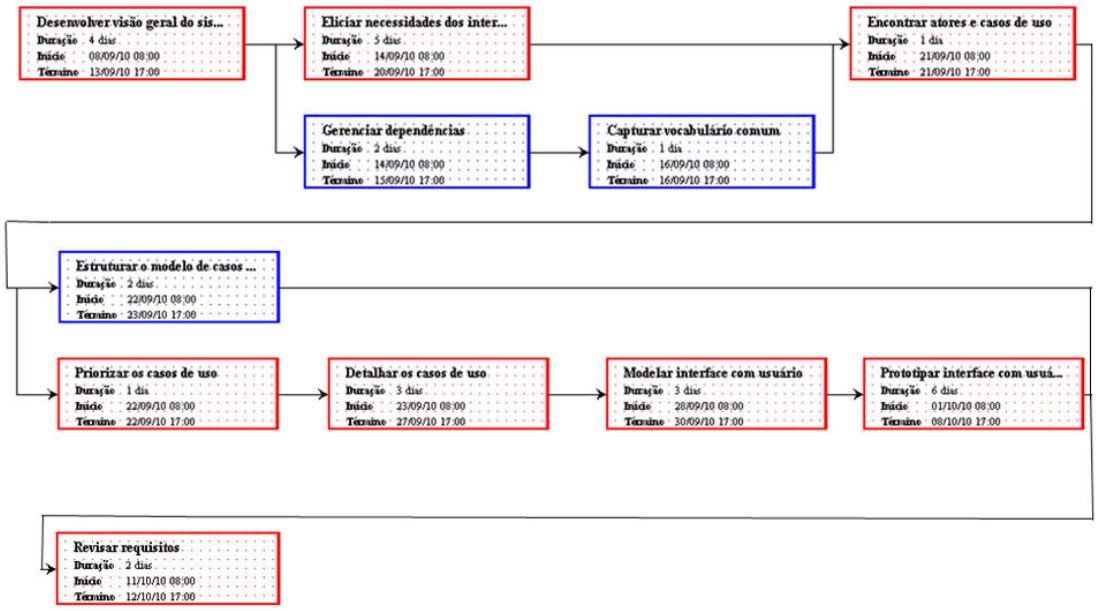

Figura 6-3: Uma rede PERT para as atividades da WBS.

[83] openproj.org/

6.5.5.2 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 irá atrasar. Este é o caminho sem folga.

As atividades, entretanto, 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 mostradas em vermelho. Esta é uma característica automática da ferramenta. As atividades que não estão no caminho crítico são mostradas em azul.

O caminho crítico pode não ser simplesmente uma linha, mas um caminho composto. Ou seja, atividades paralelas podem estar no caminho crítico.

Quando uma atividade do caminho crítico atrasa é 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. Existem três opções usuais:

a) Aumentar a jornada da equipe (o que não pode se transformar em rotina).
b) 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).
c) Eliminar alguns objetivos (artefatos) ou características de artefatos da iteração. Por exemplo, ao invés de implementar 3 casos de uso, caso haja atrasos, implementa-se apenas 2, 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 frequencia, a moral da equipe vai baixar e possivelmente atrasos serão cada vez mais frequentes.

Já o aumento do tamanho da equipe usualmente só produz resultados positivos a médio prazo, ou seja, duas ou três iterações depois daquela onde um ou mais novos membros foram adicionados. Assim, essa solução, inicialmente, deve atrasar o projeto ainda mais.

A eliminação de artefatos ou características de artefatos, que são remanejados para a lista de mudanças solicitadas para serem resolvidos oportunamente em uma iteração futura usualmente é a recomendação mais acertada nestes 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.

6.5.6 Cronograma

O cronograma do projeto usualmente é mostrado em um diagrama Gantt, que consiste em uma visualização do tempo linear transcorrido e a ocorrência das diferentes atividades ao longo deste tempo.

Em relação ao diagrama PERT, o diagrama Gantt apresenta o andamento das atividades ao longo de uma linha de tempo permitindo visualizar claramente quais atividades devem estar sendo executadas a cada dia.

A Figura 6-4 mostra um diagrama Gantt para o diagrama 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.

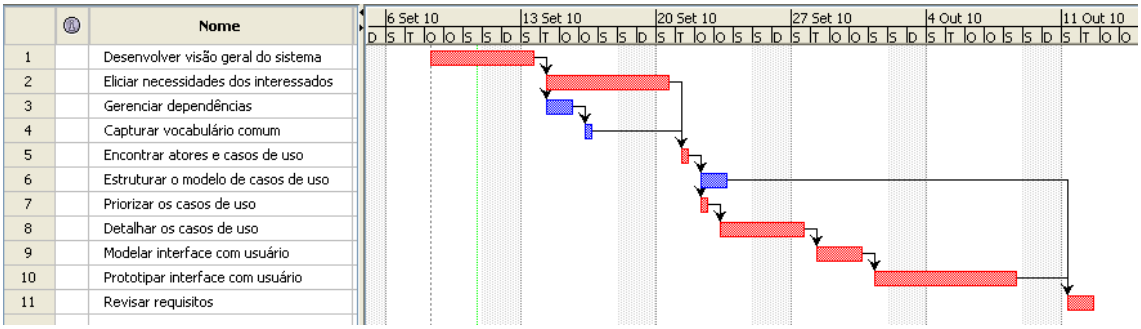

Figura 6-4: Diagrama Gantt para as atividades da WBS.

Essas atividades de planejamento, porém, de nada adiantam 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 a longo do projeto. Além disso, é mostrado adiante como o gerente deve fazer para bem conduzir um projeto durante seu desenvolvimento.

