3 Estrutura do processo de software

Conceitos-chave

Em um livro fascinante que apresenta a visão de um economista sobre software e engenharia de software, Howard Baetjer Jr. [Bae98] comenta o processo de software:

> Porque o software, como todo capital, é conhecimento incorporado, e porque esse conhecimento é, inicialmente, disperso, tácito, latente e, em considerável medida, incompleto, o desenvolvimento de software é um processo de aprendizado social. Esse processo é um diálogo no qual o conhecimento, que deverá se tornar o software, é coletado, reunido e incorporado ao software. O processo possibilita a interação entre usuários e projetistas, entre usuários e ferramentas em evolução e entre projetistas e ferramentas em evolução (tecnologia). Trata-se de um processo iterativo no qual a própria ferramenta em evolução serve como meio de comunicação, com cada nova rodada do diálogo extraindo mais conhecimento útil das pessoas envolvidas.

De fato, construir software é um processo de aprendizado social iterativo e o resultado, algo que Baetjer denominaria "capital de software", é a incorporação do conhecimento coletado, filtrado e organizado à medida que se desenvolve o processo.

Mas, do ponto de vista técnico, o que é exatamente um processo de software? No contexto deste livro, definimos *processo de software* como uma metodologia para as atividades, ações e tarefas necessárias para desenvolver

PANORAMA

O que é? Quando se elabora um produto ou sistema, é importante seguir uma série de passos previsíveis – um roteiro que ajude a criar um resultado de alta qualidade e dentro do prazo estabelecido. O roteiro é denominado "processo de software".

Quem realiza? Engenheiros de software e seus gerentes adaptam o processo às suas necessidades e então o seguem. Os solicitantes do software têm um papel a desempenhar no processo de definição, construção e teste do software.

Por que é importante? Porque propicia estabilidade, controle e organização para uma atividade que pode se tornar bastante caótica sem controle. Entretanto, uma abordagem de engenharia de software moderna deve ser "ágil". Deve demandar apenas atividades, controles e artefatos que sejam apropriados para a equipe do projeto e para o produto a ser produzido.

Quais são as etapas envolvidas? O processo adotado depende do software a ser desenvolvido. Determinado processo pode ser adequado para um software do sistema de voo de uma aeronave, enquanto um processo totalmente diferente pode ser indicado para a criação de um site.

Qual é o artefato? Do ponto de vista de um engenheiro de software, os artefatos são os programas, os documentos e os dados produzidos pelas atividades e tarefas definidas pelo processo.

Como garantir que o trabalho foi realizado corretamente? Há muitos mecanismos de avaliação que permitem às empresas determinarem o nível de "maturidade" de seu processo de software. Entretanto, a qualidade, o cumprimento de prazos e a viabilidade em longo prazo do produto que se desenvolve são os melhores indicadores da eficácia do processo utilizado.

um software de alta qualidade. "Processo" é sinônimo de "engenharia de software"? A resposta é "sim e não". Um processo de software define a abordagem adotada conforme um software é elaborado pela engenharia. Entretanto, a engenharia de software também engloba tecnologias que fazem parte do processo – métodos técnicos e ferramentas automatizadas.

Mais importante, a engenharia de software é realizada por pessoas criativas e com amplo conhecimento, e que devem adaptar um processo de software maduro de modo que fique adequado aos produtos desenvolvidos e às demandas de seu mercado.

3.1 Um modelo de processo genérico

No Capítulo 2, processo foi definido como um conjunto de atividades de trabalho, ações e tarefas realizadas quando algum artefato de software deve ser criado. Cada uma dessas atividades, ações e tarefas se alocam dentro de uma metodologia ou modelo que determina sua relação com o processo e umas com as outras.

A hierarquia de trabalho técnico, dentro do processo de software, consiste em atividades e ações abrangentes compostas por tarefas.

O processo de software está representado esquematicamente na Figura 3.1. De acordo com a figura, cada atividade metodológica é composta por um conjunto de ações de engenharia de software. Cada ação é definida por um *conjunto de tarefas*, o qual identifica as tarefas de trabalho a ser completadas, os artefatos de software que serão produzidos, os fatores de garantia da qualidade que serão exigidos e os marcos utilizados para indicar progresso.

Como discutido no Capítulo 2, uma metodologia de processo genérica para engenharia de software estabelece cinco atividades metodológicas: **comunicação**, **planejamento**, **modelagem**, **construção** e **entrega**. Além disso, um conjunto de atividades de apoio é aplicado ao longo do processo, como o acompanhamento e controle do projeto, a administração de riscos, a garantia da qualidade, o gerenciamento das configurações, as revisões técnicas, entre outras.

O que é fluxo de processo?

Um aspecto importante do processo de software ainda não foi discutido. Esse aspecto – chamado *fluxo de processo* – descreve como são organizadas as atividades metodológicas, bem como as ações e tarefas que ocorrem dentro de cada atividade em relação à sequência e ao tempo, como ilustrado na Figura 3.2.

Um *fluxo de processo linear* executa cada uma das cinco atividades metodológicas em sequência, começando com a comunicação e culminando com a entrega (Figura 3.2a). Um *fluxo de processo iterativo* repete uma ou mais das atividades antes de prosseguir para a seguinte (Figura 3.2b). Um *fluxo de processo evolucionário* executa as atividades de forma "circular". Cada volta pelas cinco atividades conduz a uma versão mais completa do software (Figura 3.2c). Um *fluxo de processo paralelo* (Figura 3.2d) executa uma ou mais atividades em paralelo com outras (por exemplo, a modelagem para um aspecto do software poderia ser executada em paralelo com a construção de outro aspecto do software).

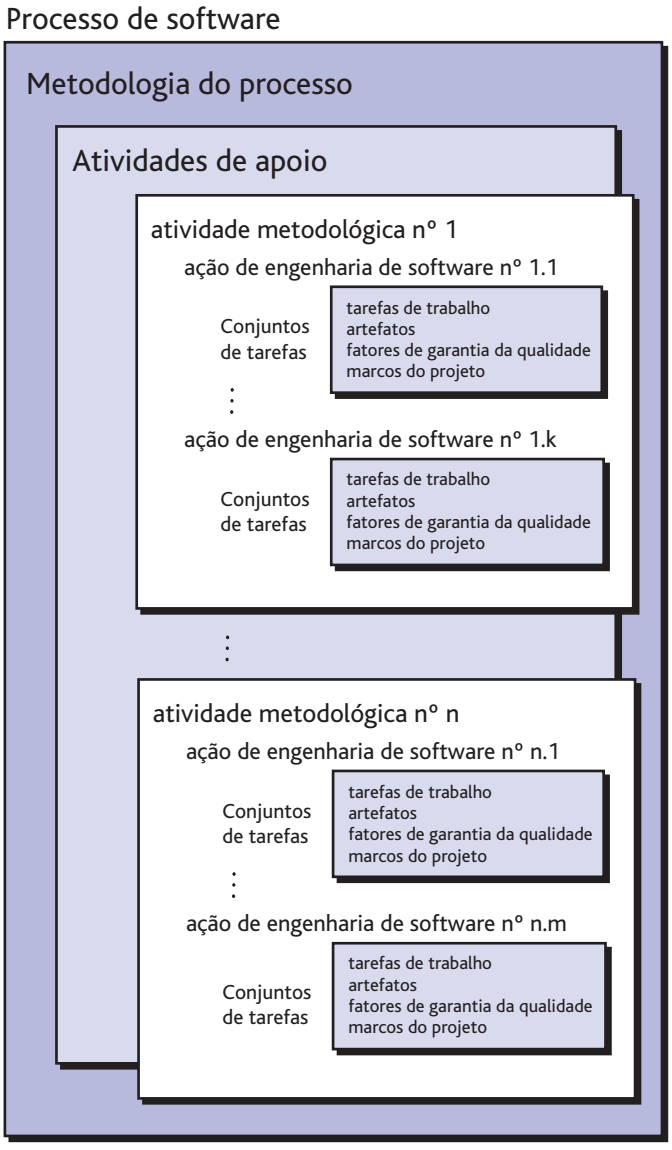

FIGURA 3.1 Uma metodologia do processo de software.

3.2 Definição de uma atividade metodológica

"Se o processo estiver correto, os resultados falarão por si mesmos."

Takashi Osada

Embora cinco atividades metodológicas tenham sido descritas e tenha-se fornecido uma definição básica de cada uma delas no Capítulo 2, uma equipe de software precisa de muito mais informações antes de poder executar qualquer uma das atividades como parte do processo de software. Assim, enfrenta-se uma questão-chave: *Quais ações são apropriadas para uma atividade metodológica, uma vez fornecidos a natureza do problema a ser solucionado, as características das pessoas que farão o trabalho e os envolvidos no projeto?*

Como uma atividade metodológica é modificada de acordo com as alterações da natureza do projeto?

Para um pequeno projeto de software solicitado por uma única pessoa (em um local distante) com requisitos simples e objetivos, a atividade de **comunicação** pode se resumir a pouco mais de um telefonema ou email para o

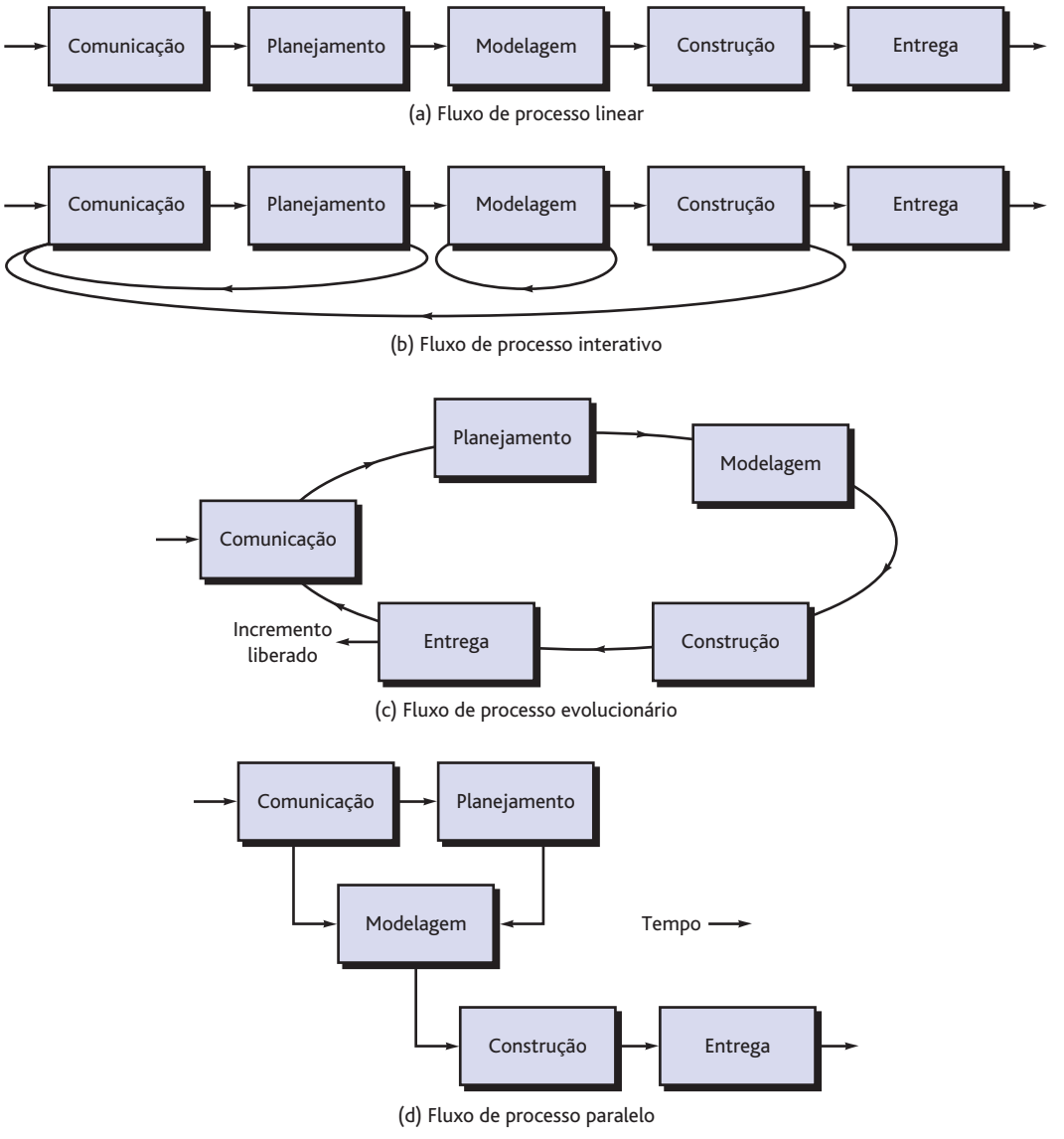

FIGURA 3.2 Fluxo de processo.

envolvido. Portanto, a única ação necessária é uma *conversa telefônica*, e as tarefas de trabalho (o *conjunto de tarefas*) que essa ação envolve são:

1. Contatar o envolvido via telefone.
2. Discutir os requisitos e gerar anotações.
3. Organizar as anotações em uma breve relação de requisitos, por escrito.
4. Enviar um email para o envolvido para revisão e aprovação.

Projetos diferentes exigem conjuntos de tarefas diferentes. A equipe de software escolhe o conjunto de tarefas baseada no problema e nas características do projeto.

Se o projeto fosse consideravelmente mais complexo, com muitos envolvidos, cada qual com um conjunto de requisitos diferentes (por vezes conflitantes), a atividade de comunicação poderia ter seis ações distintas (descritas no Capítulo 8): *concepção*, *levantamento*, *elaboração*, *negociação*, *especificação* e *validação*. Cada uma dessas ações de engenharia de software conteria muitas tarefas de trabalho e uma série de diferentes artefatos.

INFORMAÇÕES

Conjunto de tarefas

Um conjunto de tarefas define o trabalho a ser feito para atingir os objetivos de uma ação de engenharia de software. Por exemplo, *levantamento* (mais comumente denominada "levantamento de requisitos") é uma importante ação de engenharia de software que ocorre durante a atividade de **comunicação**. A meta do levantamento de requisitos é compreender o que os vários envolvidos esperam do software a ser desenvolvido.

Para um projeto pequeno e relativamente simples, o conjunto de tarefas para levantamento dos requisitos seria semelhante a este:

1. Fazer uma lista dos envolvidos no projeto.
2. Fazer uma reunião informal com todos os envolvidos.
3. Solicitar a cada envolvido uma lista com as características e funções necessárias.
4. Discutir sobre os requisitos e elaborar uma lista final.
5. Organizar os requisitos por grau de prioridade.
6. Destacar pontos de incertezas.

Para um projeto de software maior e mais complexo, é preciso usar um conjunto diferente de tarefas. Esse conjunto pode incluir as seguintes tarefas de trabalho:

1. Fazer uma lista dos envolvidos no projeto.
2. Entrevistar separadamente cada um dos envolvidos para levantamento geral de suas expectativas e necessidades.
3. Fazer uma lista preliminar das funções e características, com base nas informações fornecidas pelos envolvidos.
4. Agendar uma série de reuniões facilitadoras para especificação de aplicações.
5. Realizar reuniões.
6. Incluir cenários informais de usuários como parte de cada reunião.
7. Refinar os cenários de usuários, com base no feedback dos envolvidos.
8. Fazer uma lista revisada dos requisitos dos envolvidos.
9. Empregar técnicas de implantação de funções de qualidade para estabelecer graus de prioridade dos requisitos.
10. Agrupar os requisitos de modo que possam ser entregues em incrementos.
11. Fazer um levantamento das limitações e restrições que serão aplicadas ao sistema.
12. Discutir sobre os métodos para validação do sistema.

Esses dois conjuntos de tarefas atingem o objetivo do "levantamento de requisitos"; porém, são bem diferentes quanto ao seu grau de profundidade e formalidade. A equipe de software deve escolher o conjunto de tarefas que possibilite atingir o objetivo de cada ação, mantendo, inclusive, a qualidade e a agilidade.

3.3 Identificação de um conjunto de tarefas

Voltando novamente à Figura 3.1, cada ação de engenharia de software (por exemplo, *levantamento*, uma ação associada à atividade de **comunicação**) pode ser representada por vários e diferentes *conjuntos de tarefas* – constituídos por uma gama de tarefas de trabalho de engenharia de software, artefatos relacionados, fatores de garantia da qualidade e marcos do projeto.

Deve-se escolher um conjunto de tarefas mais adequado às necessidades do projeto e às características da equipe. Isso significa que uma ação de engenharia de software pode ser adaptada às necessidades específicas do projeto de software e às características da equipe.

3.4 Padrões de processo

O que é padrão de processo?

Toda equipe de desenvolvimento encontra problemas à medida que avança no processo de software. Seria útil se soluções comprovadas estivessem pronta-

mente à disposição da equipe, de modo que os problemas pudessem ser localizados e resolvidos rapidamente. Um *padrão de processo*[1] descreve um problema de processo encontrado durante o trabalho de engenharia de software, identificando o ambiente onde foi encontrado e sugerindo uma ou mais soluções comprovadas para o problema. Em termos mais genéricos, um padrão de processo fornece um modelo [Amb98] – um método consistente para descrever soluções de problemas no contexto do processo de software. Combinando padrões, uma equipe conseguirá solucionar problemas e elaborar um processo que melhor atenda às necessidades de um projeto.

"A repetição de padrões é algo bem diferente da repetição de partes. De fato, as partes diferentes serão únicas, pois os padrões são os mesmos."

Christopher Alexander

Padrões podem ser definidos em qualquer nível de abstração.[2] Em alguns casos, um padrão poderia ser utilizado para descrever um problema (e sua solução) associado ao modelo de processo completo (por exemplo, prototipação). Em outras situações, os padrões podem ser usados para descrever um problema (e sua solução) associado a uma atividade metodológica (por exemplo, **planejamento**) ou uma ação dentro de uma atividade metodológica (por exemplo, estimativa de custos do projeto).

Ambler [Amb98] propôs um modelo para descrever um padrão de processo:

Nome do padrão. O padrão deve receber um nome que o descreva no contexto do processo de software (por exemplo, **RevisõesTécnicas**).

Forças. Ambiente onde se encontram o padrão e as questões que tornam visível o problema e que poderiam afetar sua solução.

Um modelo de padrões propicia um meio consistente para descrever um padrão.

Tipo. É especificado o tipo de padrão. Ambler sugere três tipos:

1. ***Padrão de estágio*** – define um problema associado a uma atividade metodológica para o processo. Como uma atividade metodológica envolve várias ações e tarefas de trabalho, um padrão de estágio engloba vários padrões de tarefas (veja o próximo padrão) relevantes ao estágio (atividade metodológica). Um exemplo de padrão de estágio poderia ser **EstabelecimentoDeComunicação**. Esse padrão poderia incorporar o padrão de tarefas **LevantamentoDeRequisitos** e outros.
2. ***Padrão de tarefas*** – define um problema associado a uma ação de engenharia de software ou tarefa de trabalho relevante para a prática de engenharia de software bem-sucedida (por exemplo, **LevantamentoDeRequisitos** é um padrão de tarefas).
3. ***Padrão de fases*** – define a sequência das atividades metodológicas que ocorrem dentro do processo, mesmo quando o fluxo geral de atividades é iterativo por natureza. Um exemplo de padrão de fases seria **ModeloEspiral** ou **Prototipação**.[3]

[1] Uma discussão detalhada sobre padrões é apresentada no Capítulo 11.

[2] Os padrões são aplicáveis a várias atividades de engenharia de software. Padrões de análise, de projeto e de testes são discutidos nos Capítulos 11, 13, 15, 16 e 20. Padrões e "antipadrões" para atividades de gerenciamento de projetos são discutidos na Parte IV deste livro.

[3] Esses padrões de fases são discutidos no Capítulo 4.

"Achamos que desenvolvedores de software não percebem uma realidade essencial: a maioria das organizações não sabe o que faz. Elas acham que sabem, mas não sabem."

Tom DeMarco

Contexto inicial. Descreve as condições sob as quais o padrão se aplica. Antes do início do padrão: (1) Que atividades organizacionais ou relacionadas à equipe já ocorreram? (2) Qual é o estado inicial do processo? (3) Que informação de engenharia de software ou de projeto já existe?

Por exemplo, o padrão **Planejamento** (um padrão de estágio) exige que: (1) clientes e engenheiros de software tenham estabelecido uma comunicação colaborativa; (2) tenha ocorrido a finalização bem-sucedida de uma série de padrões de tarefas [especificados] para o padrão **Comunicação**; e (3) sejam conhecidos o escopo e as restrições do projeto, bem como os requisitos básicos do negócio.

Problema. O problema específico a ser resolvido pelo padrão.

Solução. Descreve como implementar o padrão de forma bem-sucedida. Esta seção descreve como o estado inicial do processo (que existe antes de o padrão ser implementado) é modificado como consequência do início do padrão. Descreve também como as informações de engenharia de software ou de projeto que se encontram à disposição antes do início do padrão são transformadas como consequência da execução bem-sucedida do padrão.

Contexto resultante. Descreve as condições que resultarão assim que o padrão tiver sido implementado com êxito. Após a finalização do padrão: (1) Quais atividades organizacionais ou relacionadas à equipe devem ter ocorrido? (2) Qual é o estado de saída do processo? (3) Quais informações de engenharia de software ou de projeto foram desenvolvidas?

Padrões relativos. Fornecem uma lista de todos os padrões de processo que estão diretamente relacionados ao processo em questão. Essa lista pode ser representada de forma hierárquica ou em alguma outra forma com diagramas. Por exemplo, o padrão de estágio **Comunicação** envolve os padrões de tarefas: **EquipeDeProjeto**, **DiretrizesColaborativas**, **IsolamentoDoEscopo**, **LevantamentoDeRequisitos**, **DescriçãoDasRestrições** e **CriaçãoDeCenários**.

Muitos recursos sobre padrões de processo podem ser encontrados em **www.ambysoft.com/processPatternsPage.html**.

Usos conhecidos e exemplos. Indicam as instâncias específicas a que o padrão é aplicável. Por exemplo, **Comunicação** é obrigatória no início de todo projeto de software, é recomendável ao longo de todo o projeto de software e é obrigatória assim que a atividade **Entrega** estiver em andamento.

Padrões de processo propiciam um mecanismo eficaz para a localização de problemas associados a qualquer processo de software. Os padrões permitem que se desenvolva uma descrição do processo de forma hierárquica que se inicia com nível alto de abstração (um padrão de fases). A descrição é então refinada em um conjunto de padrões de estágio que descreve atividades metodológicas e são ainda mais refinadas, de uma forma hierárquica, em padrões de tarefa mais detalhados para cada padrão de estágio. Uma vez que os padrões de processos tenham sido desenvolvidos, eles poderão ser reutilizados na definição de variantes do processo – isto é, um modelo de processo personalizado pode ser definido por uma equipe de software usando os padrões como blocos de construção para o modelo do processo.

INFORMAÇÕES

Um exemplo de padrão de processo

O padrão de processo resumido mostrado a seguir descreve uma abordagem que pode ser aplicada quando os envolvidos têm uma ideia geral do que precisa ser feito, mas estão incertos quanto aos requisitos específicos do software.

Nome do padrão. **RequisitosImprecisos**

Intuito. Esse padrão descreve uma abordagem voltada à construção de um modelo (um protótipo) passível de ser avaliado iterativamente pelos envolvidos, em um esforço para identificar ou solidificar requisitos do software.

Tipo. Padrão de fase.

Contexto inicial. As seguintes condições devem ser atendidas antes de iniciar esse padrão: (1) envolvidos identificados; (2) forma de comunicação entre envolvidos e equipe de software já determinada; (3) principal problema de software a ser resolvido já identificado pelos envolvidos; (4) compreensão inicial do escopo do projeto, dos requisitos de negócio básicos e das restrições do projeto já atingida.

Problema. Os requisitos são vagos ou inexistentes, ainda assim há o reconhecimento claro de que existe um problema a ser solucionado e ele deve ser identificado utilizando-se uma solução de software. Os envolvidos não sabem o que querem, ou seja, eles não conseguem descrever os requisitos de software em detalhe.

Solução. Uma descrição do processo de prototipação poderia ser apresentada nesta etapa – e está disponível posteriormente, na Seção 4.1.3.

Contexto resultante. Um protótipo de software que identifique os requisitos básicos (por exemplo, modos de interação, características computacionais, funções de processamento) é aprovado pelos envolvidos. Em seguida, (1) o protótipo pode evoluir por uma série de incrementos para se tornar o software de produção ou (2) o protótipo pode ser descartado, e o software de produção ser construído usando-se algum outro padrão de processos.

Padrões relacionados. Os seguintes padrões estão relacionados a esse padrão: **ComunicaçãoComOCliente**, **ProjetoIterativo**, **DesenvolvimentoIterativo**, **AvaliaçãoDoCliente**, **ExtraçãoDeRequisitos**.

Usos conhecidos e exemplos. A prototipação é recomendada quando os requisitos são incertos.

3.5 Avaliação e aperfeiçoamento de processos

A avaliação tenta compreender o atual estado do processo de software com o intuito de aperfeiçoá-lo.

A existência de um processo de software não garante que o software será entregue dentro do prazo, que estará de acordo com as necessidades do cliente ou que apresentará características técnicas que resultarão em qualidade de longo prazo (Capítulo 19). Os padrões de processo devem ser combinados com uma prática de engenharia de software confiável (Parte II deste livro). Além disso, o próprio processo pode ser avaliado para que esteja de acordo com um conjunto de critérios de processo básicos, comprovados como essenciais para uma engenharia de software bem-sucedida.[4]

Ao longo das últimas décadas foi proposta uma série de diferentes abordagens de avaliação e aperfeiçoamento dos processos de software:

SCAMPI (Standard CMMI Assessment Method for Process Improvement) (Método Padrão CMMI de Avaliação para Aperfeiçoamento de Processo da CMMI) – fornece um modelo de avaliação do processo de cinco etapas, contendo cinco fases: início, diagnóstico, estabelecimento, atuação e aprendizado. O método SCAMPI usa o CMMI da SEI como base para avaliação [SEI00].

[4] A CMMI [CMM07] da SEI descreve, de forma extremamente detalhada, as características de um processo de software e os critérios para o êxito de um processo.

"As empresas de software têm mostrado grandes deficiências em tirar proveito das experiências adquiridas com projetos executados."

NASA

CBA IPI (CMM-Based Appraisal for Internal Process Improvement) (Avaliação para Aperfeiçoamento do Processo Interno baseada na CMM) – fornece uma técnica de diagnóstico para avaliar a maturidade relativa de uma organização de software; usa a CMM da SEI como base para a avaliação [Dun01].

SPICE (ISO/IEC15504) – padrão que define um conjunto de requisitos para avaliação do processo de software. A finalidade do padrão é auxiliar as organizações no desenvolvimento de uma avaliação objetiva da eficácia de um processo qualquer de software [ISO08].

ISO 9001:2000 para Software – padrão genérico aplicável a qualquer organização que queira aperfeiçoar a qualidade global de produtos, sistemas ou serviços fornecidos. Portanto, o padrão é aplicável diretamente a organizações e empresas de software [Ant06].

Uma discussão mais detalhada sobre métodos de avaliação de software e aperfeiçoamento de processo é apresentada no Capítulo 37.

3.6 Resumo

Um modelo de processo genérico para engenharia de software consiste em um conjunto de atividades metodológicas e de apoio, ações e tarefas a realizar. Cada modelo de processo, entre os vários existentes, pode ser descrito por um fluxo de processo diferente – uma descrição de como as atividades metodológicas, ações e tarefas são organizadas, sequencial e cronologicamente. Padrões de processo são utilizados para resolver problemas comuns encontrados no processo de software.

Problemas e pontos a ponderar

3.1. Na introdução deste capítulo, Baetjer observa: "O processo oferece interação entre usuários e projetistas, entre usuários e ferramentas em evolução e entre projetistas e ferramentas [de tecnologia] em evolução". Liste cinco perguntas que (1) os projetistas deveriam fazer aos usuários, (2) os usuários deveriam fazer aos projetistas, (3) os usuários deveriam fazer a si mesmos sobre o produto de software a ser desenvolvido, (4) os projetistas deveriam fazer a si mesmos sobre o produto de software a ser construído e sobre o processo que será usado para construí-lo.

3.2. Discuta as diferenças entre os vários fluxos de processo descritos na Seção 3.1. Consegue identificar tipos de problemas que poderiam ser aplicáveis a cada um dos fluxos genéricos descritos?

3.3. Tente desenvolver um conjunto de ações para a atividade de comunicação. Selecione uma ação e defina um conjunto de tarefas para ela.

3.4. Durante a **comunicação**, um problema comum ocorre ao encontrarmos dois envolvidos com ideias conflitantes sobre como o software deve ser. Isto é, há requisitos mutuamente conflitantes. Desenvolva um padrão de processo (que seja um padrão de estágio) usando o modelo apresentado na Seção 3.4 que trata desse problema e sugira uma abordagem eficaz para ele.

Leituras e fontes de informação complementares

A maioria dos livros-texto sobre engenharia de software considera os modelos de processo com certo nível de detalhe. Livros de Sommerville (*Software Engineering,* 9ª ed., Addison-Wesley, 2010), Schach (*Object-Oriented and Classical Software Engineering,* 8ª ed., McGraw-Hill, 2010) e Pfleeger e Atlee (*Software Engineering: Theory and Practice,* 4ª ed., Prentice Hall, 2009) consideram os paradigmas tradicionais e discutem suas vantagens e desvantagens. Munch e seus colegas (*Software Process Definition and Management,* Springer, 2012) apresentam uma visão do processo e do produto da engenharia de software e de sistemas. Glass (*Facts and Fallacies of Software Engineering*, Prentice Hall, 2002) fornece uma visão simples e pragmática do processo de engenharia de software. Embora não seja especificamente dedicado a processos, Brooks (*The Mythical Man-Month*, 2ª ed., Addison-Wesley, 1995) apresenta conhecimentos sábios de projeto antigo que têm tudo a ver com processos.

Firesmith e Henderson-Sellers (*The OPEN Process Framework: An Introduction*, Addison-Wesley, 2001) apresentam um quadro geral para a criação de "processos de software flexíveis e que, ainda assim, não deixam de ser disciplinados" e discutem atributos e objetivos dos processos. Madachy (*Software Process Dynamics*, Wiley-IEEE, 2008) fala sobre técnicas de modelagem que possibilitam a análise dos elementos técnicos e sociais relacionados do processo de software. Sharpe e McDermott (*Workflow Modeling: Tools for Process Improvement and Application Development*, 2ª ed., Artech House, 2008) apresentam ferramentas para modelagem de processos de software e de negócios.

Uma ampla variedade de fontes de informação sobre engenharia de software e o processo de software está disponível na Internet. Uma lista atualizada de referências relevantes (em inglês) para o processo para o software pode ser encontrada no site: **www.mhhe.com/pressman.**