PARTE

UM

PROCESSOS DE SOFTWARE

Nesta seção você aprenderá sobre o processo que fornece uma metodologia para a prática da engenharia de software. As questões abaixo são tratadas nos capítulos seguintes:

- O que é um processo de software?
- Quais são as atividades metodológicas genéricas presentes em todos os processos de software?
- Como os processos são modelados e o que são padrões de processo?
- O que são modelos de processo prescritivo e quais são seus pontos fortes e fracos?
- Por que agilidade é um lema no trabalho da engenharia de software moderna?
- O que é desenvolvimento de software ágil e como ele difere dos modelos de processos mais tradicionais?

Respondidas tais questões, você estará mais bem preparado para compreender o contexto no qual a prática da engenharia de software é aplicada.

2 Modelos de Processo

Conceitos-Chave

Em um livro fascinante, que nos dá a visão de um economista acerca do software e da engenharia de software, Howard Baetjer, Jr. [Bae98], comenta o processo de software:

> Pelo fato de software, como todo capital, ser conhecimento incorporado, e pelo fato de esse conhecimento ser, 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 tornar-se o software, é coletado, reunido e incorporado ao software. Tal 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 conforme se desenvolve o processo.

Mas o que é exatamente um processo de software do ponto de vista técnico? No contexto desse livro, *processo de software é definido* como uma metodologia para as atividades, ações e tarefas necessárias para desenvolver 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. Mas 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 amplos conhecimentos e que devem adaptar um processo de software maduro, de forma que fique apropriado aos produtos desenvolvidos e às demandas de seu mercado.

Panorama

O que é? Quando se trabalha na elaboração de 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? Os 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 ele é importante? Porque propicia estabilidade, controle e organização para uma atividade que pode, sem controle, tornar-se bastante caótica. Entretanto, uma abordagem de engenharia de software moderna deve ser "ágil". Deve demandar apenas atividades, controles e produtos de trabalho 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. Um determinado processo pode ser apropriado para um software do sistema "aviônico" 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 produtos de trabalho são os programas, os documentos e os dados produzidos em consequência das atividades e tarefas definidas pelo processo.

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

2.1 UM MODELO DE PROCESSO GENÉRICO

No Capítulo 1, 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 alocam-se dentro de uma metodologia ou modelo que determina seu relacionamento com o processo e seu relacionamento umas com as outras.

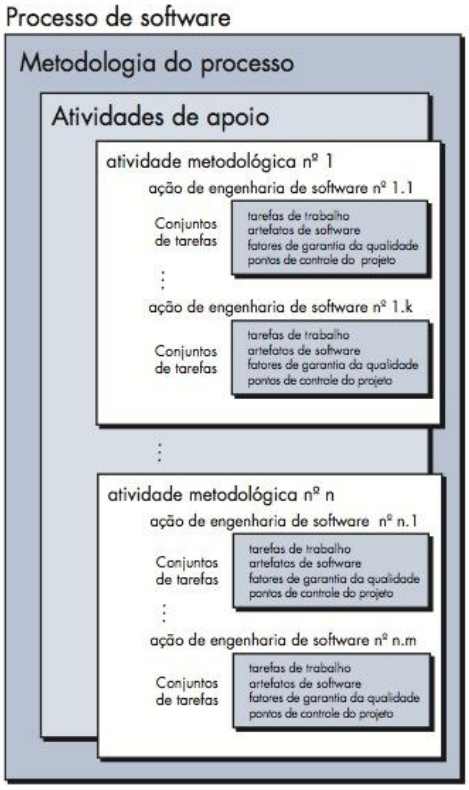
O processo de software é representado esquematicamente na Figura 2.1. De acordo com a figura, cada atividade metodológica é composta por um conjunto de ações de engenharia de software. Cada ação é definida por um *conjunto de tarefas, o qual* identifica as tarefas de trabalho a ser completadas, os artefatos de software que serão produzidos, os fatores de garantia da qualidade que serão exigidos e os marcos utilizados para indicar progresso.

PONTO-CHAVE

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

Como discutido no Capítulo 1, 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 (*umbrella activities*) são aplicadas 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 e outras.

FIGURA 2.1
Uma metodologia do processo de software



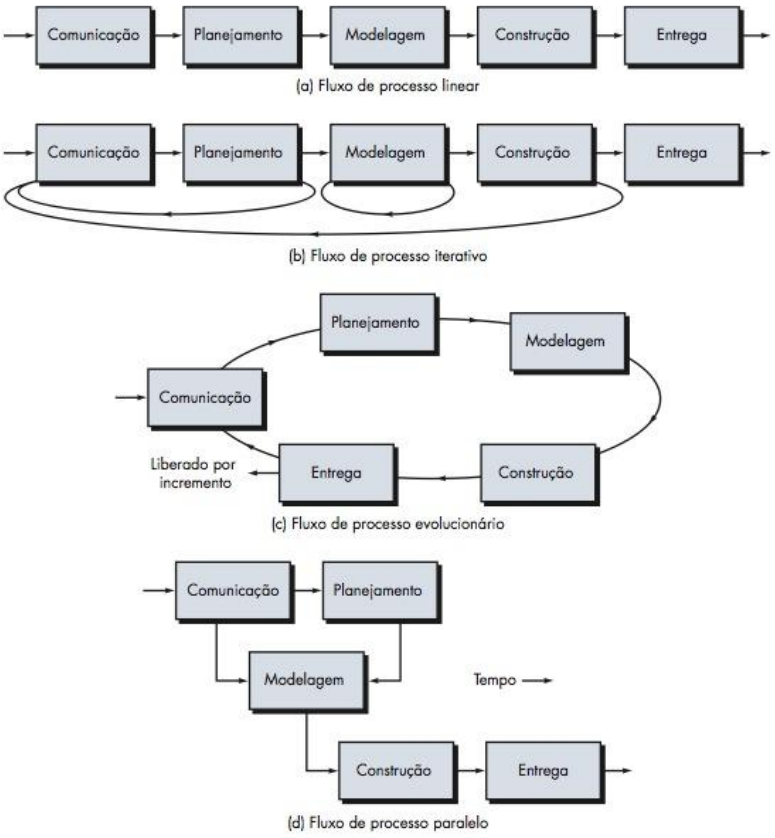
"Achamos que desenvolvedores de software não percebem uma verdade essencial: a maioria das organizações não sabe o que faz. Elas acham que sabem, mas não sabem."

Tom DeMarco

Deve-se notar que um importante aspecto 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 2.2.

Um *fluxo de processo linear* executa cada uma das cinco atividades metodológicas em sequência, começando com a de comunicação e culminando com a do emprego (Figura 2.2a). Um *fluxo de processo iterativo* repete uma ou mais das atividades antes de prosseguir para a seguinte (Figura 2.2b). Um *fluxo de processo evolucionário* executa as atividades de uma forma "circular". Cada volta pelas cinco atividades conduz a uma versão mais completa do software (Figura 2.2c). Um *fluxo de processo paralelo* (Figura 2.2d) executa uma ou mais atividades em paralelo com outras atividades (por exemplo, a modelagem para um aspecto do software poderia ser executada em paralelo com a construção de um outro aspecto do software).

FIGURA 2.2 **Fluxo de processo**



2.1.1 Definindo atividade metodológica

Embora tenham sido descritas cinco atividades metodológicas e se tenha fornecido uma definição básica de cada uma delas no Capítulo 1, uma equipe de software precisa de muito mais informações antes de poder executar apropriadamente qualquer uma dessas atividades como parte do processo de software. Consequentemente, depara-se com uma questão-chave: *Que ações são apropriadas para uma atividade metodológica, uma vez fornecidos a natureza do problema a ser solucionado, as características das pessoas que estarão executando o trabalho e os interessados que estarão propondo o 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 (numa localidade longínqua) com necessidades simples e objetivas, a atividade de comunicação pode resumir-se a pouco mais de um telefonema para o devido solicitante. Portanto, a única ação necessária é uma *conversação telefônica*, e as tarefas de trabalho (o *conjunto de tarefas*) que essa ação envolve são:

1. Contactar o interessado via telefone.
2. Discutir as necessidades e tomar notas.
3. Organizar anotações em uma breve relação de requisitos, por escrito.
4. Enviar um e-mail para o interessado para revisão e aprovação.

Se o projeto fosse consideravelmente mais complexo, com muitos interessados, cada qual com um conjunto de necessidades diferentes (por vezes conflitantes), a atividade de comunicação poderia ter seis ações distintas (descritas no Capítulo 5): *inserção, elucidação, elaboração, negociação, especificação* e *validação*. Cada uma dessas ações de engenharia de software conteria muitas tarefas de trabalho e uma série de diferentes artefatos.

PONTO-CHAVE

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

2.1.2 Identificação de um conjunto de tarefas

Referindo-se novamente à Figura 2.1, cada ação de engenharia de software (por exemplo, elucidação, *uma* ação associada à atividade de comunicação) pode ser representada por vários e diferentes *conjuntos de tarefas* — cada um constituído por uma gama de tarefas de trabalho de engenharia de software, artefatos relativos, fatores de garantia da qualidade e pontos de controle 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.

2.1.3 Padrões de processos

? 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 prontamente 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 (template) [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

1 Uma discussão detalhada sobre padrões é apresentada no Capítulo 12.

2 Os padrões são aplicáveis a várias atividades de engenharia de software. Padrões de análise, de projeto e de testes são discutidos nos Capítulos 7, 9, 10, 12 e 14. Padrões e "antipadrões" para atividades de gerenciamento de projetos são discutidos na Parte 4 deste livro.

INFORMAÇÕES

Conjunto de tarefas

Um conjunto de tarefas define o verdadeiro trabalho a ser feito para se atingir os objetivos de uma ação de engenharia de software. Por exemplo, elucidação (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 interessados esperam do software a ser desenvolvido.

Para um projeto pequeno, relativamente simples, o conjunto de tarefas para levantamento das necessidades seria semelhante ao seguinte:

1. Fazer uma lista dos envolvidos no projeto.
2. Fazer uma reunião informal com todos os interessados.
3. Solicitar para cada interessado uma lista com as características e funções necessárias.
4. Discutir sobre os requisitos e construir 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, necessita-se de um conjunto diferente de tarefas. Tal 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 interessados.
4. Agendar uma série de reuniões facilitadoras para especificação de aplicações.
5. Promover reuniões.
6. Incluir cenários informais de usuários como parte de cada reunião.
7. Aprimorar os cenários de usuários, com base no *feedback* dos interessados.
8. Fazer uma lista revisada das necessidades dos interessados.
9. Empregar técnicas de aplicação de funções de qualidade para estabelecer graus de prioridade dos requisitos.
10. Agrupar os requisitos de modo que eles possam ser entregues incrementalmente.
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 de "levantamento de necessidades", porém, são bem diferentes em relação aos graus de profundidade e formalidade. A equipe de software deve escolher o conjunto de tarefas que lhe possibilitará atingir o objetivo de cada ação, mantendo, inclusive, a qualidade e a agilidade.

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:

PONTO-CHAVE

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

Nome do Padrão. O padrão deve receber um nome significativo 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.

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 múltiplas ações e tarefas de trabalho, um padrão de estágio engloba múltiplos padrões de tarefas (veja o próximo padrão) que são relevantes ao estágio (atividade metodológica). Podemos citar como um exemplo de padrão de estágio **EstabelecendoComunicação.** Esse padrão incorpora o padrão de tarefas **LevantamentodeNecessidades** 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, **LevantamentodeNecessidades** 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]

3 Esses padrões de fases são discutidos na Seção 2.3.3.

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 o estado inicial para o processo? (3) Que informação de engenharia de software ou de projeto já existe?

Por exemplo, o padrão **Planejamento** (um padrão de estágio) requer 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 do projeto, as necessidades básicas do negócio, bem como as restrições do projeto.

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 do padrão de forma bem-sucedida.

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 para o processo?

(3) Quais informações de engenharia de software ou de projeto foram desenvolvidas?

Padrões Relativos. Forneça 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, LevantamentoDeNecessidades, DescriçãoDasRestrições** e **CriaçãoDeCenários.**

INFORMAÇÕES

Um exemplo de padrão de processo

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

Nome do padrão. RequisitosImprecisos

Intento. Este padrão descreve uma abordagem voltada para a construção de um modelo (um protótipo) passível de ser avaliado iterativamente pelos interessados, num esforço para identificar ou solidificar os requisitos de software.

Tipo. Padrão de fase.

Contexto inicial. As condições seguintes devem ser atendidas antes de iniciar esse padrão: (1) interessados identificados; (2) forma de comunicação entre interessados e equipe de software já determinada; (3) principal problema de software a ser resolvido já identificado pelos interessados; (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 um reconhecimento claro de que existe um problema a ser solucionado e este deve ser identificado utilizando-se uma solução de software. Os interessados 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, mas é descrita posteriormente na Seção 2.3.3.

Contexto resultante. Um protótipo de software que identifique os requisitos básicos (por exemplo, modos de interação, características computacionais, funções de processamento) é aprovado pelos interessados. 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 associados. Os seguintes padrões estão relacionados a esse padrão: **ComunicaçãoComOCliente, ProjetoIterativo, DesenvolvimentoIterativo, AvaliaçãoDoCliente, ExtraçãoDeRequisitos.**

Usos conhecidos e um exemplo. A prototipação é recomendada quando as necessidades são incertas.

Usos Conhecidos e Exemplos. Indicam as instâncias específicas onde 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 de emprego estiver em andamento.

WebRef
Recursos completos para padrões de processo podem ser encontrados em **www.ambysoft.com/processPatternsPage.html**.

Padrões de processo propiciam um mecanismo efetivo para localização de problemas associados a qualquer processo de software. Os padrões permitem que se desenvolva uma descrição de 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 para a definição de variantes de 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 de processo.

2.2 Avaliação e Aperfeiçoamento de Processos

PONTO-CHAVE
Tentativas de avaliação para 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 conduzirão a características de qualidade de longo prazo (Capítulos 14 e 16). Os padrões de processo devem ser combinados com uma prática de engenharia de software consistente (Parte 2 deste livro). Além disso, o próprio processo pode ser avaliado para assegurar que está de acordo com um conjunto de critérios de processo básicos comprovados como essenciais para uma engenharia de software de sucesso.[4]

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

? Quais técnicas formais estão disponíveis para avaliar o processo 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].

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].

"As organizações de software apresentaram falhas significativas quanto à habilidade em capitalizar as experiências adquiridas nos projetos finalizados."
Nasa

Uma discussão mais detalhada sobre métodos de avaliação de software e aperfeiçoamento de processo é apresentada no Capítulo 30.

2.3 Modelos de Processo Prescritivo

Originalmente, modelos de processo prescritivo foram propostos para trazer ordem ao caos existente na área de desenvolvimento de software. A história tem demonstrado que esses

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.