

22

GESTÃO DE CONFIGURAÇÃO DE SOFTWARE

CONCEITOS-CHAVE

auditoria de configuração.....	526
controle de alterações.....	524
controle de versão.....	524
gestão de conteúdo.....	530
identificação.....	522
itens de configuração de software (SCIs).....	518
itens de configuração.....	518
objetos de configuração de WebApp.....	529
processo SCM.....	521
referencial.....	516
relatório de status.....	527
repositório.....	519
WebApps.....	528

Mudanças são inevitáveis quando o software é construído. E as mudanças aumentam o nível de confusão entre os membros de uma equipe de software que estão trabalhando em um projeto. A confusão surge quando as mudanças não são analisadas antes de ser feitas, não são registradas antes de ser implementadas, não são relatadas àqueles que precisam saber, ou não são controladas de uma maneira que melhore a qualidade e reduza os erros. Babich [Bab86] discute isso quando afirma:

A arte de coordenar desenvolvimento de software para minimizar a... Confusão é chamada de gestão de configuração. A gestão de configuração é a arte de identificar, organizar e controlar modificações no software que está sendo criado por uma equipe de programação. O objetivo é maximizar a produtividade minimizando os erros.

A gestão de configuração de software (software configuration management – SCM) é uma atividade do tipo “guarda-chuva”, aplicada através de toda a gestão de qualidade. Como as mudanças podem ocorrer em qualquer instante, as atividades SCM são desenvolvidas para (1) identificar a alteração, (2) controlar a alteração, (3) assegurar que a alteração esteja sendo implementada corretamente e (4) relatar as alterações a outros interessados.

É importante fazer uma clara distinção entre suporte de software e gestão de configuração de software. Suporte é um conjunto de atividades de engenharia que ocorrem depois que o software foi fornecido ao cliente e posto em operação. Gestão de configuração é um conjunto de atividades de rastreamento e controle iniciadas quando um projeto de engenharia de software começa e termina apenas quando o software sai de operação.

Um objetivo primário da engenharia de software é incrementar a facilidade com que as alterações podem ser acomodadas e reduzir o esforço necessário quando as alterações

PANORAMA

O que é? Ao se criar software, acontecem mudanças. E por isso, precisamos gerenciá-las eficazmente. A gestão de configuração de software – (SCM), também chamada de gestão de alterações, é um conjunto de atividades destinadas a gerenciar as alterações identificando os artefatos que precisam ser alterados, estabelecendo relações entre eles, definindo mecanismos para gerenciar diferentes versões desses artefatos, controlando as alterações impostas e auditando e relatando as alterações feitas.

Quem realiza? Qualquer um que esteja envolvido na gestão de qualidade está envolvido com a gestão de alterações até certo ponto, mas muitas vezes criam-se posições especializadas de suporte para controlar o processo SCM.

Por que é importante? Se você não controlar as alterações, elas controlarão você. E isso nunca é bom. É muito fácil acontecer de uma sequência de alterações não controladas transformar um bom software em um caos. Como consequência, a qualidade do software é prejudicada e a entrega atrasa. Por essa razão, a gestão de alterações é parte essencial da gestão da qualidade.

Quais são as etapas envolvidas? Como muitos artefatos são produzidos quando o software é criado, cada um deles deve ser identificado de forma única. Feito isso, podem ser estabelecidos mecanismos para controle de versão e alteração. Para assegurar que a qualidade seja mantida quando são feitas alterações, o processo é auditado; e para assegurar que aqueles que precisam saber sobre as alterações sejam informados, são gerados relatórios.

Qual é o artefato? Um plano de gestão de configuração de software define a estratégia de projeto para a gestão das alterações. Além disso, quando é invocada a SCM formal, o processo de controle de alterações produz requisições de alteração de software, relatórios e ordens de alteração de engenharia.

Como garantir que o trabalho foi realizado corretamente? Quando cada artefato pode ser levado em conta, rastreado e controlado; quando todas as alterações podem ser rastreadas e analisadas, quando todos aqueles que precisam saber sobre as alterações já foram informados — você fez tudo certo.

tiverem de ser feitas. Neste capítulo, discutiremos as atividades específicas que lhe permitem gerenciar a alteração.

22.1 GESTÃO DE CONFIGURAÇÃO DE SOFTWARE

O resultado do processo de software são informações que podem ser divididas em três categorias principais: (1) programas de computador (tanto na forma de código-fonte quanto na forma executável), (2) produtos que descrevem os programas de computador (focado em vários interessados) e (3) dados ou conteúdo (contidos nos programas ou externos a ele). Os itens que compõem todas as informações produzidas como parte do processo de software são chamados coletivamente de *configuração de software*.

"Não há nada permanente, exceto as mudanças."
Heráclito,
500 a.C.

À medida que avança o trabalho de engenharia de software, cria-se uma hierarquia de *itens de configuração de software* (*software configuration items* – SCIs) – um elemento de informação com nome, que pode ser tão pequeno quanto um simples diagrama UML ou tão grande quanto um documento de projeto completo. Se cada SCI simplesmente conduzir a outros SCIs, resultará em pouca confusão. Infelizmente, uma outra variável entra no processo – *alteração*. A alteração pode ocorrer a qualquer momento, por qualquer razão. De fato, a *Primeira Lei da Engenharia de Sistemas* [Ber80] diz: "Não importa onde você esteja no ciclo de vida do sistema, o sistema mudará e o desejo de alterá-lo persistirá através de todo o ciclo de vida".

Qual é a origem dessas alterações? A resposta a essa pergunta é variada, assim como as próprias alterações. No entanto, há quatro fontes fundamentais de alterações:

? Qual a origem das alterações solicitadas para o software?

- Novos negócios ou condições de mercado ditam mudanças nos requisitos do produto ou nas regras comerciais.
- Novas necessidades dos interessados demandam modificação dos dados produzidos pelos sistemas de informação, funcionalidade fornecida pelos produtos ou serviços fornecidos por um sistema baseado em computador.
- Reorganização ou crescimento/enxugamento causam alterações em prioridades de projeto ou estrutura de equipe de engenharia de software.
- Restrições orçamentárias ou de cronograma causam a redefinição do sistema ou produto.

A gestão de configuração de software é um conjunto de atividades que foram desenvolvidas para gerenciar alterações através de todo o ciclo de vida de um software. A SCM pode ser vista como uma atividade de garantia de qualidade do software aplicada através de todo o processo do software. Nas seções a seguir, descrevem-se as principais tarefas da SCM e conceitos importantes que podem ajudá-lo a gerenciar as alterações.

22.1.1 Um cenário SCM¹

? Quais os objetivos e as atividades executadas pelas divisões envolvidas na gestão de alterações?

Um típico cenário operacional CM envolve um gerente de projeto encarregado de um grupo de software, um gerente de configuração encarregado dos procedimentos e políticas CM, os engenheiros de software responsáveis pelo desenvolvimento e manutenção do artefato e o cliente que usa o produto. No cenário, suponha que o produto seja um item pequeno envolvendo aproximadamente 15 mil linhas de código que está sendo desenvolvido por uma equipe de seis pessoas. (Note que são possíveis outros cenários com equipes menores ou maiores, mas, essencialmente, há problemas genéricos que cada um desses projetos enfrenta em relação à CM.)

No nível operacional, o cenário envolve vários papéis e tarefas. Para o gerente de projeto, o objetivo é garantir que o produto seja desenvolvido em certo prazo. O gerente monitora o progresso do desenvolvimento e reconhece e reage aos problemas. Isso é feito gerando e analisando relatórios sobre o estado do sistema de software e fazendo revisões no sistema.

¹ Esta seção foi extraída de [Dar01]. A permissão especial para reproduzir "Gama de Funcionalidade no Sistema CM" por Susan Dart [Dar01], © 2001 pelo Carnegie Mellon University, foi concedida pelo Software Engineering Institute.

PONTO-CHAVE

Deve haver um mecanismo para assegurar que alterações simultâneas ao mesmo componente sejam adequadamente rastreadas, gerenciadas e executadas.

As metas do gerente de configuração são garantir que sejam seguidos os procedimentos e políticas para criar, alterar e testar o código, bem como tornar acessíveis as informações sobre o projeto. Para implementar técnicas para manter controle sobre as mudanças de código, esse gerente introduz mecanismos para fazer solicitações oficiais de alterações, para avaliá-las (através de um Grupo de Controle de Alterações que é responsável pela aprovação das alterações do sistema) e para autorizar as alterações. O gerente cria e distribui listas de tarefas para os engenheiros e basicamente cria o contexto do projeto. Além disso, o gerente coleta dados estatísticos sobre os componentes do sistema de software, como, por exemplo, informações determinando que componentes do sistema são problemáticos.

Para os engenheiros de software, o objetivo é trabalhar eficazmente. Isso significa que os engenheiros não interferem uns com os outros de forma desnecessária na criação e teste do código e na produção de artefatos de suporte. Mas, ao mesmo tempo, eles tentam se comunicar e coordenar eficientemente. Os engenheiros usam ferramentas que ajudam a criar artefatos consistentes. Eles se comunicam e se coordenam notificando uns aos outros sobre as tarefas necessárias e as tarefas completadas. As alterações são propagadas por meio do trabalho dos outros mesclando arquivos. Existem mecanismos que asseguram que, para componentes submetidos a alterações simultâneas, há uma maneira de resolver conflitos e mesclar alterações. É mantido um histórico da evolução de todos os componentes do sistema juntamente com um registro (log) com as razões para as alterações e um registro do que realmente foi alterado. Os engenheiros têm seu próprio espaço de trabalho para criar, alterar, testar e integrar o código. Em certo ponto, o código é transformado em referencial com base no qual o desenvolvimento continua e por meio do qual são criadas variações para outras máquinas.

O cliente usa o produto. Como o produto está sob o controle da Gestão de Configuração (CM), o cliente segue os procedimentos formais para solicitar alterações e para indicar *bugs* no produto.

No caso ideal, um sistema de CM usado nesse cenário deveria suportar todos esses papéis e tarefas; isto é, os papéis determinam a funcionalidade requerida para um sistema de CM. O gerente de projeto vê a CM como um mecanismo de auditoria; o gerente de configuração a vê como um mecanismo de controle, rastreamento e criador de políticas; o engenheiro de software a vê como um mecanismo de alteração, criação e controle de acesso; e o cliente a vê como um mecanismo de garantia de qualidade.

22.1.2 Elementos de um sistema de gestão de configuração

Em sua publicação sobre gestão de configuração de software, Susan Dart [Dar01] identifica quatro importantes elementos que devem existir quando é desenvolvido um sistema de gestão de configuração:

- *Elementos de componente* — conjunto de ferramentas acopladas em um sistema de gestão de arquivos (por exemplo, um banco de dados) que possibilita acesso à gestão de cada item de configuração de software.
- *Elementos de processo* — coleção de ações e tarefas que definem uma abordagem eficaz da gestão de alterações (e atividades relacionadas) para todas as partes envolvidas na gestão, engenharia e uso do software.
- *Elementos de construção* — conjunto de ferramentas que automatizam a construção do software, assegurando que tenha sido montado o conjunto apropriado de componentes validados (isto é, a versão correta).
- *Elementos humanos* — conjunto de ferramentas e características de processo (abrangendo outros elementos de CM) usados pela equipe de software para implementar uma SCM eficaz.

Esses elementos (discutidos com mais detalhes em seções posteriores) não são mutuamente exclusivos. Por exemplo, elementos de componente funcionam em conjunto com elementos de cons-

trução à medida que evolui o processo do software. Elementos de processo guiam muitas atividades humanas relacionadas à SCM e podem, portanto, ser consideradas elementos humanos também.

22.1.3 Referenciais



Muitas alterações de software são justificadas, portanto não tem sentido reclamar delas. Em vez disso, esteja certo de ter os mecanismos prontos para cuidar delas.

Alteração é um fato normal no desenvolvimento de software. Clientes querem modificar requisitos. Desenvolvedores querem modificar a abordagem técnica. Gerentes querem modificar a estratégia do projeto. Por que todas essas modificações? A resposta é realmente muito simples.

À medida que o tempo passa, todas as partes envolvidas sabem mais (sobre o que eles precisam, qual será a melhor abordagem e como conseguir que seja feito e ainda ganhar dinheiro). Esse conhecimento adicional é a força motora que está por trás da maioria das alterações e leva à constatação de um fato que para muitos profissionais de engenharia de software é difícil de aceitar: *Muitas alterações são justificadas!*

Uma referência é um conceito de gestão de configuração de software que o ajuda a controlar alterações sem obstruir seriamente as alterações justificáveis. O IEEE (IEEE Std. No. 610.12-1990) define uma referência como:

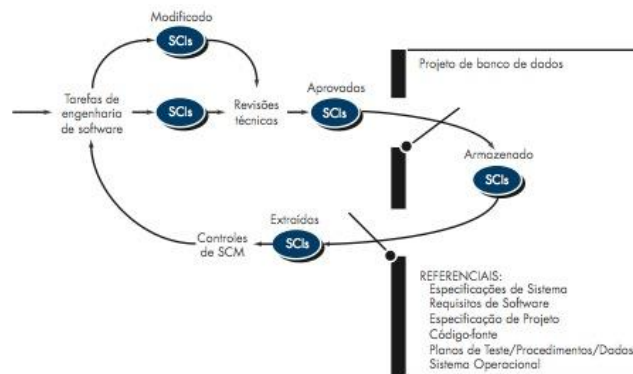
Uma especificação ou produto que tenha sido formalmente revisado e acordado, que depois serve de base para mais desenvolvimento, e pode ser alterado somente por meio de procedimentos formais de controle de alteração.

Para que um item de configuração de software se torne uma referência para o desenvolvimento, devem ser feitas alterações rápida e informalmente. No entanto, uma vez estabelecida uma referência, podem ser feitas alterações, mas deve ser aplicado um processo específico e formal para avaliar e verificar cada alteração.

No contexto de engenharia de software, uma referência é um marco no desenvolvimento de software. Uma referência é marcada pelo fornecimento de um ou mais itens de configuração de software que foram aprovados em consequência de uma revisão técnica (Capítulo 15). Por exemplo, os elementos de um modelo de projeto foram documentados e revisados. Erros foram encontrados e corrigidos. Uma vez que todas as partes do modelo foram revisadas, corrigidas e então aprovadas, o modelo do projeto torna-se uma referência. Outras alterações na arquitetura do programa (documentadas no modelo de projeto) podem ser feitas apenas depois que cada uma tenha sido avaliada e aprovada. Embora as referências possam ser definidas em qualquer nível de detalhe, as referências de software mais comuns estão na Figura 22.1.

FIGURA 22.1

SCIs que se tornaram referenciais e o banco de dados de projeto





AVISO
Certifique-se de que o banco de dados de projeto seja mantido em uma localização centralizada e controlada.

A sequência de eventos que levam a uma referência também está ilustrada na Figura 22.1. Tarefas de engenharia de software produzem uma ou mais SCIs. Depois que as SCIs são revisadas e aprovadas, são colocadas em um *banco de dados de projeto* (também chamado de *biblioteca de projeto* ou *repositório de software* e discutidos na Seção 22.2). Quando um membro de uma equipe de engenharia de software quer fazer uma modificação em uma SCI que se tornou referencial, ela é copiada do banco de dados de projeto para o espaço de trabalho privado do engenheiro. Porém, essa SCI extraída só pode ser modificada se os controles de SCM (discutidos mais adiante neste capítulo) forem seguidos. As setas na Figura 22.1 ilustram o caminho de modificação para uma SCI referencial.

22.1.4 Itens de configuração de software

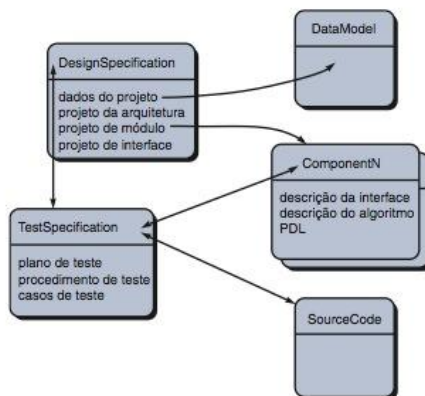
Já definimos um item de configuração de software como informação criada como parte do processo de engenharia de software. No caso extremo, uma SCI poderia ser considerada uma única seção de uma grande especificação ou um caso de teste em um grande conjunto de testes. Mais realisticamente, uma SCI é todo ou parte de um artefato (por exemplo, um documento, um conjunto inteiro de casos de teste ou um programa ou componente com nome).

Além das SCIs derivadas dos artefatos de software, muitas organizações de engenharia de software também colocam ferramentas de software sob o controle de configuração. Isto é, versões específicas de editores, compiladores, browsers e outras ferramentas automáticas são "congeladas" como parte da configuração do software. Como essas ferramentas foram usadas para produzir documentação, código-fonte e dados, elas devem estar disponíveis quando alterações forem feitas na configuração do software. Embora os problemas sejam raros, é possível que uma nova versão de uma ferramenta (por exemplo, um compilador) possa produzir resultados diferentes daqueles da versão original. Por essa razão, as ferramentas, assim como o software que elas ajudam a produzir, podem ser referenciadas como parte de um processo bem especificado de gestão de configuração.

Na realidade, as SCIs são organizadas para formar objetos de configuração que podem ser catalogados no banco de dados do projeto com um nome único. Um *objeto de configuração* tem um nome, atributos e é "conectado" a outros objetos por relações. De acordo com a Figura 22.2, os objetos de configuração, **DesignSpecification**, **DataModel**, **ComponentN**, **SourceCode** e **TestSpecification** são definidos separadamente. No entanto, cada um dos objetos está relacionado com os outros, como mostram as setas. Uma seta curva indica uma relação de composição. Isto é, **DataModel** e **ComponentN** são parte do objeto **DesignSpecification**. Uma seta

FIGURA 22.2

Objetos de configuração



reta bidirecional indica uma inter-relação. Se for feita uma alteração no objeto **SourceCode**, as inter-relações lhe permitem determinar que outros objetos (e SCIs) podem ser afetados.²

22.2 O REPOSITÓRIO SCM

Nos primórdios da engenharia de software, os itens de configuração eram mantidos na forma de documentos impressos (ou cartões perfurados!), colocados em pastas de arquivos ou naquelas pastas de três furos, e armazenados em armários de aço. Essa abordagem era problemática por várias razões: (1) encontrar um item de configuração quando necessário era muitas vezes difícil, (2) determinar quais itens foram alterados, quando e por quem, era em geral um desafio, (3) criar uma nova versão de um programa existente era um processo demorado e sujeito a erros e (4) descrever relações detalhadas ou complexas entre itens de configuração era praticamente impossível.

Atualmente, as SCIs são mantidas em um banco de dados de projeto ou repositório. O Dicionário Webster define a palavra *repository* (repositório) como “qualquer coisa ou pessoa considerada um centro de acumulação ou armazenagem”. Nos primeiros tempos da engenharia de software, o repositório era sem dúvida uma pessoa — o programador que tinha de se lembrar da localização de todas as informações relevantes a um projeto de software, aquele que tinha de se lembrar de todas as informações que nunca foram escritas e reconstruir informações perdidas. Infelizmente, o uso de uma pessoa como “centro de acumulação e armazenagem” (embora esteja de acordo com a definição *Webster's Dictionary*) não funciona muito bem. Hoje, o repositório é uma “coisa” — um banco de dados que age como o centro de acumulação e de armazenagem de informações de engenharia de software. O papel da pessoa (o engenheiro de software) é interagir com o repositório usando ferramentas integradas com ele.

22.2.1 O papel do repositório

O repositório de SCM é um conjunto de mecanismos e estruturas de dados que permitem a uma equipe de software gerenciar alterações de maneira eficaz. Ele proporciona as funções óbvias de um sistema moderno de gestão de banco de dados garantindo a integridade dos dados, compartilhamento e integração. Além disso, o repositório de SCM proporciona um centralizador (*hub*) para a integração das ferramentas de software, está no centro do fluxo do processo de software e pode impor estrutura e formato uniformes para os artefatos.

Para tanto, o repositório é definido em termos de metamodelo. O *metamodelo* determina como as informações são armazenadas no repositório, como os dados podem ser acessados pelas ferramentas e visualizados pelos engenheiros de software, quão bem pode ser mantida a segurança e a integridade dos dados e com que facilidade o modelo existente pode ser estendido para satisfazer a novas necessidades.

22.2.2 Características gerais e conteúdo

As características e o conteúdo do repositório são mais bem compreendidas quando são observadas a partir de duas perspectivas: o que tem de ser armazenado no repositório e quais os serviços específicos que são fornecidos pelo repositório. Na Figura 22.3 está uma divisão detalhada dos tipos de representações, documentos e outros produtos que são armazenados no repositório.

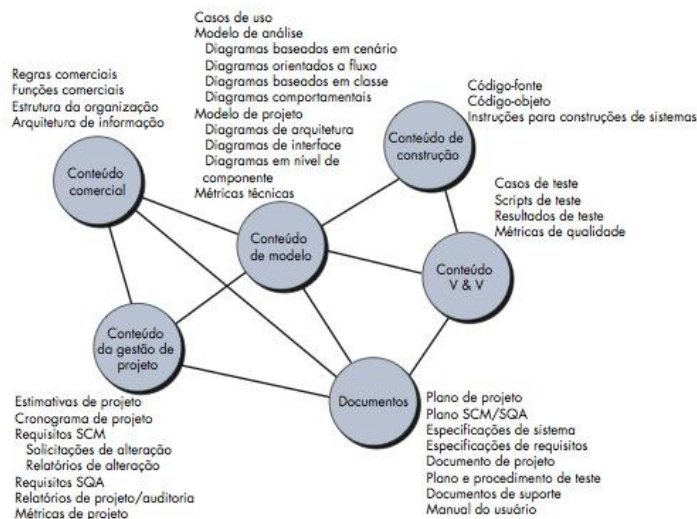
Um repositório robusto proporciona duas classes diferentes de serviços: (1) os mesmos tipos de serviços que podem ser esperados de qualquer sistema sofisticado de gerenciamento de banco de dados (2) serviços que são específicos ao ambiente de engenharia de software.

Um repositório que sirva a uma equipe de engenharia de software deve também ter as seguintes características (1) integrar com ou suportar diretamente as funções de gestão de processo,

WebRef

Um exemplo de um repositório disponível comercialmente pode ser obtido em www.oracle.com/technology/products/repository/index.html.

² Essas relações são definidas no banco de dados. A estrutura do banco de dados (repositório) é discutida em mais detalhes na Seção 22.2.

FIGURA 22.3**Conteúdo do repositório**

(2) suportar regras específicas que governam a função SCM e os dados mantidos no repositório, (3) proporcionar uma interface para outras ferramentas de engenharia de software, e (4) acomodar o armazenamento de objetos de dados sofisticados (por exemplo, texto, gráficos, vídeo, áudio).

22.2.3 Características SCM

Para suportar a SCM, o repositório deve ter um conjunto de ferramentas que proporcione suporte para as seguintes características:

Versões. À medida que um projeto avança, serão criadas muitas versões (Seção 22.3.2) dos artefatos individuais. O repositório deve ser capaz de salvar todas essas versões para possibilitar uma gestão eficaz das versões do produto e permitir aos desenvolvedores retroceder a versões anteriores durante o teste e depuração.

O repositório deve ser capaz de controlar uma grande variedade de tipos de objetos, incluindo texto, gráficos, bitmaps, documentos complexos e objetos especiais como definições de tela e relatório, arquivos de objeto, dados de testes e resultados. Um repositório desenvolvido rastreia versões de objetos com níveis arbitrários de granularidade; por exemplo, podem ser rastreados uma definição de dados especial ou um conjunto de módulos.

Acompanhamento de dependências e gestão de alterações. O repositório gerencia uma grande variedade de relações entre os elementos de dados armazenados nele. Isso inclui relações entre entidades e processos corporativos, entre as partes do projeto de uma aplicação, entre componentes de projeto e arquitetura de informações corporativas, entre elementos de projeto e outros artefatos, e assim por diante. Algumas dessas relações são meramente associações e outras são relações de dependências ou de obrigatoriedade.

PONTO-CHAVE

O repositório deve ser capaz de manter SCIs relacionadas com muitas versões diferentes do software. Mais importante ainda, deve proporcionar os mecanismos para montagem desses SCIs em uma configuração específica de versão.

A habilidade em manter controle de todas essas relações é crucial para a integridade das informações armazenadas no repositório e para a geração de outros produtos baseados nele e é uma das contribuições mais importantes do conceito de repositório para o aperfeiçoamento do processo de software. Por exemplo, se um diagrama de classe UML é modificado, o repositório pode detectar se as classes relacionadas, as descrições de interface e os componentes de código também requerem modificações e podem chamar a atenção do desenvolvedor para as CSIs afetadas.

Controle de requisitos. Essa função especial depende da gestão de link e proporciona a habilidade para controlar todos os componentes de projeto e construção e outros produtos que resultam de uma especificação especial de requisitos (acompanhamento adiante). Além disso, ela proporciona a habilidade para identificar que requisitos geraram determinado produto (retroacompanhamento).

Gestão de configuração. O recurso de gestão de configuração mantém controle de uma série de configurações representando marcos de projeto específico ou versões de produção.

Pistas de auditoria. Uma pista de auditoria estabelece informações adicionais sobre quando, por que e por quem foram feitas as alterações. As informações sobre a origem das alterações podem ser colocadas como atributos de objetos específicos no repositório. Um mecanismo de disparo do repositório é útil para avisar o desenvolvedor ou a ferramenta que está sendo usada para iniciar a aquisição de informações de auditoria (como, por exemplo, a razão de uma alteração) sempre que um elemento de projeto for modificado.

22.3 O PROCESSO SCM

“Qualquer alteração, mesmo uma alteração para melhor, é acompanhada de contratempos e desconfortos.”

Arnold Bennett

Que questões o processo SCM está designado a responder?

O processo de gestão de configuração de software define uma série de tarefas que têm quatro objetivos primários: (1) identificar todos os itens que coletivamente definem a configuração do software, (2) gerenciar alterações de um ou mais desses itens, (3) facilitar a construção de diferentes versões de uma aplicação e (4) assegurar que a qualidade do software seja mantida à medida que a configuração evolui com o tempo.

Um processo que atinja esses objetivos não precisa ser burocrático ou pesado, mas deve ser caracterizado de maneira que permita a equipe de software desenvolver respostas a uma série de questões complexas:

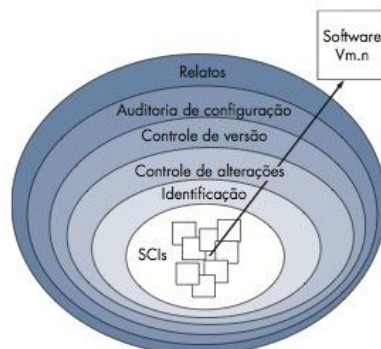
- Como uma equipe de software identifica os elementos discretos de uma configuração de software?
- Como uma organização lida com as várias versões de um programa (e sua documentação) de maneira que venha a permitir que a alteração seja acomodada eficientemente?
- Como uma organização controla alterações antes e depois que o software é entregue ao cliente?
- Quem tem a responsabilidade de aprovar e classificar as alterações solicitadas?
- Como podemos assegurar que as alterações foram feitas corretamente?
- Que mecanismo é usado para alertar outras pessoas sobre as alterações que são feitas?

Essas questões levam à definição de cinco tarefas SCM — identificação, controle de versão, controle de alteração, auditoria de configuração e relatos — ilustradas na Figura 22.4.

De acordo com a figura, as tarefas SCM podem ser vistas como camadas concêntricas. As SCIs fluem para fora através dessas camadas por toda a sua vida útil, tornando-se finalmente parte da configuração do software de uma ou mais versões da aplicação ou sistema. À medida que uma SCI se move através de uma camada, as ações deduzidas por cada tarefa SCM podem ser ou não aplicáveis. Por exemplo, quando uma nova SCI é criada, ela deve ser identificada. No entanto, se não forem solicitadas alterações para a SCI, a camada de controle de alteração não se aplica. A SCI é atribuída a uma versão específica do software (aqui entram em ação os mecanismos de controle de versão). É mantido um registro da SCI (seu nome, data de criação,

FIGURA 22.4

Camadas do processo SCM



designação da versão etc.) para fins de auditoria da configuração e relato para aqueles que precisam ter conhecimento. Nas próximas seções, examinaremos cada uma dessas camadas de processo SCM em mais detalhes.

22.3.1 Identificação de objetos na configuração de software

Para controlar e gerenciar itens de configuração de software, cada um deles deverá ser nomeado separadamente e depois organizado usando uma abordagem orientada a objeto. Podem ser identificados dois tipos de objeto [Cho89]: objetos básicos e objetos agregados.³ O *objeto básico* é uma unidade de informação que se cria durante a análise, projeto, codificação ou teste. Por exemplo, o objeto básico pode ser uma seção de especificação de requisitos, parte de um modelo de projeto, código-fonte para um componente ou um conjunto de casos de teste usados para exercitar o código. O *objeto agregado* é uma coleção de objetos básicos e outros objetos agregados. Por exemplo, uma **DesignSpecification** é um objeto agregado. Conceitualmente, ela pode ser vista como uma lista nomeada (identificada) de ponteiros que especificam objetos agregados como, por exemplo, **ArchitecturalModel** e **DataModel**, e *objetos básicos* como **ComponentN** e **UMLClassDiagramN**.

Cada objeto tem um conjunto de características distintas que o identificam de forma única: um nome, uma descrição, uma lista de recursos e uma “realização”. O nome do objeto é uma sequência de caracteres que o identifica de forma não ambígua. A descrição do objeto é uma lista de itens de dados que identifica o tipo de SCI (por exemplo, elemento de modelo, programa, dados) representado pelo objeto, um identificador de projeto e informações sobre alteração e/ou versão. Recursos são “entidades fornecidas, processadas, referenciadas ou de qualquer outra forma requeridas pelo objeto” [Cho89]. Por exemplo, tipos de dados, funções específicas ou mesmo nomes de variáveis podem ser considerados recursos de objeto. A realização pode ser um ponteiro para a “unidade de texto”, para um objeto básico null ou para um objeto agregado.

A identificação do objeto de configuração pode também considerar as relações entre objetos nomeados. Por exemplo, usando a notação simples

```
Class diagram <part-of> requirements model;
Requirements model <part-of> requirements specification;
```

você pode criar uma hierarquia de SCIs.

PONTO-CHAVE

As interrelações estabelecidas para objetos de configuração lhe permitem avaliar o impacto da alteração.

³ O conceito de objeto agregado [Gus89] tem sido proposto como mecanismo para representar uma versão completa de uma configuração de software.



Mesmo que o banco de dados do projeto tenha habilidade para estabelecer essas relações, elas demoram para se estabelecer e são difíceis de manter atualizadas. Embora muito úteis para análise de impacto, não são essenciais para a gestão geral das alterações.

Em muitos casos, objetos são inter-relacionados através de ramificações da hierarquia do objeto. Essas relações que cruzam a estrutura podem ser representadas da seguinte maneira:

```
DataModel <interrelated> DataFlowModel
DataModel <interrelated> TestCaseClassM
```

No primeiro caso, a inter-relação é entre um objeto composto, enquanto a segunda relação é entre um objeto agregado (**DataModel**) e um objeto básico (**TestCaseClassM**).

O esquema de identificação para objetos de software deve reconhecer que objetos evoluem através do processo de software. Antes que um objeto seja um referencial, ele pode mudar muitas vezes, e mesmo após um referencial ter sido estabelecido, as mudanças podem ser muito frequentes.

22.3.2 Controle de versão

O controle de versão combina procedimentos e ferramentas para gerenciar diferentes versões dos objetos de configuração criados durante o processo de software. Um sistema de controle de versão implementa ou está diretamente integrado com quatro recursos principais: (1) um banco de dados de projeto (repositório) que armazena todos os objetos de configuração relevantes, (2) um recurso de *gestão de versão* que armazena todas as versões de um objeto de configuração (ou permite que qualquer versão seja construída usando diferenças das versões anteriores), (3) uma facilidade de construir que permite coletar todos os objetos de configuração relevantes e construir uma versão específica do software. Além disso, os sistemas de controle de versão e controle de alteração muitas vezes implementam um recurso chamado acompanhamento de tópicos (também conhecido como acompanhamento de *bug*), que permite à equipe de software registrar e acompanhar o status de todos os problemas pendentes associados com cada objeto de configuração.

Alguns sistemas de controle de versão criam um conjunto de modificações — uma coleção de todas as alterações (em relação a alguma configuração referencial) que são necessárias para criar uma versão específica do software. Dart [Dar91] observa que um conjunto de modificações “captura todas as alterações a todos os arquivos na configuração juntamente com a razão para aquelas alterações e os detalhes de quem as fez e quando”.

Alguns conjuntos de modificações que receberam denominação podem ser identificados para uma aplicação ou sistema. Isso permite construir uma versão do software especificando os conjuntos de modificações (pelo nome) que devem ser aplicados à configuração referencial. Para isso, aplica-se uma abordagem de *modelagem de sistema*. O modelo de sistema contém: (1) um gabarito que inclui hierarquia de componentes e uma “ordem de construção” para os componentes descrevendo como o sistema deve ser construído, (2) regras de construção e (3) regras de verificação.⁴

Durante as últimas décadas foram propostas muitas abordagens automáticas diferentes para o controle de versão. A diferença primária entre as abordagens é a sofisticação dos atributos usados para construir versões específicas e variantes de um sistema e os mecanismos do processo de construção.

Mesmo com essas limitações, o CVS “é um sistema de controle de versão predominante de código aberto transparente à rede [que] é útil para qualquer um, desde desenvolvedores individuais até grandes equipes distribuídas” [CVS07]. Sua estrutura cliente-servidor permite aos usuários acessar arquivos via conexões na Internet, e sua filosofia de código aberto o torna disponível às plataformas mais populares.

O CVS está disponível sem custo para ambientes Windows, Mac OS, LINUX e UNIX. Ver [CVS07] para mais detalhes.

⁴ É possível também consultar o modelo do sistema para avaliar como uma alteração em um componente afeta outros componentes.

FERRAMENTAS DO SOFTWARE

**O sistema de versões concorrentes (concurrent versions system – CVS)**

O uso de ferramentas para deter o controle de versão é essencial para uma gestão eficaz das alterações. O sistema de versões concorrentes (CVS) é uma ferramenta largamente utilizada para controle de versão. Projetada originalmente para código-fonte, mas útil para qualquer arquivo baseado em texto, o sistema CVS (1) estabelece um repositório simples, (2) mantém todas as versões de um arquivo sob um único nome de arquivo, armazenando apenas as diferenças entre versões progressivas do arquivo original e (3) protege contra

alterações simultâneas de um arquivo, estabelecendo diferentes diretórios para cada desenvolvedor, isolando assim uns dos outros. O CVS mescla as alterações quando cada desenvolvedor completa seu trabalho.

É importante notar que o CVS não é um sistema de construção; ele não constrói uma versão específica do software. Outras ferramentas (por exemplo, *Makefile*) devem ser integradas ao CVS para conseguir isso. O CVS não implementa um processo de controle de alteração (por exemplo, solicitações de alterações, relatos de alterações, acompanhamento de bugs).

22.3.3 Controle de alterações

A realidade do controle de alterações em um moderno contexto de engenharia de software foi resumida elegantemente por James Bach [Bac98]:

O controle de alterações é vital. Mas as forças que o tornam necessário também o tornam inconveniente. Temos medo das alterações porque uma pequena perturbação no código pode criar uma enorme falha no produto. Mas elas podem também reparar uma grande falha ou habilitar novos e maravilhosos recursos. Temos medo das alterações porque um único desenvolvedor irresponsável poderia afundar o projeto todo; embora ideias brilhantes possam surgir nas mentes desses brincalhões, um controle de processo de alterações pesado poderia efetivamente desencorajá-los no seu trabalho criativo.

Bach reconhece que temos aqui uma lei de equilíbrio. Se tivermos muito controle das alterações, criaremos problemas. Se tivermos pouco controle, criaremos outros problemas.

Em um grande projeto de software, alterações não controladas levam rapidamente ao caos. Para projetos assim, o controle de alterações combina procedimentos humanos e ferramentas automatizadas, proporcionando um mecanismo para o controle de alterações. O processo de controle de alterações está ilustrado esquematicamente na Figura 22.5. Uma *solicitação de alteração* é apresentada e avaliada para determinar o mérito técnico, efeitos colaterais potenciais, impacto global sobre outros objetos de configuração e funções do sistema e o custo projetado da alteração. Os resultados da avaliação são apresentados como um *relatório de alterações*, usado por uma *autoridade de controle de alterações* (*change control authority* — CCA) — uma pessoa ou grupo de pessoas que toma a decisão final sobre o status e prioridade da alteração. Uma *ordem de alteração de engenharia* (*engineering change order* — ECO) é gerada para cada alteração aprovada. A ECO descreve a alteração a ser feita, as restrições que devem ser respeitadas e o critério para revisar e auditar.

Os objetos a ser alterados podem ser colocados em um diretório que é controlado apenas pelo engenheiro de software que está fazendo a alteração. Um sistema de controle de versão (ver o quadro *Ferramentas de software* sobre CVS) atualiza o arquivo original logo que a alteração foi feita. Como alternativa, os objetos a ser alterados podem ser “retirados” do banco de dados do projeto (repositório), as alterações ser feitas e aplicadas às atividades SQA apropriadas. Os objetos são então “colocados” no banco de dados e usados mecanismos de controle de versão apropriados (Seção 22.3.2) para criar a próxima versão do software.

Esses mecanismos de controle de versão, integrados ao processo de controle de alterações, implementam dois elementos importantes da gestão de alterações — controle de acesso e controle de sincronização. O *controle de acesso* controla quais os engenheiros de software têm autoridade para acessar e modificar um objeto de configuração particular. O *controle de sincronização* ajuda a assegurar que alterações paralelas, executadas por duas pessoas diferentes, não sobrescrevam uma à outra.

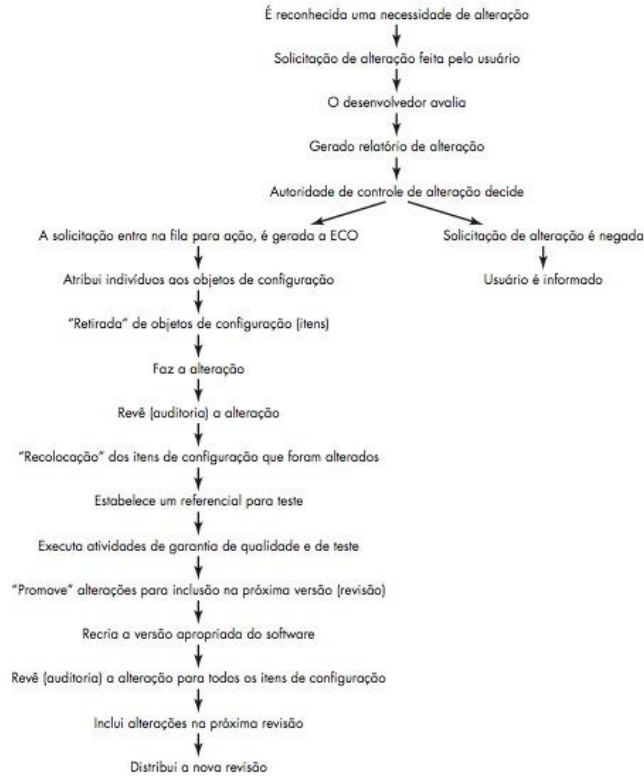
Você pode se sentir incomodado pelo nível de burocracia gerado pela descrição do processo de controle de alteração mostrado na Figura 22.5. Essa sensação não é incomum. Sem as

“A arte do progresso é preservar a ordem nas alterações e preservar as alterações na ordem.”

Alfred North Whitehead

PONTO-CHAVE

Deve-se observar que muitas solicitações de alterações podem ser combinadas para resultar em uma única ECO e que ECOs resultam tipicamente em alterações a múltiplos objetos de configuração.

FIGURA 22.5**O processo de controle de alterações**

condições de segurança apropriadas, o controle de alterações pode retardar o progresso e criar barreiras desnecessárias. Grande parte dos desenvolvedores de software que usam mecanismos de controle de alterações (infelizmente, muitos não usam nenhum) já criou uma série de camadas de controle para ajudar a evitar os problemas mencionados aqui.



AVISO
 Opte por um pouco mais de controle de alterações do que você acha que precisará. É provável que a dose certa seja bem maior.

Antes de uma SCI se tornar um referencial, só é necessário usar o *controle informal de alteração*. O desenvolvedor do objeto de configuração (SCI) em questão pode fazer todas as alterações justificáveis pelo projeto e pelos requisitos técnicos (desde que as alterações não afetem requisitos mais amplos do sistema que estejam fora do escopo de trabalho do desenvolvedor). Uma vez que o objeto tenha passado pela revisão técnica e tenha sido aprovado, pode ser criado um referencial.⁵ Uma vez que uma SCI se torna um referencial, é implementado o *controle de alterações em nível de projeto*. Agora, para fazer uma alteração, o desenvolvedor precisa ter novamente a aprovação do gerente de projeto (se a alteração for "local") ou da CCA se a alteração afetar outras SCIs. Em alguns casos, são necessárias solicitações formais de geração de altera-

5 Um referencial pode ser criado por outras razões também. Por exemplo, quando são criadas "construções diárias", todos os componentes verificados por determinado tempo se tornam o referencial para o trabalho do dia seguinte.

ções, relatórios de alterações e ECOS. No entanto, é feita a avaliação de cada alteração e todas as alterações são acompanhadas e revisadas.

Quando o artefato de software é liberado para os clientes, institui-se o *controle formal de alterações*. O procedimento formal de controle de alterações foi resumido na Figura 22.5.

A autoridade de controle de alterações desempenha um papel ativo no segundo e terceiro níveis de controle. Dependendo do tamanho e do tipo de projeto de software, a CCA pode ser composta por uma pessoa — o gerente de projeto — ou um grupo de pessoas (por exemplo, representantes do software, hardware, engenharia do banco de dados, suporte, marketing). O papel da CCA é assumir uma visão global, isto é, avaliar o impacto das alterações além da SCI em questão. Como a alteração afetará o hardware? Como a alteração afetará o desempenho? Como a alteração modificará a percepção do cliente em relação ao produto? Como a alteração afetará a qualidade e a confiabilidade do produto? Essas e muitas outras questões são resolvidas pela CCA.

22.3.4 Auditoria de configuração

Identificação, controle de versão e controle de alterações ajudam a manter a ordem naquilo que de outra forma seria uma situação caótica. No entanto, mesmo os melhores mecanismos de controle rastreiam uma alteração somente até que seja gerada uma ECO. Como a equipe de software pode assegurar que a alteração foi implementada corretamente? A resposta é dupla: (1) revisões técnicas e (2) a auditoria de configuração de software.

A revisão técnica (Capítulo 15) focaliza a exatidão técnica do objeto de configuração modificado. Os revisores avaliam a SCI para determinar a consistência com outras SCIs, omissões ou efeitos colaterais potenciais. Deverá ser feita uma revisão técnica para todas as alterações, exceto as mais triviais.

Uma *auditoria de configuração de software* complementa a revisão técnica avaliando o objeto de configuração quanto a características que em geral não são consideradas durante a revisão. A auditoria propõe e responde as seguintes questões:

“O troco é inevitável, exceto para as máquinas automáticas de refrigerantes.”

Adesivo em para-choque

CASA SEGURA



Problemas SCM

Cena: Escritório de Doug Miller no início do projeto de software CasaSegura.

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

Conversa:

Doug: Eu sei que ainda é cedo para isso, mas precisamos falar sobre gestão de alterações.

Vinod (rindo): Dificilmente. O pessoal de marketing ligou esta manhã e eles tinham algumas “segundas intenções”. Nada importante, mas é só o começo.

Jamie: Nós fomos muito informais a respeito de gestão de alterações em projetos anteriores.

Doug: Eu sei, mas este é maior e mais visível, e pelo que me lembro...

Vinod (balançando a cabeça): Nos matamos por alterações descontroladas no projeto de controle de luz ambiental... Lembra-me dos atrasos...

Doug (franzindo a testa): Um pesadelo que eu prefiro não lembrar.

Jamie: Então o que fazemos?

Doug: Penso que devemos fazer três coisas. Primeiro temos que desenvolver — ou tomar emprestado — um processo de controle de alterações.

Jamie: Você quer dizer, o modo como as pessoas solicitam alterações?

Vinod: Sim, mas também como avaliamos a alteração, como decidimos quem faz (se é que nós decidimos sobre isso) e como mantemos os registros do que é afetado pela alteração.

Doug: Em segundo lugar, precisamos realmente arranjar uma boa ferramenta de SCM para controle de versão e alteração.

Jamie: Podemos criar um banco de dados para todos os nossos artefatos.

Vinod: Nesse contexto, elas são chamadas de SCIs, e há muitas boas ferramentas que proporcionam suporte para isso.

Doug: É um bom começo, agora temos que...

Jamie: Ei, Doug, você disse que eram três coisas...

Doug (sorrindo): Terceiro — todos nós temos que seguir os processos de gestão de alterações e usar as ferramentas — não importa o quê, OK?

Quais as perguntas primárias que fazemos durante uma auditoria de configuração?

1. Foi feita a alteração especificada na ECO? Alguma modificação adicional foi incorporada?
2. Foi feita uma revisão técnica para avaliar a exatidão técnica?
3. Seguiu-se o processo do software e os padrões de engenharia de software foram aplicados adequadamente?
4. A alteração foi "destacada" no SCI? A data e o autor da alteração foram especificados? Os atributos do objeto de configuração refletem a alteração?
5. Seguiram-se os procedimentos da SCM para anotar a alteração, registrá-la e relatá-la?
6. Todos os SCIs relacionados foram adequadamente atualizados?

Em alguns casos, as perguntas de auditoria são formuladas como parte da revisão técnica. No entanto, quando a SCM é uma atividade formal, a auditoria de configuração é conduzida separadamente pelo grupo de garantia de qualidade. Essas auditorias de configuração formais também asseguram que os SCIs corretos (por versão) tenham sido incorporados em uma construção específica e que toda a documentação esteja atualizada e consistente com a versão construída.

22.3.5 Relatório de status

O relatório de status de configuração (às vezes chamado de contabilidade de status) é uma tarefa da SCM que responde às seguintes questões: (1) O que aconteceu? (2) Quem fez? (3) Quando aconteceu? (4) O que mais será afetado?

O fluxo de informações para o relatório de status de configuração (CSR) está ilustrado na Figura 22.5. A cada vez que a um SCI é atribuída uma identificação nova ou atualizada, faz-se uma entrada no CSR. Cada vez que uma alteração é aprovada pela CCA (isto é, é gerada uma ECO), é feita uma entrada no CSR. Cada vez que se executa uma auditoria de configuração, os resultados são relatados como parte da tarefa do CSR. A saída do CSR pode ser colocada em um banco de dados on-line ou em um site, de forma que os desenvolvedores de software ou pessoal de suporte possam acessar as informações de alterações por categoria de palavra-chave. Além disso, é gerado um relatório do CSR regularmente; ele se destina a manter a gerência e os profissionais informados sobre alterações importantes.



Desenvolva uma lista do tipo "precisa saber" para todo objeto de configuração e mantenha-a atualizada. Quando é feita uma alteração, certifique-se de que todos os que estão na lista sejam notificados.

FERRAMENTAS DO SOFTWARE



Suporte de SCM

Objetivo: as ferramentas de SCM proporcionam suporte para uma ou mais das atividades de processo discutidas na Seção 22.3.

Mecânica: muitas ferramentas de SCM modernas funcionam em conjunto com um repositório (um sistema de banco de dados) e proporcionam mecanismos para identificação, versão e controle de alterações, auditoria e relatórios.

Ferramentas representativas:⁶

CCC/Harvest, distribuída pela Computer Associates (www.cai.com), é um sistema de SCM multiplataforma.

ClearCase, desenvolvida pela Rational, proporciona uma família de funções de SCM (www-306.ibm.com/software/awdtools/clearcase/index.html).

Serena ChangeMan ZMF, distribuída pela Serena (www.serena.com/US/products/zmf/index.aspx),

contém um conjunto completo de ferramentas de SCM aplicáveis tanto ao software convencional quanto às WebApps.

SourceForge, distribuída pela VA Software (sourceforge.net), contém gerenciamento de versão, capacidades de construção, rastreamento de problema/bug e muitas outras características de gestão.


SurroundSCM, desenvolvida pela Seapine Software, proporciona recursos completos de gestão de alterações (www.seapine.com).

Vesta, distribuída pela Compac, é um sistema de SCM de domínio público que pode suportar projetos pequenos (<10 KLOC) e grandes (10.000 KLOC) (www.ves-tasys.org).

Uma lista bem organizada de ferramentas e ambientes de SCM comerciais pode ser encontrada em www.cmtoday.com/yp/commercial.html.

⁶ As ferramentas aqui apresentadas não significam um aval, mas sim uma amostra dessa categoria. Na maioria dos casos, seus nomes são marcas registradas pelos respectivos desenvolvedores.

22.4 GESTÃO DE CONFIGURAÇÃO PARA WEBAPPS

 **Que impacto uma alteração não controlada tem sobre uma WebApp?**

Neste livro, é discutida a natureza especial das aplicações para Web e de métodos especializados (chamados de *métodos de engenharia Web*⁷) necessários para criá-las. Dentre as muitas características que diferenciam as WebApps do software tradicional está a natureza onipresente da alteração.

Os desenvolvedores para WebApp muitas vezes usam um modelo de processo iterativo, incremental, que aplica muitos princípios derivados do desenvolvimento ágil de software (Capítulo 3). Por meio dessa abordagem, uma equipe de engenharia muitas vezes desenvolve um incremento para WebApp em um período de tempo muito curto usando uma abordagem focada no cliente. Incrementos subsequentes adicionam conteúdo e funcionalidade, e cada um deles tende a implementar alterações que levam a um conteúdo aperfeiçoado, melhor utilização, melhor estética, melhor navegação, melhor desempenho e maior segurança. Portanto, no mundo ágil das WebApps, a alteração é vista de forma um tanto diferente.

Se você é um membro de uma equipe para WebApp, tem de adotar as alterações. E ainda mais, uma equipe ágil típica evita todas as coisas que parecem ser intensivas que tornam processo pesado burocrático e formal. A gestão de configuração de software é vista com frequência (embora incorretamente) como detentora dessas características. Essa contradição é remediada, não pela rejeição dos princípios, práticas e ferramentas de SCM, mas sim moldando-as para satisfazerem às necessidades especiais dos projetos de WebApp.

22.4.1 Problemas dominantes

Na medida em que as WebApps se tornam cada vez mais importantes para a sobrevivência e crescimento dos negócios, crescem as necessidades da gestão de configuração. Por quê? Porque sem controles eficazes, alterações impróprias a uma WebApp (lembre-se de que o imediatismo e a evolução contínua são os atributos dominantes de muitas WebApps) podem levar a: uma colocação não autorizada de informações sobre novos produtos, funcionalidade errônea ou mal testada que causa frustração nos visitantes de um site, brechas na segurança que põem em risco os sistemas internos da empresa e outras consequências economicamente desagradáveis ou até mesmo desastrosas.

As estratégias gerais para gestão de configuração de software (SCM) descritas neste capítulo são aplicáveis, mas as táticas e as ferramentas devem ser adaptadas para se conformarem com a natureza especial das WebApps. Quatro aspectos [Dar99] deverão ser considerados ao desenvolvermos táticas para gestão de configuração de WebApp.

Conteúdo. Uma WebApp típica contém um vasto conjunto de conteúdo — texto, gráficos, applets, scripts, arquivos de áudio/vídeo, formulários, elementos de página ativos, tabelas, dados encadeados e muitos outros. O desafio é organizar esse mar de conteúdo em um conjunto racional de objetos de configuração (Seção 22.1.4) e, então, estabelecer mecanismos de controle de configuração apropriados para esses objetos. Uma abordagem é modelar o conteúdo da WebApp usando técnicas convencionais de modelagem de dados (Capítulo 6), anexando um conjunto de propriedades especializadas a cada objeto. A natureza estática/dinâmica de cada objeto e sua longevidade projetada (por exemplo, objeto temporário, de existência fixa ou permanente) são exemplos de propriedades necessárias para estabelecer uma abordagem de SCM eficaz. Por exemplo, se um item de conteúdo é alterado a cada hora, ele tem longevidade temporária. Os mecanismos de controle para esse item seriam diferentes (menos formais) daqueles aplicados a um componente de formulários que é um objeto permanente.

Pessoas. Devido ao fato de que uma porcentagem significativa do desenvolvimento de WebApp continua a ser executada de maneira *ad hoc*, qualquer pessoa envolvida na WebApp pode criar conteúdo (e frequentemente o faz). Muitos criadores de conteúdo não possuem conhecimentos

⁷ Ver em [Pre08] uma discussão simples dos métodos de engenharia para Web.

em engenharia de software e ignoram completamente a necessidade de gestão de configuração. Consequentemente, a aplicação cresce e é alterada de maneira não controlada.

Escalabilidade. As técnicas e controles aplicados a uma pequena WebApp não são bem escaláveis. Não é raro uma simples WebApp crescer significativamente enquanto são implementadas interconexões com sistemas de informação existentes, bancos de dados, armazém de dados e gateways de portais. À medida que cresce o tamanho e a complexidade, pequenas mudanças podem ter efeitos amplos e inesperados que podem se tornar problemáticos. Portanto, o rigor dos mecanismos de configuração deverá ser diretamente proporcional à escala de aplicação.

Políticas. Quem é o “dono” de uma WebApp? Essa é uma pergunta feita em empresas grandes e pequenas, e sua resposta tem um impacto significativo sobre as atividades de gerenciamento e controle. Em alguns casos os desenvolvedores para Web estão instalados fora da organização de TI, criando dificuldades potenciais de comunicação. Dart [Dar99] sugere as seguintes perguntas para ajudar a entender as políticas associadas com engenharia para Web:

- Quem assume a responsabilidade pela exatidão das informações no site?
- Quem garante que os processos de controle de qualidade foram obedecidos antes que as informações fossem publicadas no site?
- Quem é responsável por fazer alterações?
- Quem assume o custo da alteração?

As respostas a essas perguntas ajudam a determinar as pessoas na organização que devem adotar um processo de gestão de configuração para WebApps.

A gestão de configuração para WebApps continua a evoluir (por exemplo, [Ngu06]). Um processo de SCM convencional pode ser muito desajeitado, mas uma nova geração de *ferramentas de gestão de conteúdo* especificamente projetadas para a engenharia surgiu nos últimos anos. Essas ferramentas estabelecem um processo que adquire as informações existentes (de uma ampla variedade de objetos WebApp), gerencia as alterações nos objetos, estrutura essas alterações para que possam ser apresentadas a um usuário final e as apresenta ao ambiente do lado do cliente para ser exibidas.

22.4.2 Objetos de configuração de WebApp

As WebApps abrangem grande variedade de objetos de configuração — objetos de conteúdo (por exemplo, texto, gráficos, imagens, vídeo e áudio), componentes funcionais (por exemplo, scripts, applets) e objetos de interface (por exemplo, COM ou CORBA). Os objetos da WebApp podem ser identificados (podem ser atribuídos nomes de arquivo para eles) de qualquer forma que seja apropriada para a organização. No entanto, recomendam-se as seguintes convenções para assegurar que seja mantida a compatibilidade entre plataformas: nomes de arquivos deverão ser limitados a 32 caracteres, deverão ser evitados nomes de arquivos com misturas de caracteres maiúsculos e minúsculos ou nomes com todas as letras em maiúscula e deverá ser evitado também o uso de underlines em nomes de arquivos. Além disso, referências a URL (links) dentro de um objeto de configuração devem sempre usar caminhos relativos (por exemplo, ../products/alarmsensors.html).

Todo o conteúdo da WebApp tem formato e estrutura. Os formatos de arquivos internos são ditados pelo ambiente de computação no qual o conteúdo está armazenado. No entanto, o *formato de renderização* (muitas vezes chamado de *formato de exibição*) é definido pelo estilo estético e regras de design estabelecidas para WebApp. A *estrutura de conteúdo* define uma arquitetura de conteúdo; ela define a maneira pela qual são montados os objetos de conteúdo para apresentar informações claras ao usuário final. Boiko [Boi04] define estrutura como “mapas que você coloca sobre um conjunto de conteúdo [objetos] para organizá-los e torná-los acessíveis às pessoas que precisam deles”.

 Como determinar quem tem a responsabilidade pela CM da WebApp?

"Gestão de conteúdo é um antídoto para o emaranhado de informações de hoje."

Bob Boiko

22.4.3 Gestão de conteúdo

A *gestão de conteúdo* está relacionada com a gestão de configuração no sentido de que um sistema de gestão de conteúdo (CMS) estabelece um processo (suportado por ferramentas apropriadas) que adquire o conteúdo existente (de uma ampla variedade de objetos de configuração de WebApp), estrutura esse conteúdo de maneira que ele possa ser apresentado a um usuário final e, então, fornece-o ao ambiente no lado do cliente para ser exibido.

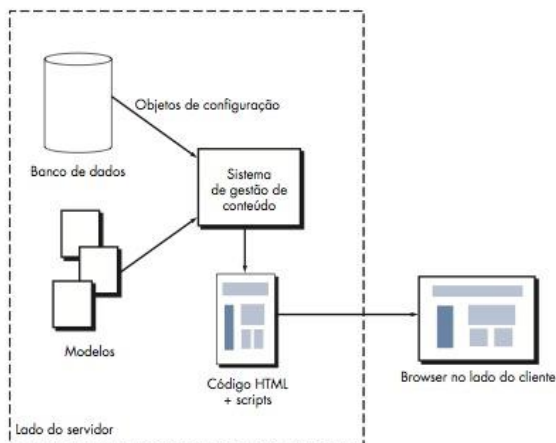
O uso mais comum de um sistema de gestão de conteúdo ocorre quando é criada uma WebApp dinâmica. As WebApps dinâmicas criam páginas Web "dinamicamente". Isto é, o usuário tipicamente consulta a WebApp solicitando informações específicas. A WebApp consulta um banco de dados, formata as informações corretamente e as apresenta ao usuário. Por exemplo, uma empresa de música fornece uma biblioteca de CDs para venda. Quando o usuário solicita um CD ou seu arquivo de música equivalente, é consultado um banco de dados, e uma variedade de informações sobre o artista, o CD (por exemplo, sua imagem ou elementos gráficos), o conteúdo musical e uma amostra de áudio, tudo isso é baixado (download) e configurado em um modelo de conteúdo padrão. A página Web resultante é criada no lado do servidor e passada para o lado do browser cliente para ser examinada pelo usuário final. Uma representação genérica disso está na Figura 22.6.

No sentido mais geral, um CMS "configura" conteúdo para o usuário final por meio da invocação de três subsistemas integrados: um subsistema de coleção, um subsistema de gestão e um subsistema de publicação [Boi04].

O subsistema de coleção. O conteúdo é extraído dos dados e de informações que devem ser criados ou adquiridos por um gerador de conteúdo. O *subsistema de coleção* abrange todas as ações necessárias para criar e/ou adquirir conteúdo e as funções técnicas que são necessárias para (1) converter conteúdo de maneira que possa ser representado por uma linguagem de marcação (por exemplo, HTML, XML), e (2) organiza o conteúdo em pacotes que podem ser efetivamente mostrados no lado do cliente.

FIGURA 22.6

Sistema de gestão de conteúdo



PONTO-CHAVE

O subsistema de coleção abrange todas as ações necessárias para adquirir, e/ou converter conteúdo de maneira que possa ser apresentado no lado do cliente.

Criação e aquisição de conteúdo (também chamada de *autoria*) ocorre muitas vezes em paralelo com outras atividades de desenvolvimento de WebApp e em geral é conduzido por criadores de conteúdo não técnicos. Essa atividade combina elementos de criatividade e pesquisa e é suportada por ferramentas que permitem ao autor do conteúdo caracterizá-lo de modo que possa ser padronizado para uso dentro da WebApp.

Uma vez existindo o conteúdo, ele deve ser convertido para se adaptar aos requisitos de um CMS. Isso implica eliminar o conteúdo bruto de quaisquer informações desnecessárias (por exemplo, representações gráficas redundantes), formatação do conteúdo para se adaptar aos requisitos do CMS e mapear os resultados em uma estrutura de informações que permita que seja gerenciado e publicado.

PONTO-CHAVE

O subsistema de gestão implementa um repositório para todo o conteúdo. A gestão de configuração é executada nesse subsistema.

O subsistema de gestão. Uma vez existindo o conteúdo, ele deve ser armazenado em um repositório, catalogado para aquisição e uso subsequente, e rotulado para definir (1) status atual (por exemplo, o objeto de conteúdo está completo ou em desenvolvimento?), (2) a versão apropriada do objeto de conteúdo, e (3) objetos de conteúdo relacionados. Portanto, o *subsistema de gestão* implementa um repositório que abrange os seguintes elementos:

- *Banco de dados de conteúdo* — a estrutura de informações estabelecida para armazenar todos os objetos de conteúdo.
- *Recursos de banco de dados* — funções que permitem ao CMS pesquisar objetos de conteúdo específicos (ou categorias de objetos), armazenar e recuperar objetos e gerenciar a estrutura de arquivo estabelecida para o conteúdo.
- *Funções de gestão de configuração* — os elementos funcionais e o workflow associado que suporta identificação do objeto de conteúdo, controle de versão, gestão de alterações, gestão de auditoria e relatos.

Além desses elementos, o subsistema de gestão implementa uma função de administração que abrange os metadados e regras que controlam a estrutura global do conteúdo e a maneira pela qual ele é suportado.

PONTO-CHAVE

O subsistema de publicação extrai o conteúdo do repositório e o fornece aos navegadores do lado do cliente.

O subsistema de publicação. O conteúdo deve ser extraído de um repositório, convertido para uma forma que seja conveniente para a publicação e formatado de maneira que possa ser transmitido aos navegadores do lado do cliente. O subsistema de publicação executa essas tarefas usando uma série de modelos (templates). Cada *modelo* é uma função que cria uma publicação por meio de um dentre três componentes diferentes [Boi04]:

- *Elementos estáticos* — texto, gráficos, mídia e scripts que não requerem outros processamentos são transmitidos diretamente para o lado do cliente.
- *Serviços de publicação* — chamadas de função para serviços específicos de acesso e formatação que personalizam o conteúdo (usando regras predefinidas), executam a conversão dos dados e criam links de navegação apropriados.
- *Serviços externos* — fornecem acesso à infraestrutura de informação corporativa externa como, por exemplo, aplicações de dados ou de retaguarda da empresa.

Um subsistema de gestão de conteúdo que abrange cada um desses subsistemas é aplicável à maior parte dos projetos para WebApp. No entanto, a filosofia e funcionalidade básicas associadas com um CMS são aplicáveis a todas as WebApps dinâmicas.

22.4.4 Gestão de alterações

O fluxo de trabalho associado ao controle de alterações para software convencional (Seção 22.3.3) em geral é muito ponderado para desenvolvimento de WebApp. É pouco provável que a solicitação de alteração, relato da alteração e sequência de ordem de mudança de engenharia possam ser conseguidos de forma ágil que seja aceitável para a maioria dos projetos de desenvolvimento de WebApp. Como podemos então controlar uma corrente contínua de alterações solicitadas para o conteúdo e funcionalidade da WebApp?

FERRAMENTAS DO SOFTWARE

**Gestão de conteúdo**

Objetivo: ajudar os engenheiros de software e criadores de conteúdo na gestão de conteúdo que é incorporado nas WebApps.

Mecânica: ferramentas nesta categoria permitem aos engenheiros da Web e criadores de conteúdo atualizar o conteúdo da WebApp de forma controlada. Muitos estabelecem um simples sistema de gestão de arquivo que atribui permissões de atualizações página por página para vários tipos de conteúdo da WebApp. Outros mantêm um sistema de controle de versões para que uma versão anterior de um conteúdo possa ser arquivada para fins históricos.

Ferramentas representativas:⁸

Vignette Content Management, desenvolvida pela Vignette (www.vignette.com/us/Products), é um conjunto de ferramentas de gestão de conteúdo corporativo.

ektron-CMS300, desenvolvida pela ektron (www.ektron.com), é um conjunto de ferramentas que proporciona recursos de gestão de conteúdo, como também ferramentas de desenvolvimento para Web.

OmniUpdate, desenvolvida pela WebsiteASP, Inc. (www.omniupdate.com), é uma ferramenta que permite que os provedores de conteúdo autorizado desenvolvam atualizações controladas para conteúdo específico da WebApp.

Informações adicionais sobre SCM e ferramentas de gestão de conteúdo para engenharia para Web podem ser encontradas em um ou mais dos seguintes sites: *Web Developer's Virtual Encyclopedia* (www.wdlv.com), *WebDeveloper* (www.webdeveloper.com), *Developer Shed* (www.devshed.com), *webknowhow.net* (www.webknowhow.net) ou *WebReference* (www.webreference.com).

Para implementar um gerenciamento efetivo de alterações segundo a filosofia de "codifique e vá em frente" que continua a dominar o desenvolvimento de WebApp, o processo convencional de controle de alterações deve ser modificado. Cada alteração deverá ser classificada em uma dentre quatro classes:

Classe 1 — alteração de conteúdo ou função que corrige um erro ou melhora o conteúdo ou a funcionalidade local.

Classe 2 — alteração de conteúdo ou função que tenha impacto sobre outros objetos de conteúdo ou sobre os componentes funcionais.

Classe 3 — alteração de conteúdo ou função que tenha um amplo impacto através de uma WebApp (por exemplo, extensão ou funcionalidade principais, melhora significativa ou redução em conteúdo, alterações importantes necessárias na navegação).

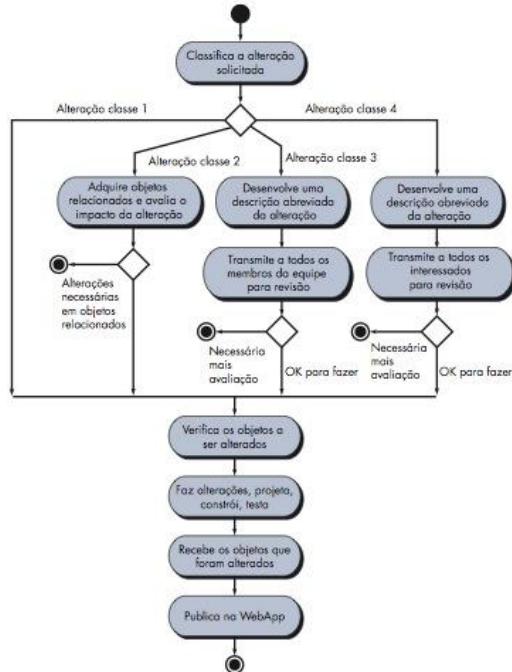
Classe 4 — alteração importante de projeto (por exemplo, alteração na abordagem do projeto da interface ou na abordagem de navegação) que será notada imediatamente por uma ou mais categorias de usuário.

Uma vez classificada a solicitação de alteração, ela pode ser processada de acordo com o algoritmo mostrado na Figura 22.7.

De acordo com a figura, as alterações classe 1 e classe 2 são tratadas informalmente e são manipuladas de modo ágil. Para uma alteração classe 1, você avaliaria o impacto da mudança, mas não é necessária nenhuma revisão externa ou documentação. À medida que a alteração é feita, os procedimentos-padrão de entrada (check-in) e saída (check-out) são apoiados por ferramentas de repositório de configuração. Para alterações classe 2, você deveria revisar o impacto da alteração sobre objetos relacionados (ou pedir a outros desenvolvedores responsáveis por aqueles objetos que o façam). Se a alteração pode ser feita sem necessidade de alterações significativas em outros objetos, a modificação ocorre sem revisão ou documentação adicional. Se forem necessárias alterações substanciais, mais avaliação e planejamento serão exigidos.

Alterações classe 3 e 4 também são tratadas de uma forma ágil, mas é necessária alguma documentação descritiva e procedimentos de revisão mais formais. Para as alterações classe 3 é desenvolvida uma *descrição de alteração* — descrevendo a alteração e fornecendo uma breve avaliação do impacto da alteração. A descrição é distribuída a todos os membros da equipe que a

⁸ As ferramentas aqui apresentadas não significam um aval, mas sim uma amostra dessa categoria. Na maioria dos casos, seus nomes são marcas registradas pelos respectivos desenvolvedores.

FIGURA 22.7**Gerenciando alterações para WebApps****FERRAMENTAS DO SOFTWARE****Gestão de alterações**

Objetivo: ajudar os projetistas da Web e criadores de conteúdo na gestão de alterações à medida que são feitas nos objetos de configuração para WebApp.

Mecânica: as ferramentas desta categoria foram desenvolvidas originalmente para software convencional, mas podem ser adaptadas para ser usadas pelos engenheiros da Web e criadores de conteúdo para fazer alterações controladas nas WebApps. Elas suportam entrada e saída automáticas, controle de versão e desfazer alterações (rollback), relatórios e outras funções da SCM.

Ferramentas representativas:⁹

ChangeMan WCM, desenvolvida pela Serena (www.serena.com), é um conjunto de ferramentas de gestão de alteração que proporcionam recursos de SCM completos.

ClearCase, desenvolvida pela Rational (www-306.ibm.com/software/rational/sw-atoz/indexC.html), é um conjunto de ferramentas que proporcionam recursos completos de gestão de configuração para WebApps.

Source Integrity, desenvolvida pela mks (www.mks.com), é uma ferramenta de SCM que pode ser integrada a ambientes de desenvolvimento selecionados.

examinam para melhor avaliar seu impacto. É desenvolvida também uma descrição de alteração para as alterações classe 4, mas nesse caso a revisão é conduzida por todos os interessados.

⁹ As ferramentas aqui apresentadas não significam um aval, mas sim uma amostra dessa categoria. Na maioria dos casos, seus nomes são marcas registradas pelos respectivos desenvolvedores.

22.4.5 Controle de versão

À medida que uma WebApp evolui por meio de uma série de incrementos, podem existir várias versões diferentes ao mesmo tempo. Uma versão (a WebApp operacional corrente) está disponível na Internet para os usuários finais; outra versão (o próximo incremento da WebApp) pode estar nos estágios finais de teste antes da distribuição/instalação (*deployment*); uma terceira versão está em desenvolvimento e representa uma grande atualização em conteúdo, estética de interface e funcionalidade. Objetos de configuração podem estar claramente definidos para que cada um possa ser associado à versão apropriada. Além disso, devem ser estabelecidos mecanismos de controle. Dreilinger [Dre99] discute a importância do controle de versão (e alteração) quando escreve:

Em um site *não controlado* no qual múltiplos autores têm acesso para editar e contribuir, surge um potencial para conflitos e problemas — mais ainda quando esses autores trabalham em locais diferentes em horários diferentes do dia e da noite. Você pode passar o dia melhorando o arquivo *index.html* para um cliente. Depois que você fez as suas alterações, outro desenvolvedor que trabalha em casa após o horário comercial ou em outro escritório, pode gastar a noite fazendo o uploading de sua própria nova versão revisada do arquivo *index.html*, sobrescrevendo completamente o seu trabalho sem maneira de recuperá-lo!

É provável que você já tenha passado por uma situação assim. Para tanto, é necessário um processo de controle de versão.

1. *Deverá ser estabelecido um repositório central para o projeto de WebApp.* O repositório terá as versões correntes de todos os objetos de configuração de WebApp (conteúdo, componentes funcionais e outros).
2. *Cada projetista da Web cria sua própria pasta de trabalho.* A pasta contém aqueles objetos que estão sendo criados ou alterados em determinado instante.
3. *Os relógios das estações de trabalho de todos os desenvolvedores deverão estar sincronizados.* Isso é feito para evitar conflitos de sobrescrita quando dois desenvolvedores fazem alterações com horários muito próximos.
4. *Na medida em que novos objetos de configuração são desenvolvidos ou objetos existentes são alterados, eles são importados para o repositório central.* A ferramenta de controle de versão (veja discussão sobre CVS na barra lateral) irá gerenciar todas as funções de check-in (entrada) e check-out (saída) das pastas de trabalho de cada desenvolvedor da WebApp. A ferramenta também fornecerá atualizações automáticas de e-mail a todas as partes interessadas quando forem feitas alterações no repositório.
5. *À medida que objetos são importados ou exportados do repositório, é gerada uma mensagem automática com data e hora.* Isso proporciona informações úteis para auditoria e pode se tornar parte de um esquema eficaz de relatórios.

A ferramenta de controle de versão mantém diferentes versões da WebApp e pode reverter para uma versão mais antiga se necessário.

22.4.6 Auditoria e relatório

Para melhorar a agilidade, as funções de auditoria e relatório não são enfatizadas no trabalho de engenharia Web¹⁰. No entanto, elas não são todas eliminadas. Todos os objetos que entram (check-in) ou saem (check-out) do repositório são registrados em um log (registro) que pode ser revisto quando se desejar. Pode ser criado um relatório completo de forma que todos os membros da equipe da WebApp tenham uma cronologia das alterações durante um período definido. Além disso, uma notificação automática por e-mail (endereçada a todos os desenvolvedores e interessados que tenham interesse) pode ser enviada todas as vezes que um objeto entra ou sai do repositório.

¹⁰ Isso está começando a mudar. Há uma ênfase cada vez maior no SCM como um elemento da segurança WebApp [Sar06]. Proporcionando um mecanismo para rastrear e relatar todas as alterações feitas em um objeto WebApp, uma ferramenta de gestão de alterações pode proporcionar uma valiosa proteção contra alterações mal-intencionadas.



Normas de SCM

A seguir é apresentada uma lista de normas de SCM (extraída em parte do site www.12207.com) razoavelmente abrangente:

IEEE Standards	standards.ieee.org/catalog/olis/	EIA CMB6-5	Textbook for Configuration Status Accounting
IEEE 828	Software Configuration Management Plans	EIA CMB7-1	Electronic Interchange of Configuration Management Data
IEEE 1042	Software Configuration Management	U.S. Military Standards mil	Information of MIL standards: www.library.itsi.disa.mil
ISO Standards	www.iso.ch/iso/en/ISOOnline.frontpage	DoD MIL STD-973	Configuration Management
ISO 10007-1995	Quality Management, Guidance for CM	MIL-HDBK-61	Configuration Management Guidance
ISO/IEC 12207	Information Technology-Software Life Cycle Processes	Outras normas	
ISO/IEC TR 15271	Guide for ISO/IEC 12207	DO-178B	Guidelines for the Development of Aviation Software
ISO/IEC TR 15846	Software Engineering-Software Life Cycle Process-Configuration Management for Software Order	ESA PSS-05-09	Guide to Software Configuration Management
EIA Standards	www.eia.org/	AECL CE-1001-STD	rev.1 Standard for Software Engineering of Safety Critical Software
EIA 649	National Consensus Standard for Configuration Management	DOE SCM checklist:	http://cio.doe.gov/ITReform/sqse/download/cmcklst.doc
EIA CMB4-1A	Configuration Management Definitions for Digital Computer Programs	BS-6488	British Std., Configuration Management of Computer-Based Systems
EIA CMB4-2	Configuration Identification for Digital Computer Programs	Best Practice — UK	Office of Government Commerce: www.ogc.gov.uk
EIA CMB4-3	Computer Software Libraries	CMII	Institute of CM Best Practices: www.icmhq.com
EIA CMB4-4	Configuration Change Control for Digital Computer Programs		
EIA CMB6-1C	Configuration and Data Management References Order		
EIA CMB6-3	Configuration Identification		
EIA CMB6-4	Configuration Control		

Um guia de recursos de gestão de configuração (*Configuration Management Resource Guide*) proporciona informações complementares para aqueles interessados nos processos e prática de gestão de alterações (CM). O guia está disponível no site **www.quality.org/config/cm-guide.html**.

22.5 RESUMO

A gestão de configuração de software (SCM) é uma atividade abrangente aplicada em todo o processo de software. A SCM identifica, controla, faz auditoria e relata modificações que invariavelmente ocorrem enquanto o software está sendo desenvolvido e depois que foi entregue ao cliente. Todos os produtos criados como parte da engenharia de software tornam-se parte de uma configuração de software. A configuração é organizada de maneira que permite controle ordenado das alterações.

A configuração de software é composta por uma série de objetos inter-relacionados, também chamados de itens de configuração de software (SCIs), que são produzidos como resultado de alguma atividade de engenharia de software. Além dos documentos, programas e dados, o ambiente de desenvolvimento usado para criar software também pode ser colocado sob o controle de configuração. Todas os SCIs são armazenados em um repositório que implementa uma série de mecanismos e estruturas de dados para assegurar a integridade dos dados, proporcionar suporte de integração para outras ferramentas de software, suportar compartilhamento de informações entre todos os membros da equipe de software e implementar funções no suporte do controle de versão e alteração.

Uma vez desenvolvido e revisado um objeto de configuração, ele se torna uma referência. Alterações em um objeto referencial resultam na criação de uma nova versão daquele objeto.

A evolução de um programa pode ser acompanhada examinando-se o histórico de revisão de todos os objetos de configuração. O controle de versão é uma série de procedimentos e ferramentas para gerenciar o uso desses objetos.

O controle de alteração é uma atividade procedimental que assegura qualidade e consistência quando são feitas alterações em um objeto de configuração. O processo de controle de alterações começa com uma solicitação de alteração, leva a uma decisão sobre fazer ou rejeitar a solicitação de alteração e culmina com uma atualização controlada do SCI que deve ser alteada.

A auditoria de configuração é uma atividade de SQA que ajuda a assegurar que a qualidade seja mantida quando feitas alterações. Os relatórios de status fornecem informações sobre cada alteração para aqueles que precisam ter conhecimento do assunto.

A gestão de configuração para WebApps é similar em muitos aspectos à de SCM para software convencional. No entanto, cada uma das tarefas centrais de SCM deverá ser agilizada para torná-la o mais leve possível e devem ser implementadas provisões especiais para gestão de conteúdo.

PROBLEMAS E PONTOS A PONDERAR

- 22.1.** Por que a Primeira Lei da Engenharia de Software é verdadeira? Forneça exemplos específicos para cada uma das quatro razões fundamentais para alterações.
- 22.2.** Quais são os quatro elementos que existem quando é implementado um sistema de SCM eficaz? Discuta rapidamente cada um.
- 22.3.** Discuta as razões para referenciais (baselines) com suas próprias palavras.
- 22.4.** Suponha que você seja o gerente de um pequeno projeto. Que referenciais definiria para o projeto e como os controlaria?
- 22.5.** Desenvolva um sistema de banco de dados de projeto (repositório) que permitiria a um engenheiro de software armazenar, estabelecer referências cruzadas, acompanhar, atualizar e alterar todos os itens importantes da configuração de software. Como o banco de dados trataria com diferentes versões do mesmo programa? O código-fonte seria tratado de forma diferente da documentação? Como dois desenvolvedores seriam impedidos de fazer alterações diferentes no mesmo SCI ao mesmo tempo?
- 22.6.** Pesquise uma ferramenta de SCM existente e descreva como ela implementa o controle para versões, variantes e objetos de configuração em geral.
- 22.7.** As relações <parte-de> e <inter-relacionado> representam relações simples entre objetos de configuração. Descreva cinco relações adicionais que podem ser úteis no contexto de um repositório de SCM.
- 22.8.** Pesquise uma ferramenta de SCM existente e descreva como ela implementa o mecanismo do controle de versão. Como alternativa, leia duas ou três publicações sobre a SCM e descreva as diferentes estruturas de dados e mecanismos de referência usados para o controle de versão.
- 22.9.** Desenvolva uma lista de checagem (*checklist*) para usar durante as auditorias de configuração.
- 22.10.** Qual é a diferença entre uma auditoria de SCM e uma revisão técnica? Podem suas funções serem incluídas em uma revisão? Quais são os prós e os contras?
- 22.11.** Descreva rapidamente as diferenças entre a SCM para software convencional e a SCM para WebApps.
- 22.12.** O que é gestão de conteúdo? Use a Web para pesquisar as características de uma ferramenta de gestão de conteúdo e forneça um breve resumo.

LEITURAS E FONTES DE INFORMAÇÃO COMPLEMENTARES

Entre as ofertas mais recentes sobre SCM estão Leon (*Software Configuration Management Handbook*, 2d ed., Artech House Publishers, 2005), Maraia (*The Build Master: Microsoft's Software Configuration Management Best Practices*, Addison-Wesley, 2005), Keyes (*Software Configuration Management*, Auerbach, 2004) e Hass (*Configuration Management Principles and Practice*, Addison-Wesley, 2002). Cada um desses livros apresenta todo o processo de SCM com detalhes substanciais. Maraia (*Software Configuration Management Implementation Roadmap*, Wiley, 2004) fornece um guia do tipo "como fazer para..." para aqueles que devem implementar a SCM em uma organização. Lyon (*Practical CM*, Raven Publishing, 2003, disponível em www.configuration.org) descreveu um guia simples para o profissional de CM incluindo diretrizes pragmáticas para implementar todos os aspectos de um sistema de gestão de configuração (atualizado anualmente). White e Clemm (*Software Configuration Management Strategies and Rational ClearCase*, Addison-Wesley, 2000) apresentam a SCM dentro do contexto de uma ou mais ferramentas populares de SCM.

Berczuk e Appleton (*Software Configuration Management Patterns*, Addison-Wesley, 2002) mostram uma variedade de padrões úteis que ajudam a entender a SCM e a implementar sistemas de SCM eficazes. Brown et al. (*Anti-Patterns and Patterns in Software Configuration Management*, Wiley, 1999) discutem o que não se deve fazer (antipadrões) ao implementar um processo de SCM e, em seguida, trata dos remédios. Bays (*Software Release Methodology*, Prentice-Hall, 1999) focaliza o mecanismo de "versão bem-sucedida de produtos", um complemento importante para uma SCM eficaz.

Conforme as WebApps se tornam mais dinâmicas, a gestão de conteúdo tem se tornado um tópico essencial para engenheiros da Web. Livros de White (*The Content Management Handbook*, Curtin University Books, 2005), Jenkins e seus colegas (*Enterprise Content Management Methods*, Open Text Corporation, 2005), Boiko [Boi04], Mauthe e Thomas (*Professional Content Management Systems*, Wiley, 2004), Addey e seus colegas (*Content Management Systems*, Glasshaus, 2003), Rockley (*Managing Enterprise Content*, New Riders Press, 2002), Hackos (*Content Management for Dynamic Web Delivery*, Wiley, 2002) e Nakano (*Web Content Management*, Addison-Wesley, 2001) apresentam bons tratamentos do assunto.

Além das discussões do tópico, Lim e seus colegas (*Enhancing Microsoft Content Management Server with ASP.NET 2.0*, Packt Publishing, 2006), Ferguson (*Creating Content Management Systems in Java*, Charles River Media, 2006), IBM Redbooks (*IBM Workplace Web Content Management for Portal 5.1 and IBM Workplace Web Content Management 2.5*, Vivante, 2006), Fritz e seus colegas (*Typo3: Enterprise Content Management*, Packt Publishing, 2005) e Forta (*Reality ColdFusion: Intranets and Content Management*, Pearson Education, 2002) abordam gestão de conteúdo no contexto de ferramentas e linguagens específicas.

Uma ampla variedade de fontes de informação sobre gestão de configuração de software e gestão de conteúdo está disponível na Internet. Uma lista atualizada das referências na Web, relevantes à gestão de configuração de software, pode ser encontrada no site www.mhhe.com/engcs/compsci/pressman/professional/olc/ser.htm.