

12

PROJETO BASEADO EM PADRÕES

CONCEITOS-CHAVE

erros de projeto .	326
estruturas	320
forças	318
granularidade . . .	334
linguagem de padrões	321
padrões comportamentais .	327
criacionais	319
de arquitetura . . .	327
de componentes . .	329
estruturais	319
generativos	318
para interfaces do usuário	333
para WebApps .	333

Todos já nos deparamos com um problema de projeto e, silenciosamente, pensamos: Será que alguém já desenvolveu uma solução para este problema? A resposta é quase sempre — sim! O problema é encontrar a solução; garantir que, de fato, se adapte ao problema em questão; entender as restrições que talvez limitem a maneira pela qual a solução é aplicada e, por fim, traduzir a solução proposta para seu ambiente de projeto.

Mas o que acontece se a solução fosse de alguma forma codificada? E se existisse uma maneira padronizada de descrever um problema (de tal forma que pudéssemos pesquisar-la) e um método organizado para representar a solução para o problema? Acaba evidenciando-se que problemas de software foram codificados e descritos usando-se um template padronizado e que foram propostas soluções (juntamente com restrições) para eles. Denominado *padrões de projeto*, esse método codificado para descrição de problemas e suas soluções permite aos profissionais da engenharia de software adquirir conhecimento de projeto para possibilitar que ele seja reutilizado.

O início da história dos padrões de software não se inicia com um cientista da computação, mas com um arquiteto, Christopher Alexander, que reconheceu o fato de um conjunto de problemas recorrentes ser encontrado toda vez que um edifício era projetado. Ele caracterizou esses problemas recorrentes e suas soluções como *padrões*, descrevendo-os da seguinte maneira [Ale77]:

PANORAMA

O que é? O projeto baseado em padrões cria uma nova aplicação através da busca de um conjunto de soluções comprovadas para um conjunto de problemas claramente delineados. Cada problema (e sua solução) é descrito por um padrão de projeto que foi catalogado e investigado por outros engenheiros de software que depararam com o problema e implementaram a solução ao projetarem outras aplicações. Cada padrão de projeto nos oferece uma abordagem comprovada para parte do problema a ser resolvido.

Quem realiza? Um engenheiro de software examina cada problema que surge para uma nova aplicação e tenta encontrar uma solução relevante por meio de pesquisa em um ou mais repositórios de padrões.

Por que é importante? Você já ouviu a frase “reinventar a roda”? Ela ocorre toda hora em desenvolvimento de software e é uma perda de tempo e energia. Ao usarmos padrões de projeto, podemos encontrar uma solução comprovada para um problema específico. À medida que cada padrão é aplicado, são integradas soluções, e a aplicação a ser construída se aproxima cada vez mais de um projeto completo.

Quais são as etapas envolvidas? O modelo de requisitos é examinado para isolar o conjun-

to hierárquico de problemas a ser resolvido. O espaço de problemas é subdividido de modo que subconjuntos de problemas associados a funções e características de software específicas possam ser identificados. Os problemas também podem ser organizados por tipo: de arquitetura, de componentes, algorítmicos, de interfaces do usuário etc. Uma vez definido um subconjunto de problemas, pesquisam-se um ou mais repositórios de padrões para determinar se um padrão de projeto existente, representado em um nível de abstração apropriado, existe. Padrões aplicáveis são adaptados às necessidades específicas do software a ser construído. A solução de problemas personalizados é aplicada em situações em que não foi encontrado nenhum padrão.

Qual é o artefato? É desenvolvido um modelo de projeto que represente a estrutura da arquitetura, a interface do usuário e detalhes em nível de componentes.

Como garantir que o trabalho foi realizado corretamente? À medida que cada padrão de projeto é traduzido em algum elemento do modelo de projeto, os artefatos são revistos em termos de clareza, correção, completude e consistência em relação às necessidades e entre si.

Cada padrão descreve um problema que ocorre repetidamente em nosso ambiente e então descreve o cerne de uma solução para aquele problema para podermos usar a solução repetidamente um milhão de vezes sem jamais ter de fazer a mesma coisa duas vezes.

As ideias de Alexander foram primeiro traduzidas para o mundo do software em livros como os de Gamma [Gam95], Buschmann [Bus96] e seus vários colegas¹. Hoje, existem dezenas de repositórios de padrões, e projetos baseados em padrões podem ser aplicados em diversos domínios de aplicação.

12.1 PADRÃO DE PROJETO

PONTO-CHAVE

Forças são aquelas características do problema e atributos de uma solução que restringem a maneira pela qual o projeto pode ser desenvolvido.

Um padrão de projeto pode ser caracterizado como “uma regra de três partes que expressa uma relação entre um contexto, um problema e uma solução” [Ale79]. Para projeto de software, o contexto permite ao leitor compreender o ambiente em que o problema reside e qual solução poderia ser apropriada nesse ambiente. Um conjunto de requisitos, incluindo limitações e restrições, atua como um sistema de forças que influencia a maneira pela qual o problema pode ser interpretado em seu contexto e como a solução pode ser efetivamente aplicada.

Para compreendermos melhor esses conceitos, consideremos a situação² em que uma pessoa tenha de viajar de Nova York a Los Angeles. Nesse contexto, a viagem ocorrerá em um país industrializado (os Estados Unidos), usando uma infraestrutura de transporte existente (por exemplo, estradas, linhas aéreas, ferrovias). O sistema de forças que afetará a maneira pela qual o problema de viagem é solucionado abrangerá: com que rapidez a pessoa quer ir de Nova York a LA, se a viagem incluirá ou não pontos turísticos ou de parada, quanto dinheiro a pessoa pode gastar, se a viagem se destina a cumprir um propósito específico e os veículos pessoais que a pessoa tem à sua disposição. Dadas essas forças, o problema (viajar de Nova York a LA) pode ser mais bem definido. Por exemplo, uma investigação (levantamento de requisitos) indica que a pessoa tem pouquíssimo dinheiro, tem apenas uma bicicleta (e é um ciclista entusiasta), quer fazer a viagem para arrecadar fundos para sua instituição de caridade preferida e tem muito tempo disponível. A solução para o problema, dado o contexto e o sistema de forças, poderia ser uma viagem de bicicleta atravessando o país de ponta a ponta. Se as forças fossem diferentes (por exemplo, o tempo de viagem deve ser minimizado e o propósito da viagem é uma reunião de negócios), talvez fosse mais apropriada uma outra solução.

É razoável argumentar que a maioria dos problemas possui várias soluções, porém uma solução é efetiva apenas se for apropriada no contexto do problema existente.

É o sistema de forças que faz com que um projetista escolha uma solução específica. O intuito é fornecer uma solução que melhor atenda o sistema de forças, mesmo quando essas forças forem contrárias. Por fim, toda solução tem consequências que poderiam ter um impacto sobre outros aspectos do software e ela própria poderia fazer parte do sistema de forças para outros problemas a ser resolvidos no sistema mais amplo. Coplien [Cop05] caracteriza um padrão de projeto eficaz da seguinte maneira:

- *Ele soluciona um problema:* Os padrões capturam soluções, não apenas estratégias ou princípios abstratos.
- *Ele é um conceito comprovado:* Os padrões apreendem soluções com um histórico, não teorias ou especulação.
- *Uma solução não é óbvia:* Muitas técnicas para resolução de problemas (como paradigmas ou métodos de projeto de software) tentam obter soluções com base nos primeiros princípios. Os melhores padrões geram uma solução para um problema indiretamente — uma abordagem necessária para os problemas mais difíceis de projeto.

“Nossa responsabilidade é fazer o que podemos, aprender o que podemos, aperfeiçoar as soluções e passá-las adiante.”

Richard P. Feynman

¹ Existem discussões iniciais de padrões de software, porém esses dois clássicos foram os primeiros tratados coesos sobre o assunto.

² Esse exemplo foi adaptado de [Cor98].

- *Ele descreve uma relação:* Os padrões não apenas descrevem módulos, como também estruturas e mecanismos de sistema mais profundos.
- *O padrão possui um componente humano significativo (minimizar a intervenção humana).* Todo software visa atender o conforto humano ou a qualidade de vida; os melhores padrões apelam explicitamente à estética e utilidade.

Declarado de uma forma ainda mais pragmática, um padrão de projeto adequado captura conhecimento de projeto pragmático e ganho com muito suor, de forma que outros possam reutilizar esse conhecimento “uma milhão de vezes sem jamais ter de fazer a mesma coisa duas vezes”. Um padrão de projeto evita que tenhamos de “reinventar a roda”, ou pior ainda, inventar uma “nova roda” que não será perfeitamente redonda, será muito pequena para o uso pretendido e muito estreita para o terreno onde irá rodar. Os padrões de projeto, se usados de maneira efetiva, invariavelmente o tornarão um melhor projetista de software.

12.1.1 Tipos de padrões

Uma das razões para os engenheiros de software interessarem-se (e intrigados) em padrões de projeto é o fato de os seres humanos serem inerentemente bons no reconhecimento de padrões. Se não fossemos, teríamos parado no tempo e no espaço — incapazes de aprender de experiência passada, desinteressados em nos aventurarmos devido à nossa inabilidade de reconhecer situações que talvez nos levassem a correr altos riscos, tornados por um mundo que parece não ter regularidade ou consistência lógica. Felizmente, nada disso acontece porque efetivamente reconhecemos padrões em praticamente todos os aspectos de nossas vidas.

No mundo real, os padrões que reconhecemos são aprendidos ao longo de toda uma vida de experiência. Reconhecemos instantaneamente e compreendemos inerentemente seus significados e como eles poderiam ser usados. Alguns desses padrões nos dão uma melhor visão do fenômeno da recorrência. Por exemplo, você está voltando do trabalho para casa na Marginal quando seu sistema de navegação (ou o rádio do carro) informa-lhe que um grave acidente ocorreu na Marginal no sentido oposto. Você se encontra a 6 quilômetros do acidente, porém já começa a perceber tráfego lento, reconhecendo um padrão que chamaremos **RubberNecking (olhar com curiosidade)**. Os motoristas deslocando-se na pista expressa na sua direção estão diminuindo de velocidade, para ter uma melhor visão do que aconteceu no sentido contrário. O padrão **RubberNecking** produz resultados notavelmente previsíveis (um congestionamento), mas nada mais faz que descrever um fenômeno. No jargão dos padrões, ele poderia ser denominado padrão *não generativo* pois descreve um contexto e um problema, mas não fornece nenhuma solução explícita.

Quando são considerados padrões de projeto de software, nos esforçamos para identificar e documentar padrões *generativos*. Ou seja, identificamos um padrão que descreve um aspecto importante e repetível de um sistema e que nos dá uma maneira de construir esse aspecto em um sistema de forças que são únicas em determinado contexto. Em um ambiente ideal, um conjunto de padrões de projeto generativos poderia ser usado para “gerar” uma aplicação ou sistema computacional cuja arquitetura permitisse que se adaptasse à mudança. Algumas vezes chamada *generatividade*, “a aplicação sucessiva de vários padrões, cada um deles encapsulando seu próprio problema e forças, se desdobra em uma solução mais ampla que emerge indiretamente como resultado das soluções menores” [App00].

Os padrões de projeto abrangem um amplo espectro de abstração e aplicação. Os *padrões de arquitetura* descrevem problemas de projeto de caráter amplo e diverso resolvidos usando-se uma abordagem estrutural. Os *padrões de dados* descrevem problemas orientados a dados recorrentes e as soluções de modelagem de dados que podem ser usadas para resolvê-los. Os *padrões de componentes* (também conhecidos como *padrões de projeto*) tratam de problemas associados ao desenvolvimento de subsistemas e componentes, a maneira através da qual eles se comunicam entre si e seu posicionamento em uma arquitetura maior. Os *padrões de projeto de interfaces* descrevem problemas comuns de interface do usuário e suas soluções com

PONTO-CHAVE

Um padrão “generativo” descreve o problema, um contexto e forças, mas também descreve uma solução pragmática para o problema.

um sistema de forças que incluía características específicas de usuários finais. Os *padrões para WebApp* tratam de um conjunto de problemas encontrados ao se construir WebApps e em geral incorpora muitas das demais categorias de padrões que acabamos de mencionar. Em um nível de abstração mais baixo, os *idiomas* descrevem como implementar todos ou parte de um algoritmo específico ou estrutura de dados para um componente de software no contexto de uma linguagem de programação específica.

Em seu livro seminal sobre padrões de projeto, Gamma e seus colegas³ [Gam95] focalizam três tipos de padrões particularmente relevantes em projetos orientados a objetos: padrões criacionais, padrões estruturais e padrões comportamentais.

 Existe uma maneira de classificar tipos de padrão?



Padrões criacionais, estruturais e comportamentais

Foi proposta uma ampla variedade de padrões de projeto que se encaixam nas categorias criacional, estrutural e comportamental e podem ser encontradas na Web.

A Wikipedia (www.wikipedia.org) cita a seguinte relação:

Padrões criacionais

- **Padrão de fábrica abstrata (factory pattern):** centraliza a decisão de que fábrica instanciar.
- **Padrão de métodos de fábrica (factory method pattern):** centraliza a criação de um objeto de um tipo específico escolhendo uma de várias implementações.
- **Padrão construtor (builder pattern):** separa a construção de um objeto complexo de sua representação de modo que o mesmo processo de construção possa criar diferentes representações.
- **Padrão de protótipo (prototype pattern):** usado quando o custo inerente de criação de um novo objeto da maneira padrão (por exemplo, usando a palavra-chave "new") é proibitivo para uma dada aplicação.
- **Padrão único (singleton pattern):** restringe a instanciação de uma classe a um único objeto.

Padrões estruturais

- **Padrão adaptador (adapter pattern):** "adapta" uma interface de uma classe para uma que um cliente espera.
- **Padrão de agregação (aggregate pattern):** uma versão do padrão de composição com métodos para agregação de objetos filhos.
- **Padrão ponte (bridge pattern):** desacopla uma abstração de sua implementação de modo que as duas possam variar independentemente.
- **Padrão de composição (composite pattern):** uma estrutura de objetos em forma de árvore onde cada objeto possui a mesma interface.
- **Padrão container (container pattern):** cria objetos com o propósito exclusivo de conter outros objetos e gerenciá-los.

INFORMAÇÕES

- **Padrão proxy (proxy pattern):** uma classe atuando como uma interface para outra.
- **tubos e filtros (pipes and filters patterns):** uma cadeia de processos onde a saída de cada processo é a entrada do seguinte.

Padrões comportamentais

- **Padrão de cadeia de responsabilidades (chain of responsibility pattern):** Objetos e comando são manipulados ou passados para outros objetos através de objetos de processamento com lógica.
- **Padrão de comandos (command pattern):** Objetos de comando encapsulam uma ação e seus parâmetros.
- **Escutador de eventos (event listener):** Os dados são distribuídos a objetos que se registraram para recebê-los.
- **Padrão interpretador (interpreter pattern):** Implementa uma linguagem computacional especializada para resolver rapidamente um conjunto de problemas específicos.
- **Padrão iterador (iterator pattern):** Os iteradores são usados para acessar sequencialmente os elementos de um objeto agregado sem expor sua representação subjacente.
- **Padrão mediador (mediator pattern):** fornece uma interface unificada para um conjunto de interfaces em um subsistema.
- **Padrão visitante (visitor pattern):** Uma forma de separar um algoritmo de um objeto.
- **Padrão visitante para atendimento único (single-serving visitor pattern):** Otimiza a implementação de um visitante que é alocado, usado apenas uma vez e depois deletado.
- **Padrão visitante hierárquico (hierarchical visitor pattern):** Fornece uma forma de visitar todos os nós em uma estrutura de dados hierárquica como, por exemplo, uma árvore.

Descrições detalhadas de cada um desses padrões podem ser obtidas via links em www.wikipedia.org.

3 Gamma e seus colegas [Gam95] são normalmente conhecidos como a "Gang of Four" (GoF) na literatura sobre padrões.

Os *padrões criacionais* concentram-se na “criação, composição e representação” de objetos. Gamma e seus colegas [Gam95] observam que os padrões criacionais “encapsulam o conhecimento sobre quais classes concretas o sistema usa”, porém, ao mesmo tempo, “ocultam como instâncias dessas classes são criadas e combinadas”. Os padrões criacionais dispõem de mecanismos que facilitam a instanciação de objetos em um sistema e impõem “restrições sobre o tipo e número de objetos que podem ser criados em um sistema” [Maa07].

Os *padrões estruturais* focalizam problemas e soluções associadas a como classes e objetos são organizados e integrados para construir uma estrutura maior. Em essência, ajudam a estabelecer relações entre entidades em um sistema. Por exemplo, padrões estruturais que se concentram em questões orientadas a classes poderiam fornecer mecanismos de herança que levem a interfaces de programa mais eficazes. Padrões estruturais que se concentram em objetos sugerem técnicas para combinar objetos em outros objetos ou integrar objetos em uma estrutura maior.

Os *padrões comportamentais* tratam de problemas associados à atribuição de responsabilidade entre objetos e a maneira pela qual a comunicação é efetuada entre objetos.

12.1.2 Estruturas de uso de padrões de projeto

Os próprios padrões talvez não sejam suficientes para desenvolver um projeto completo. Em alguns casos pode ser necessário fornecer uma infraestrutura mínima específica a uma implementação, para trabalho de projeto. Podemos selecionar uma “*mini-arquitetura reutilizável*” que fornece a estrutura genérica e o comportamento para uma família de abstrações de software, juntamente com um contexto... Que especifica sua colaboração e uso em um domínio de dados” [Amb98].

Essa estrutura de uso de padrões⁴ não é um padrão de arquitetura, mas sim um esqueleto com um conjunto de “pontos de conexão” (também chamados *ganchos* e *encaixes*) que permitem-lhe ser adaptada a um domínio de problemas específico. Os pontos de conexão possibilitam que integremos ao esqueleto classes ou funcionalidades específicas de um problema. Em um contexto orientado a objetos, uma estrutura é um conjunto de classes que cooperam entre si.

Gamma e seus colegas [Gam95] descrevem as diferenças entre padrões de projeto e estruturas de uso de padrões da seguinte maneira:

1. Os *padrões de projeto* são mais abstratos que as *estruturas de uso*. As estruturas de uso de padrões podem ser incorporadas ao código, mas apenas *exemplos* de padrões podem ser incorporados ao código. Um ponto forte das estruturas de uso de padrões é que podem ser escritas em linguagens de programação e não só estudadas, mas executadas e reutilizadas diretamente.
2. Os *padrões de projeto* são *elementos arquiteturais menores* que as *estruturas de uso de padrões*. Essas estruturas podem conter vários padrões de projeto, mas o contrário jamais é verdadeiro.
3. Os *padrões de projeto* são *menos especializados* que as *estruturas de uso de padrões*. Elas apresentam um domínio de aplicação particular. Diferentemente, os padrões de projeto podem ser usados em praticamente qualquer tipo de aplicação. Embora sejam possíveis padrões de projeto mais especializados, mesmos estes não ditariam uma arquitetura de aplicação.

Em essência, o projetista de uma estrutura de uso de padrão argumentará que uma *mini-arquitetura reutilizável* se aplica a todo software a ser desenvolvido em um domínio de aplicação limitado. Para serem mais efetivas, essas estruturas são aplicadas sem nenhuma alteração. Podem-se acrescentar outros elementos de projeto, mas apenas através de pontos de conexão que permitem ao projetista dar corpo ao esqueleto dessas estruturas.

12.1.3 Descrição de padrões

O projeto baseado em padrões começa com o reconhecimento de padrões na aplicação que se pretende construir, continua com a pesquisa para determinar se outros trataram o padrão


PONTO-CHAVE

Estrutura de uso de padrões é uma “mini-arquitetura” reutilizável que serve como base e a partir da qual outros padrões de projeto podem ser aplicados.

⁴ N. de R.T.: Uma estrutura de uso de padrões inclui aspectos estáticos/estruturais e dinâmicos/comportamentais que dão suporte ao uso dos padrões de projeto.

e termina com a aplicação de um padrão apropriado para o problema em questão. Em geral, a segunda dessas três tarefas é a mais difícil. Como encontrar padrões que atendam nossas necessidades?

A resposta deve depender da comunicação efetiva do problema que o padrão trata, do contexto em que o padrão reside, do sistema de forças que molda o contexto e da solução proposta. Para transmitir essas informações de forma inequívoca, é necessário um formulário ou template padronizado para descrições de padrão. Embora tenham sido propostos vários templates de padrão distintos, quase todos contêm um subconjunto principal do conteúdo sugerido por Gamma e seus colegas [Gam95]. No quadro a seguir é mostrado um template de padrão simplificado.

Template de padrões de projeto		INFORMAÇÕES
	<p>Nome do padrão — descreve a essência do padrão em um nome curto mas expressivo.</p> <p>Problema — descreve o problema de que o padrão trata.</p> <p>Motivação — dá um exemplo do problema.</p> <p>Contexto — descreve o ambiente onde o problema reside incluindo o domínio de aplicação.</p> <p>Forças — enumera o sistema de forças que afetam a maneira pela qual o problema deve ser resolvido; inclui uma discussão das limitações e restrições que devem ser consideradas.</p> <p>Solução — fornece uma descrição detalhada de uma solução proposta para o problema.</p>	<p>Objetivo — descreve o padrão e o que ele faz.</p> <p>Colaborações — descrevem como outros padrões contribuem para uma solução.</p> <p>Consequências — descrevem os possíveis prós e contras que devem ser considerados quando o padrão é implementado e as consequências do uso do padrão.</p> <p>Implementação — identifica questões especiais que devem ser consideradas ao se implementar o padrão.</p> <p>Usos conhecidos — dá exemplos de usos práticos do padrão de projeto em aplicações reais.</p> <p>Padrões relacionados — remetem a padrões de projeto relacionados.</p>

“Os padrões são semiprontos — isso significa que sempre temos de finalizá-los e adaptá-los ao nosso ambiente.”
Martin Fowler

Os nomes de padrões de projeto devem ser escolhidos com cuidado. Um dos principais problemas técnicos em projeto baseado em padrões é a inabilidade de encontrar padrões existentes quando há centenas ou milhares de candidatos. A busca do padrão “correto” é facilitada inmensuravelmente por um nome de padrão significativo.

Um template de padrões fornece um meio padronizado para descrição de padrões de projeto. Cada uma das entradas do template representa características do padrão de projeto que podem ser pesquisadas (por exemplo, por um banco de dados) e a maneira pela qual o padrão apropriado pode ser encontrado.

12.1.4 Linguagens e repositórios de padrões

Quando usamos o termo *linguagem*, a primeira coisa que nos vêm à mente é uma linguagem natural (por exemplo, inglês, espanhol, chinês) ou uma linguagem de programação (por exemplo, C++, Java). Em ambos os casos, a linguagem possui uma sintaxe e semântica usadas para transmitir ideias ou instruções procedurais de forma efetiva.

Quando se usa o termo *linguagem* no contexto de padrões de projeto, ele assume um significado ligeiramente diferente. Uma *linguagem de padrões* engloba um conjunto de padrões, cada qual descrito através do uso de um template de padrões (Seção 12.1.3) e inter-relacionados para mostrar como esses padrões colaboram para solucionar problemas em um domínio de aplicação.⁵

Em uma linguagem natural, as palavras são organizadas em sentenças que transmitem significado. A estrutura de sentenças é descrita pela sintaxe da linguagem. Em uma linguagem de padrões, os padrões de projeto são organizados para fornecer um “método estruturado para descrição de práticas de projeto adequadas em um domínio”.⁶

⁵ Christopher Alexander propôs originalmente linguagens de padrões para arquitetura e planejamento urbano. Hoje, as linguagens de padrões foram desenvolvidas para diversos setores, das ciências sociais ao processo de engenharia de software.

⁶ Essa descrição da Wikipedia pode ser encontrada em http://en.wikipedia.org/wiki/Pattern_language.



Caso não consiga encontrar uma linguagem de padrões que trate do domínio de seu problema, procure analogias em um outro conjunto de padrões.

WebRef

Para uma lista de linguagens de padrões ótimos refaça o c2.com/ppr/titles.html. Informações adicionais podem ser obtidas em hillside.net/patterns/.

De certa maneira, uma linguagem de padrões é análoga a um manual de instruções em hipertexto para resolução de problemas em áreas de aplicação específicas. O domínio do problema é descrito primeiro em termos hierárquicos, começando com problemas de projeto abrangentes associados ao domínio e então refinando-se cada um dos problemas abrangentes em níveis de abstração menores. No contexto de software, problemas de projeto abrangentes tendem a ser problemas de arquitetura por natureza e tratam a estrutura geral da aplicação e os dados ou conteúdo que o atendem. Os problemas de arquitetura são refinados para níveis de abstração menores, levando a padrões de projeto que resolvem subproblemas e colaboram entre si no nível de componentes (ou classes). Em vez de uma lista sequencial de padrões, a linguagem de padrões representa um conjunto interconectado em que o usuário pode partir de um problema de projeto abrangente e “ir fundo” para revelar problemas específicos e suas soluções.

Dezenas de linguagens de padrões foram propostas para projeto de software [Hil08]. Na maioria dos casos, os padrões de projeto que fazem parte da linguagem de padrões são armazenados em um repositório de padrões acessado via Web (por exemplo, [Boo08], [Cha03], [HPR02]). O repositório fornece um índice de todos os padrões de projeto e contém links de hipermídia que permitem ao usuário compreender as colaborações entre padrões.

12.2 PROJETO DE SOFTWARE BASEADO EM PADRÕES

Os melhores projetistas de qualquer área têm uma habilidade extraordinária de vislumbrar padrões que caracterizam um problema e padrões correspondentes que podem ser combinados para criar a solução. Os desenvolvedores de software da Microsoft [Mic04] discutem isso ao escreverem:

Embora o projeto baseado em padrões seja relativamente novo no campo de desenvolvimento de software, a tecnologia industrial tem usado projeto baseado em padrões há décadas, talvez há séculos. Catálogos de mecanismos e configurações padronizadas fornecem elementos de projeto que são usados para criar automóveis, aeronaves, máquinas-ferramenta e robôs. A aplicação de projeto baseado em padrões ao desenvolvimento de software promete os mesmos benefícios para o software como os já proporcionados à tecnologia industrial: previsibilidade, redução de riscos e maior produtividade.

Ao longo do processo de projeto, devemos buscar toda oportunidade de aplicar padrões de projeto existentes (quando atenderem às necessidades do projeto) em vez de criar novos.

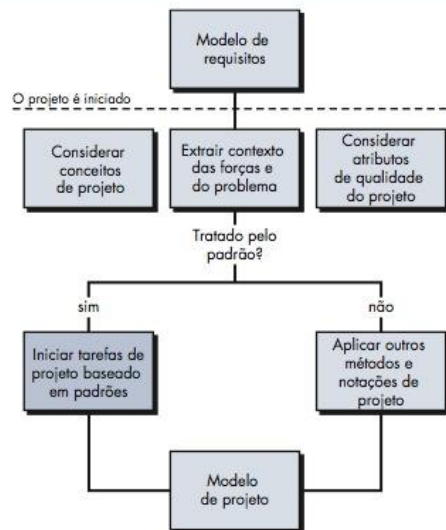
12.2.1 Contexto do projeto baseado em padrões

O projeto baseado em padrões não é utilizado isoladamente. Os conceitos e as técnicas discutidas para projeto da arquitetura, de componentes e para interfaces do usuário (Capítulos 9 a 11) são usados em conjunto com uma abordagem baseada em padrões. No Capítulo 8, citei que um conjunto de diretrizes e atributos de qualidade serve como base para todas as decisões de projeto de software. As próprias decisões são influenciadas por um conjunto de conceitos de projeto fundamentais (por exemplo, separação de preocupações, refinamento gradual, independência funcional) que são atingidos usando-se heurística que evoluiu ao longo de várias décadas e práticas melhores (por exemplo, técnicas, notação de modelagem) propostas para fazer com que o projeto seja mais fácil de ser realizado e mais efetivo como base para a construção.

O papel do projeto baseado em padrões está ilustrado na Figura 12.1. Um projetista de software inicia com um modelo de requisitos (seja ele explícito, seja implícito) que apresenta uma representação abstrata do sistema. O modelo de requisitos descreve o conjunto de problemas, estabelece o contexto e identifica o sistema de forças que exerce domínio. Talvez sugira o projeto de maneira abstrata, mas o modelo de requisitos faz pouco para representar o projeto explicitamente.

Ao iniciar seu trabalho como projetista, é sempre importante manter os atributos de qualidade em mente. Esses atributos (por exemplo, um projeto deve implementar todos os requisitos

FIGURA 12.1
Contexto do
projeto baseado
em padrões



explícitos tratados no modelo de requisitos) estabelecem uma maneira de avaliar a qualidade do software, mas pouco fazem para ajudar a atingi-lo efetivamente. O projeto criado deve apresentar os conceitos fundamentais de projeto discutidos no Capítulo 8. Consequentemente, devemos aplicar técnicas comprovadas para traduzir as abstrações contidas no modelo de requisitos de maneira mais concreta que é o projeto de software. Para tanto, usaremos os métodos e as ferramentas de modelagem disponíveis para projeto da arquitetura, de componentes e para interfaces. Mas apenas quando depararmos com um problema, um contexto e um sistema de forças que ainda não foram resolvidos anteriormente. Se já existir uma solução, use-a! E isso significa aplicar uma abordagem de projeto baseado em padrões.

12.2.2 Pensando em termos de padrões


Em um excelente livro sobre projeto baseado em padrões, Shalloway e Trott [Sha05] comentam sobre uma “nova maneira de pensar” ao usarmos padrões como parte da atividade de projeto:

Eu tinha que me tornar mais receptivo a uma nova maneira de pensar. E quando fiz isso, ouvi [Christopher] Alexander dizer que “um bom projeto de software não pode ser alcançado simplesmente juntando peças operantes”.

Um bom projeto começa levando-se em conta o contexto — a visão geral. Quando o contexto é avaliado, extraímos uma hierarquia de problemas que devem ser resolvidos. Alguns desses problemas serão de natureza global, enquanto outros tratarão características e funções específicas do software. Todos serão afetados por um sistema de forças que irá influenciar a natureza de uma solução proposta.

Shalloway e Trott [Sha05] sugerem a seguinte abordagem⁷ que permite a um projetista pensar em termos de padrões:

⁷ Baseado no trabalho de Christopher Alexander [Ale79].

 O projeto baseado em padrões parece interessante para o problema que devo resolver. Por onde devo começar?

1. Certifique-se de ter entendido o quadro geral – o contexto em que o software a ser construído reside. O modelo de requisitos deve comunicar isso a você.
2. Examine o quadro geral, extraia os padrões presentes neste nível de abstração.
3. Inicie seu projeto com os padrões do “quadro geral” que estabeleçam um contexto ou esqueleto para trabalho de projeto posterior.
4. “Trabalhe em direção à essência, partindo do contexto” [Sha05] buscando padrões em níveis de abstração mais baixos que contribuam para a solução de projeto.
5. Repita os passos 1 a 4 até que o projeto completo ganhe corpo.
6. Refina o projeto adaptando cada padrão às especificidades do software que está tentando construir.

É importante notar que os padrões não são entidades independentes. Os padrões de projeto que estão presentes em nível de abstração elevado influenciarão invariavelmente a maneira pela qual outros padrões serão aplicados em níveis de abstração mais baixos. Além disso, os padrões em geral colaboram entre si. A implicação — ao selecionar um padrão de arquitetura, ele poderá influenciar os padrões de projeto escolhidos no nível de componentes. Da mesma forma, ao selecionar um padrão de projeto específico para interfaces, muitas vezes você se vê forçado a usar outros padrões que colaboram com ele.

Para ilustrarmos, consideremos a WebApp **CasaSeguraGarantida.com**. Se considerarmos o quadro geral, a WebApp deve tratar uma série de problemas fundamentais como:


- Como fornecer informações sobre os produtos e serviços *CasaSegura*.
- Como vender aos clientes os produtos e serviços *CasaSegura*.
- Como estabelecer monitoramento e controle baseado em Internet de um sistema de segurança instalado.

Cada um desses problemas fundamentais pode ser refinado ainda mais em um conjunto de subproblemas. Por exemplo, *Como vender via Internet* implica um padrão **E-commerce (para comércio eletrônico)** que por si só implica um grande número de padrões em níveis de abstração mais baixos. O padrão **E-commerce** (provavelmente, um padrão de arquitetura) implica mecanismos para configurar uma conta de cliente, exibir os produtos a ser vendidos, selecionar produtos para compra e assim por diante. Portanto, se pensarmos em padrões, é importante determinar se um padrão para configurar uma conta existe ou não. Se **ConfigurarConta** estiver disponível como um padrão viável para o contexto do problema, talvez ele colabore com outros padrões como **CriarFormulárioDeEntrada**, **GerenciarPreenchimentoDeFormulários** e **ValidarPreenchimentoFormulários**. Cada um desses padrões delinea problemas a ser resolvidos e soluções que possam ser aplicadas.

12.2.3 Tarefas de projeto

As tarefas de projeto a seguir são aplicadas quando se adota uma filosofia de projeto baseado em padrões:

1. **Examine o modelo de requisitos e desenvolva uma hierarquia de problemas.**
Descreva cada problema e subproblema isolando o problema, o contexto e o sistema de forças que se aplicam. Trabalhe inicialmente em problemas mais abrangentes (nível de abstração elevado) indo depois para subproblemas menores (em níveis de abstração mais baixos).
2. **Determine se uma linguagem de padrões confiável foi desenvolvida para o domínio do problema.** Conforme observado na Seção 12.1.4, uma linguagem de padrões trata problemas associados a um domínio de aplicação específico. A equipe de software do *CasaSegura* procuraria uma linguagem de padrões desenvolvida especificamente para produtos de segurança domiciliar. Se esse nível de especificidade de linguagem de padrões não puder ser encontrado, a equipe subdividiria o problema de software *CasaSegura* em uma série de

 Quais são as tarefas necessárias para criar um projeto baseado em padrões?

domínios de problemas genéricos (por exemplo, problemas de monitoramento de dispositivos digitais, problemas de interfaces do usuário, problemas de gerenciamento de vídeo digital) e buscaria linguagens de padrões apropriadas.

3. **Inicie com um problema abrangente, determine se um ou mais padrões de arquitetura se encontram disponíveis para ele.** Se existir um padrão de arquitetura, certifique-se de examinar todos os padrões colaboradores. Se o padrão for apropriado, adapte a solução de projeto proposta e construa um elemento de modelo de projeto que o represente adequadamente. Conforme observado na Seção 12.2.2, um problema abrangente para a Web-App **CasaSeguraGarantida.com** é tratada com um padrão **E-commerce** (de comércio eletrônico). Este sugerirá uma arquitetura específica para lidar com os requisitos de comércio eletrônico.
4. **Use as colaborações fornecidas para o padrão de arquitetura, examine problemas de subsistemas ou de componentes e procure padrões apropriados para tratar deles.** Talvez seja necessário pesquisar outros repositórios de padrões, bem como a lista de padrões que corresponde à solução de arquitetura. Se for encontrado um padrão apropriado, adapte a solução de projeto proposta e construa um elemento de modelo de projeto que o represente adequadamente. Certifique-se de aplicar o passo 7.
5. **Repita os passos 2 a 5 até que todos os problemas mais amplos tenham sido tratados.** A implicação é começar com o quadro geral e elaborar para resolver problemas em níveis cada vez mais detalhados.
6. **Se os problemas de projeto para interfaces do usuário tiverem sido isolados (quase sempre esse é o caso), pesquise os vários repositórios de padrões de projeto para interfaces do usuário em busca de padrões apropriados.** Prossiga de maneira similar para os passos 3, 4 e 5.
7. **Independentemente de seu nível de abstração, se uma linguagem de padrões e/ou repositório de padrões ou padrão individual se mostrarem promissores, compare o problema a ser resolvido em relação ao(s) padrão(ões) existente(s) apresentado(s).** Certifique-se de examinar o contexto e as forças para garantir que o padrão forneça, de fato, uma solução suscetível para o problema.
8. **Certifique-se de refinar o projeto à medida que é obtido de padrões usando critérios de qualidade de projeto como guia.**

Embora essa abordagem de projeto seja *top-down*, as soluções de projeto na vida real são, algumas vezes, mais complexas. Gillis [Gil06] tece comentários sobre isso ao escrever:

Os padrões de projeto na engenharia de software destinam-se ao uso racional e dedutivo. Portanto, temos o problema ou requisito geral, X, o padrão de projeto Y resolve X, consequentemente usamos Y. Agora, ao refletir sobre meu próprio processo — e tenho razões para acreditar que não sou o único a pensar assim — descubro que é mais orgânico que isso, mais indutivo que dedutivo, mais *bottom-up* que *top-down*. Obviamente, há um equilíbrio a ser atingido. Quando um projeto se encontra na fase inicial de partida e estou tentando passar dos requisitos abstratos para uma solução de projeto concreta, em geral realizarei uma busca inicial de grande amplitude... Achei que os padrões de projeto são úteis, permitindo-me rapidamente formular o problema de projeto em termos concretos.

Além disso, a abordagem baseada em padrões deve ser usada em conjunto com outros conceitos e técnicas de projeto de software.



As entradas na tabela podem ser complementadas com uma indicação da aplicabilidade relativa do padrão.

12.2.4 Construção de uma tabela para organização de padrões

À medida que o projeto baseado em padrões prossegue, talvez encontremos problemas para organizar e classificar prováveis padrões contidos em vários repositórios e linguagens de padrões. Para auxiliar a organizar nossa avaliação de prováveis padrões, a Microsoft [Mic04] sugere a criação de uma *tabela para organização de padrões* que assume a forma geral indicada na Figura 12.2.

FIGURA 12.2

Uma tabela para organização de padrões

Fonte: Adaptado de (Mic04)

	Banco de dados	Aplicação	Implementação	Infraestrutura
Dados/Conteúdo				
Enunciado do problema...	Nome(s)do(s)Padrão(ões)		Nome(s)do(s)Padrão(ões)	
Enunciado do problema...		Nome(s)do(s)Padrão(ões)		Nome(s)do(s)Padrão(ões)
Enunciado do Problema...	Nome(s)do(s)Padrão(ões)			Nome(s)do(s)Padrão(ões)
Arquitetura				
Enunciado do problema...		Nome(s)do(s)Padrão(ões)		
Enunciado do problema...		Nome(s)do(s)Padrão(ões)		Nome(s)do(s)Padrão(ões)
Enunciado do problema...				
Componentes				
Enunciado do problema...		Nome(s)do(s)Padrão(ões)	Nome(s)do(s)Padrão(ões)	
Enunciado do problema...				Nome(s)do(s)Padrão(ões)
Enunciado do problema...		Nome(s)do(s)Padrão(ões)	Nome(s)do(s)Padrão(ões)	
Interface do usuário				
Enunciado do problema...		Nome(s)do(s)Padrão(ões)	Nome(s)do(s)Padrão(ões)	
Enunciado do problema...		Nome(s)do(s)Padrão(ões)	Nome(s)do(s)Padrão(ões)	
Enunciado do problema...		Nome(s)do(s)Padrão(ões)	Nome(s)do(s)Padrão(ões)	

Uma tabela para organização de padrões pode ser implementada como um modelo de planilha usando o formato mostrado na figura. Uma lista resumida dos enunciados dos problemas, organizados por tópicos como dados/conteúdo, arquitetura, componentes e interfaces do usuário, é apresentada na coluna mais à esquerda (sobra mais escura). Quatro tipos padrão — bancos de dados, aplicação, implementação e infraestrutura — são listados ao longo da linha superior. Os nomes de prováveis padrões são indicados nas células da tabela.

Para fornecermos entradas para a tabela organizadora, pesquisaremos várias linguagens e repositórios de padrões em busca de padrões que atendam determinado enunciado de problema. Quando forem encontrados um ou mais prováveis padrões, estes são introduzidos na linha correspondente ao enunciado do problema e a coluna correspondente ao tipo de padrão. O nome do padrão é introduzido como um hiperlink para o URL do endereço Web que contém uma descrição completa do padrão.

12.2.5 Erros comuns de projeto

O projeto baseado em padrões pode lhe tornar um melhor projetista de software, mas ele *não* é uma panaceia. Como qualquer método de projeto, temos de começar com os primeiros princípios, enfatizando os fundamentos da qualidade de software e garantindo que o projeto atenda, de fato, às necessidades expressas pelo modelo de requisitos.

Uma série de erros comuns ocorre quando se usa projeto baseado em padrões. Em alguns casos, não se dedicou tempo suficiente para entender o problema subjacente e seu contexto e forças e, como consequência, escolhe-se um padrão que parece adequado, porém, é inadequado para a solução exigida. Assim que o padrão incorreto é escolhido, nos recusamos a admitir o erro e forçamos a adequação do padrão. Em outros casos, o problema possui forças não consideradas pelo padrão escolhido, resultando uma adequação deficiente ou errônea.

Algumas vezes um padrão é aplicado de forma demasiadamente literal e as adaptações necessárias para o espaço do problema não são implementadas. Esses erros poderiam ser evita-



AVISO
Não force um padrão, mesmo que atenda ao problema em questão. Se o contexto e as forças estiverem errados, procure outro padrão.

dos? Na maioria dos casos a resposta é “sim”. Todo bom projetista pede uma segunda opinião e aceita revisões em seu trabalho. As técnicas de revisão discutidas no Capítulo 15 podem ajudar a garantir que o projeto baseado em padrões que desenvolvemos resultará em uma solução de alta qualidade para o problema de software a ser solucionado.

12.3 PADRÕES DE ARQUITETURA

PONTO-CHAVE

Uma arquitetura de software pode ter uma série de padrões de arquitetura que tratam questões como concorrência, persistência e distribuição.

? Quais os domínios de padrões de arquitetura típicos?

Se um engenheiro civil decide construir uma casa colonial com hall central, existe um único estilo arquitetônico a ser aplicado. Os detalhes do estilo (por exemplo, número de lajeiras, fachada da casa, posição das portas e janelas) podem variar consideravelmente, mas uma vez tomada a decisão sobre a arquitetura geral da casa, o estilo é imposto sobre o projeto.⁸

Os padrões de arquitetura são um pouco diferentes. Por exemplo, toda casa (e todo estilo arquitetônico para casas) usa um padrão **Kitchen** (cozinha). O padrão **Kitchen** e padrões com os quais ele colabora atendem problemas associados ao armazenamento e à preparação de alimentos, os utensílios necessários para cumprir essas tarefas e regras para a colocação dos utensílios em relação ao fluxo no ambiente. Além disso, o padrão poderia atender problemas associados a balcões, iluminação, interruptores, uma ilha central, pisos e assim por diante. Obviamente, há mais de um projeto para uma cozinha, em geral ditado pelo contexto e sistema de forças. Mas todo projeto pode ser concebido no contexto da “solução” sugerida pelo padrão **Kitchen**.

Conforme citado, os padrões de arquitetura para software definem uma abordagem específica para tratar alguma característica do sistema. Bosch [Bos00] e Booch [Boo08] definem uma série de domínios de padrões de arquitetura. Exemplos representativos são apresentados nos parágrafos seguintes:

Controle de acesso. Há várias situações em que o acesso a dados, recursos e funcionalidade fornecidos por uma aplicação é limitado a usuários finais especificamente definidos. Do ponto de vista da arquitetura, o acesso a alguma parte da arquitetura de software deve ser rigorosamente controlado.

Concorrência. Muitas aplicações têm de tratar múltiplas tarefas em um modo que simule paralelismo (isso ocorre sempre que vários componentes ou tarefas “paralelos” são administrados por um único processador). Há uma série de maneiras diferentes com a qual uma aplicação pode tratar a concorrência, e cada uma delas pode ser apresentada por um padrão de arquitetura distinto. Por exemplo, uma abordagem é usar um padrão **Operating-System-ProcessManagement** (sistema operacional-gerenciamento de processos) que fornece recursos embutidos no sistema operacional que permitem aos componentes executarem de forma concorrente. O padrão também incorpora funcionalidade que gerencia a comunicação entre processos, agendamento e outras capacidades exigidas para alcançar a concorrência. Uma outra abordagem poderia ser definir um agendador de tarefas no nível da aplicação. Um padrão **TaskScheduler** (agendador de tarefas) contém um conjunto de objetos ativos em que cada um contém uma operação *tick()* [Bos00]. O agendador chama periodicamente *tick()* para cada objeto, que então realiza as funções que ele deve realizar antes de retornar o controle de volta para o agendador, que então chama a operação *tick()* para o próximo objeto concorrente.

Distribuição. O problema da distribuição trata a maneira pela qual os sistemas ou componentes nos sistemas se comunicam entre si em um ambiente distribuído. São considerados dois subproblemas: (1) a maneira pela qual as entidades se conectam entre si e (2) a natureza

⁸ Isso implica que haverá um hall central e um corredor, que as salas serão colocadas à esquerda e à direita do hall, que a casa terá dois (ou mais) pisos, que os quartos da casa estarão no piso de cima e assim por diante. Essas “regras” são impostas uma vez tomada a decisão de adotar o estilo colonial com hall central.

da comunicação que ocorre. O padrão de arquitetura mais comum estabelecido para tratar o problema de distribuição é o padrão **Broker** (agente). Um agente atua com um “intermediário” entre o componente-cliente e o componente-servidor. O cliente envia uma mensagem ao agente (contendo todas as informações apropriadas para a comunicação a ser efetuada) e o agente completa a conexão.

Persistência. Os dados persistem se sobreviverem depois da execução do processo que o criou. Os dados persistentes armazenados em um banco de dados ou arquivo podem ser lidos ou modificados por outros processos posteriormente. Em ambientes orientados a objetos, a ideia de um objeto persistente estende um pouco mais o conceito de persistência. Os valores de todos os atributos do objeto, o estado geral do objeto e outras informações complementares são armazenados para recuperação e uso futuros. Em geral, usam-se dois padrões de arquitetura para obter a persistência — o padrão **DatabaseManagementSystem** (sistema de gerenciamento de bancos de dados), que aplica o recurso de armazenamento e recuperação de um DBMS à arquitetura da aplicação, ou o padrão **Application Level-Persistence** (persistência no nível de aplicação) que constrói recursos de persistência na arquitetura da aplicação (por exemplo, software de processamento de texto que gerencia sua própria estrutura de documentos).

Antes que qualquer um dos padrões de arquitetura representativos citados nos parágrafos anteriores possa ser escolhido, ele deve ser avaliado em termos de sua adequação para a aplicação e estilo de arquitetura geral, bem como o contexto e o sistema de forças que ele especifica.



Repositórios de padrões de projeto

Existem várias fontes de padrões de projeto disponíveis na Web. Alguns padrões podem ser obtidos de linguagens de padrões publicadas individualmente, enquanto outros se encontram disponíveis como parte de um portal ou repositório de padrões. Vale a pena dar uma olhada nas seguintes fontes na Web:

Hillside.net <http://hillside.net/patterns/> Um dos conjuntos mais completos da Web em termos de padrões e linguagens de padrões.

Portland Pattern Repository

<http://c2.com/ppr/index.html>

Contém links para uma ampla gama de recursos e coleções de padrões.

Pattern Index <http://c2.com/cgi/wiki?PatternIndex>

Uma “coleção de padrões eclética”

Booch's Architecture Patterns Handbook

www.booch.com/architecture/index.jsp

Referência bibliográfica para centenas de padrões de projeto de arquitetura e componentes

Coleções de padrões para interfaces do usuário

Padrões para UI/HCI

www.hcipatterns.org/patterns.html

Jennifer Tidwell's UI patterns

www.time-tripper.com/uipatterns/

Padrões de projeto para interfaces do usuário móveis

<http://patterns.littlespringsdesign.com/wikka.php?wakka=Mobile>

Padrões

Linguagem de padrões para projeto de interfaces do usuário

www.maplefish.com/todd/papers/Experiences.html

Biblioteca de projeto interativo para jogos

www.eelke.com/research/usability.html

Padrões de projeto para interfaces do usuário

www.cs.helsinki.fi/u/salaakso/patterns/

Padrões de projeto especializados:

Aviônica

<http://g.oswego.edu/dl/acs/acs.html>

Sistemas de informações de negócios

www.objectarchitects.de/arcus/cookbook/

Processamento distribuído

www.cs.wustl.edu/~schmidt/

Padrões IBM para e-Business

www128.ibm.com/developerworks/patterns/

Biblioteca de padrões de projeto do Yahoo!

<http://developer.yahoo.com/ypatterns/>

WebPatterns.org <http://webpatterns.org/>

INFORMAÇÕES

12.4 PADRÕES DE PROJETO DE COMPONENTES

Os padrões de projeto de componentes nos dão soluções comprovadas que tratam um ou mais subproblemas extraídos do modelo de requisitos. Em muitos casos, os padrões de projeto desse tipo se concentram em algum elemento funcional de um sistema. Por exemplo, a aplicação **CasaSeguraGarantida.com** deve tratar o seguinte subproblema de projeto: *Como podemos obter especificações de produtos e informações relacionadas para qualquer dispositivo do CasaSegura?*

Tendo enunciado o subproblema que deve ser resolvido, devemos considerar agora o contexto e o sistema de forças que afetam uma solução. Examinando-se o caso de uso de modelo de requisitos apropriado, constatamos que o consumidor usa a especificação para um dispositivo do CasaSegura (por exemplo, um sensor ou câmera de segurança) para fins de informação. Entretanto, outras informações relacionadas à especificação (por exemplo, determinação de preços) poderiam ser empregadas quando a funcionalidade de comércio eletrônico fosse selecionada.

A solução para o subproblema envolve uma pesquisa. Como pesquisar é um problema muito comum, não deve ser nenhuma surpresa a existência de muitos padrões relacionados à pesquisa. Pesquisando uma série de repositórios de padrões, descobriremos os seguintes padrões, juntamente com o problema que cada um resolve:

AdvancedSearch. Os usuários precisam encontrar um item específico em um grande conjunto de itens.

HelpWizard. Os usuários precisam de ajuda sobre certo tópico relativo ao site ou quando eles precisam encontrar uma página específica dentro desse site.

SearchArea. Os usuários precisam encontrar uma página Web.

SearchTips. Os usuários precisam saber como controlar o mecanismo de busca.

SearchResults. Os usuários têm de processar uma lista de resultados de uma busca.

SearchBox. Os usuários têm de encontrar um item ou informações específicas.

Para **CasaSeguraGarantida.com** o número de produtos não é particularmente grande e cada um deles possui uma classificação relativamente simples, de modo que **AdvancedSearch** e **HelpWizard** provavelmente não sejam necessários. Similarmente, a busca é relativamente simples para não precisar de **SearchTips**. A descrição de **SearchBox**, entretanto, é dada (em parte) como:

Search Box

(Adaptado de www.welie.com/patterns/showPattern.php?patternID=search.)

Problema: Os usuários precisam encontrar um item ou informações específicos.

Motivação: Qualquer situação em que uma busca com palavra-chave é aplicada em um conjunto de objetos de conteúdo organizados na forma de páginas Web.

Contexto: Em vez de usar navegação para obter informações ou conteúdo, o usuário quer fazer uma busca direta em conteúdo contido em várias páginas Web. Qualquer site que já possui recursos primários de navegação. O usuário poderia querer procurar um item numa categoria. Ele poderia querer especificar uma consulta de forma mais detalhada.

Forças: O site já possui recursos primários de navegação. Os usuários talvez queiram procurar um item de determinada categoria. Eles poderiam querer especificar uma consulta, de forma mais detalhada, através de operadores booleanos simples.

Solução: Oferecer funcionalidade de busca formada por um identificador de busca, um campo para palavra-chave, um filtro se aplicável e um botão “avançar”. Pressionar a tecla Enter tem a mesma função que selecionar o botão Avançar. Também fornece Dicas de Busca e exemplos em uma página separada. Um link para essa página é colocado próximo à funcionalidade de busca. A caixa de edição para o termo de pesquisa é suficientemente grande para acomodar três consultas de usuário típicas (normalmente, por volta de 20 caracteres). Se o número de filtros for maior que 2, use uma caixa de combinação para seleção de filtros ou um botão de rádio.

Os resultados da pesquisa são apresentados em uma nova página com um identificador claro contendo pelo menos “Resultados da busca” ou algo similar. A função de busca é repetida na parte superior da página com as palavras-chave anteriormente introduzidas, de modo que os usuários saibam quais eram elas.

A descrição de padrões continua com outras entradas conforme descrito na Seção 12.1.3.

O padrão prossegue para descrever como os resultados da busca são acessados, apresentados, correspondidos e assim por diante. Baseado nisso, a equipe do **CasaSeguraGarantida.com** pode projetar os componentes necessários para implementar a busca ou (mais provavelmente) obter componentes reutilizáveis existentes.

CASA SEGURA



Aplicação de padrões

Cena: Discussão informal durante o projeto de um incremento do software que implementa controle de sensores via Internet para o **CasaSeguraGarantida.com**.

Atores: Jamie (responsável pelo projeto) e Vinod (arquiteto-chefe do sistema **CasaSeguraGarantida.com**).

Conversa:

Vinod: Então, como está indo o projeto da interface para controle de câmeras?

Jamie: Nada mal. Já desenhei a maior parte da capacidade de conexão com os verdadeiros sensores sem grandes problemas. Já comecei também a pensar sobre a interface para os usuários poderem efetivamente movimentar, deslocar e ampliar as câmeras de uma página Web remota, porém ainda não estou certo de tê-la feito corretamente.

Vinod: Qual sua ideia?

Jamie: Bem, entre os requisitos, o controle de câmeras precisa ser altamente interativo — à medida que o usuário movimenta o controle, a câmera deve se movimentar o quanto antes possível. Portanto, estava imaginando um conjunto de botões dispostos como uma câmera comum, porém, ao clicá-los, o usuário controlaria a câmera.

Vinod: Hummm. Sim, isso funcionaria, mas não estou certo de que esteja correto — para cada clique de um controle precisaríamos esperar que ocorresse toda a comunicação cliente/servidor e, portanto, não teríamos a sensação de uma resposta imediata.

Jamie: Foi isso que imaginei — pelo fato de não estar muito satisfeito com a abordagem, mas não sei exatamente como poderia fazê-lo de outra forma.

Vinod: Bem, por que não simplesmente usa o padrão **InteractiveDeviceControl**?

Jamie: Hummm — o que é isso? Creio nunca ter ouvido falar dele.

Vinod: Basicamente trata-se de um padrão destinado exatamente para o problema que você está me descrevendo. A solução que ele propõe é criar uma conexão de controle do servidor com o dispositivo, por meio da qual comandos de controle poderiam ser enviados. Dessa maneira, não seria preciso enviar solicitações HTTP normais. E o padrão ainda indicaria como implementar isso usando algumas técnicas AJAX simples. Temos alguns JavaScripts simples no cliente que se comunicam diretamente com o servidor e transmitem os comandos assim que o usuário fizer alguma coisa.

Jamie: Legal! Era exatamente isso que eu precisava para resolver essa questão. Onde posso encontrá-lo?

Vinod: Ele se encontra à disposição em um repositório on-line. Eis a URL.

Jamie: Vou verificar isso.

Vinod: Sim — mas lembre-se de verificar as consequências para o padrão. Parece que eu me lembro de haver algo lá sobre precisar tomar cuidado com questões de segurança. Imagino que possa ser devido à criação de um canal de controle separado e, portanto, evitando os mecanismos de segurança comuns da Web.

Jamie: Excelente observação. Provavelmente não teria atentado a esse fato! Obrigado.

12.5 PADRÕES DE PROJETO PARA INTERFACES DO USUÁRIO

Foram propostas centenas de padrões para interfaces do usuário (UI, em inglês) nos últimos anos. A maior parte deles cai em uma das seguintes dez categorias de padrões (discutidas com um exemplo representativo⁹) conforme descrito por Tidwell [Tid02] e vanWelie [Wel01]:

Toda a Interface com o Usuário. Fornece orientação para estrutura de alto nível e navegação por toda a interface.

Padrão: TopLevelNavigation

Descrição: Usada quando um site ou uma aplicação implementa uma série de funções principais. Oferece um menu de alto nível, geralmente acoplado a um logo ou imagem identificadora, que possibilita a navegação direta para qualquer uma das principais funções do sistema.

Detalhes: Funções principais (em geral limitadas a quatro a sete nomes de função) são listadas na parte superior da tela (possível também formatos de colunas verticais) em uma linha horizontal de texto. Cada nome fornece um link para uma fonte de informações ou função apropriada. Geralmente, usada com o padrão **BreadCrumbs** discutido mais à frente.

Elementos de navegação: Cada nome de função/conteúdo representa um link para a função ou conteúdo apropriado.

Layout da página. Trata a organização geral das páginas (para sites) ou exibições em tela distintas (para aplicações interativas).

Padrão: CardStack

Descrição: Usado quando uma série de subfunções ou categorias de conteúdo específicas relacionadas com um recurso ou função deve ser selecionada em ordem aleatória. Dá a aparência de uma pilha de fichas indexadoras, cada uma delas selecionável com um clique de mouse e cada qual representando subfunções ou categorias de conteúdo específicas.

Detalhes: As fichas indexadoras são uma metáfora bem compreendida e fácil para o usuário manipular. Cada ficha indexadora (separador) pode ter um formato ligeiramente diverso. Alguns poderão exigir entrada de dados e possuir botões ou outros mecanismos de navegação; outros podem ser informativos. Poderiam ser combinados com outros padrões como **DropDownList**, **Fill-in-the-Blanks** e outros.

Elementos de navegação: Um clique de mouse em uma guia faz com que a ficha apropriada apareça. Os recursos de navegação contidos na ficha também poderiam estar presentes, porém, em geral, estas deveriam iniciar uma função relacionada aos dados da ficha e não provocar um link real para alguma outra tela.

Formulários e introdução de dados. Considera uma grande variedade de técnicas de projeto para realizar a introdução de dados em formulários.

Padrão: Fill-in-the-Blanks

Descrição: Possibilita que dados alfanuméricos sejam introduzidos em uma "caixa de texto."

Detalhes: Os dados poderiam ser introduzidos em uma caixa de texto. Em geral, são validados e processados após a seleção de algum indicador de texto ou gráfico (por exemplo, um botão contendo "avançar", "submeter", "próximo"). Em muitos casos, esse padrão pode ser combinado com uma lista suspensa ou outros padrões (por exemplo, SEARCH <drop down list> FOR <fill-in-the-blanks text box>).

Elementos de navegação: Um indicador de texto ou gráfico que inicia a validação e o processamento.

Tabelas. Fornece orientação de projeto para a criação e manipulação de dados tabulares de toda espécie.

⁹ Um template-padrão sintetizado é usado aqui. Descrições de padrões completas (juntamente com dezenas de outros padrões) podem ser encontradas em [Tid02] e [Wel01].

Padrão: SortableTable

Descrição: Mostra uma longa lista de registros que podem ser ordenados através da seleção de um mecanismo comutador para qualquer rótulo de coluna.

Detalhes: Cada linha da tabela representa um registro completo. Cada coluna representa um campo no registro. Cada título de coluna é, na verdade, um botão selecionável que pode ser comutado para iniciar uma ordem crescente ou decrescente no campo associado à coluna para todos os registros exibidos. Em geral a tabela é redimensionável e poderá ter um mecanismo de rolagem caso o número de registros seja maior que o espaço de janela disponível.

Elementos de navegação: Cada cabeçalho de coluna inicia uma classificação de todos os registros. Nenhuma outra navegação é fornecida, embora em alguns casos, cada registro poderia por si só conter links de navegação para outro conteúdo ou funcionalidade.

Manipulação de dados diretos. Lida com a edição, a modificação e a transformação de dados.

Padrão: BreadCrumbs

Descrição: Fornece um caminho de navegação completo quando o usuário está trabalhando com uma hierarquia de páginas ou telas complexa.

Detalhes: É atribuído um identificador exclusivo a cada página ou tela. O caminho de navegação para o local atual é especificado em um local predefinido para qualquer tela. O caminho assume a forma: **homepage>página de tópico principal>página de subtópico>página específica>página atual**.

Elementos de navegação: Qualquer uma das entradas contidas na tela Bread Crumbs pode ser usada como um ponteiro para um link de retorno para um nível hierárquico mais elevado.

Navegação. Ajuda o usuário na navegação através de menus hierárquicos, páginas Web e telas interativas.

Padrão: EditInPlace

Descrição: Fornece um recurso de edição simples para certos tipos de conteúdo no local em que é exibido. Nenhuma necessidade de o usuário introduzir explicitamente uma função ou modo de edição de texto.

Detalhes: O usuário vê o conteúdo na tela que deve ser alterado. Um clique duplo com o mouse sobre o conteúdo indica ao sistema que se deseja editar. O conteúdo é realçado para significar que o modo de edição está disponível e o usuário faz as mudanças apropriadas.

Elementos de navegação: Nenhum.

Searching. Possibilita buscas de conteúdo específicas nas informações mantidas em um site ou contidas em repositórios de dados persistentes que podem ser acessados via aplicação interativa.

Padrão: SimpleSearch

Descrição: Oferece a capacidade de pesquisar em um site ou fonte de dados persistentes para um dado simples descrito em uma string alfanumérica.

Detalhes: Oferece a capacidade de pesquisar local (uma página ou um arquivo) ou globalmente (o site todo ou um banco de dados completo) por uma string de busca. Gera uma lista de "acertos" em ordem de probabilidade para atender às necessidades do usuário. Não oferece buscas de itens múltiplos ou operações booleanas especiais (veja *padrão de busca avançado*).

Elementos de navegação: Cada entrada na lista de acertos representa um link de navegação com os dados referidos pela entrada.

Elementos de página. Implementa elementos específicos de uma página Web ou tela.

Padrão: Wizard

Descrição: Conduz o usuário, passo a passo, através de uma tarefa complexa, dando orientação para a finalização da tarefa por meio de uma série de telas de janelas simples.

Detalhes: O exemplo clássico é um processo de registro em quatro etapas. O padrão Wizard gera uma janela para cada etapa, solicitando informações específicas do usuário, um passo por vez.

Elementos de navegação: Navegação para a frente e para trás que permite ao usuário revisitar cada passo dado no processo de assistente.

E-commerce. Específicos para sites, esses padrões implementam elementos recorrentes de aplicações para comércio eletrônico.

Padrão: ShoppingCart

Descrição: Fornece uma lista de itens selecionados para compra.

Detalhes: Lista informações de itens, quantidade, código de produto, disponibilidade (disponível, esgotado), preço, informações para entrega, custos de remessa e outras informações de compra relevantes. Também oferece a habilidade de editar (por exemplo, remover, modificar quantidade).

Elementos de navegação: Contém a capacidade de prosseguir com a compra ou sair para encerrar as compras.

Miscellaneous. Padrões que não se encaixam perfeitamente em uma das categorias anteriores. Em alguns casos, dependem do domínio ou ocorrem apenas para classes de usuário específicas.

Padrão: ProgressIndicator

Descrição: Dá uma indicação do progresso quando uma operação leva mais do que n segundos.

Detalhes: Representado na forma de um ícone de animação ou uma caixa de mensagens contendo alguma indicação visual (por exemplo, um "poste de barbeiro" girante, um controle deslizante com um indicador da porcentagem completada) de que o processamento está em andamento. Também pode conter uma indicação de conteúdo de texto do estado do processamento.

Elementos de navegação: Em geral contém um botão que permite ao usuário pausar ou cancelar processamento.

Cada um dos exemplos de padrões anteriores (e todos os padrões em cada categoria) também teria um projeto completo de componentes, incluindo classes de projeto, atributos, operações e interfaces.

Uma discussão completa para interfaces do usuário vai além do escopo deste livro. Caso tenha maior interesse, veja [Duy02], [Bor01], [Tid02] e [Wel01] para mais informações.

12.6 PADRÕES DE PROJETO PARA WEBAPPS

Ao longo deste capítulo vimos que há diferentes tipos de padrões e várias maneiras distintas em que podem ser classificados. Ao considerarmos os problemas de projeto a ser solucionados quando uma WebApp é construída, vale a pena considerar categorias de padrão concentrando-nos em duas dimensões: o foco de projeto do padrão e seu nível de granularidade. O *foco do projeto* identifica qual aspecto do modelo de projeto é relevante (por exemplo, arquitetura das informações, navegação, interação). A *granularidade* identifica o nível de abstração que está sendo considerado (por exemplo, o padrão se aplica a toda a WebApp ou a uma única página Web, a um subsistema ou um componente WebApp individual?).

PONTO-CHAVE

O foco se torna "mais limitado" quanto mais se avança no projeto.

12.6.1 Foco do projeto

Em capítulos anteriores enfatizei uma progressão de projeto que inicia considerando-se questões de arquitetura e de componentes e representações de interfaces do usuário. A cada etapa, os problemas considerados e as soluções propostas começam com um alto nível de abstração e lentamente se tornam mais detalhados e específicos. Em outras palavras, o foco do

projeto se torna “mais limitado” à medida que avançamos no projeto. Os problemas (e soluções) que encontraremos ao projetarmos uma arquitetura de informações para uma WebApp são diferentes dos problemas (e soluções) encontrados ao realizarmos o projeto de interfaces. Consequentemente, não será nenhuma surpresa que os padrões para o projeto de WebApps possa ser desenvolvido para diferentes níveis de foco de projeto, para podermos tratar os problemas únicos (e soluções relativas) encontrados em cada nível. Os padrões para WebApps podem ser classificados usando-se os seguintes níveis de foco do projeto:

- **Padrões de arquitetura de informações** relacionam-se à estrutura geral do espaço de informações e as maneiras pelas quais os usuários irão interagir com as informações.
- **Padrões de navegação** definem estruturas de links de navegação como hierarquias, anéis, *tours* e assim por diante.
- **Padrões de interação** contribuem para o projeto da interface do usuário. Os padrões nessa categoria tratam a maneira pela qual a interface informa o usuário das consequências de uma ação específica, como um usuário expande conteúdo baseado no contexto de uso e nos desejos dos usuários, como descrever melhor o destino definido por um link, como informar o usuário sobre o estado de uma interação em andamento e questões relacionadas com interfaces.
- **Padrões de apresentação** ajudam na apresentação do conteúdo à medida que é apresentado ao usuário via interface. Padrões nessa categoria tratam como organizar as funções de controle de interfaces do usuário para melhor usabilidade, como mostrar a relação entre uma ação de interface e os objetos de conteúdo que ela afeta e como estabelecer hierarquias de conteúdo efetivas.
- **Padrões funcionais** definem os fluxos de trabalho, comportamentos, processamento, comunicação e outros elementos algorítmicos contidos em uma WebApp.

Na maioria dos casos, seria infrutífero explorar o conjunto dos padrões de arquitetura de informações quando se encontra um problema no projeto de interação. Teríamos de examinar padrões de interação, pois esse é o foco do projeto relevante ao trabalho que está sendo realizado.

12.6.2 Granularidade do projeto

Quando um problema envolve questões de “quadro geral”, devemos tentar desenvolver soluções (e usar padrões relevantes) que se concentram no quadro geral. Quando o foco é muito limitado (por exemplo, selecionando unicamente um item de um pequeno conjunto de cinco ou menos itens), a solução (e o padrão correspondente) é atingida com pouca margem de erro. Em termos do nível de granularidade, os padrões podem ser descritos nos seguintes níveis:

- **Padrões de arquitetura.** Esse nível de abstração irá, tipicamente, se relacionar a padrões que definem a estrutura geral da WebApp, indicam os relacionamentos entre os diferentes componentes ou incrementos e definem as regras para especificar as relações entre os elementos (páginas, pacotes, componentes, subsistemas) da arquitetura.
- **Padrões de projeto.** Tratam um elemento específico do projeto como uma agregação de componentes para resolver algum problema de projeto, relações entre os elementos em uma página ou os mecanismos para efetuar a comunicação componente-para-componente. Um exemplo poderia ser o padrão **Broadsheet** para o layout da homepage de uma WebApp.
- **Padrões de componentes.** Esse nível de abstração relaciona-se a elementos individuais de pequena escala de uma WebApp. Entre os exemplos, temos elementos de interação individual (por exemplo, botões de rádio), itens de navegação (por exemplo, como formataríamos links?) ou elementos funcionais (por exemplo, algoritmos específicos). Também é possível definir a relevância de diferentes padrões para diferentes áreas ou categorias de aplicação. Por exemplo, um conjunto de padrões (em diferentes níveis da granularidade e foco do projeto) poderia ser particularmente relevante ao comércio eletrônico.

INFORMAÇÕES



Repositórios de padrões de projeto para hipermídia

O site IAWiki (<http://iawiki.net/WebsitePatterns>), um espaço de discussão colaborativa para arquitetos da informação, contém diversos recursos úteis. Entre eles temos links para uma série de catálogos de padrões e repositórios para hipermídia. Centenas de padrões de projeto estão representados:

Repositório de padrões de projeto para hipermídia

www.designpattern.lu.unisi.ch/

InteractionPatterns de Tom Erickson

www.pliant.org/personal/Tom_Erickson/InteractionPatterns.html

Padrões de projeto para a Web de Martijn van Welie

www.welie.com/patterns/

Padrões Web para projeto de interfaces do usuário

http://harbinger.sims.berkeley.edu/ui_designpatterns/webpatterns2/

webpatterns/home.php

Padrões para sites pessoais

www.rdrop.com/%7Ehalf/Creations/Writings/Web.patterns/index.html

Aperfeiçoamento de sistemas de informações na Web com padrões de navegação

<http://www8.org/w8-papers/5b-hypertext-media/improving/improving.html>

Uma linguagem de padrões HTML 2.0

www.anamorph.com/docs/patterns/default.html

Pontos em comum — uma linguagem de padrões para projeto de HCLs

www.mit.edu/~jtidwell/interaction_patterns.html

Padrões para sites pessoais www.rdrop.com/~half/Creations/Writings/Web.patterns/index.html

Indexação de linguagem de padrões www.cs.brown.edu/~rms/InformationStructures/Indexing/Overview.html

12.7 RESUMO

Os padrões de projeto fornecem um mecanismo codificado para descrever problemas e suas soluções de maneira que permita à comunidade da engenharia de software capturar conhecimento de projeto para que possa ser reutilizado. Um padrão descreve um problema, indica o contexto, permitindo ao usuário compreender o ambiente em que o problema reside e listar um sistema de forças que indicam como o problema pode ser interpretado no seu contexto e como uma solução pode ser aplicada. No trabalho de engenharia de software, identificamos e documentamos padrões generativos que descrevem um aspecto importante e repetível de um sistema, dando-nos então uma forma de construir esse aspecto em um sistema de forças que seja único a um dado contexto.

Os padrões de arquitetura descrevem problemas de projeto abrangentes resolvidos usando-se uma abordagem estrutural. Os padrões de dados descrevem problemas recorrentes orientados a dados e as soluções de modelagem de dados que podem ser usadas para resolvê-las. Os padrões de componentes (também conhecidos como padrões de projeto) tratam de problemas associados ao desenvolvimento de subsistemas e componentes, a maneira pela qual eles se comunicam entre si e seu posicionamento em uma arquitetura mais ampla. Os padrões de projeto para interfaces descrevem problemas comuns de interfaces do usuário e suas soluções com um sistema de forças que inclui as características específicas dos usuários finais. Os padrões para WebApps lidam com um conjunto de problemas encontrado ao se construírem WebApps e muitas vezes incorporam muitas das demais categorias de padrões mencionados. Uma estrutura fornece a infraestrutura em que talvez possam residir padrões, e idiomas descrevem detalhes de implementação específicos a uma linguagem de programação para o todo ou parte de uma estrutura de dados ou algoritmo específico. Um formulário ou template-padrão é usado para descrições de padrões. Uma linguagem de padrões engloba um conjunto de padrões, cada qual descrito por meio do uso de um template padronizado e inter-relacionado para mostrar como os padrões colaboram para solucionar problemas em um campo de aplicação.

O projeto baseado em padrões é usado em conjunto com métodos de projeto para arquitetura, para a componentização e para interfaces do usuário. A abordagem de projeto começa com um exame do modelo de requisitos para isolar problemas, definir o contexto e descrever o siste-

ma de forças. Em seguida, buscam-se linguagens de padrões para o domínio do problema para determinar se há padrões para os problemas que foram isolados. Uma vez encontrados padrões apropriados, eles são usados como um guia de projeto.

PROBLEMAS E PONTOS A PONDERAR

- 12.1. Discuta as três "partes" de um padrão de projeto e forneça um exemplo concreto de cada um deles de algum outro campo que não seja o de software.
- 12.2. Qual a diferença entre um padrão não generativo e generativo?
- 12.3. Como os padrões de arquitetura diferem dos padrões de componentes?
- 12.4. O que é uma estrutura de uso de padrões e como ela difere de um padrão? O que é um idioma e como ele difere de um padrão?
- 12.5. Usando o template de padrões de projeto apresentado na Seção 12.1.3, desenvolva uma descrição de padrão completa para um padrão sugerido pelo seu professor.
- 12.6. Desenvolva uma linguagem de padrões estrutural para um esporte com o qual você esteja familiarizado. Você pode começar lidando com o contexto, o sistema de forças e os problemas abrangentes que um técnico e a equipe devem solucionar. É preciso não apenas especificar os nomes de padrão, mas também uma descrição em uma sentença para cada padrão.
- 12.7. Encontre cinco repositórios de padrões e apresente uma descrição abreviada dos tipos de padrões contidos em cada um deles.
- 12.8. Quando Christopher Alexander diz "um bom projeto de software não pode ser alcançado simplesmente juntando-se peças operantes", o que você acha que ele quis dizer com isso?
- 12.9. Usando as tarefas para projeto baseado em padrões citadas na Seção 12.2.3, desenvolva um esboço de projeto para o "sistema de design de interiores" descrito na Seção 11.3.2.
- 12.10. Construa uma tabela para organizar padrões para os padrões utilizados no Problema 12.9.
- 12.11. Usando o template de padrões de projeto apresentado na Seção 12.1.3, desenvolva uma descrição completa de padrões para o padrão **Kitchen** mencionado na Seção 12.3.
- 12.12. A "gangue dos quatro" [Gam95] propôs uma variedade de padrões de componentes aplicáveis a sistemas orientados a objetos. Escolha um (estes se encontram à disposição na Web) e discuta-o.
- 12.13. Encontre três repositórios de padrões para padrões de interfaces do usuário. Escolha um padrão de cada e apresente uma descrição abreviada dele.
- 12.14. Encontre três repositórios de padrões para padrões de WebApps. Escolha um padrão de cada e apresente uma descrição abreviada dele.

LEITURAS E FONTES DE INFORMAÇÃO COMPLEMENTARES

Ao longo da última década, lançaram-se vários livros sobre projeto baseado em padrões destinados a engenheiros de software. Gamma e seus colegas [Gam95] escreveram o livro seminal sobre o tema. Entre algumas contribuições mais recentes, temos obras como as de Lasater (*Design Patterns*, Wordware Publishing, Inc., 2007), Holzner (*Design Patterns for Dummies*, For Dummies, 2006), Freeman e seus colegas (*Head First Design Patterns*, O'Reilly Media, Inc., 2005) e Shalloway e Trott (*Design Patterns Explained*, 2. ed., Addison-Wesley, 2004). Uma edição especial do *IEEE Software* (julho/agosto, 2007) discute uma variedade de tópicos de padrões de software. Kent Beck (*Implementation Patterns*, Addison-Wesley, 2008) trata de padrões para problemas de codificação e implementação encontrados durante a atividade de construção.

Outros livros focalizam padrões de projeto à medida que são fornecidos em ambientes de linguagem e desenvolvimento de aplicações específicos. Entre as contribuições nessa área, temos: Bowers (*Pro CSS and HTML Design Patterns*, Apress, 2007), Tropashko e Burleson (*SQL Design Patterns: Expert Guide to SQL Programming*, Rampant Techpress, 2007), Mahemoff (*Ajax Design Patterns*, O'Reilly Media, Inc., 2006), Metsker e Wake (*Design Patterns in Java*, Addison-Wesley, 2006), Nilsson (*Applying Domain-Driven Design and Patterns: With Examples in C# and .NET*, Addison-Wesley, 2006), Sweat (*PHPArchitect's Guide to PHP Design Patterns*, Marco Tabini & Associates, Inc., 2005), Metsker (*Design Patterns C#*, Addison-Wesley, 2004), Grand e Merrill (*Visual Basic .NET Design Patterns*, Wiley, 2003), Crawford e Kaplan (*J2EE Design Patterns*, O'Reilly Media, Inc., 2003), Juric et al. (*J2EE Design Patterns Applied*, Wrox Press, 2002) e Marinescu e Roman (*EJB Design Patterns*, Wiley, 2002).

Outras obras ainda tratam de campos de aplicação específicos. Entre elas estão a de Kuchana (*Software Architecture Design Patterns in Java*, Auerbach, 2004), Joshi (*C++ Design Patterns and Derivatives Pricing*, Cambridge University Press, 2004), Douglass (*Real-Time Design Patterns*, Addison-Wesley, 2002) e Schmidt e Rising (*Design Patterns in Communication Software*, Cambridge University Press, 2001).

Clássicos do arquiteto Christopher Alexander (*Notes on the Synthesis of Form*, Harvard University Press, 1964 e *A Pattern Language: Towns, Buildings, Construction*, Oxford University Press, 1977) é uma leitura que vale a pena para um projetista de software que pretenda entender completamente os padrões de projeto.

Uma ampla gama de fontes de informação sobre projeto baseado em padrões se encontra à disposição na Internet. Uma lista atualizada de referências na Web, relevante para o projeto baseado em padrões, pode ser encontrada no site www.mhhe.com/engcs/compsci/pressman/professional/olc/ser.htm.