Capítulo 2

Padrões de projeto orientado a objetos

Alexander, Ishikawa e Silverstein (1977) perceberam, ao longo de seus primeiros trabalhos com arquitetura civil, que, entre um projeto e outro, as dificuldades enfrentadas eram parecidas. A aplicação de determinado tipo de técnica era recorrente de um projeto para outro. A partir desse ponto, decidiram criar um catálogo com as soluções apresentadas para os problemas habituais. O intuito era ganhar tempo, partindo de soluções já validadas na prática.

Se Alexander, Ishikawa e Silverstein (1977) criaram um catálogo de soluções adotadas em projetos de arquitetura civil, engenheiros de software pelo mundo continuam utilizando essa prática para documentar problemas recorrentes em seus projetos. A essa documentação das soluções de problemas que costumam se repetir chamamos pattern,

ou padrão, em português. Vamos entender melhor o que são padrões, com foco em padrões de projetos, ou design patterns, ao longo deste capítulo.

1 O que é um padrão de projeto orientado a objetos?

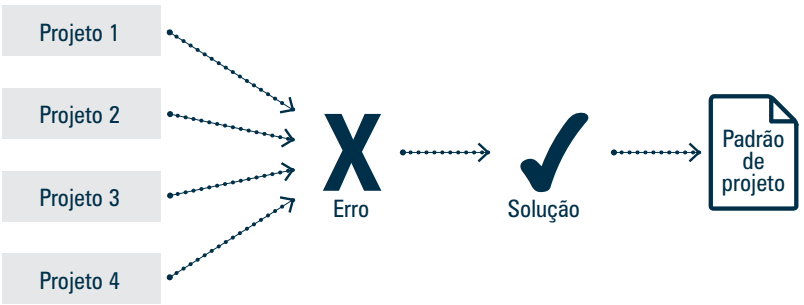
A construção de um projeto orientado a objetos apresenta muitos desafios. Desenvolvedores e engenheiros de software defendem que um software orientado a objetos é reutilizável, mas a verdade é que a criação de um software desse tipo deve levar em consideração algumas questões, como: se os objetos são pertinentes ao problema, qual a profundidade da hierarquia de classes, os relacionamentos entre classes e interfaces, entre outros pontos. Essa é uma tarefa difícil principalmente quando estão envolvidos profissionais pouco experientes no manejo desse tipo de modelagem (GAMMA *et al.*, 2009).

Nesse ponto, considerando todas as questões de um projeto orientado a objetos, como é possível torná-lo reutilizável a novos projetos? O que grande parte dos profissionais mais experientes da área descobrem com o passar do tempo é que, para tornar isso possível, não se deve criar nenhum projeto totalmente do zero. Para isso, necessita-se reutilizar soluções que obtiveram resultados bons em momentos passados (FOWLER, 2002; GAMMA *et al.*, 2009). Nesse ponto nascem os padrões de projetos. Mas o que são eles?

Alexander, Ishikawa e Silverstein (1977) dizem que um padrão deve descrever o problema de determinado cenário e a essência de sua solução, fazendo com que a solução possa ser utilizada diversas outras vezes, mas nunca da mesma maneira. E isso é totalmente aplicável a projetos orientados a objetos. Basta uma análise de cenários em que as janelas e portas são expressas em objetos e interfaces (GAMMA *et al.*, 2009).

A partir de uma solução utilizada repetidas vezes, é criado um padrão, que pode ser reutilizado em diversos outros projetos. Assim, é obtido um nível de amadurecimento para criar um sistema reutilizável, portanto mais flexível. A figura 1 demonstra o processo de criação de um padrão de projeto.

Figura 1 – Processo de criação de padrões de projeto



Na figura 1 temos quatro projetos, que são executados em tempos diferentes por uma mesma equipe. Ao longo do desenvolvimento, percebe-se um erro, problema ou até mesmo uma barreira para encarar e superar. As alternativas de solução foram muito similares apesar de não ter sido utilizada a mesma ferramenta ou linguagem de programação. A partir dessa percepção, documenta-se todo esse passo a passo para uma sugestão de solução ao problema. Essa documentação é o padrão de projeto.

Perceba que não se fala em utilizar o código efetivo do objeto, mas sim a maneira como ele foi concebido (GAMMA *et al.*, 2009). Essa é a essência da definição de Alexander, Ishikawa e Silverstein (1977) ao dizerem que a solução poderá ser aproveitada em outros momentos, mas de formas diferentes.

Os padrões de projeto têm como objetivo tornar mais fácil a reutilização de projetos e arquiteturas bem-sucedidas. Eles não devem descrever um projeto ou implementações sem que tenham ocorrido testes antes. A descrição de um padrão de projeto deve focar em soluções que

já tenham sido utilizadas múltiplas vezes em diversos projetos diferentes (FOWLER, 2002; GAMMA *et al.*, 2009).

Padrões de projetos devem possuir em sua descrição quatro atributos/propriedades essenciais:

- **Nome:** referência que possa ser utilizada para descrever um problema, sua solução e as consequências do uso. Cada nome dado a um padrão aumenta o vocabulário de um projeto. Esse vocabulário é importante para ajudar a determinar, dentre os diversos padrões a serem escolhidos, onde e como aplicar cada um.

- **Problema:** descreve o cenário em que o padrão deve ser aplicado. Além de informar o problema, é importante descrever qual o contexto em que ele deve ser considerado. Esse atributo pode conter informações específicas de projetos ou até mesmo uma lista de condições que devem ser atendidas em um projeto ou cenário.

- **Solução:** descreve os elementos que um padrão de projeto precisa ter para conseguir, a partir de seus relacionamentos, suas responsabilidades e colaborações, atender ao problema. Não deve ser concreta, muito menos tão específica na parte técnica. Deve ser uma recomendação de como solucionar o problema, deixando o desenvolvedor livre para utilizá-la ou não no projeto.

- **Consequências:** descrevem as vantagens e desvantagens da aplicação do padrão para solucionar o problema. Por mais que não entrem, em geral, nas decisões de um projeto, as consequências são termômetros para avaliar um padrão de projeto. Um dos pontos que sempre são discutidos nas consequências de um padrão é a questão entre o desempenho de tempo e da memória. Questões sobre implementações em determinadas linguagens também podem ser encontradas nesse atributo.

O mais importante ao construir ou utilizar um padrão é que ele não deve ser considerado uma norma ou regra, cuja implementação é

obrigatória em determinado cenário. O padrão de projeto é uma boa prática que se recomenda seguir quando certo problema é encontrado. E o problema, vale lembrar, não é necessariamente um erro, podendo ser entendido simplesmente como um cenário de um novo projeto parecido com o anterior.

2 Tipos de padrões

Padrões de projeto podem ser segmentados por setores de aplicação, como aplicativos móveis, sistemas cliente-servidor, aplicações desktop ou aplicações corporativas. Por mais que a grande maioria possa ser aplicada aos diversos segmentos, alguns são mais relevantes para segmentos específicos (FOWLER, 2002).

Apesar da especificidade dos padrões para cada tipo de segmento, existe uma classificação que possibilita agrupar os padrões de projeto em três tipos: padrões de criação, estruturais e comportamentais.

2.1 Padrões de criação

Padrões de criação estão relacionados ao processo de instância dos objetos. Em geral, utiliza-se a herança para variar as classes que são instanciadas ao longo da execução do sistema. Esses padrões são importantes à medida que o sistema evolui, no sentido de depender mais da composição de objetos, e não da herança das classes. A composição é algo que precisa ser sempre priorizado, pois isso diminui a dependência entre os objetos, permitindo que o sistema seja escalável (FOWLER, 2002; GAMMA *et al.*, 2009).

A ênfase se desloca da codificação rígida de um conjunto fixo de comportamentos para uma definição de um conjunto menor de comportamentos fundamentais. Os dois temas mais recorrentes entre os padrões de criação são: todos encapsulam conhecimento das classes concretas que são utilizadas pelo sistema; todos ocultam o modo como as instâncias dessas classes são criadas e compostas (GAMMA *et al.*, 2009).

Os padrões de criação propiciam muita flexibilidade ao objeto criado, no que diz respeito a como e quando é criado, e a quem o cria, permitindo, assim, configurar um sistema com "objetos-produtos" que variam amplamente em estrutura e funcionalidade. São alguns exemplos de padrões de criação: Prototype, Abstract Factory, Builder, Singleton e Factory Method (FOWLER, 2002; GAMMA *et al.*, 2009).

2.2 Padrões estruturais

Os padrões estruturais ocupam-se de como a classe e os objetos são compostos para formação de estruturas maiores e utilizam herança para compor interfaces ou implementações. Eles são úteis para fazer bibliotecas de objetos que são desenvolvidas independentemente trabalharem juntas. No lugar de compor interfaces ou implementações, os padrões estruturais de objetos descrevem maneiras de compor objetos para obter novas funcionalidades (GAMMA *et al.*, 2009).

A flexibilidade na composição dos objetos provém da capacidade de mudar a composição em tempo de execução, o que é impossível com a composição estática de classes. Alguns exemplos de padrões estruturais são: Adapter, Composite, Proxy, Flyweight, Bridge, Decorator e Facade (GAMMA *et al.*, 2009).

2.3 Padrões comportamentais

Os padrões comportamentais têm foco nos algoritmos e na atribuição de responsabilidades entre os objetos. Não descrevem apenas o objeto, mas sim a comunicação entre eles. Caracterizam fluxos de controle difíceis de seguir em tempo de execução. Sendo assim, a ideia é concentrar-se apenas na maneira como os objetos são interconectados. Geralmente utilizam a herança para distribuir o comportamento entre as classes (FOWLER, 2002; GAMMA *et al.*, 2009).

Os padrões comportamentais utilizam a composição de objetos em vez da herança. Alguns descrevem como um grupo de objetos pares cooperam para a execução de uma tarefa que nenhum objeto sozinho poderia executar. Alguns padrões comportamentais são: Template Method, Interpreter, Mediator, Chain of Responsibility, Observer, Strategy, Command, State e Iterator (GAMMA *et al.*, 2009).

3 Aplicações de software voltadas à web e a dispositivos móveis

Aplicações web podem ser divididas, hoje em dia, entre front-end (parte de interface gráfica com usuário) e back-end (parte que é executada no servidor). Os aplicativos móveis possuem padrões mais próximos do front-end nas aplicações web. Isso porque envolvem comportamentos de interface no geral, como navegação de tela, apresentação de elementos, entre outros (FOWLER, 2002; NUDELMAN, 2013).

Os padrões de projetos de aplicações hoje, como sistemas corporativos, estão mais focados na arquitetura de software e na parte do back-end.

Considerações finais

Neste capítulo, apresentamos o conceito de padrões de projeto. Com eles podemos criar um projeto a partir da experiência vivenciada não só por uma equipe, mas por diversos outros desenvolvedores. É uma maneira de começarmos a construir o projeto já com informações de arquitetura e componentes, em vez de começar do zero. O uso de padrões de projeto é essencial para manter um sistema reutilizável e escalável.

Referências

ALEXANDER, Christopher; ISHIKAWA, Sara; SILVERSTEIN, Murray. **A pattern language**: towns, buildings, construction. Oxford: Oxford University Press, 1977.

FOWLER, Martin. **Patterns of enterprise application architecture**. Boston: Addison-Wesley, 2002.

GAMMA, Erich *et al.* **Design patterns elements of reusable object-oriented software**. Boston: Addison-Wesley, 2009.

NUDELMAN, Greg. **Padrões de projeto para o Android**: soluções de projetos de interação para desenvolvedores. São Paulo: Novatec, 2013.