Atualmente um desenvolvimento sério de software utiliza orientação a objetos, pois esta permite extensibilidade e reuso de código, porém utilizá-la não é garantia de qualidade.

Quando nos deparamos com um novo projeto é fácil perceber que uma solução semelhante, se não igual, já foi implementada anteriormente, porém devido à deficiência de documentação e/ou da qualidade do código o reuso/extensão torna-se inviável.

Todo projetista avançado sabe que o que não deve fazer é elaborar uma solução do 0. Da mesma forma que numa solução matemática usa-se todo o conhecimento disponível para se chegar ao resultado final, um programa deve ser composto por partes menores de código, já testadas e funcionais. Por esse motivo os design patterns têm ganhado fama.

Os padrões de projetos tornam mais fácil reutilizar soluções e arquiteturas bem sucedidas para construir softwares orientados a objetos de forma flexível e fácil de manter. O uso de padrões de projeto pode reduzir a complexidade do processo de projetar software. Além disso, o software orientado a objetos bem projetado possibilita aos projetistas reutilizar e empregar componentes preexistentes em sistemas futuros.

Livro útil do tema: Design patterns – elements of reusable object-oriented software, Addison Wesley Longman

O livro é um catálogo que descreve 23 padrões de projeto cada um fornecendo uma solução para um problema de software, seu contexto, aplicação e suas eventuais conseqüências, dividindo-os em 3 categorias: padrões de criação, estruturais, e de comportamento.

Se temos uma solução num certo contexto, ela não necessariamente torna-se um padrão, pois uma característica necessária é a regularidade

**finalidade** - reflete o que um padrão faz. Os padrões podem ter finalidades de criação, comportamento e estrutural. Os padrões de criação descrevem as técnicas para instanciar objetos (ou grupos de objetos), e possibilitam organizar classes e objetos em estrutura maiores, os de comportamento se caracterizam pela maneira pelas quais classes ou objetos interagem e distribuem responsabilidades e os estruturais lidam com a composição de classes ou objetos. O segundo critério é o **escopo** - especifica se o padrão é aplicado à classe ou objeto.

Embora um padrão seja a descrição de um problema, de uma solução genérica e sua justificativa, isso não significa que qualquer solução conhecida para um problema possa constituir um padrão, pois existem características obrigatórias que devem ser atendidas pelos padrões:

1. Devem possuir um nome, que descreva o problema, as soluções e conseqüências. Um nome permiti definir o vocabulário a ser utilizado pelos projetistas e desenvolvedores em um nível mais alto de abstração.
2. Todo padrão deve relatar de maneira clara a qual (is) problema(s) ele deve ser aplicado, ou seja, quais são os problemas que quando inserido em um determinado contexto o padrão conseguirá resolve-lo.Alguns podendo exigir pré-condições.
3. Solução descreve os elementos que compõem o projeto, seus relacionamentos, responsabilidades e colaborações. Um padrão deve ser uma solução concreta, ele deve ser exprimido em forma de gabarito (algoritmo) que, no entanto pode ser aplicado de maneiras diferentes.
4. Todo padrão deve relatar quais são as suas conseqüências para que possa ser analisada a solução alternativa de projetos e para a compreensão dos benefícios da aplicação do projeto.

Não pode ser considerado um padrão de projeto trecho de códigos específicos, mesmo que para o seu criador ele reflita um padrão, que soluciona um determinado problema, porque os padrões devem estar a um nível maior de abstração e não limitado a recursos de programação. Um padrão de projeto nomeia, abstrai e identifica os aspectos chaves de uma estrutura de projeto comum para torna-la útil para a criação de um projeto orientado a objetos reutilizável.

A utilização dos padrões proporciona um aumento na flexibilidade do sistema, entretanto pode deixa-lo mais complexo ou degradar a performance. Algumas perdas são toleráveis, mas subestimar os efeitos colaterais da adoção dos patterns, é um erro comum, principalmente daqueles que tomam o uso como um diferencial e não pela real necessidade.

As abstrações que surgem durante um projeto são as chaves para torna-lo flexível. Os padrões de projeto ajudam a identificar abstrações menos óbvias bem como os objetos que podem capturá-las. Por exemplo, objetos que representam processo ou algoritmo não ocorrem na natureza, no entanto, eles são uma parte crucial de projetos flexíveis. Esses objetos são raramente encontrados durante a análise ou mesmo durante os estágios iniciais de um projeto; eles são descobertos mais tarde, durante o processo de tornar um projeto mais flexível e reutilizável.

Escolher dentre os padrões existentes aquele que melhor soluciona um problema do projeto, sem cometer o erro de escolher de forma errônea e torná-lo inviável, é uma das tarefas mais difíceis. Em suma, a escolha de um padrão de projeto a ser utilizado, pode ser baseada nos seguintes critérios:

1. Considerar como os padrões de projeto solucionam problemas de projeto.
2. Examinar qual a intenção do padrão, ou seja, o que faz de fato o padrão de projeto, quais seus princípios e que tópico ou problema particular de projeto ele trata (soluciona).
3. Estudar como os padrões se relacionam.
4. Estudar as semelhanças existentes entre os padrões.
5. Examinar uma causa de reformulação de projeto.
6. Considerar o que deveria ser variável no seu projeto, ou seja, ao invés de considerar o que pode forçar uma mudança em um projeto, considerar o que você quer ser capaz de mudar sem reprojetá-lo.