正如任何其他抽象的特点，设计模式的特点就是为了使工作更加容易。系统中总是有一些东西在变化—这可能是在软件项目生命周期中代码的变化，或许是在某个程序执行的生命周期期间某些对象的变化。找出变化的东西，利用设计模式封装这些变化，并使这些变化能够得到控制。

人们在进行程序设计时很容易迷恋于使用某个特定的设计模式，并且如果因为刚刚知道如何做就贸然去做也将给自己带来烦恼。最难做到的是什么？有点讽刺意味，是遵循《极限编程》（《Extreme Programming》）中的那句格言：“只要能用，就做最简单的。”仅仅做最简单的东西，不仅能够最快速的实现设计，而且其设计也很容易维护。如果这种最简单的东西不能完成工作，读者很快就会发现，除了花费时间编写复杂的实现方法之外，它们还是不起作用的。

当发现需要添加新的类型到一个系统中时，最明智的首要步骤就是用多态机制为这些新类型创建一个共同的接口。用这种方法可以将系统中其余的代码与新添加的特定类型的代码分开。新类型的添加并不会扰乱已存在的代码……或者至少看上去如此。起初它似乎只需要在继承新类的地方修改代码，但这并非完全正确。仍须创建新类型的对象，在创建对象的地方必须指定要使用的准确的构造函数。因此，如果创建对象的代码遍布整个应用程序，在增加新类型时将会遇到同样的问题—仍然必须找出代码中所有与新类型相关的地方。这是由类的创建而不是类的使用（类型的使用问题已被多态机制解决了）而引起，但是效果是一样的：添加新类型将导致问题的出现。

这个问题的解决方法就是强制用一个通用的工厂（factory）来创建对象，而不允许将创建对象的代码散布于整个系统。如果程序中所有需要创建对象的代码都转到这个工厂执行，那么在增加新对象时所要做的全部工作就是只需修改工厂。这种设计是众所周知的工厂方法（Factory Method）模式的一种变体。由于每个面向对象应用程序都需要创建对象，并且由于人们可能通过添加新类型来扩展应用程序，工厂模式可能是所有设计模式中最有用的模式之一。