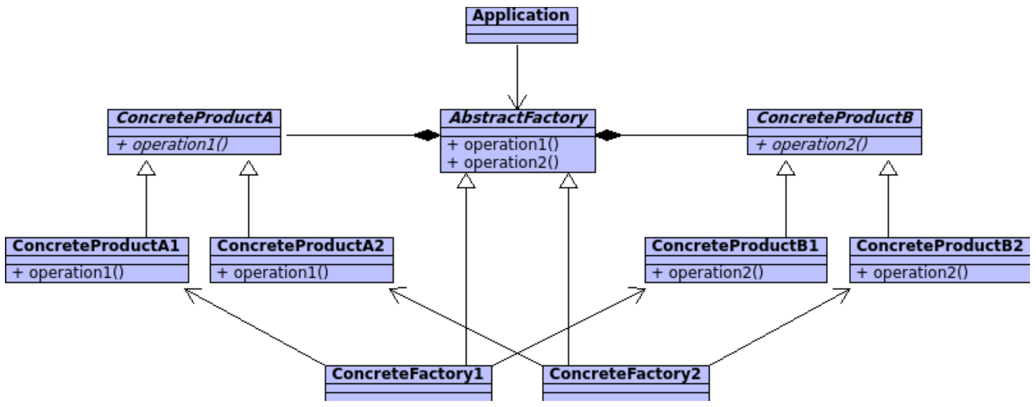
Evan Cummings
CSCI 441 – Advanced Programming
Fall '12 – Joel Henry

Abstract Factory and Bridge Pattern Integration

When using the Abstract Factory by itself you must have access to the classes that the factory produces; by integrating the Abstract Factory and Bridge patterns, you allow the application to call methods from a single object. By eliminating this requirement the user can still access the methods and information that it needs, while simplifying the task of creating new concrete factories. This leads to cleaner-looking and easier to maintain code, as the user is required to traverse less lines of code to find a solution.

Because the abstract factory class has been converted from an interface to an abstract class, the derived classes need to implement less, if any methods from the parent. One downside of this is that the abstract factory class must now contain the concrete products, and it may be confusing that the abstract factory is being used for something other than creating concrete factories. However, the usefulness of this pattern becomes clear when making the desired calls to concrete products, which are completely hidden from the user.

Maintaining and testing remains unchanged from the Abstract Factory pattern; there are the same number of connections to test. The pattern is moderately complex and does require a fair amount of time to comprehend – this is the pattern's main weakness (see figure).


Application
ConcreteProductA
+ operation1()
AbstractFactory
+ operation1()
+ operation2()
ConcreteProductB
+ operation2()
ConcreteProductA1
+ operation1()
ConcreteProductA2
+ operation1()
ConcreteProductB1
+ operation2()
ConcreteProductB2
+ operation2()
ConcreteFactory1
ConcreteFactory2