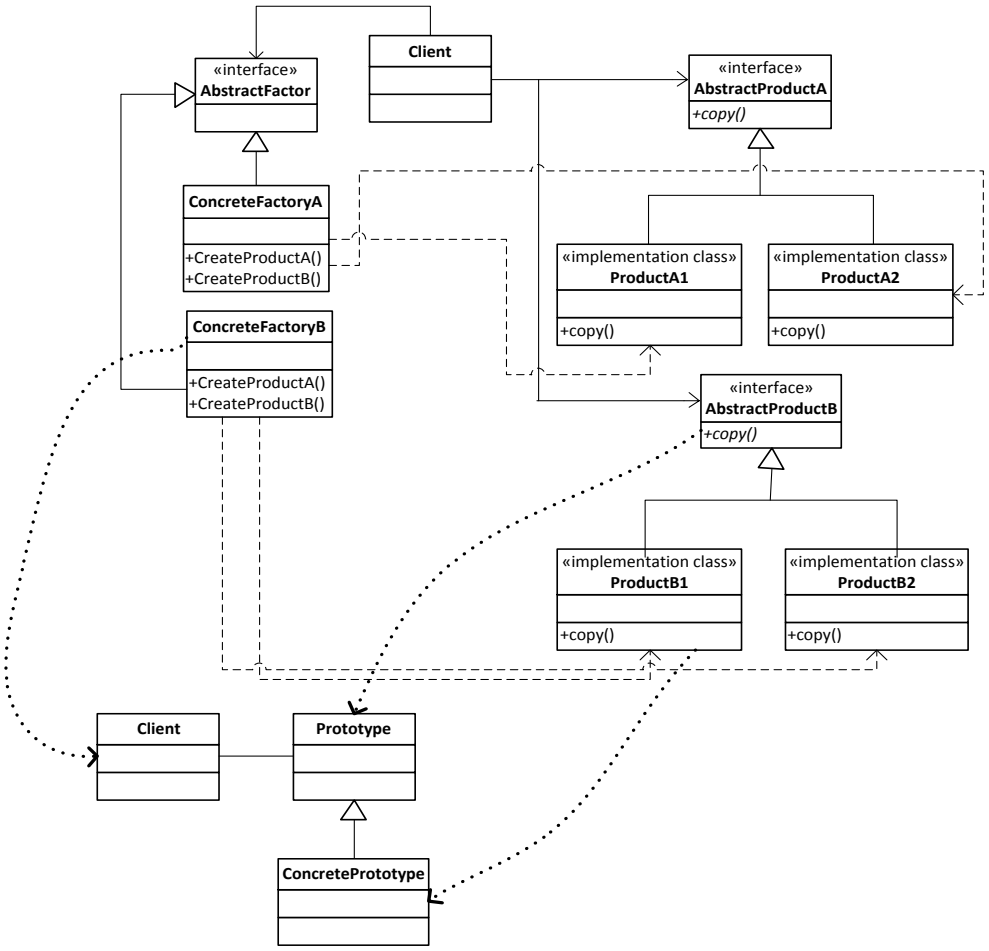
5. Prototype [Gamma et al]

When dealing with large objects, usually the cost factor is an impact on the performance. There are certain situations where we generates copies of objects instead of fresh ones. The prototype pattern is of such use where objects is cloned instead of newly created. The object to be cloned is termed as prototypical instance [1]. The clonning is advantegous in the sense that the process of building an object is skiped. Ready made clones are used enabling the system un aware of its objects being instantiated and represented. Usually clonning is provided as a language feature [15] in modern languages like C# and Java. Clonning of objects can be achieved by Shallow Copy and Deep Copy [54]. Shallow Copy is a default implementation of cloning in modern languages where as implementing Deep Copy need special care. Following is the variant of Prototype Pattern:

5.1 Abstract Factory / Prototype Compound [53]

This variant is a compound implementation extended from the concept provide by pluggable factories [21] [22]. Here both the pattern instances of abstract factory and prototype overlap each other and exist in a more standard form. In this variant the role of prototype participants are substituted by abstract factory particpants. Relationships between design patterns is best sketched by [55].



5.1 UML Diagram of prototype in compound implementation. [53]