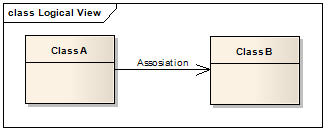
[**UML Class Diagram: Association, Aggregation and Composition**](http://aviadezra.blogspot.ie/2009/05/uml-association-aggregation-composition.html)

The UML Class diagram is used to visually describe the problem domain in terms of types of object (classes) related to each other in different ways.

There are three primary inter-object relationships: *association*, *aggregation*, and *composition.* Using the **right** relationship line is important for placing implicit restrictions on the visibility and propagation of changes to the related classes, matter which play major role in reducing system complexity.

**Association**

The most abstract way to describe static relationship between classes is using the ‘Association’ link, which simply states that there is some kind of a link or a dependency between two classes or more.

[](http://lh5.ggpht.com/_aUOgqE3fGXc/Sh32OSBGAdI/AAAAAAAAAZ4/HYw8B9VNlMs/s1600-h/image%5B4%5D.png)

**Weak Association**

ClassA may be linked to ClassB in order to show that one of its methods includes parameter of ClassB instance, or returns instance of ClassB.

[](http://lh3.ggpht.com/_aUOgqE3fGXc/Sh32SJJ1K6I/AAAAAAAAAaA/unIYlYJmUoI/s1600-h/image%5B9%5D.png)

**Strong Association**

ClassA may also be linked to ClassB in order to show that it holds reference to ClassB instance.

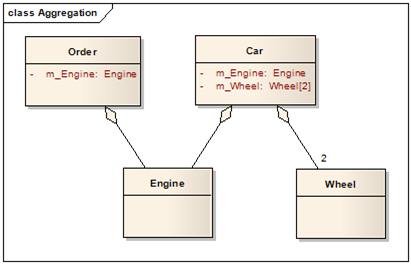
[](http://lh5.ggpht.com/_aUOgqE3fGXc/Sh32Wod9dyI/AAAAAAAAAaI/cWKKrsxaQ-8/s1600-h/image%5B14%5D.png)

**Aggregation (Shared Association)**

In cases where there’s a part-of relationship between ClassA (whole) and ClassB (part), we can be more specific and use the aggregation link instead of the association link, taking special notice that ClassB can also be aggregated by other classes in the application (therefore aggregation is also known as shared association).

[](http://lh6.ggpht.com/_aUOgqE3fGXc/Sh32ZWzvg3I/AAAAAAAAAaQ/mtsE0R03LwA/s1600-h/image%5B34%5D.png)

So basically, the aggregation link **doesn’t state** in any way that ClassA owns ClassB **nor** that there is a parent-child relationship (when parent deleted all its child’s are being deleted as a result) between the two. Actually, quite the opposite! The aggregation link usually used to stress the point that ClassA is not the exclusive container of ClassB, as in fact ClassB has another container.

[](http://lh4.ggpht.com/_aUOgqE3fGXc/Sh32cQ4pjVI/AAAAAAAAAaY/2lq7mZumWM8/s1600-h/image%5B24%5D.png)

**Aggregation v.s. Association**

The association link can replace the aggregation link in every situation, while aggregation cannot replace association in situations were there is only a ‘weak link’ between the classes, i.e. ClassA has method/s that contain parameter of ClassB but ClassA doesn’t hold reference to ClassB instance.

*Martin Fowler suggest that the aggregation link should not be used at all because it has no added value and it disturb consistency, Quoting  Jim Rumbaugh "Think of it as a modeling placebo".*

**Composition (Not-Shared Association)**

In cases where in addition to the part-of relationship between ClassA and ClassB - there’s a strong life cycle dependency between the two, meaning that when ClassA is deleted then ClassB is also deleted as a result, we should be more specific and use the composition link instead of the aggregation link or the association link.

[](http://lh5.ggpht.com/_aUOgqE3fGXc/Sh35YNKDw9I/AAAAAAAAAao/E4v4uDJcD5w/s1600-h/image%5B5%5D.png)

The composition link shows that a class (container, whole) has exclusive ownership over other class/s (parts), meaning that the container object and its parts constitute a parent-child/s relationship.

Unlike association and aggregation, in the composition relationship, the composed class cannot appear as a return type or parameter type of the composite class,  thus changes in the composed class cannot be propagated to the rest of the system. Consequently, usage of composition limits complexity growth as the system grows.

**Measuring system complexity**

System complexity can be measured simply by looking at a UML class diagram and evaluating the association, aggregation, and composition relationship lines. The way to measure complexity is to determine how many classes can be affected by changing a particular class. If class *A* exposes class *B*, then any given class that uses class *A* can theoretically be affected by changes to class *B*. The sum of the number of potentially affected classes for every class in the system is the total system complexity.