Chapter VI
Object Aggregation

ABSTRACT

This chapter describes the information and meanings that emerge from aggregates. It shows how the concepts like containment and subtyping are configured from the concept of location.

Classes, subclasses, compositions, and relationships are collections of object instances. So are systems, cars, and households. They are all examples of *aggregate objects.* Aggregate objects are collections of parts—parts that are also objects—structured, unstructured, or collected into sets based on a multitude of criteria. These collections have properties that are distinct from the properties of the members that constitute them.[1] We have found these collections everywhere in the metamodel of knowledge—in perspectives, in relationships, in domains of values, in patterns of things; they are found even in the concept of object class itself, the root from which the tree of knowledge grows.

Let us start by recapitulating what we already know about aggregate objects:

1. Aggregate objects are also object instances. This implies that each instance of an aggregate object must have a unique identifier and history (except in the case of a domain, which may not have history). The class of insurance claims is a collection of claims. An instance of the collection is an instance of an object, and the many instances of insurance claims in it are also instances of objects. Further, aggregate objects may themselves be aggregations of aggregate objects.
2. Aggregate objects can be any collection of objects, structured or loose. Some examples are:
   - Patterns:
   - Perspectives: Perspectives are topoi[2]—consistent, self contained, complete structures of knowledge. They are compositions of components, relationships, and rules valid within a scope (see Box 2.5).
   - Sets and lists: Sets do not distinguish between multiples of the same object among its members; lists do.
   - Object classes: Members of object classes share attributes and effects. Object classes do not distinguish between multiples of the same object among its members.

- Domains and value sets: Domains and value sets are aggregates of values.
- Compositions and other aggregates of objects.

3. Aggregate objects have emergent properties (see Box 4.1), which are different and distinct from the properties of their constituent objects, but are derived from them on the following basis:
   - The enumeration of its members is a universal emergent property of all aggregate objects.
   - Enumeration is an attribute of the aggregate, not of its constituent members. Indeed, each combination in Figure 5.4 may be considered to be an aggregate, and it is clear that some of these combinations can contain others. For example, the combination of three represented by the lowest double headed arrow in Figure 5.4, implicitly contains all the other combinations, and the combinations of two in the same Figure implicitly contain two object classes.
   - Enumeration constraints on aggregates may also include enumeration constraints on combinations of aggregates—cardinalities, degrees, and the order of a combination. Constraints on the order of an aggregate are constraints on the number of different object classes from which instances may be aggregated at any given moment.
   - Order, degree, and cardinality are emergent properties of aggregate objects; relationships inherit them from aggregates. Relationships are subtypes of aggregates—subtypes with added information on structures and meanings of structures.
   - The overall state of the aggregate is determined by the states of its contents.
   - Objects inside aggregate objects may be events. Aggregate objects may change state spontaneously if invisible internal events or events beyond the scope of the model change the state of the aggregate object.

Emergent properties are often distinguished from resultant properties of aggregates.[3]

An emergent property is a property of the aggregate that is independent of the properties of its parts, whereas a resultant property is derived from properties of the parts of an aggregate. The horsepower of an engine belongs to the engine alone and is not directly derived from attributes of its parts, whereas the weight of the engine is the sum of the weights of individual parts and is therefore derived from them. As such, *Horsepower* would be an emergent property of *Engine*, whereas its weight would be a resultant property.

However, in this book, we will not make this distinction. We will not distinguish between emergent and resultant properties because both are items of information conveyed by the existence of the aggregate. The only difference between them is that a resultant property conveys information on its derivation, whereas the emergent property does not. The emergent property seems to pop up magically because the structural details of the composition within the aggregate are unknown. The aggregate is a pattern. We may not know the pattern in its entirety (and are at liberty to discard even what we do know).

The only reason the horsepower of the engine seems to pop up magically from the aggregate of its parts, rather than being logically derived from known properties of those parts, is that we have ignored the structure—the pattern of parts that make the engine. Resultant properties may convey more information than emergent properties (information about their derivation from parts), but distinguishing between resultant and emergent properties when we recognize the "unknown" value is redundant. Therefore, unless it is explicitly stated otherwise, emergent and resultant properties (of aggregates) will mean the same in this book.