**GraphAlgorithm** : GraphAlgorithm object process information on graphs that is either the graph structure or specific information on nodes, edges and the graph itself. Each graph has typed input and output information

**Graph Information**: With this we mean the graph structure information that is modelled by the nodes and edges and the specialized information stored in nodes, edges and the graph under different keys. Graph Information are specific data types that are introduced in a specific graph processing context and are produced and consumed by the various stages of graph processing. Each graph algorithm has inputs and outputs with specific type registered in the **processing context**. Each different registered variable has a unique key in the processing context so as to be possible to be addressed by every algorithm of the processing context. The information storing into the graph is done using “object” type reference thus appropriate casting must be done by the algorithm that produces and consumes the variable.

**Processing Context** : Processing context is a processing flow on graphs. This includes the graphs that are produced and consumed, the information that is produced and consumed on the graphs as typed variables, non-graph processing and the corresponding data.

**Information** : The information that involves a graph processing context is raw typed data ( includes build-in and composite data types ) and graph data distributed in the graph structure ( nodes, edges, graph ). Information is represented as tuples <key,dataprovider> of type <object,object>.

* Graphs : For graphs the “key” member of the tuple refers to the key by which the graph information becomes available from the graph data storage ( nodes, edges, graphs ). In this case the “dataprovider” is the graph that stores the information
* Raw Typed Data : For data produced and consumed from processing not involving graphs. The “key” member of the tuple is NULL for this case and the dataprovider is the typed data.

It is also recommended to give a separate InfoKey for each separate property that we want to access from the input graph i.e use a separate InfoKey for Color property. In this case each InfoKey accesses a sole property thus the key can be named **PropertyKey**.

**Addressing the Information** : To address the information we use <key,dataprovider> tuples called PropertyKeys which are the regular input of the graph algorithm. However as described in **Graph Information,** the information that it points it must be casted appropriately.