## Sufficiency, Completeness and Primitiveness

Every component of a software system should be sufficient, complete, and primitive

**'Sufficient'** means that the component should capture those characteristics of an abstraction that are necessary to permit a meaningful and efficient interaction with the component.

**'Completeness'** means that a component should capture all relevant characteristics of its abstraction.

**'Primitiveness'** means that all the operations a component can perform can be implemented easily.

## Separation of Interface and Implementation

*An interface* defines the functionality provided by the component, interface is accessible by the clients of the component.

*An implementation* part that includes the actual code for the functionality provided by the component. The implementation part may also comprise additional functions and data structures that are only used internally to the component. The implementation part is not accessible by the component's clients.

## Separation of Policy and Implementation

A component of a software system should deal with policy or implementation, but not both

**A policy component** deals with context-sensitive decisions, knowledge about the semantics and interpretation of information.

***An implementation* component** deals with the execution of a fully specified algorithm in which no context-sensitive decisions have to be made.

If it is not possible to separate policy and implementation into different components within a software architecture, there should at least be a clear separation of policy and implementation functionality within a component.