04 - Component Software

Component software is a composite system made of software components. It allows software reuse, thus reduce the time spent and the money required. A software component is a unit of composition (black box, often compiled) with contractually specified interfaces and explicit context dependencies only (specification of the deployment and run-time environments, e.g. platforms, tools...). A software component can be deployed independently and is subject to composition by third party.

A component model defines a conceptual framework containing how the components are described and implemented. They should have:

- Component interface (what): operations (method calls, messages...);
- Composition mechanism (how): how components are composed (e.g. message passing);
- Component platform (where): platform for development and execution of components;

Indeed, components are generally composed of:

- Specification: description of the behaviour;
- Interface: definition of the behaviours offered;
- Implementation: realization of the specification;
- Installed Component: deployed copy of the implementation;
- Component Object: instance of the installed component (runtime concept).

Finally, CBSE (Component-Based Software Engineering) is the branch of SE that provides methods and tools to work with components.

Modules

Instead, a module is a part of a program to support development of large applications. They are the "conceptual parent" of classes: can be viewed as a collection of data with operations defined on them. It supports information hiding by encapsulating variables, data types and subroutines in a package:

- Objects inside of it are visible to each other;
- To make something visible outside, it must be exported;
- Depending on visibility politics:
  - Open scope: objects outside are visible;
  - Closed scope: objects outside are visible only when imported;
  - Selectively open scope: visible with qualified name `ModuleName.EntityName`;

Modules and components are different: modules are part of a program, component are part of a system (they could be written in different languages and already compiled, including static resources).

While Component Oriented Programming (COP) is oriented towards reuse, OOP is instead oriented towards representing appropriately domains using objects, classes, inheritance...