Notes Démarrage Bloc Programmation Système :

La qualité d'un logiciel peut s'évaluer à partir d'un ensemble de facteurs tels que :
- le logiciel répond-il aux besoins exprimés ?
- le logiciel demande-t-il peu d'efforts pour évoluer aux regards de nouveaux besoins ?
- le logiciel peut-il facilement être transféré d'une plate-forme à une autre ?

Les principes de conception :
Le principe de séparation des responsabilités (separation of concerns) vise à organiser un logiciel en plusieurs sous-parties, chacune ayant une responsabilité bien définie.

Le principe de responsabilité unique (single responsibility principle) stipule quant à lui que chaque sous-partie atomique d'un logiciel (exemple : une classe) doit avoir une unique responsabilité (une raison de changer) ou bien être elle-même décomposée en sous-parties.

Exemples d'applications des deux principes.
- Une sous-partie qui s'occupe des affichages à l'écran ne devrait pas comporter de traitements métier, ni de code en rapport avec l'accès aux données.
- Un composant de traitements métier (calcul scientifique ou financier, etc.) ne doit pas s'intéresser ni à l'affichage des données qu'il manipule ni à leur stockage.
- Une classe d'accès à une base de données (connexion, exécution de requêtes) ne devrait faire ni traitements métier ni affichage des informations.
- Une classe qui aurait deux raisons de changer devrait être scindée en deux classes distinctes.

Encapsulation max :
Au niveau d'une classe, cela consiste à ne donner le niveau d'accessibilité publique qu'à un nombre minimal de membres, qui seront le plus souvent des méthodes.
Au niveau d'une sous-partie d'application composée de plusieurs classes, cela consiste à rendre certaines classes privées afin d'interdire leur utilisation par le reste de l'application.

DRY est l'acronyme de Don't Repeat Yourself. Ce principe vise à éviter la redondance au travers de l'ensemble de l'application. Cette redondance est en effet l'un des principaux ennemis du développeur. Elle a les conséquences néfastes suivantes :
- augmentation du volume de code ;
- diminution de sa lisibilité
- risque d'apparition de bogues dus à des modifications incomplètes.

KISS est un autre acronyme signifiant Keep It Simple, Stupid et que l'on peut traduire par « Ne complique pas les choses ». Ce principe vise à privilégier autant que possible la simplicité lors de la construction d'une application.

Ce troisième acronyme signifie You Ain't Gonna Need It. Corollaire du précédent, il consiste à ne pas se baser sur d'hypothétiques évolutions futures pour faire les choix du présent

Il n'existe pas une architecture logicielle parfaite qui s'adapterait à toutes les exigences.
Au fil du temps et des projets, plusieurs architectures-types se sont dégagées.