# HAI913I - Évolution et restructuration des logiciels - 2

Cours: 2
Date: 04/10/2022
Professeur/intervenant: Seriai

---

## Notes de Cours :

Nécessité de comprendre le logiciel, donc le code source
Problème : les développeurs qui s'occupent du développement d'un logiciel ne sont souvent pas les développeurs qui feront la maintenance de ce même logiciel. Même problème pour les tests.
Objectifs de la compréhension du programme :

- Sa structure
- Ses fonctionnalités
- Son comportement dynamique
- Sa raison d'être

Le compréhension est une tâche difficile car il peut y avoir :

- un écart entre le cahier des charges et le fonctionnement réel
- un écart avec la documentation
- des erreurs
- etc

Différentes approches dans la compréhension :

- descendante -> on part des connaissances (fonctionnement global du logiciel) et on les projettes sur le code (mapper)
- ascendante -> on part des parties du code que l'on a analysés pour les combiner et comprendre le fonctionnement de l'application
- opportuniste -> Un mélange des deux autres cad que l'on connait quelques parties et une partie du fonctionnement du logiciel, et on assemble les deux pour essayer de comprendre le tout

Souvent, le seul moyen de comprendre un logiciel est son code source ou binaire.

Pour collecter des informations, on a les parseurs, les débogueurs, les profileurs (plugins qui se mettent sur la VM pour collecter des infos)...
Faire des modèles **compréhensibles et de haut niveau**

Il y a 3 différents concepts qui définissent une extraction d'architecture logicielle :

- composants
- connecteurs
- typologie

Un composant est une sorte de boîte noire dans laquelle on connait les interfaces qu'elle fournit, mais aussi les interfaces dont elle a besoin. Càd les fonctionnalités qu'elle déploit, mais aussi celles dont elle a besoin.

Deux types de classes dans un composant :

- Soit une classe qui appartient au noyau
  Ce type de classe n'a aucun lien avec des classes extérieures au composant
- Soit une casse qui appartient à l'interface
  Ce type de classe a un ou plusieurs lien(s) avec des classes extérieures au composant

Le principe est de découper une architecture logicielle en composants (partitions)
Ce découpage des composants doit avoir un sens
C'est à dire que les classes qui appartiennent à un composant ne sont pas choisies au hasard

Un composant est considéré composable si il définit clairement les services qu'il fournit et ceux qu'il requiert, à travers des interfaces
Il sera plus facile à assembler avec un autre si, dans chacune de ses interfaces, les services sont cohérents
Il est complètement autonome s'il ne possède pas d'interface requise

---

## Définitions :

Modèle statique : Modèle obtenu en examinant le code source et les documents connexes
Modèle dynamique : Modèle obtenu en exécutant l'application

Composant : Unité de composition possédant des interfaces spécifiées par contrat et des dépendances explicite avec le contexte. Il peut être déployé indépendamment et composé par un tiers. / Un élement logiciel composable sans modification, distribué de manière autonome et qui encapsule une fonctionnalité qui adhère au modèle de composant