I.1 Les Composants logiciels

Selon Wikipédia, un composant logiciel est un "élément constitutif d'un logiciel destiné à être incorporé en tant que pièce détachée dans des applications". On distingue plusieurs formats de composants logiciels comme par exemple Les packages (ex : fichier jar en Java), les bibliothèques (ex : DLL en C#), les fichiers exécutables (EXE en C#), etc.

Un composant peut se trouve sous la forme d'une extension qui se charge d'apporter des fonctionnalités spéciales à un logiciel particulier comme un plugin ou un driver.

De cette perspective, une simple fonction du langage C qui se trouve dans une bibliothèque (studio.h par exemple) est considérée comme un composant logiciel.

Dans ce cours, on va se focaliser sur les composants suivants :

- Composants .Net : EXE et DLL.
- JavaBeans.
- Entreprise Java Beans (EJBs).

La figure nous montre deux composants dans une format abstraite sous le modèle UML 2.0. Les composant "Checkout" et "CardProcesing" facilitent la tâche d'une application qui assure un service de payement. Le composant "Checkout" à besoin des services du composant "CardProcesing".

Les composants se communiquent à travers des interfaces qui sont représenté par des symboles spécifiques, à savoir le "open socket" qui symbolise l'interface du demandeur de service, et le "lollipop" qui symbolise celle du fournisseur.

I.2 L'intérêt des composants logiciels

L'objectif majeur qui motive la conception de composants logiciels est :

- La création des applications par composition. Le composant créé par un développeur va être **réutilisable** dans ses applications d'une part, ainsi que dans les applications des autres développeurs d'une autre part.
- Les composants nous permettent le découpage de l'application. De ce fait on peut séparer les rôles lors du développement (composants techniques et composants métier).

I.3 Les Caractéristiques d'un composant

Les caractéristiques d'un composant peuvent être regroupées en fonction de trois axes principaux :

Taille : elle varie d'un composant à l'autre, dont on trouve un composant qui encapsule une seule classe, ainsi il existe un composant qui encapsule toute une application.

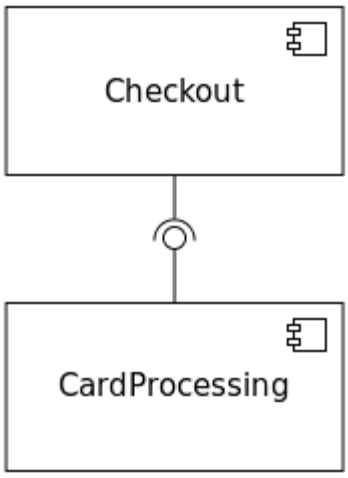
Granularité : représente le nombre d'éléments encapsulés dans le composant.

Domaine : ou la fonction du composant est le type de rôle qui assure au sein de l'application, à savoir :

- Fonctions techniques : tous ce qui concerne l'implémentation (ex : connexion réseau, base de données, affichage, etc.).
- Fonctions métiers : tous ce qui concerne la logique métier (business logique) de l'application vis-à-vis un domaine particulier (science, finance, etc.)

Généricité : signifie l'adaptabilité du composant (sans le modifier) vis-à-vis l'application qui va l'intégrer.

L'abstraction : impose au composant d'avoir une interface claire qui expose aux applications (locales ou distantes) tous les services qu'il peut offrir sans consulter son code source.



FigI : Le modèle abstrait d'un composant selon UML 2.0.

I.4 Les éléments d'un composant

Un composant logiciel doit avoir certains éléments à savoir :

- **Une interface :** à travers la quelle il communique avec les autres composants sur la même application. L'interface généralement montre la signature d'une classe pour facilite aux clients de connaitre les services offerts par le composant sans avoir besoin de consulter son implémentation.
- **Des attributs :** ce sont eux qui identifient le composant.
- **Une implémentation :** c'est le code attribué aux différentes méthodes du composant.

I.5 La programmation orientée composant (POC)

La POC est l'approche de programmation qui s'articule sur l'usage de composants logiciels. En anglais, on trouve souvent Component-based software engineering (CBSE), components-based development (CBD), ou Component-oriented Programming font tous référence à la POC.
La POC permet au développeurs une conception des applications qui se base majoritairement sur l'agrégation de composants logiciels déjà existantes.

Cette démarche offre aux applications une meilleure conception et une meilleure maintenance.
Ce type de programmation ressemble au domaine de l'électronique dont les appareils sont fabriqués à travers l'assemblage de plusieurs composants tels que les transistors, les diodes, les résistances, etc. Les deux avantages majeurs de cette approche sont :

- Lors de la conception, il suffit d'intégrer et d'assembler les composants (existants) nécessaires.
- Lors de la maintenance (ou mise à jour), il suffit de retirer un composant et de le remplacer par un autre.

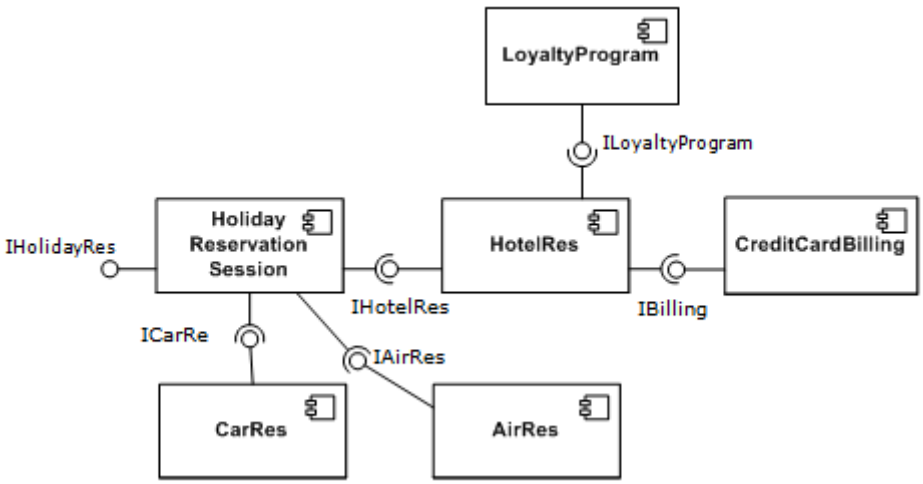


Fig2 : Plusieurs composants qui interagissent entre eux selon UML 2.0.

I.6 La POC vs La POO

La programmation orientée composants n'est jamais la démarche de programmation qui remplace la programmation orientée objet, mais plutôt c'est la nouvelle démarche qui fait évoluer la POO.
La POC utilise une approche objet pour le développement, l'intégration, et le déploiement des applications à base de composants logiciel.

Dans les prochains chapitres, nous nous assurerons que les composants qu'on va rencontrer ne sont rien d'autres que des objets.
Par rapport à la POO, la POC est spécialement pratique pour faciliter le travail en équipes dont chaque équipe assure le développement d'une partie de l'application qui va ensuite être assemblé avec les autres partie pour construire une application complète. Chaque partie peut être développée selon la POC comme un composant autonome, persistant, réutilisable et remplaçable.