Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Génie logiciel

Résumé / prise de note rush

Auteurs :
Marc Roten

Professeur :
Julien Tscherrig

26 septembre 2018

Table des matières

1 Introduction au génie logiciel

les attentes des clients sont de plus en plus grandissantes. Le rôle des logiciel est de plus en plus important. Il est donc essentiel que le client et le programmeur aient la meme vision du futur produit. Un projet de développement ou il y a des mésententes, le temps sera décuplé et donc le budget sera dépassé et risque de rupture ou refus du contrat par le client!!!

Le code doit être facilement modifiable, et en vue d'anticiper la scalabilité future du produit.

1.1 Problèmes liés à la mise en place système

1 système logiciel complexe

2 multi tache

3 Evolutivité du système

4 User interface adaptation

5 Connexion avec l'extérieur pour application de patch. L'objectif de ce cours est donc de *comprendreetdesynthétiser* les attentes des différents participants.

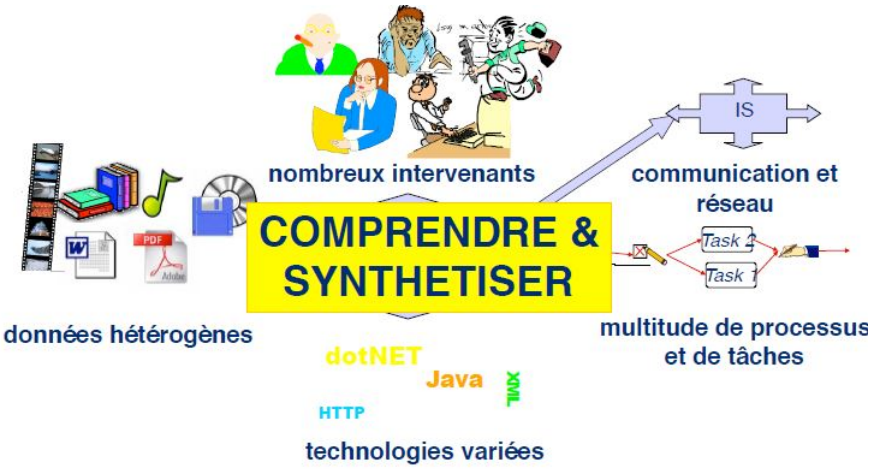


FIGURE 1 – comprendre et synthétiser

1.2 Problème de langage

Le manager utilisera un langage de manager, l'expert aura un langage technique, des divergences, incompréhension peuvent apparaitre. Il faura donc instaurer de la *communication*!!! entre les différents participants. Attention de ne pas oublier la communication avec le client final.

FIGURE 2 – la comunication la base du succès

1.3 Le génie logiciel définition

- Ensemble des programmes, procédés et règles, et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitement de données.
  - Dictionnaire Larousse

- En informatique, un logiciel est un ensemble d'informations relatives à des traitements effectués automatiquement par un appareil informatique. Y sont incluses les instructions de traitement, regroupées sous forme de programmes, des données et de la documentation.
  - Wikipedia

FIGURE 3 – définition d'un logiciel

En bref la définition ci-dessous suffit :

Construction collective d'un système complexe. Nécessité de pouvoir travailler en équipe, de pouvoir séparer le problème en différentes parties avant de synthétiser avec les collègues.

- Ce qui est fabriqué **répond aux besoins** des utilisateurs (correction fonctionnelle) et possède les **qualités attendues**
- La qualité correspond au contrat de service initial, en termes de **performance**, sûreté de **fonctionnement** et **robustesse**, sécurité, facilité d'évolution, portabilité, convivialité (« user friendliness »)…
- Les **coûts** et **délais** restent dans les limites prévues au départ

FIGURE 4 – concept

1.4 Réalisation du produit logiciel

- Utilise des méthodes et outils spécifiques à la réalisation de produits logiciels
  - Méthode COO (Conception Orienté Objet)
  - UML (Unified Modeling Language)
- C'est la partie technique du travail : suivi des activités
  - Analyse, Conception, Codage, Test

FIGURE 5 – réalisation logiciel

1.5 Raison des echecs

- Illusion : l'outil va résoudre le problème…
- Promesses non-tenues des standards : ne garantit pas le résultat mais garantit que vous suivez un processus (centré sur la méthode pas sur le résultat)
- Le risque d'échec est proportionnel à la taille du logiciel développé

FIGURE 6 – raison des échecs

1.6 Mise en garde client

Malgré un cahier des charges, il est nécessaire de parler au cient pour comprendre réelement ses attentes avant de développer le développement logiciel.

Les modification tardives sont beaucoup plus couteuses malgré que le logiciel aie été conçu en vue d'une scalabilité ultérieure ! !

1.7 Mise en garde développeurs

La maintenance est une partie fondamentale du développement logiciel.

La documentation n'est pas du temps perdu

1.8 Quand utiliser géniel logiciel

fabriquer une niche : non

fabriquer un garage : nécessité de prévoir mais on ne construit pas un garage sur plusieurs années

Fabriquer une villa : nombreux corps de métier devant travailler en cohésion

par là j'entends : Plus on s'oriente vers un projet complexe plus le génie logiciel aura une place prépondérante.

1.9 Qualités désirées du logiciel

- **Validité :** réalise exactement les tâches définies par sa spécification
- **Fiabilité :** assure de manière continue le service attendu
- **Robustesse :** fonctionne même dans des conditions anormales
- **Extensibilité :** s'adapte facilement aux changements de spécification
- **Ré-utilisabilité :** peut être réutilisé à d'autres fins (tout ou en partie)
- **Compatibilité :** peut être combiné avec d'autres logicielles
- **Efficacité :** utilise bien les ressources matérielles (mémoire, U.C., etc.)
- **Portabilité :** peut être porté sur de nouveaux environnements matériels et/ou logiciels
- **Traçabilité :** Identification/suivit d'un élément du cahier des charges lié à un composant logiciel
- **Vérifiabilité :** Facilité de préparation des procédures de recette et de certification
- **Intégrité :** Aptitude d'un logiciel à protéger ses différents composants contre des accès
- **Intelligibilité :** Facile à comprendre à utiliser à maintenir, documenté
- **Évolutivité:** On peut facilement rajouter des méthodes ou algorithmes
- **Documenté :** Il existe une documentation du programme et du processus de développement

FIGURE 7 – Qualité attendur d'un logiciel

2 Etapes de réalisation d'un logiciel

2.1 Construction d'une application

- S'assurer d'avoir bien compris le besoin de l'utilisateur afin de réaliser un code qui le satisfasse
- S'assurer d'avoir réaliser un code facilement modifiable, permettant de prendre en compte des évolutions futures
- Définir l'architecture d'application afin de bien comprendre les relations entre les composantes
- Réaliser une batterie de tests afin de mesure la « qualité » du code
- Effectuer un suivi des besoins de l'utilisateur afin d'intégrer des améliorations
- Effectuer des corrections de bogues - ils sont inévitables de toute façon
- Écrire la documentation utilisateur
- Écrire la documentation d'installation
- ...

FIGURE 8 – Checklist developpement application

2.1.1 Schéma bloc de la construction d'une appli

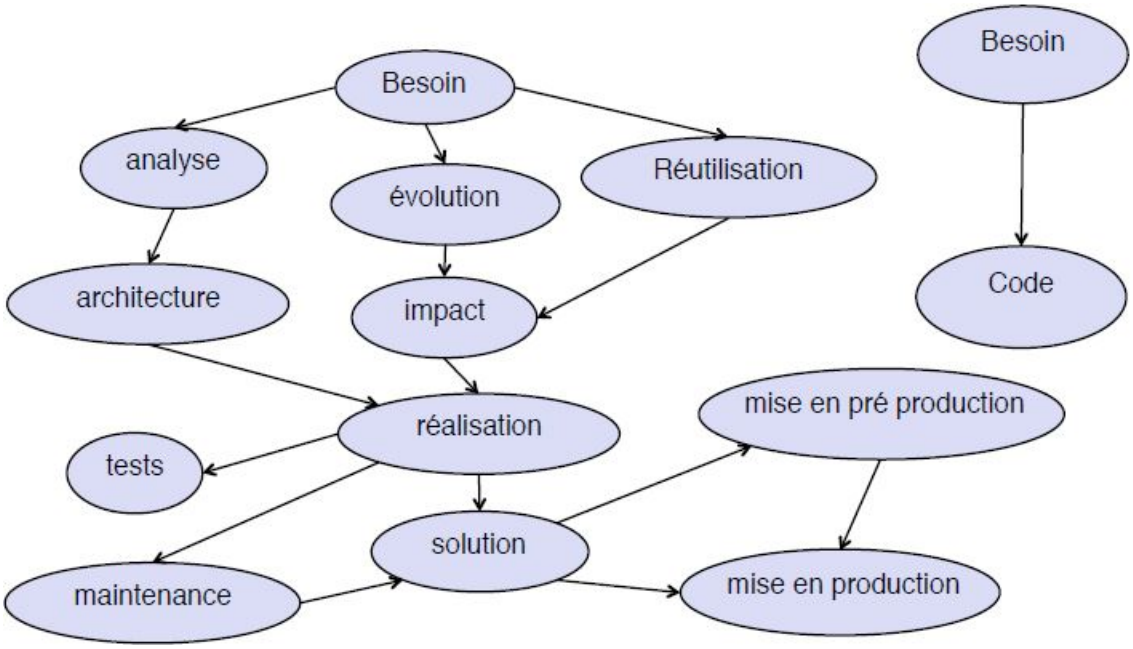


FIGURE 9 – Schéma bloc

Le code est la finalité, mais il ne faut pas que coder, que faire en cas d'erreur ? il faut une certaine logique derrière, toute une organisation.

2.2 Elaboration d'un projet

Voir cours de gestion de projet aux chapitre sur le SDLC, WaterFall, Effet tunnel.

3 Modélisation

Les modèles d'applications informatiques sont des représentations, à différents niveaux d'abstraction et selon plusieurs vues, de l'information nécessaire à la production et à l'évolution des applications

3.1 Pourquoi est-il nécessaire de modéliser

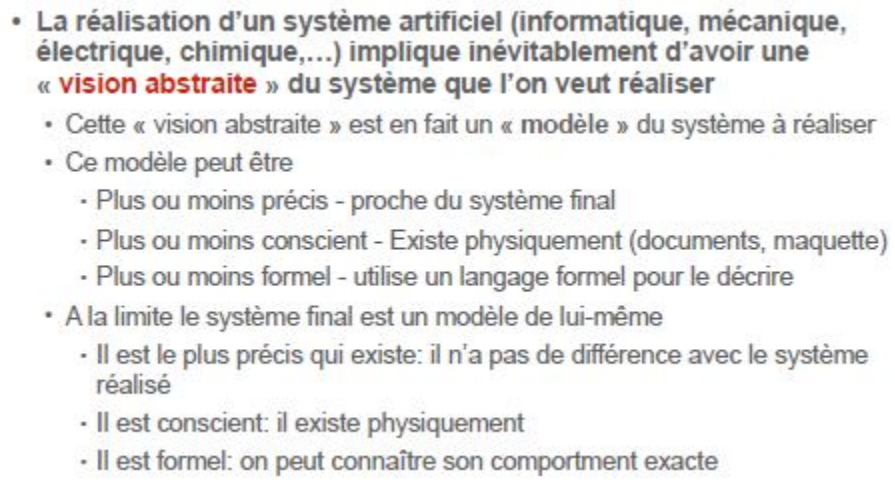


FIGURE 10 – nécessité de modéliser

Pourquoi modéliser ?

```
Mieux comprendre le fonctionnement du système.
Maîtriser la complexité et assurer la cohérence.
Vecteur privilégié pour communiquer.
Mieux répartir les tâches et dautomatiser certaines dentre elles.
Facteur de réduction des coûts et des délais (ex : génération automatique de code).
Assurer un bon niveau de qualité et une maintenance efficace.
```

3.2 Le bon et le mauvais modèle

- **Un modèle est une abstraction de la réalité…**
  - Oui mais qu'est-ce qu'une abstraction ??
    - Une abstraction est un processus intellectuel qui consiste à identifier les caractéristiques intéressantes d'une entité, **en vue d'une utilisation précise**
    - Le mot « abstraction » désigne également le résultat de ce processus.
      - On dira: ceci est une abstraction (un modèle) de cela.
- **Un « bon » modèle est une vue subjective mais pertinente de la réalité**
  - Subjective
    - Ce n'est pas un vue totalement générale et objective mais une vue « biaisée » par nos buts
  - Pertinente
    - Il doit retenir toutes les caractéristiques utiles à nos buts mais ignorer les caractéristiques qui n'ont pas d'intérêt pour nos buts

FIGURE 11 – un bon modèle

3.3 Code ou explication visuelle ?

3.3.1 UML

Génie Logiciel

```
Ensemble des activités de modélisation et de mise en oeuvre des
produits et des procédures tendant à rationaliser la production du logiciel
et son suivi
```

UML signifie "Langage de Modélisation Unifié"

```
Cest un langage
Visualiser, concevoir, développer, documenter, etc.
Cest générique
Il est indépendant de méthodes ou langages de programmation
Cest un langage visuel (diagrammes)
```

3.4 Comment représente-t-on l'UML

- **UML utilise une notation basée sur des diagrammes**
  - Les différents diagrammes d'UML permettent d'exprimer différentes « vues » du système
    - On fait souvent la distinction entre vision **statique** et vision **dynamique**
  - Aucun diagramme d'UML ne permet, à lui tout seul, de modéliser tous les aspects du système
    - Il faut utiliser plusieurs diagrammes pour modéliser un système

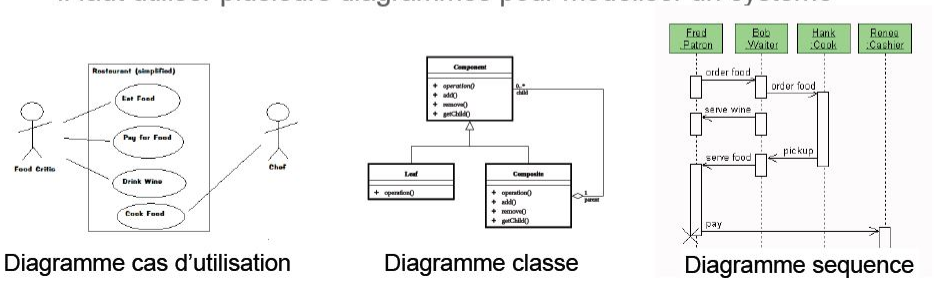


FIGURE 12 – Représentation de l'UML

4 Use Case

Définition

```
Un cas dutilisation (Use Case) représente un ensemble de
séquences dactions réalisées par le système et produisant
un résultat observable intéressant pour un acteur particulier.

Il modélise un service rendu par le système et exprime des
interaction acteurs/système qui apportent une valeur
ajoutée notable a lacteur concerné.
```

4.1 Analyse

```
Définitions des besoins

Découvrir et documenter les objets

Communication entre objets
```

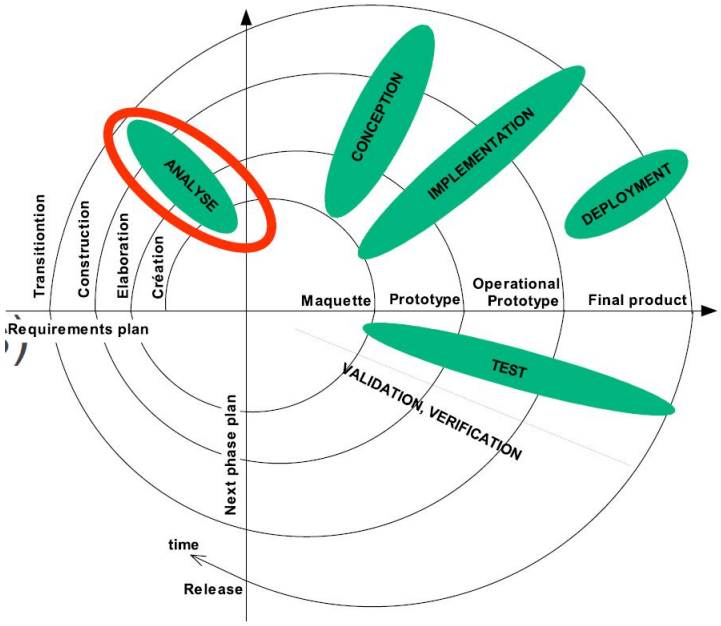


FIGURE 13 – Diagrammes de séquence et de collaboration

4.1.1 Diagramme Use Case

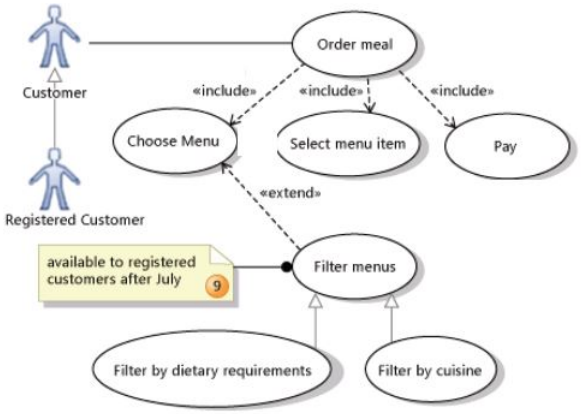


FIGURE 14 – Diagrammes des Use Case

4.2 Objectif des Use Case

Objectifs

```
Comprendre et structurer les besoins du client et les objectifs
correspondants dun système.

Décrire des interactions du système avec son environnement

On se concentre ici sur les préoccupations réelles des utilisateurs
```

4.2.1 En Bref

en bref

```
Les cas dutilisation serviront de base à la validation et à la traçabilité du système. Ils seront le fil
    conducteur du développement.
```

4.3 Découverte des UseCase

- **Comment découvrir les cas d'utilisation lors de la lecture d'un problème :**
  - Quelles sont les tâches de chaque acteur ?
  - Quels acteurs ou cas d'utilisation vont créer, stocker, modifier, supprimer ou lire de l'information ?
  - Est-ce que certains acteurs auront besoin d'informer le système ou d'être informés par le système sur des modifications soudaines, externes ?
  - Quels cas d'utilisation vont faire vivre et maintenir le système ?
  - Est-ce que tous les comportments principaux du système sont présents dans les cas d'utilisation ?

FIGURE 15 – Découverte des UseCase

4.4 Syntaxe

4.4.1 Syntaxe des useCase

- **Cas d'utilisation: est représenté par une ellipse**
  - Dialogue entre l'acteur et le système
  - Représente en général un service rendu (tâche exécutée par le système) à l'acteur qui interagit avec le système
  - Décrit par un ensemble de scénarios

- **Le cas d'utilisation (use case) montre les besoins de l'utilisateur**
  - Il décrit les processus que le système réalise par rapport aux acteurs.

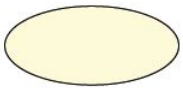


FIGURE 16 – Syntaxe des Use Cases

4.4.2 Syntaxe des Acteurs

Définition acteur

- **Acteur: est représenté par une icône dans le diagramme**
  - Représente le rôle joué par l'élément externe au système.
  - Un élément externe peut jouer plusieurs rôles : plusieurs acteurs
  - Plusieurs éléments externes peuvent jouer le même rôle : un seul acteur

- **Un acteur représente un rôle joué par une personne ou une chose qui interagit avec le système**
  - La même personne physique peut jouer le rôle de plusieurs acteurs
  - Un même rôle peut être jouer par différentes personnes physiques

FIGURE 17 – Syntaxe des Acteurs

lacteurestoujoursconsédérécommeélémentexterne

4.5 Découverte des Acteurs

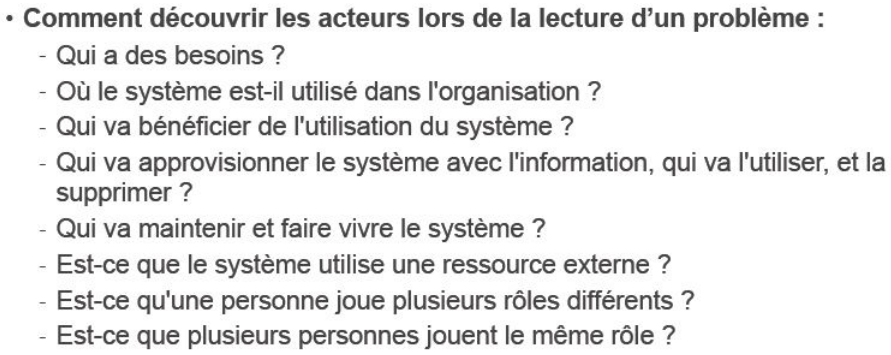


FIGURE 18 – Découverte des acteurs

4.6 Exercice

4.6.1 Exercice A du guichet médiathèque

- **Buts et besoins (point de vue)**
  - Il s'agira d'un système de guichet automatique où l'utilisateur pourra s'inscrire et se désinscrire à la médiathèque. Lorsqu'il sera inscrit, il pourra choisir dans un menu informatisé le document qu'il désire emprunter et prendre le document. Pour le retour de documents, l'utilisateur dépose simplement le document dans le guichet.
  - *Pour l'instant il est possible d'emprunter les documents suivants: des livres, des CD-Audio, des cassette audio, des jeux sur CD*

- **Exercices**
  - Trouver les acteurs de la médiathèque et décrire leurs interactions possible avec le système
  - Identifier les cas d'utilisation de la médiathèque et les associer aux acteurs concernés

FIGURE 19 – Exercice Guichet médiathèque

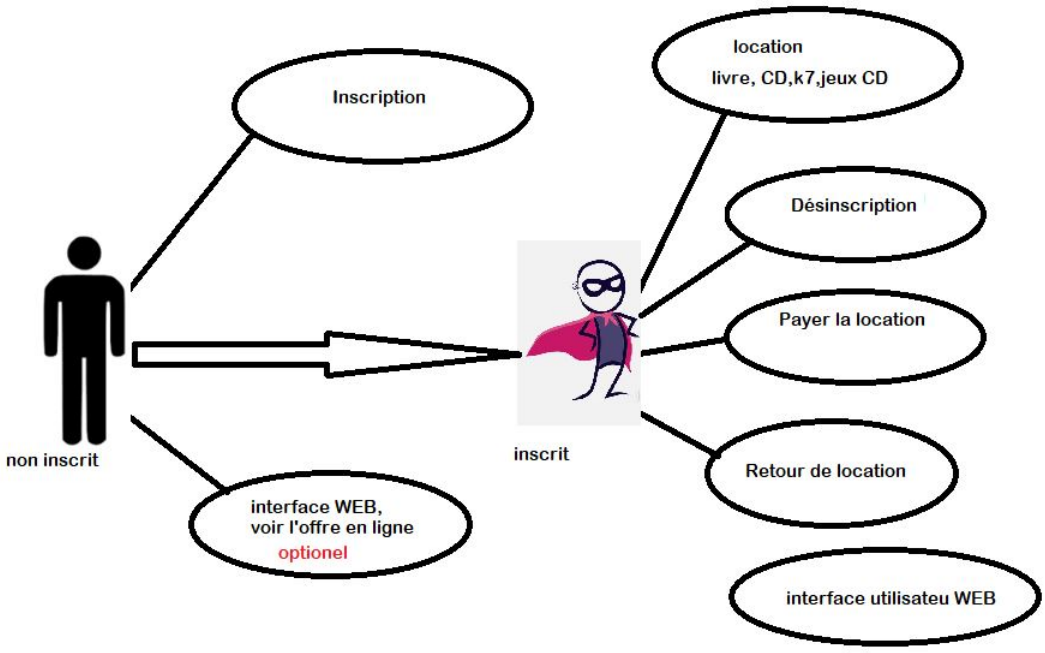


FIGURE 20 – Réponse

4.6.2 Exercices B

demander la correction

_____

_____

- **En se basant sur l'exercice A, on précise que**
  - Lors de chaque utilisation de la médiathèque l'utilisateur doit se loguer.
  - lors de l'utilisation (emprunt) ou de l'inscription, l'utilisateur peut, sur demande, consulter une aide en ligne.

- **Exercice**
  - On demande de compléter le diagramme de cas d'utilisation de la médiathèque pour tenir compte de ces ajouts

FIGURE 21 – donnée

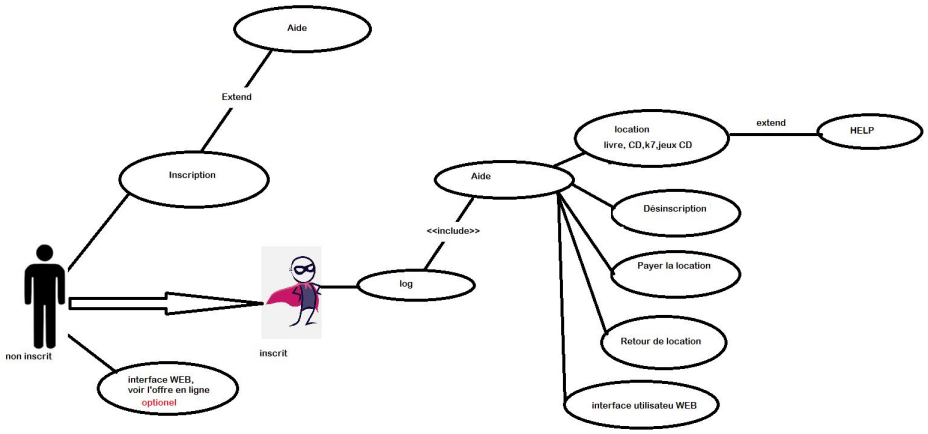


FIGURE 22 – réponse

4.6.3 Exercice C

- **Exercice**
  - En se basant sur l'exercice B, proposez un diagramme de cas d'utilisation pour la médiathèque qui utilise la généralisation

FIGURE 23 – donnée

4.7 Lien entre acteur et usecase

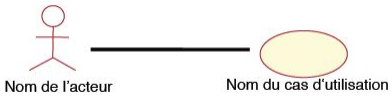
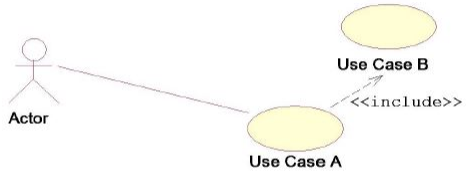


FIGURE 24 – lien entre les deux

4.7.1 Include

La relation <<include>> existe quand un comportement ou un fragment de comportement est similaire dans plusieurs cas d'utilisations



Le Use Case A inclut le Use Case B

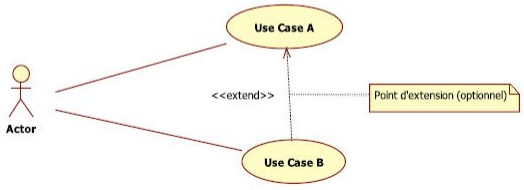
Attention : Ne pas tomber dans le travers consistant à faire du fonctionnel

Je découpe mes comportements en petits comportements et j'utilise la relation d'inclusion comme moyen d'exprimer un décomposition fonctionnelle

FIGURE 25 – Include

4.7.2 Extend

- **L'extension permet d'ajouter un comportement (optionnel) à un cas d'utilisation.**
  - Ces extensions correspondent à des activées optionnelles
  - Ils peuvent être activé par des « points d'extensions » (ou points de variance)
    - *Les point d'extensions peuvent être indiqués sur la relation d'extension (flèche) qui relie les cas d'utilisation.*



Le Use Case B etend le Use Case A

FIGURE 26 – Extend

4.7.3 inclusion vs généralisation

Il peut-être parfois difficile de décider s'il faut utiliser la généralisation ou l'extension.

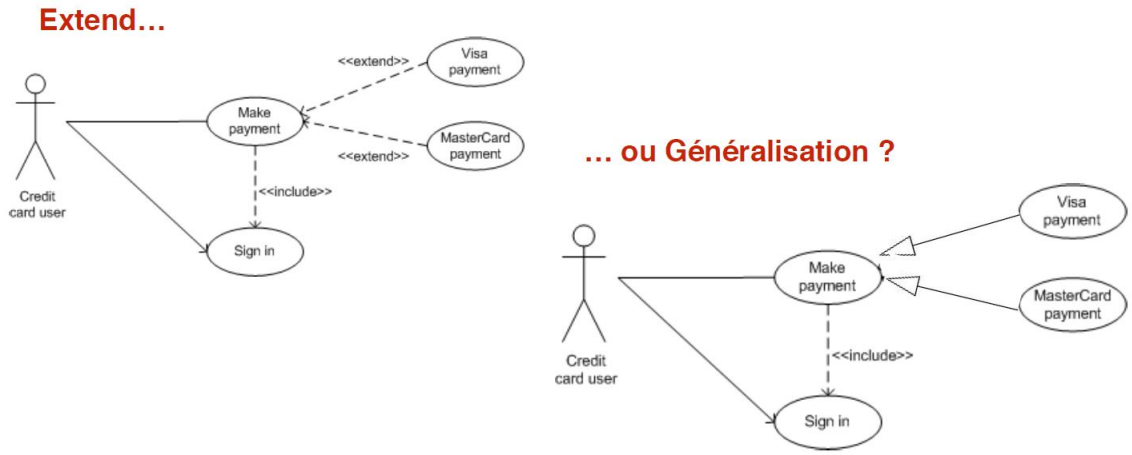


FIGURE 27 – include ou généralisation