La conception.

La conception consiste à décrire le **comment** sans aller jusqu'à la réalisation complète du système. Le modèle de conception décrit les objets, leurs relations, leurs interactions réciproques, l'allocation et la répartition sur les ressources physiques de traitement ainsi que les activités concurrentes, leur ordonnancement et leur synchronisation dans le système.
La conception va refléter la manière dont l'application (logiciel, système) va être réalisée, elle met en évidence la réflexion du concepteur (informaticien) par rapport à la réalisation. Dans la phase de conception, on ne peut pas faire abstraction des différents logiciels et matériels qui seront utilisés pour la réalisation.
Dans cette partie, on retrouve également le découpage du système en modules

Objectifs de la conception :

1- acquérir une compréhension approfondie des questions concernant les exigences non fonctionnelles et les contraintes liées aux langages de programmation, à la réutilisation des composants, aux systèmes d'exploitation, etc.
2- constitue un point d'entrée pour l'implémentation, au sens ou l'implémentation est un raffinement direct de la conception.
3- décomposer le système en plusieurs sous-systèmes
4- déterminer les principales interfaces des sous-systèmes.

Il existe deux niveaux de conception :

- Générale ou d'architecture (dite également de haut niveau): Elle définit les mécanismes communs, et les choix importants du système. Le diagramme de classes global et le diagramme de déploiement peuvent être utilisés
- Détaillée, elle s'intéresse au niveau élémentaire (classes, relations, et on utilise également le diagramme de classes et les diagrammes dynamiques: séquences, état ..)

I. Le modèle de conception :

C'est un modèle objet décrivant la réalisation physique des cas d'utilisation. Il constitue une entrée majeure à l'implémentation.

Dans le modèle de conception on retrouve **:**

- le raffinement des cas d'utilisation : le but est de rechercher une solution à la problématique. Les cas d'utilisation doivent être raffinés afin de faire ressortir les concepts qui appartiennent au domaine de la solution

- définir la présentation du logiciel : le but de cette activité est décrire d'une manière logique l'interface de présentation du logiciel. Dans la plus part des cas cela signifie la présentation de l'interface graphique.
- Définir le diagramme de classes : c'est à partir de classes dégagées des cas d'utilisation que l'on peut identifier. Les classes à ce niveau sont dites « classe de conception »
- Définir les bases de données : pour une application de grande envergure, elles nécessitent dans la plus part des cas de définir le modèle de base de données

II. **Classe de conception** :

Une classe de conception est une abstraction d'une classe d'implémentation du système.

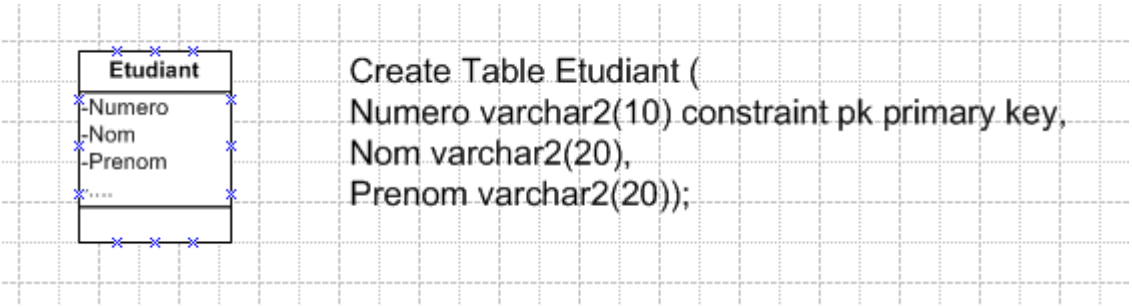
- La spécification d'une classe de conception utilise le même langage que le langage de programmation (paramètre, attribut, méthodes...)
- La visibilité des attributs est définie au niveau de la classe de conception (proected, private, public)
- Les relations liant une classe à une autre trouve un signification directe lors de l'implémentation, surtout les relation d'héritage
- Les méthodes de classe de conception présentent des liens directs avec les méthodes programmées.

Modèle de conception utilisant SQL : Passage du modèle de classes (Classes avec données persistantes ou classes «entités» à SQL

Règle 1 :Les classes.

Chaque classe devient une relation. Il faudra choisir des attributs qui pourront jouer le rôle de clé primaire

Exemple

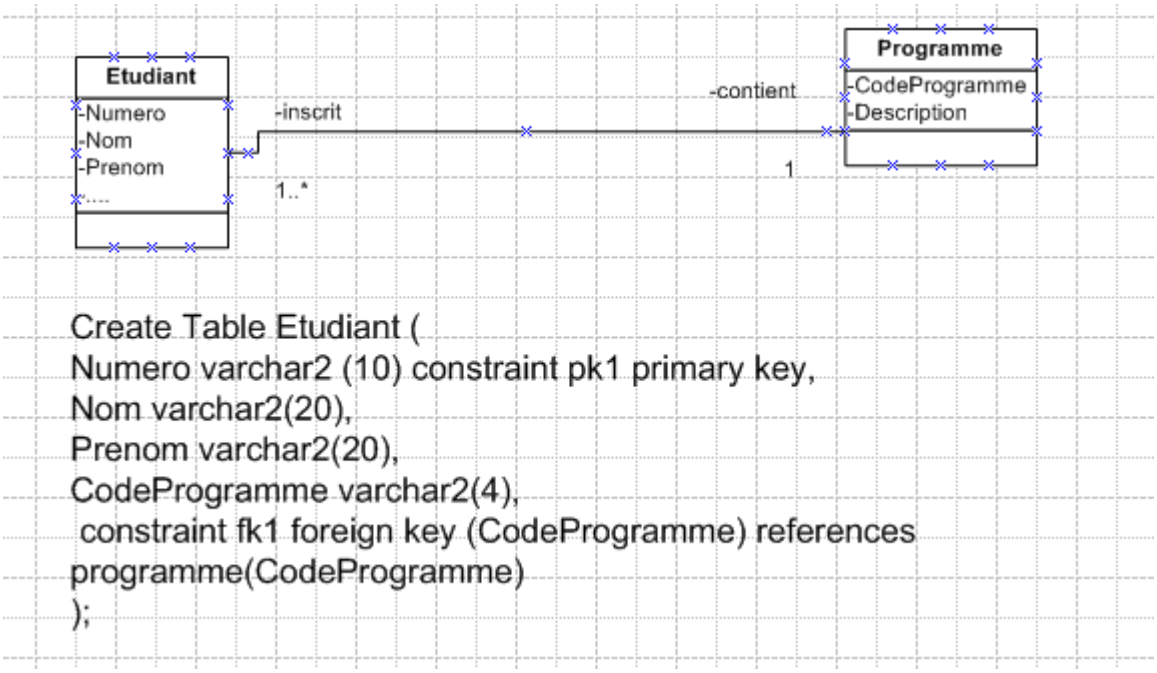


Règle 2 : Traduction des associations

1. **Cas d'une multiplicité (m,n) d'un coté et de (0,1 ou 1,1 ou encore 1) de l'autre côté.**

On ajoute la clé de la calasse ayant de son côté la multiplicité (1,1) ou (0,1) ou 1 dans l'autre table. La clé ajoutée est une clé étrangère.

Exemple:



(Évidemment il faut créer la table Programme avec les champs CodeProgramme (clé primaire) et description)

2. **Cas d'une multiplicité (m,n) d'un coté et de (m,n) de l'autre côté.**

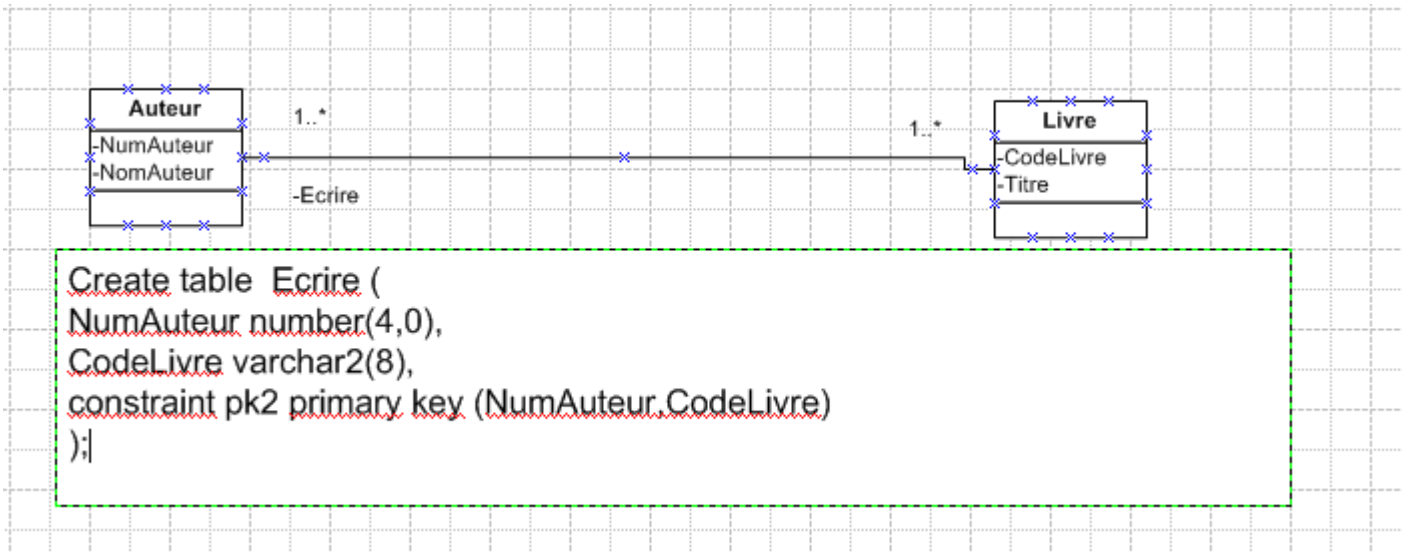
Ajouter une table qui aura comme champs les deux clés primaire des deux tatbles.

Exemple

Il est clair que les tables Auteur et Livre doivent être crées.

Pour **Auteur** nous aurons les champs NumAuteur comme clé primaire et NomAuteur

Pour la table Livre, nous aurons les champs CodeLivre comme clé primaire et Titre.

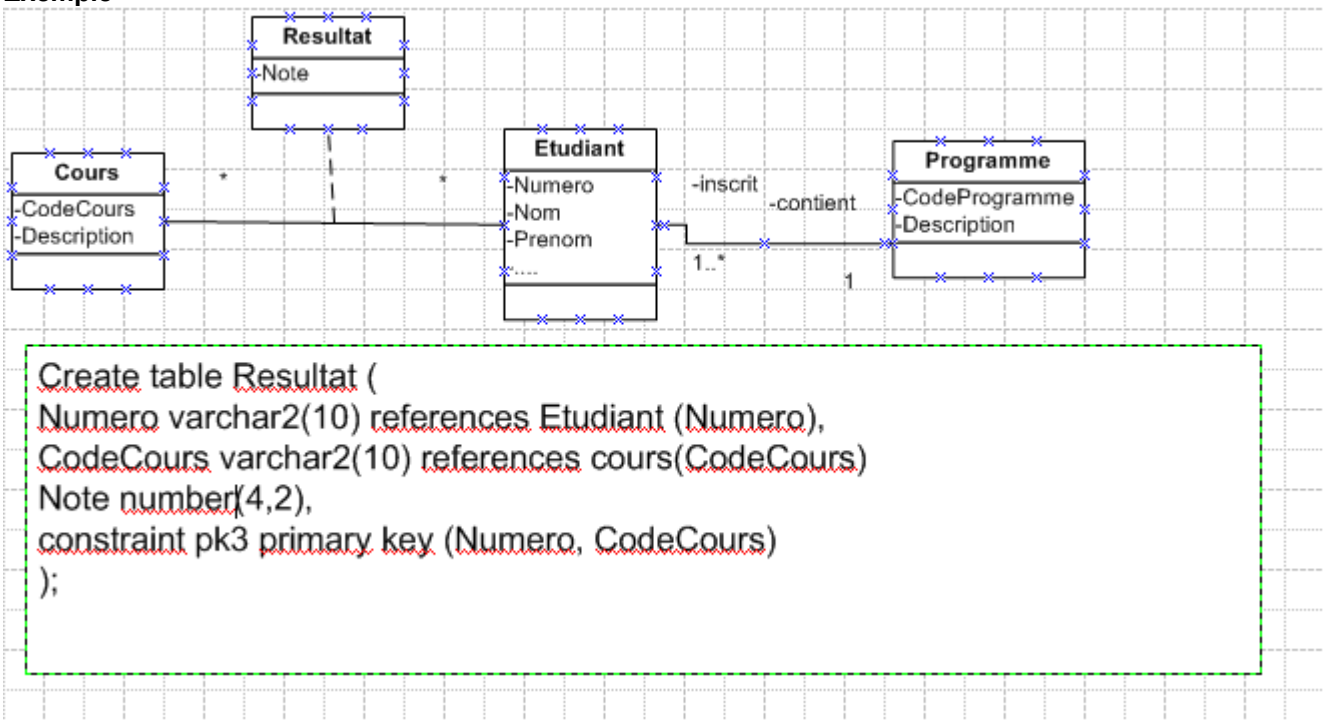



```
Create table  Ecrire (
NumAuteur number(4,0),
CodeLivre varchar2(8),
constraint pk2 primary key (NumAuteur,CodeLivre)
);
```

Règle 3 : Classe d'association

La classe d'association devient une table, ses champs sont les attributs de cette relation ET les deux clés primaires issues des tables d'association. Elle a comme clé primaire une cléc primaire composée. (la composition des deux clés primaires des deux autres classes)

Exemple

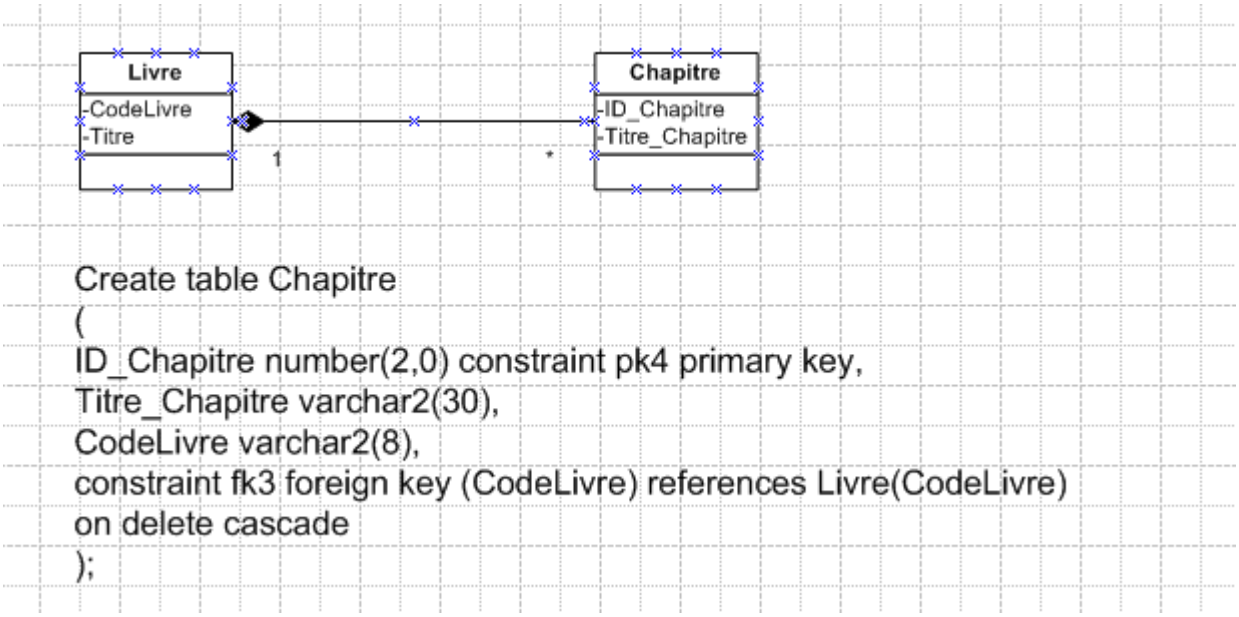



```
Create table Resultat (
Numero varchar2(10) references Etudiant (Numero),
CodeCours varchar2(10) references cours(CodeCours)
Note number(4,2),
constraint pk3 primary key (Numero, CodeCours)
);
```

Règle 4 : relation de composition.

Cette règle s'utilise de la même manière que la règle 2, sauf que dans le cas **d'une composition** forte, il faudra utiliser le ON DELETE CASCADE.

Exemple



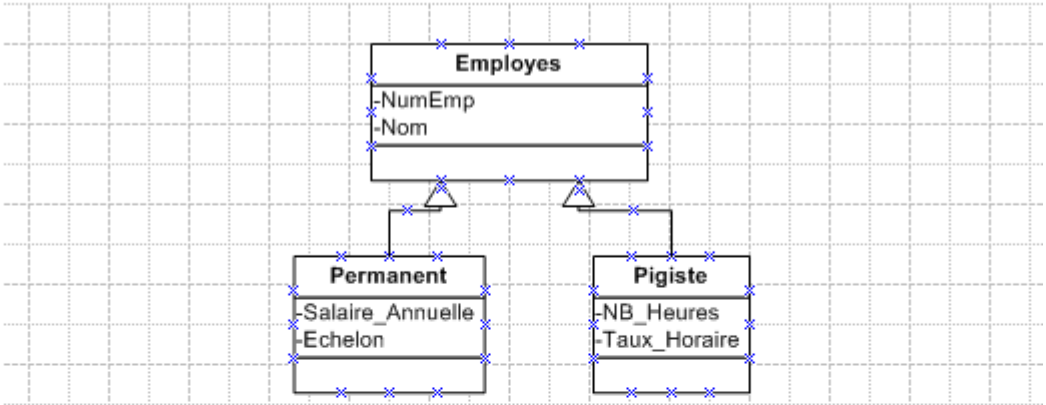
Il faudra créer la table Livre normalement, nous aurons les champs CodeLivre comme clé primaire et Titre comme champs sans contrainte particulière.

Règle 5 : relation de généralisation et de composition.

Deux solutions sont disponibles pour cette règle. Il s'agira d'en choisir une selon les besoins du système à l'étude.

Solution 1 : créer une table qui aura pour nom le nom de la super-classe (classe parent)et inclure TOUT les attributs des classes dérivées (classes enfants) dans cette table. Le problème de cette solution est que nous risquons d'avoir beaucoup de rubriques NULLES.

Exemple




```
Create table employes (
NumEmp number(6,0) constraint pk8 primary key,
Nom ------------------------
Salaire_Annuelle    ----------
Echelon ---------
NB_Heures ---------
Taux_Horaire --------
);
```

Solution 2 :

Avoir une table correspondant à la super-classe.(classe parent). Les champs de cette table, sont les attributs de la classe dont elle est issue.

Créer autant de tables qu'il ya de classes enfants. Les champs de tables sont les attributs des classes dont elles sont issues ET la clé primaire de la classe parent.

```
Create table employes (
NumEmp number(6,0) constraint pk8 primary key,
Nom ------------------------
);

Create table Permanent (
NumEmp number(6,0) constraint pk8 primary key,
Salaire_Annuelle---------
Echelon ------
);

Create table Pigiste (
NumEmp number(6,0) constraint pk8 primary key,
NB_Heures ---------
Taux_Horaire --------
);
```

III. Le modèle d'analyse VS le modèle de conception.

Modèle d'analyse	Modèle de conception
Modèle conceptuel, car c'est une abstraction du système qui évite les questions d'implémentation	Modèle physique, puisque c'est un plan d'élaboration et de construction de l'implémentation
Générique à la conception : applicable à plusieurs niveaux	Non générique : Spécifique à une implémentation
Trois stéréotypes de classes : Entités, frontière (interface) et de contrôle	Autant de stéréotypes (physiques) de classes que le permet le langage de programmation
Moins formel	Plus formel
Peu coûteux	Plus coûteux (5 à1 par rapport d'analyse)
Dynamique	Dynamique, avec intérêt à la séquence
Trace les grandes lignes de la conception, y compris son architecture	Exprime la conception du système
Émerge d'un « marathon » entres les ateliers et les divers lieux de réflexion	Emerge principalement de la programmation visuelle « roud-trip engineering », on effectue des allées retours entre le modèle de conception et le modèle d'implémentation
Risque de ne pas être conservé tout au long du développement du système	Doit être conservé tout au long du développement du système
Définit une structure constituant une base essentielle pour la création du système	Façonne le système, tout en essayant de respecter autant que possible ce qui est défini par le modèle d'analyse

Source : Le processus unifié de développement de logiciels par Ivar Jacobson, Grady Booch et James Rumbaugh

IV. Réalisation-conception du cas d'utilisation :

Une réalisation-conception de cas d'utilisation fournit une réalisation physique de la réalisation-analyse de cas d'utilisation et prend en compte les exigences non fonctionnelles (exigences particulières). On y définie :

- Diagramme de classes, en utilisant les classes de conception
- Diagrammes d'interaction : la séquence d'action d'un cas d'utilisation débute lorsqu'un acteur invoque celui-ci en envoyant un message au système. En conception, on préfère représenter cette interaction par le biais d'un diagramme de séquences

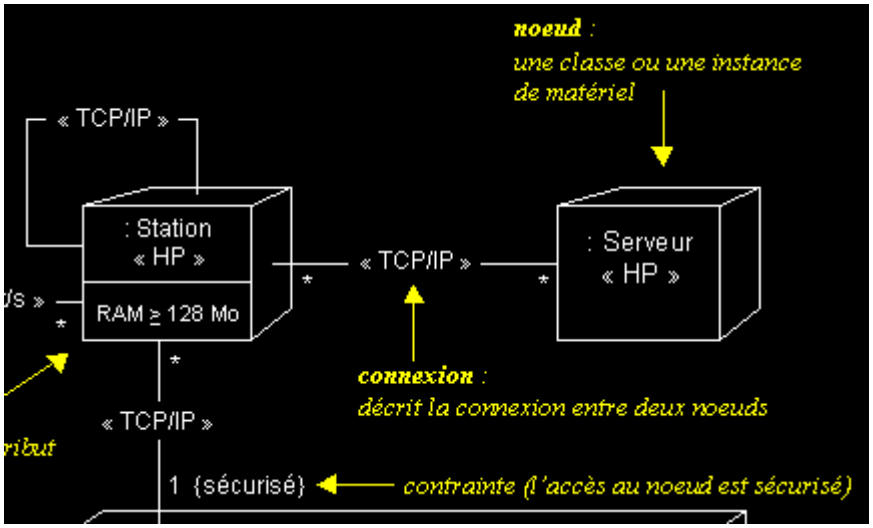
- Flot d'événement : description textuelles du diagramme d'interaction des cas d'utilisation
- Exigences d'implémentation : présentées sous forme d'une description textuelle.

V. Le modèle de déploiement

Le modèle de déploiement représente la distribution des composants sur différents matériels. Il constitue une entrée importante pour le modèle de conception et d'implémentation.
Notez que le déploiement du système chez le client doit se faire conformément au diagramme de déploiement conçu par l'équipe de développement. On peut représenter le modèle de déploiement par un diagramme de déploiement.
Un diagramme de déploiement illustre l'architecture physique du matériel et du logiciel du système

- Les diagrammes de déploiement montrent la disposition physique des matériels qui composent le système et la répartition des composants sur ces matériels.
- Les ressources matérielles sont représentées sous forme de noeuds.
- Les noeuds sont connectés entre eux, à l'aide d'un support de communication. La nature des lignes de communication et leurs caractéristiques peuvent être précisées.
- Les diagrammes de déploiement peuvent montrer des instances de noeuds (un matériel précis) ou des classes de noeuds.

Exemple



Source : http://uml.free.fr/

Contenu d'un doucement de conception

Nous l'avons mentionné, au début du cours précèdent que les méthodes orientées objets favorisent le développement par raffinement.. UML propose que les concepts et les notations utilisées durant l'analyse continuent à être utilisés pour la conception. Cette approche s'appelle ***l'élaboration. L*****es modèles de l'analyse et de la conception ne diffèrent que par le niveau de détail des spécifications des éléments utilisés.** UML ne définit pas de barrière entre les activités d'analyse et de la conception, c'est une approche simplificatrice.

- Introduction
- Historique
- Droits d'auteur
- Références
- Introduction, rappel de la description de l'application (Système, logiciel..)
- Liste des équipements matériels et logiciels qui seront utilisés
- Liste des librairies à utiliser
- Diagramme de déploiement
- Découpage de l'application en modules
    - Pour chaque module :
        - Diagramme de classes au niveau conception donc en tenant compte du langage d'implémentation. S'il s'agit d'un modèle de classes «entité» alors il faudra tenir compte du SGBD à utiliser à la phase d'implémentation
        - Description du diagramme de classes
        - Mettre en évidence les relations d'héritage
        - Diagrammes dynamiques pour les méthodes les plus importantes (séquences, états ....)
        - Interfaces
- Conclusion
- Annexes