Conception Orientée Objet

Objectifs du Cours

prof : Kim Mens

Thèmes

- Introduction aux bases de données et modélisation de données
- Conception de programmes orientés objet
- Méthodologie d'aide au développement de programmes
- Réalisation (analyse et implémentation) de programmes Java (android) de complexité moyenne

Durée

9 semaines à raison de 2H/semaine

Projet - EZMeal

1. Ecran de Login
2. Gestion du profil utilisateur
3. Menu d'accueil
4. Recettes recommandées
5. Catalogue de recettes
6. Détail d'une recette
7. Recherche de recettes
8. différentes extensions possibles

Contenu du cours

1. Gestionn de données
2. Conception orientée Objet
3. Programmation Android

Evaluation

50 % projet et 50 % Travail
Il faut un minimum de 10/20 à chaqu'un d'entre eux pour réussir

Examen

Modélisation et conception d'une application similaire à l'étude de cas développée dans les séances pratiques

5 questions sur des sujets vus au cours

ORM, SQL, diagrammes de classes, diagrammes de séquences, ...

» Question type: à partir d'un modèle incomplet
» corrigez / critiquez / discutez ce modèle
» complétez ou raffinez le modèle pour une extension donnée

Introduction

Une Base de Donnée : est un système informatique de stockage d'informations. Les plus utilisés s'appuient sur le **modèle relationnel** et utilisent le **language SQL**

```
select Item1, Item2, Item3
from DataBase
where condition
```

ceci retourne un tableau de 3 colonnes et d'autant de lignes qu'il y a d'éléments dans la BD qui satisfont la condition.

```
select D1.ITEM2, D2.ITEM3, sum(D.QUANTITE*P.PRIX)
from CLIENT D1 , DETAIL D , PRODUIT P
where C.CLI + D.NCLI
and D.NPRO = P.NPRO
group by C.LOCALITE, P.NPRO
```

Une **base de données** est constituée d'un ensemble de **tables**.
Une **table** contient les données relatives à une **collection d 'entités** de même nature.
Chaque **ligne** d'une table reprend les données relatives à une **entité**.
Chaque **colonne** d'une table décrit une propriété commune des entités.
Un **identifiant** de la table est Le jeu de colonnes dont les valeurs sont uniques.
Une **clé étrangère** vers cette autre table sont les lignes d'une table peuvent faire référence chacune à une ligne d'une autre table.
On évite d'enregistrer des données qu'il est possible de calculer à partir d'autres données enregistrées.

Les SGBD (Systèmes de Gestion de Bases de Donées)

Un système de gestion de bases de données est un système informatique permettant de gérer une BD.(maintenir la DB & répondre à des requêtes)

» Organisation des données
» Gestion des données
» Accès aux données
» Protection contre les accidents
» Gestion des accès concurrents
» Contrôle des accès

SQLite

Le moteur de **base de données relationnelles** est accessible par le language SQL.

» il doit être intégré aux programmes et n'a pas besoin de serveur séparé.
» Sa particularité est de stocker toutes ses données dans un seul fichier .sqlite sur le disque
» Sqlite est utilisé pour les données des applications de type Android et IPhone

Univers de discours (UoD)

L'UoD décrit le domaine de l'application. Cet univers décrit typiquement une partie du monde "réel".

ORM

ORM = Object-Role Modeling
Base de données = ensemble de relations n-aires
Schéma de base de données = constitué de l'ensemble des signatures des relations de la base.

Le modèle relationnel ne se base pas sur la notation positionelle (1,2,3... comme un tableau) mais sur les identificateur pour identifier les entitées.

L'ORM est *fact-oriented* = modélise l'info comme des **faits** suivant des **règles**.

Exemples

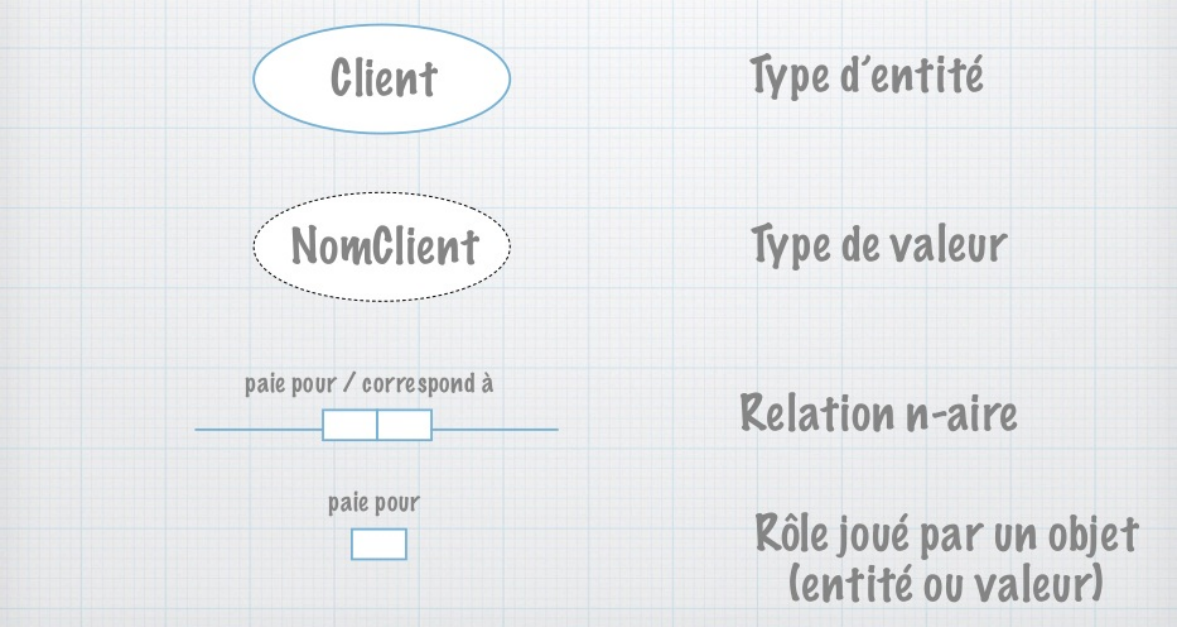
(12,Antoine,Lambert,SINF12BA) : **Tuple**
(ID,NOM,PRENOM,UCL_INFO) : **Shéma de la relation**
STUDENT(ID,NOM,PRENOM,UCL_INFO) : **Signature de la relation n-aire**

Modélisation ORM

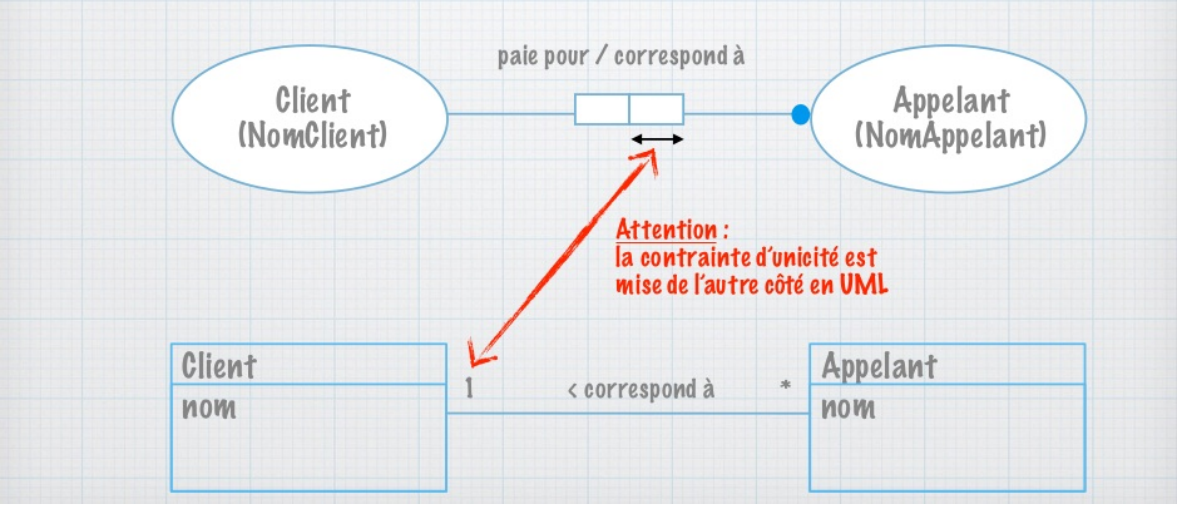
Suivant le processus *Conceptual Schema Design Procedure*

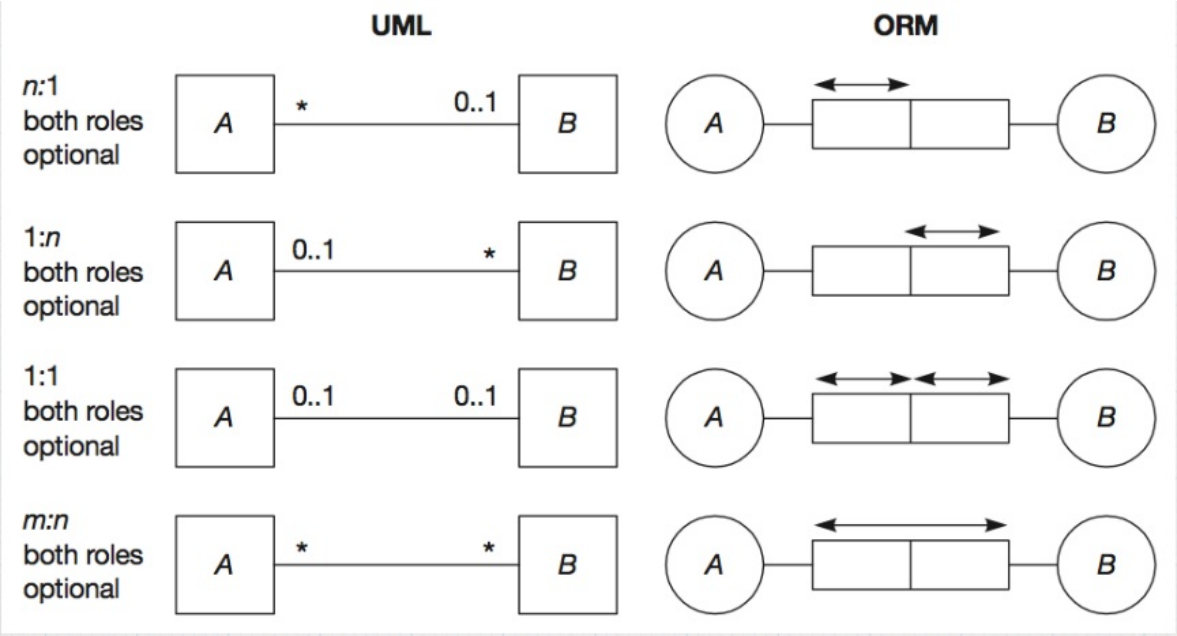
1. Trouver les **faits élémentaires** à partir des **exemples de données**.
   Les entités : Type('Client') - (Mode de référence('nom'))- Valeur('Bob')
   Les faits élémentaires expriment des relations entre les objets et ne peuvent être divisés.
   Ils peuvent être unaire (propriété de l'objet[...est...]), binaire (relation entre 2 objets[...a...]) ou n-aire

2. Dessiner les **types de faits** et ajouter une **population** au diagramme

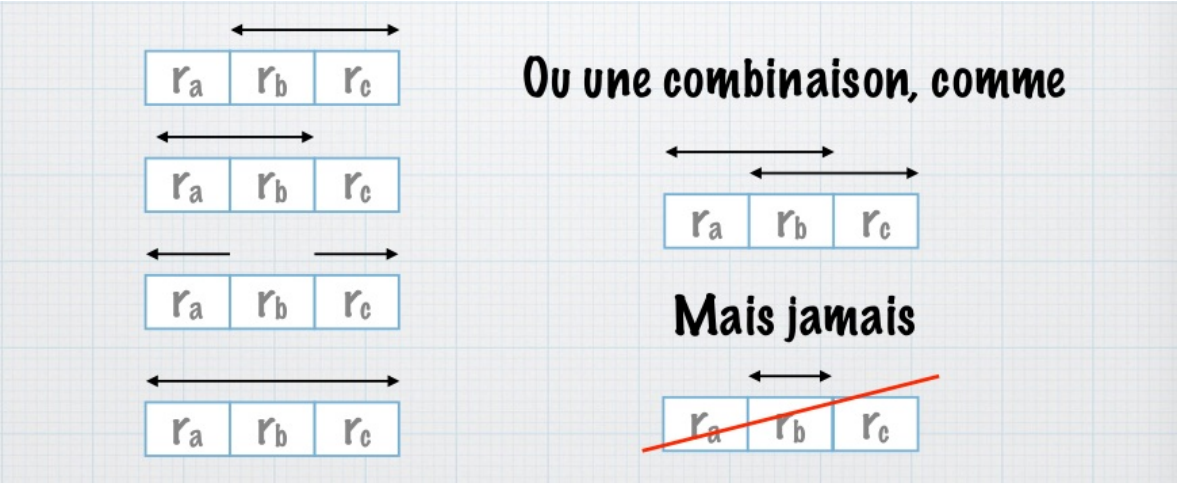


3. Combiner des types d'entités et noter des dérivations arithmétiques
4. Ajouter des contraintes d'unicité et vérifier les arités des types de faits
   Les **contraintes d'unicité** définissent quelles colonnes peuvent avoir des doublons. Elles sont représentées par une flèche au dessus d'un rôle.

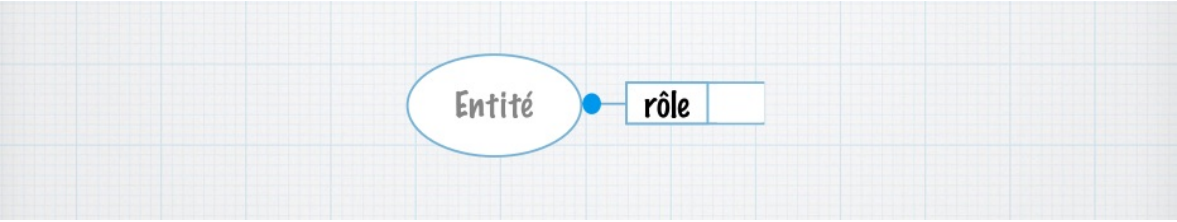




Une relation ternaire ne peut avoir de contrainte d'unicité simple. Si tel est le cas on peut la split en 2 relation binaires.



L'arité des contraintes doit toujours être de n-1. Sinon on peut les subdiviser en plusieurs autres.
!!Attention !! : l'emplacement des contriantes est l'inverse du diagramme UML



Ce rôle est **obligatoire** (arité de l'uml = 1 voir plus)

Dans une Base de donnée, on utilise souvent des ID's pour les données (pas mis dans l'ORM sauf si ID existe déjà dans l'univers du discours) (ex: Noma, ISBN)

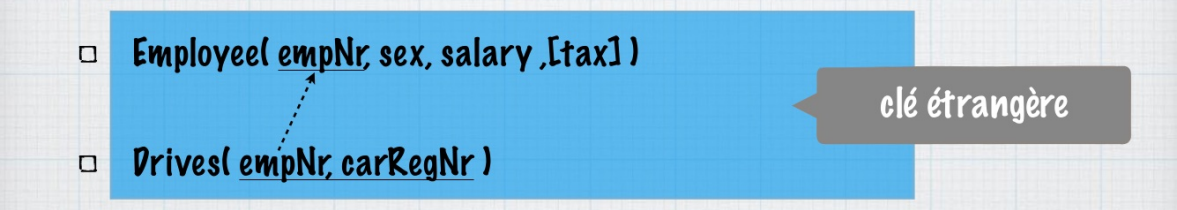
Les **clés candidates** sont, dans le shémas de la BDD, on souligne le nom de colonnes (ou de paires de colonnes) qui permettent d'identifier de manière unique chaque élément.

Il peut exister une série de clés candidates, il est donc important de définir une clé candidate qui deviendra une **clé primaire**. (elle est soulignée deux fois)

Si une clé est composée de plusieurs colonnes qui ne sont pas listées consécutivement, des *flèches doivent être ajoutés pour indiquer quels colonnes font parties d'une même clé*

Une clé candidate ne permets **pas de valeur nulle**
Les colonnes optionnelles sont indiqués entre […], (ex. :Employee(empNr, empName, address, sex, [phone]))

Règles d'intégrité du modèle relationnel



1. Règle d'intégrité des entités

La règle de l'intégrité référentielle : chaque valeur non-null d'une clé étrangère doit correspondre à

la valeur d'une clé primaire

2. Règle d'intégrité référentielle

chaque valeur non-nulle d'une clé étrangère doit correspondre à la valeur d'une clé primaire.

Le Sql

Le language DDL (Data Description Language)

Le sous language DDL permets de créer des structures de données et les modifier.

Creation d'une table

```
create table CLIENT (   NCLI char(10) not null primary,
                    NOM char(32) not null  default 'John',
                    ADRESSE char(60) unique,
);
```

```
create table COMMANDE ( NCOM char(10) not null primary key,
                        NCLI char(10) not null reference CLIENT,
                        DATECOM date not  null,
);
```

Les types

valeur	sql
Valeur null	NULL
numérique exact	INTEGER
numérique approché	REAL
chaîne de bits	BLOB
chaîne de caractères	TEXT

temps : TEXT, REAL, ITEGER
boolean : INTEGER (0 ou 1)

Suppression d'une table

```
DROP TABLE (IF EXISTS) (database-name.)table-name
```

Bien vérifier que la table n'est plus référencée par une clé étrangère

Modification d'une table

Renomer la table

```
ALTER TABLE (database-name.)table-name RENAME TO new-table-name
```

Ajouter une colonne

```
ALTER TABLE (database-name.)table-name ADD (COLUMN) column_def
```

En Sql on ne peut :
renommer /supprimer une colonne
ajouter ou supprimer des contraintes d'une table

Le language DML (Data Manipulation Language)

Toute requête *select* renvoie un résultat sous la forme d'une table

Extraction

```
select QUOI_AFFICHER
from TABLE
where CONDITION
```

select distinct QUOI_AFFICHER -> évite les doublons

Conditions

Sur valeurs null
null ne peut être comparé que avec : is/is not (ex:NUM is null, NUM is not null)

Condition in / between
- in /not in (ex: CAT in ('C1','C2','C3','C4'))
- between (ex: between 00 and 200)

Conditions avec masques
avec l'indicatif like
- _ (ex: Cat like 'B_') -> masque un caractère à cet endroit précisément
- % (ex: CAT like '%truc%') -> masque une chaîne de caracrères

caractère d'échapement
- !

Conditions logiques
Cond **and** Cond

alias
Une colonne peut avoir un alias : a+b as Truc remplace la colone a+b par truc même et le résultat de cette colonne est a+b

fonctions sql
- Arithmétique : abs, random, round
- Chaînes de caractère : lenght, replace, trim, upper, lower, substr
- Temps : date, time, datetime
- Agrègative : avg, count, max, min, sum, total

count produit 0 mais les autres fonctions agrégatives produisent null
attention ou l'on utilise le *distinct*

```
select count(NCLI), CAT
from CLIENT
group by CAT
```

Les sous-requêtes

Les sous requêtes utilisent le retour sous forme de table d'un select pour imbriquer et ainsi avoir une sélection plus précise

Les jointures

COMMANDE.NCLI = CLIENT.NCLI -> condition de jointure

Les conditions de jointure peuvent s'additionner avec des conditions de sélection

Les sous-requêtes peuvent avoir la même finalité que les jointures

Insertion

```
insert into DETAIL
values('a','b','c')


insert into DETAIL (NOM,PRENOM,AGE)
values('a','b','c')

insert into CLIENT
select ... +condition de table
```

Delete

```
delete from TABLE
where COND
```

Update

```
update TABLE
set INFO-A-AJOUTER='BOB'
where COND
```

Android (implémentation voir cours

Utilisation du **MVC** (Model-view-controler) qui consiste en la séparation de la représentation de l'info par rapport à la présentation à l'utilisateur. Ce qui rends les applications flexibles et extensibles.
Modèle : ce que l'app fait
vue : présentation de l'info
controler : couche de code liant les vues et les modèles (évennement, choix de la vue à afficher

La persistance des données est assurée par JDBC (java Database Connectivity)

Processus de développement (orienté objet)

Le processus dépend de plusieurs facteurs.
Dans le language orienté objet, c'est l'UML qui est généralement utilisé pour représenter des objet et des classes.
Pourquoi L'UML ? car c'est une notation graphique standard

Processus de développement

Analyse (quoi faire ?)

Quel est le problème et quels sont les besoins ?
Il s'agit d'une analyse des besoins (fonctionnels et non-fonctionnels)
ainsi qu'une analyse orientée objet

Conception (comment faire ?)

Mise en place d'une solution conceptuelle, via diagramme de classes, schéma de base de données...

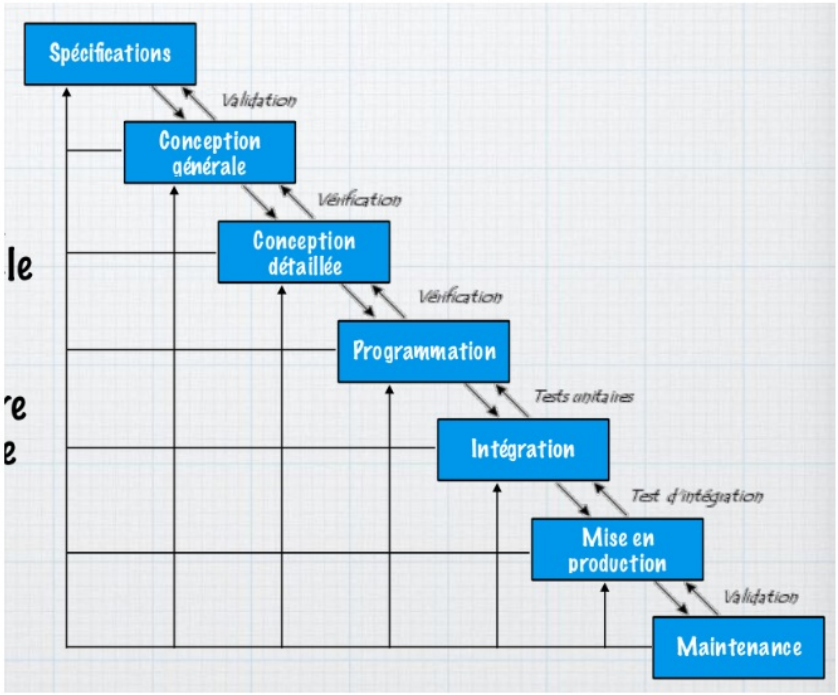
Implémentation

Mise en oeuvre, en incluant les détails de bas niveau dans un language orienté objet en particulier

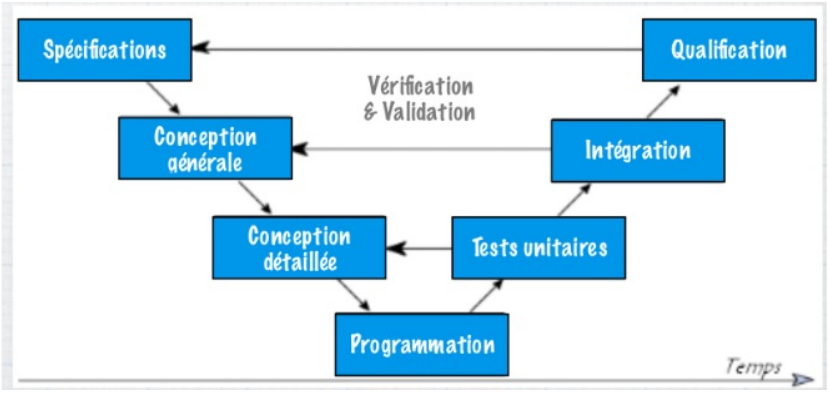
Tests et documentation

Cycles de développement

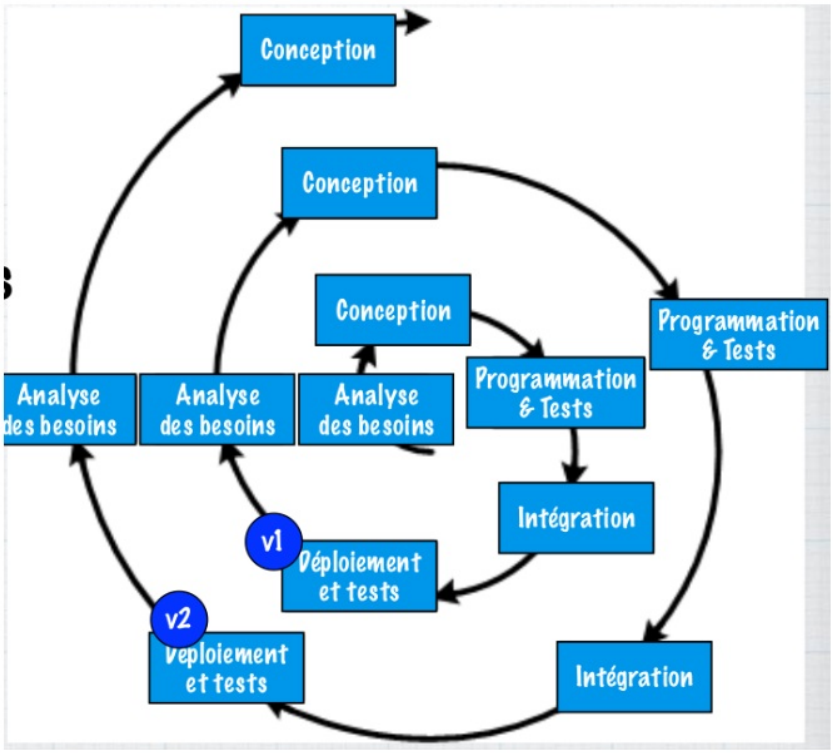
Modèle en cascade avec retour



Modèle en v



La spirale



Le cycle semi-iteratif

- Inception : description de l'objectf et de l'ampleur du projet
- Elaboration : analyse, architecture, planing
- Fabrication : la construction itérative et incrémental du projet
- Transition : tout ce qui va faciliter la transmission au client : beta-testing, user trainig, performance tuning

Dans la phase de fabrication, on morcele le logiciel et on produit chaque morceau via des itérations.

Iterations

- Analyse (récits d'utilisateurs, carte CRC)
- Conception (Diagrammes de classes UML, Diagrammes de séquence UML)
- Implémentations
- Tests

Analyse des besoins

Une des causes répandues de l'echec de projets est l'identification des besoins.

Rates of Software Engineering Failure

Requirements analysis	Very High
Specification	Low
Design	Low
Implémentation	Low
Installation	High

Operation	Enormous
Maintenance	Very High

Types de besoins

Besoins fonctionnels

» comportement d'une application

Besoins non-fonctionnels

» besoin de fiabilité (fréquence de bugs)
» besoin de performance (temps de réponse)
» besoin d'implémentation (système sous linux)

Récits utilisateur

"user story"
typiquement utilisé avec les méthodes agiles et le développement itératif

```
En tant que ROLE
je veux ACTION
afin de OBJECTIF
```

sans language technique
chaque user-story a un nom, une estimation de durée et une description

Invest

» Independant
» Negotiable
» Valuable
» Estimable
» Small
» Testable

Plannification

poker planning
choisir les récits à implémenter, plannifier et allouer des têches à effectuer pour chaque itération.

Les cartes CRC

CRC = Classes, responsabilités et collaborations
objectif : déterminer **les classe**s les plus importantes de l'application et leurs **responsabilités** + **collaborations** potentielles

Une carte CRC se compose de : Classe, Description, Responsabilités, Collaborations

Conception orienté objet - UML

Diagrammes de classes UML et **Diagrammes de séquence UML**
L'UML résulte d'une standardisation de l'OMG(Object Management Group)

Diagramme de classes

Représente la **structure statique** d'un système.

Classes : nom, attributs, opérations

Attributs : visibilité nom:Type [multiplicité] = ValeurInitiale
Opérations : visibilité nom(nomArg:Type=Valeur,...) : Type
Les opérations inutiles ne sont pas obligatoires (ex: CRUD)

visibilité :
Public (+) : toutes les classes
Protégé (#) : par les sous classes
* Privé (-) : par la classe seule

(...not finished)