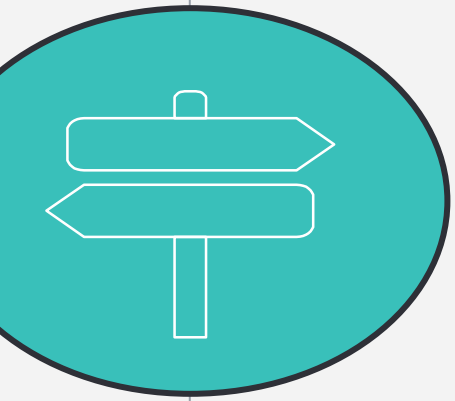
INTRODUCTION

Analyse 3

2021-2022



3 ans d'analyse...

...et ce n'est que le début !

PLAN DU COURS SUR 3 ANS

Q2: Introduction sur l'analyse
Diagramme d'activité
Diagramme de classes

Q3: Diagramme de classes et MCD
Diagramme de cas d'utilisation et MCT
Spécification des cas d'utilisation et diagramme d'activité
Tests fonctionnels élémentaires
UC Realization - Diagramme de séquence
Diagramme de classes technique
Design Patterns

Q4: MVC, MVP, MVVM

Q5: UML, génie logiciel, gestion d'équipe et méthode agile

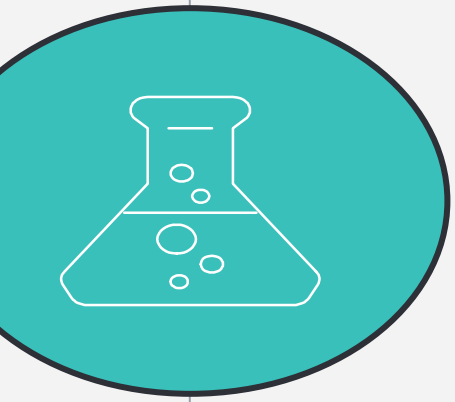
PLANNING DE CETTE ANNÉE

	B1	B2	B3
Théorie	**4h/s**		
Laboratoires	**2h/s**	**4h/s**	
Laboratoires bis*			**2h/s (ATLG4)**

* Uniquement pour les étudiants de gestion

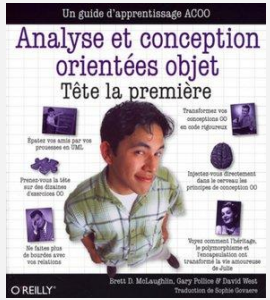
Méthode pédagogique

	Théorie	Exposé théorique - démonstration
	Laboratoires	B1 : Divers petits exercices non évalués B2: Projet en groupe évalué



Cas malheureux

Jeu de dés (inspiré du livre “Analyse et conception OO” Série Tête la première.)



Jeu de dés

On lance deux dés, si le total obtenu vaut 7, c'est gagné!

Version 0

le jeu de dés: 7
3 et 5
Perdu :(

le jeu de dés: 7
2 et 5
Gagné :)

Client

```java
public class Main {

  public static void main(String[] args) {

    System.out.println("le jeu de dés: 7");

      int de1 = (int) (Math.random() * 5) + 1;
      int de2 = (int) (Math.random() * 5) + 1;

    System.out.println( de1 + " et " + de2);

    if ((de1 + de2) == 7) {
        System.out.println("Gagné :)");
    } else {
        System.out.println("Perdu :(");
    }

  }
}
```

Collègue

Jeu de dés

Première règle:

Assurez-vous que le logiciel fait ce que le client veut qu'il fasse.

- priorité au client
- identifier les vrais besoins

Jeu de dés

On peut lancer plusieurs fois les dés. Le nom du joueur et son score sont affichés

Version 01

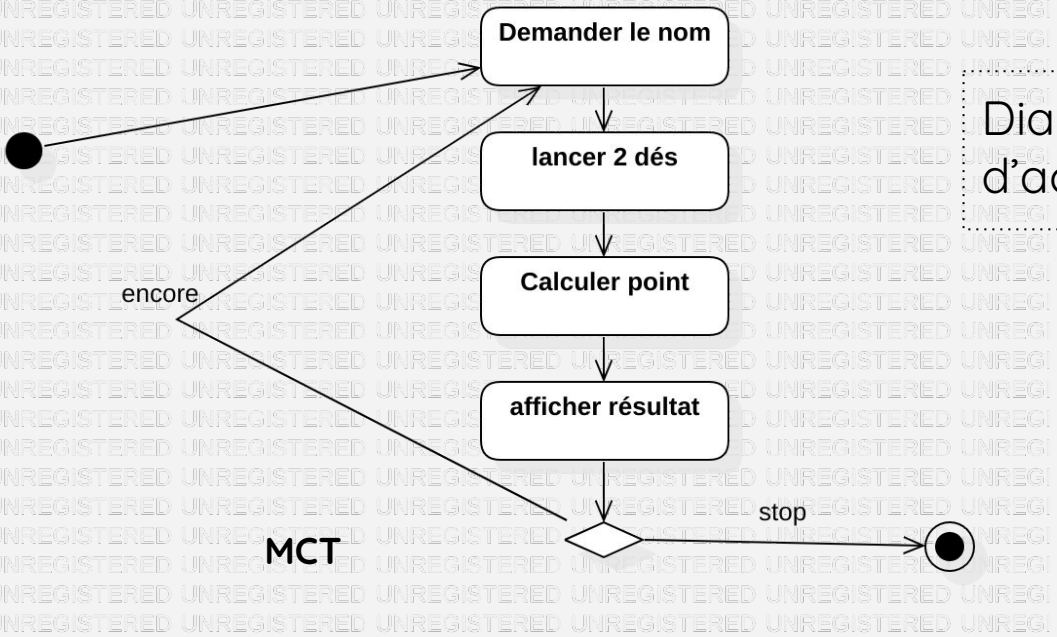


Diagramme d'activité

Jeu de dés

On peut lancer plusieurs fois les dés. Le nom du joueur et son score sont affichés

Version 01

```
le jeu de dés: 7
nom:
Geneviève
3 et 4
Gagné :)
1 point(s)
encore un lancer
(true/false)
true
...

2 point(s)
encore un lancer
(true/false)
false
```

Client

```java
public class Main {

  public static void main(String[] args) {

      System.out.println("le jeu de dés: 7");

      Scanner sc = new Scanner(System.in);
      System.out.println("nom: ");
      String nom = sc.nextLine();
      boolean encore = true;
      int point = 0;

      while (encore) {
         int de1 = (int) (Math.random() * 5) + 1;
         int de2 = (int) (Math.random() * 5) + 1;

         System.out.println( de1 + " et " + de2);

         if ((de1 + de2) == 7) {
            System.out.println("Gagné :)");
         } else {
            System.out.println("Perdu :(");
         }

         point = (de1 + de2 == 7)? point + 1:point;
         System.out.println(point + " point(s)");

         System.out.println("encore un lancer (true/false)");
         encore = sc.nextBoolean() ;

    }
  }
```

Collègue

Jeu de dés

Deuxième règle:

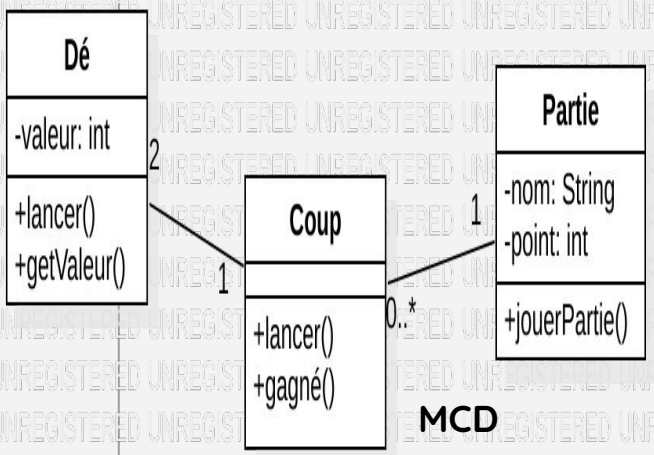
Appliquez les principes OO de base pour ajouter de la souplesse.

- encapsulation
- héritage
- polymorphisme

Jeu de dés

Version OO

Le collègue demande un programme OO !



MCD

```java
public class Dé {
   private int valeur;
   public Dé() {
      this.valeur = 0;
   }
   public void lancer() {
      valeur = (int) (Math.random() * 5) + 1;
   }
   public Integer getValeur() {
      return valeur;
   }
}
```

```java
public class Coup {
   private Dé dé1;
   private Dé dé2;
   public Coup(){
      dé1 = new Dé();
      dé2 = new Dé();
   }
   public void lancer(){
      dé1.lancer();
      dé2.lancer();
   }
   public boolean gagné(){
      return dé1.getValeur() + dé2.getValeur() == 7;
   }
   public String toString(){
      return ("dés = " + dé1.getValeur() + " , " + dé2.getValeur());
   }
}
```

```java
public class Partie {
   private String nom;
   private boolean encore;
   private int point;
   private Coup coup;
   private Scanner sc;
   public Partie() {
      sc = new Scanner(System.in);
      System.out.println("nom : ");
      nom = sc.nextLine();
      encore = true;
      point = 0;
      coup = new Coup();
   }
   public void jouerPartie() {
      while (encore) {
         coup.lancer();
         System.out.println(coup);
         if (coup.gagné()) {
            point++;
            System.out.println(nom + " a gagné :)");
         } else {
            System.out.println(nom + " a perdu :(");
         }
         System.out.println("point(s) = "+point);
         System.out.println("encore un lancer (true/false)");
         encore = sc.nextBoolean();
      }
   }
}
```

Client

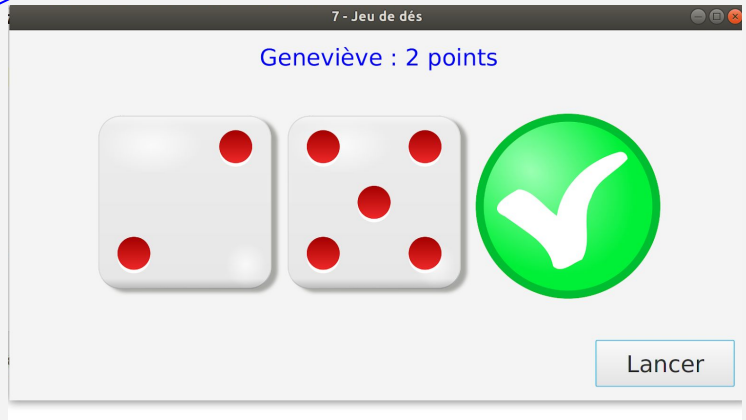
```
nom :
Geneviève
dés = 5 , 3
Geneviève a perdu :(
point(s) = 0
encore un lancer (true/false)
true
....
```

Collègue

Jeu de dés

Besoin d'une interface graphique

Version FX




```java
public class Dé {
  private int valeur;
  public Dé() {
    this.valeur = 0;
  }
  public void lancer() {
    valeur = (int) (Math.random() * 5) + 1;
  }
  public Integer getValeur() {
    return valeur;
  }
}
```

```java
public class Partie {
  private String nom;
  private int point;
  private Coup coup;
  public Partie() {...
  }
  public void lancer(){...
  }
  ....
```

```java
public class Coup {
  private Dé dé1;
  private Dé dé2;
  public Coup(){
    ...
  }
  public void lancer(){
    ...
  }
  public boolean gagné(){
    ...
  }
  public String toString(){
    ...
  }
}
```

```java
public class Main extends Application {

  private Image images[] ;
  private ImageView ivs[];
  private Text t;
  private HBox hbox;
  private HBox hBoxImg;
  private Label label1;
  private TextField textField;
  private HBox hBoxTxt;
  private String family = "Helvetica";
  private double size = 50;
  private Button button;
  private BorderPane root;

  private Partie partie;
  private boolean debut = true;
  @Override
  public void start(Stage primaryStage)
throws Exception{

    partie = new Partie();

    creerImages();
    creerText();
    creerHBoxImg();
    creerTextField();
    creerHBoxTxt();
    creerBouton();
    creerPane();
....
```

😐

Client

😠

Collègue

Jeu de dés

Troisième règle:

Oeuvrez pour une conception facile à maintenir et à réutiliser.

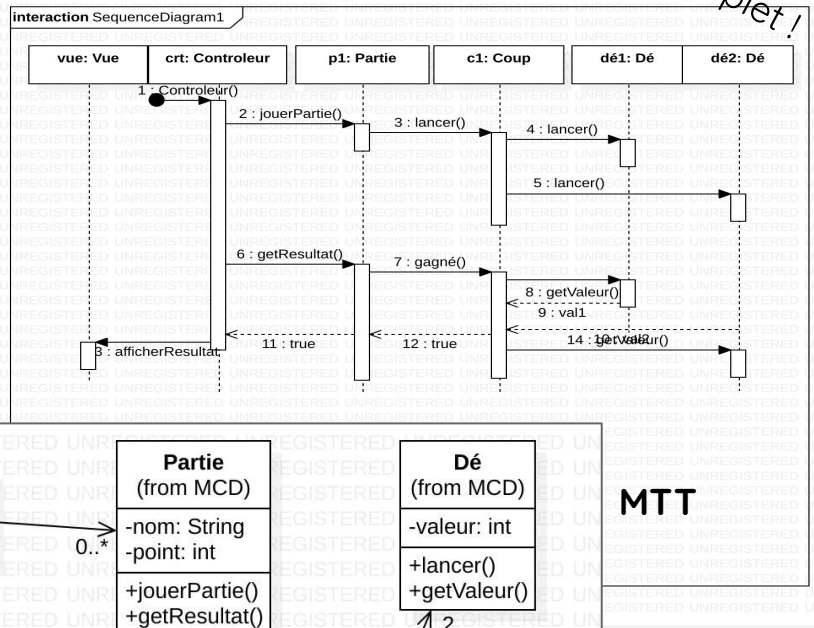
- principes GRASP (délégation...)
- design pattern

Jeu de dés

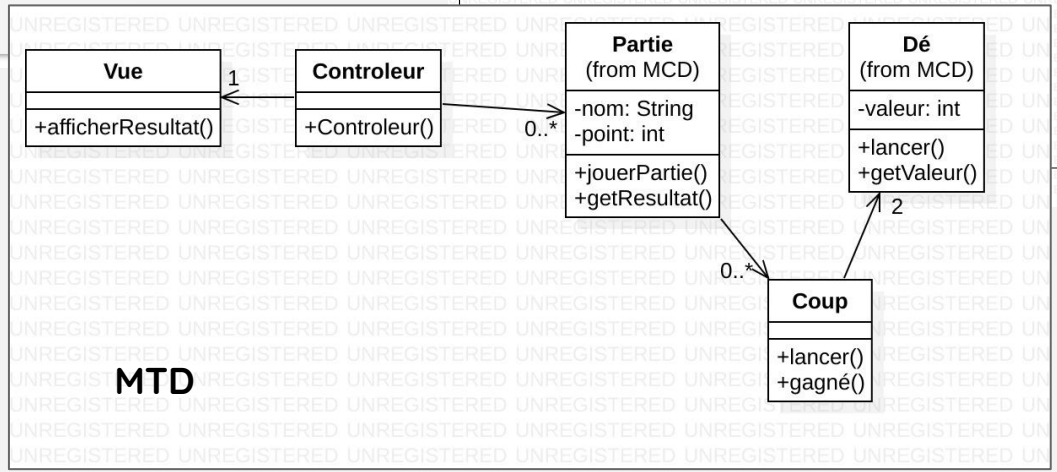
Une architecture MVP

Version MVP

incomplet !



MTT



MTD

😐

Client

😐

Collègue

Jeu de dés

A retenir:

1. Assurez-vous que le logiciel fait ce que le **client** veut qu'il fasse.
2. Appliquez les **principes OO** de base pour ajouter de la souplesse.
3. Oeuvrez pour une conception **facile à maintenir et à réutiliser.**

RÔLE DE L'ANALYSTE

Traduire les demandes des utilisateurs en modèles que les développeurs utiliseront pour construire une solution informatique.

Importance de l'utilisateur en tant que demandeur

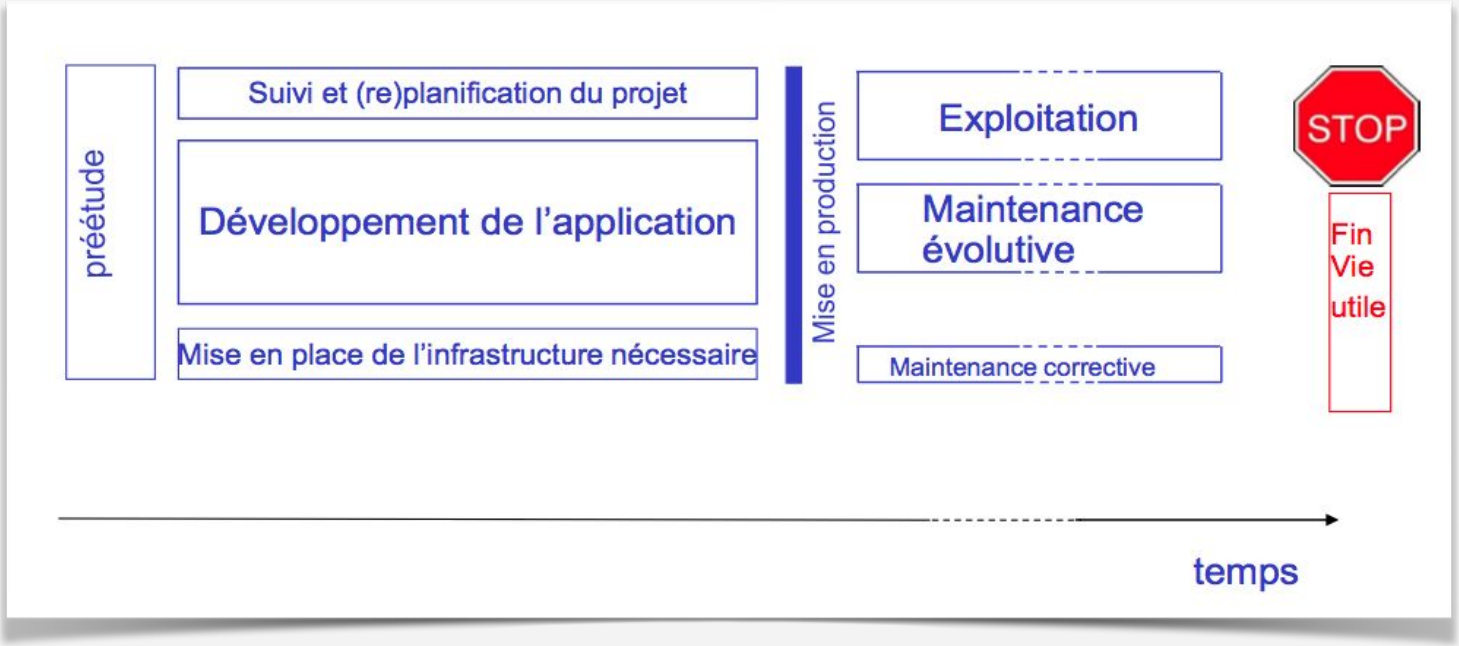
PROJET, SI ET SIA

Projet tout ce qui touche à la réalisation d'un (d'une partie d'un) système informatique par une équipe de développement. Il y a un début et une fin, avec des délivrables.

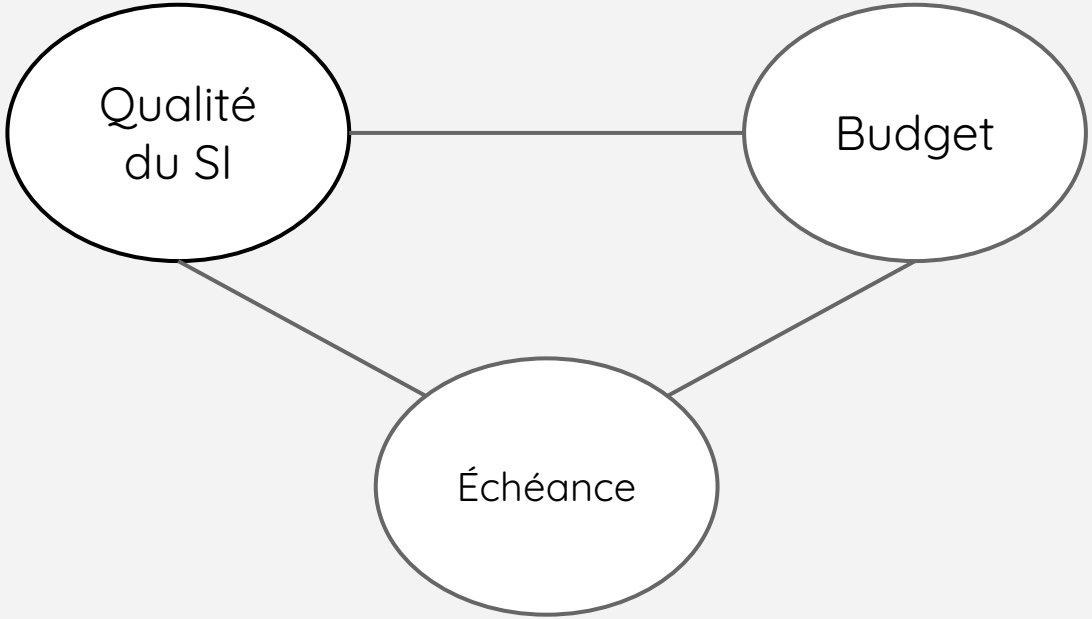
Système d'information (SI) : toutes les informations *utiles* collectées sur le projet. Les besoins de l'utilisateur final dans le logiciel mais aussi l'organisation de l'entreprise, l'esprit de l'entreprise, les règles internes de l'entreprise, ...

Système d'information automatisé (SIA) : la partie automatisée du système d'information (SI), tout ce qui tourne sur un processeur.

CYCLE DE VIE D'UN PROJET



QUALITÉ D'UN PROJET



QUALITÉ DU SI

«Ensemble des caractéristiques d'une entité qui lui confèrent l'aptitude à satisfaire des besoins exprimés ou implicites », ISO 8402

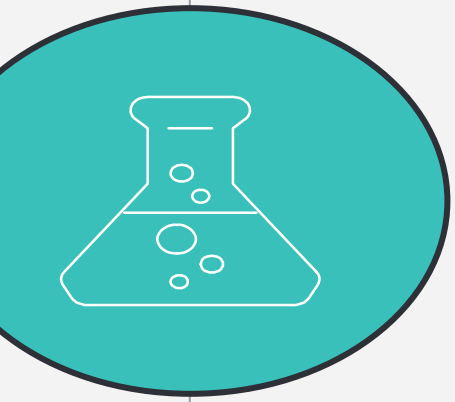
QUALITÉ DU SI

Critères d'évaluation (par l'utilisateur)

- Pertinence (demandes de l'utilisateur)
- Apport de bénéfices
- Fiabilité
- Performances
- Convivialité

Critères techniques permettant d'atteindre cette qualité:

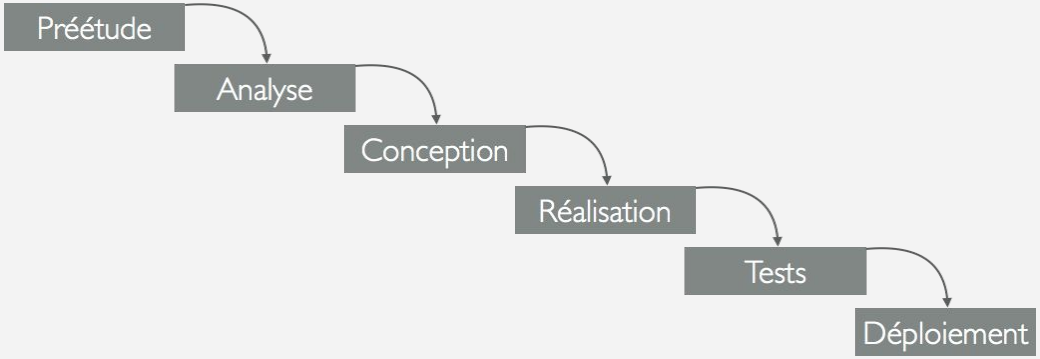
- Évolutivité
- Réutilisabilité
- Portabilité



Méthode

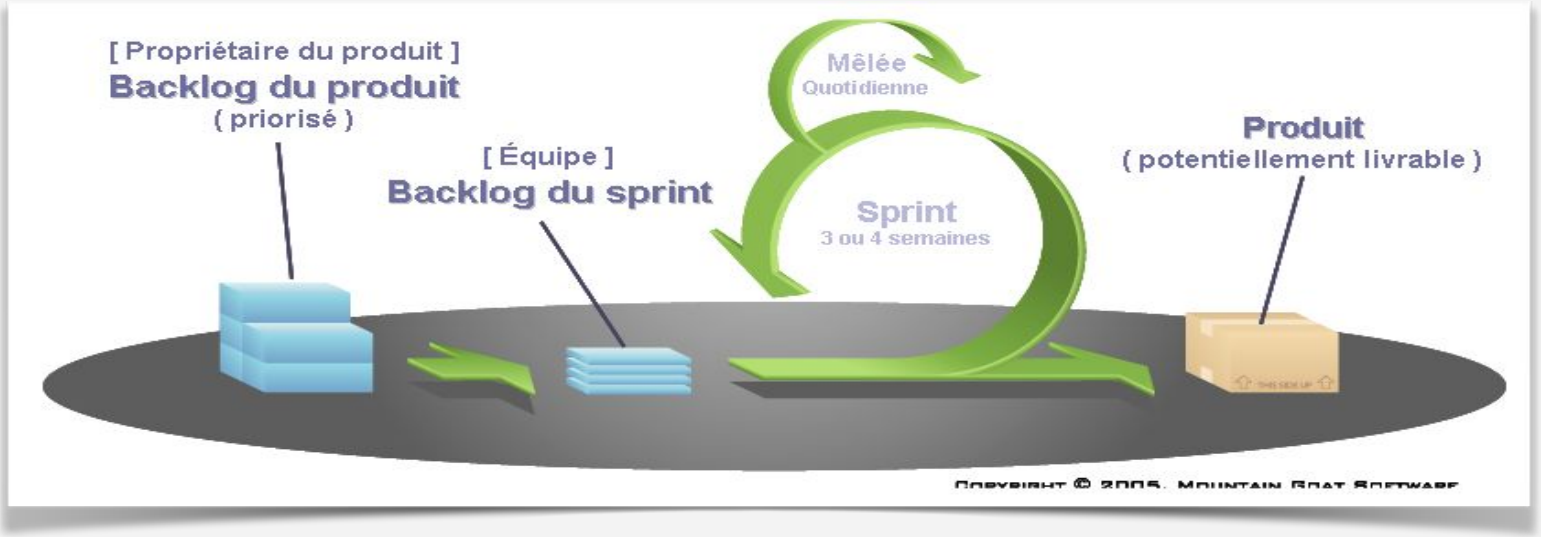
Quelle méthode va-t-on utiliser pendant les laboratoires ?

Cascade


Préétude
Analyse
Conception
Réalisation
Tests
Déploiement

TYPES DE MÉTHODOLOGIE

Méthodologie agile SCRUM



MÉTHODE DES LABORATOIRES

Découpe des besoins en 3 niveaux

Conceptuel
Trouver les données métiers et les traitements

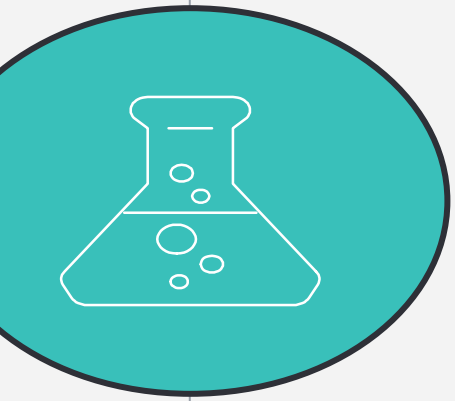
Fonctionnel
Trouver les détails fonctionnels

Technique
Trouver les détails techniques
(interfaces, volumes de données, langage, …)

MÉTHODE DES LABORATOIRES

Découpe des besoins en 3 niveaux

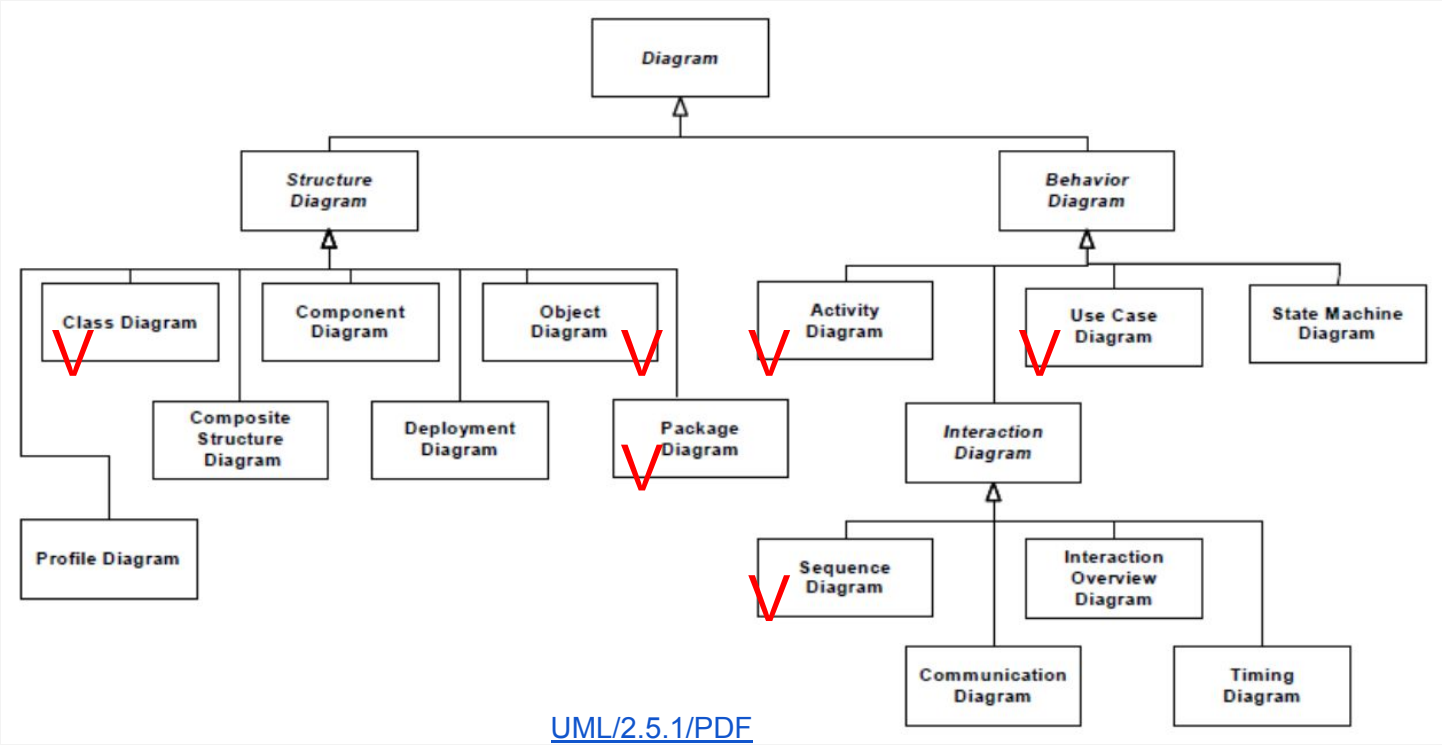
Conceptuel Trouver les données métiers et les traitements	**Modèle conceptuel des données** **MCD** **Modèle conceptuel des traitements** **MCT**
Fonctionnel Trouver les détails fonctionnels	**UC Spécifications** **PTFE**
Technique Trouver les détails techniques	**Modèle technique des données** **MTD** **Modèle technique des traitement** **MTT**



Langage

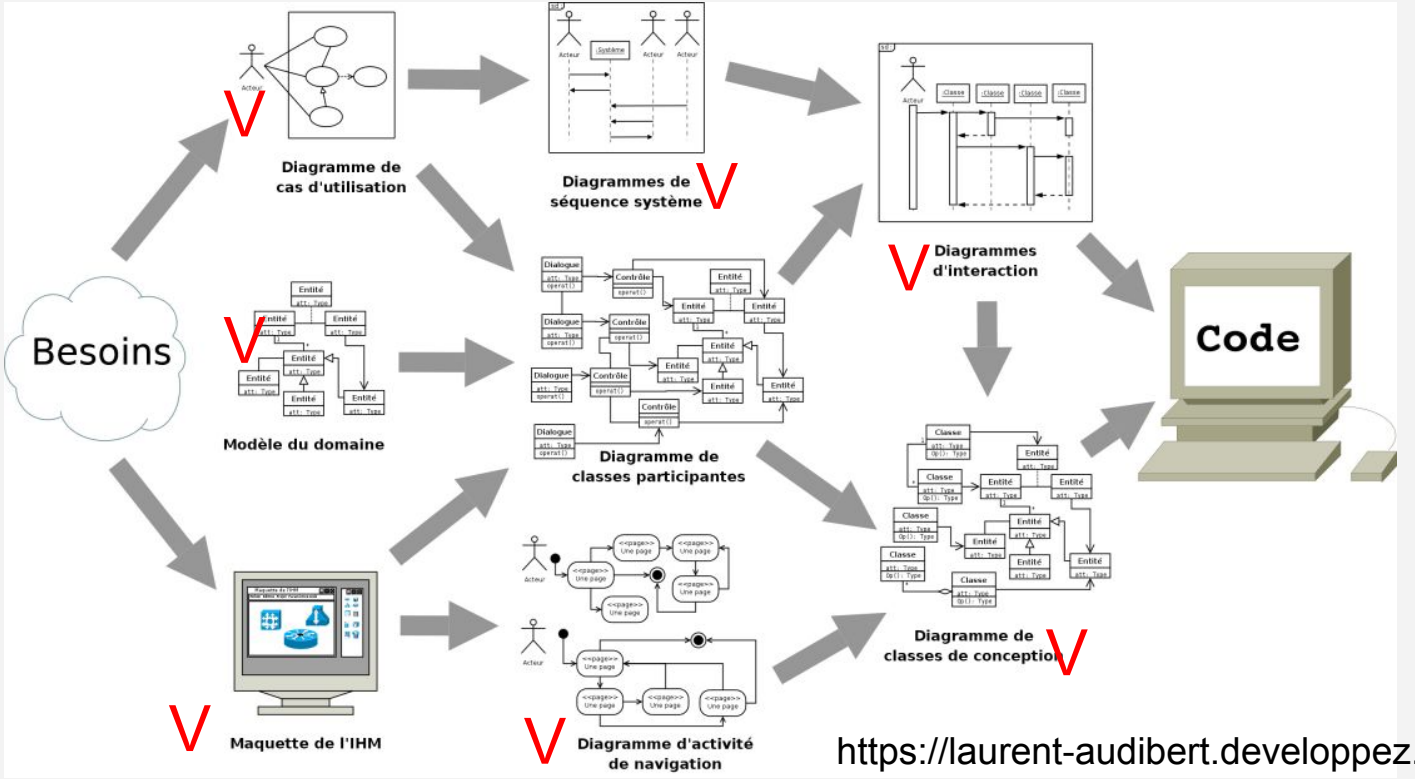
Quel langage va-t-on utiliser?

UML 2.x : 14 DIAGRAMMES, DONT 6 VUS DANS CE COURS

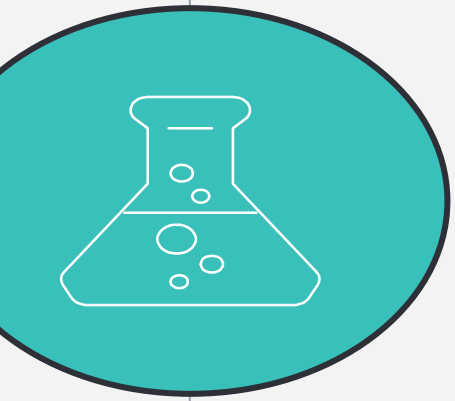


UML/2.5.1/PDF

UML 2.x : 14 DIAGRAMMES, DONT 6 VUS DANS CE COURS



https://laurent-audibert.developpez.com



Outils

Quels outils va-t-on utiliser?

OUTILS

MockFlow

PLAN

MCD
Modèle conceptuel des données
Diagramme de classes
(rappels) Documentation

MCT
Modèle conceptuel des traitements
Diagramme de Use Cases (UC)
Documentation

Conceptuel

UC Specification
Documentation de UC
Interface utilisateur
Diagramme d'activité (rappel)

PTFE
Plan de tests fonctionnels élémentaires
Documentation

Fonctionnel

MTD-MTT

Design Pattern

UC Realization
Diagramme de séquence
Diagramme de classes techniques

Technique

Méthodes