Genie Logiciel

Def:
est un ensemble des methode, technique et des outils, didiés a la conception au, developement et à maintenance

Objectif de Genie Logiciel

1 - Adéquation au bessoins du client
2 - Respect du delai de réalisation prévus
3 - Maximation des performance et fiabilité
4 - Folliciation de Maintenance et évolution ultérieurs.

(1)

logiciel composé de:

- document (gestion projet)
- specisification (list fonctionL)
- Conception (plan)
- Code source (Implementer)

Principe du Genie Logiciel

1 - Separation responsabilité: décomposition travoille
2 - Réutilisation exploiter Composant logiciel pré-fabriquer
3 - Encapsulation maximal: donner niveau accessibilité public qu'a nombre maximal de nombre
4 - couplage faible une grande souplesse de mise a jour (peut modifier)
5 - cohésion forte ce principe recommande de rassembler ensemble elemt ayant même rôle

(2)

Qualité attendue logiciel

(Bessoin non fonctionnel)

- Utilité (bessoins utilisateur et les fonction)
- Utilisabilité: (Eficasité)
- fiabilité (resultat sont attendu)
- Compabilité (logiciel interagir avec d'autre logiciel)
- Performance: (logicil doit Satisfaire aux contraite temps execution)

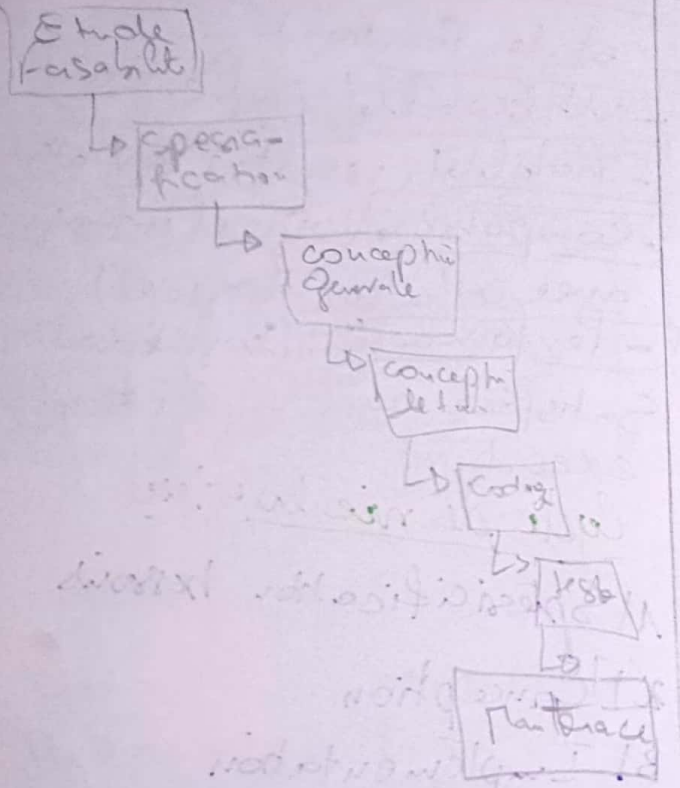
Cycle de vie logiciel

1| Spécisification bessoins
2| Conception
3| Implémentation
4| Verification et Validation
5| Maintenance

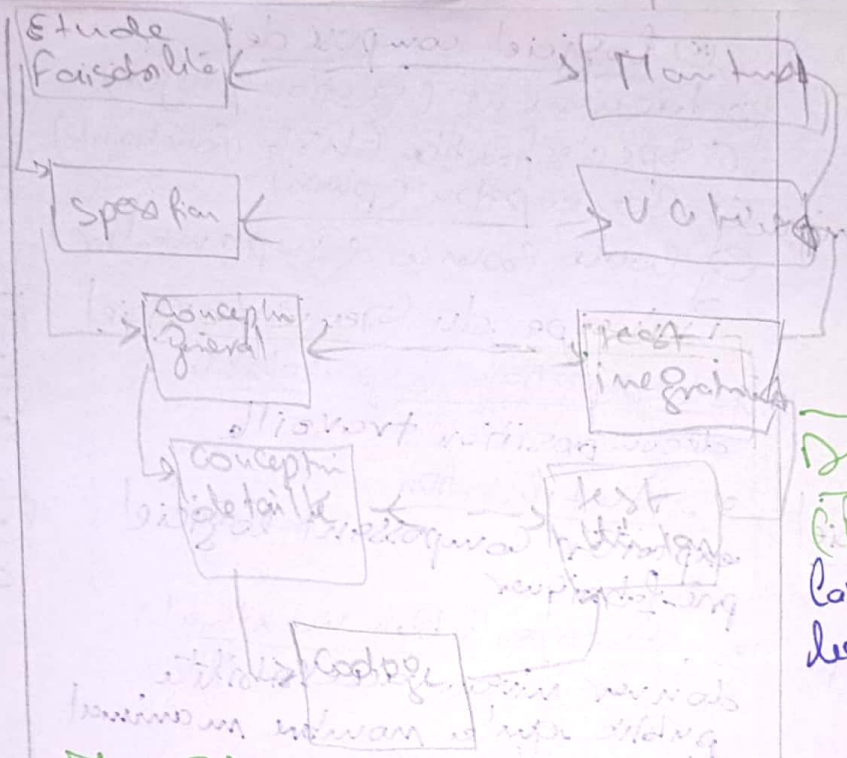
(3)

Modèle Cycle de vie

1| Cycle de vie Cascade



2| Cycle de vie En V



Modélisation

1| Modèle:
est une abstraction d'objet a la réalité

2| UML: (Unifire Model language)
est un langage modélisation Oriente Objet

typediogramme

- Statique (Architecture)
  - classe
  - objet
  - Cas Utilisation
  - Composaint
- dynamique
  - Composement
  - sequence
  - collaborat
  - Activité

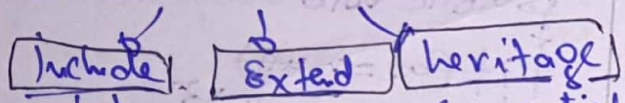
Diogramme Cas Utilisation (Fonctionnel)

Cas Utilisation exprime les Actions qui deroule Diogramme Cas

- Acteur: personne, chose qui interagit avec sys
- Cas Utilisation: Action qui fait

① Relation Entre Cas Utilisation et Acteur : Association, heritage

② Relation Entre Cas Utilisation



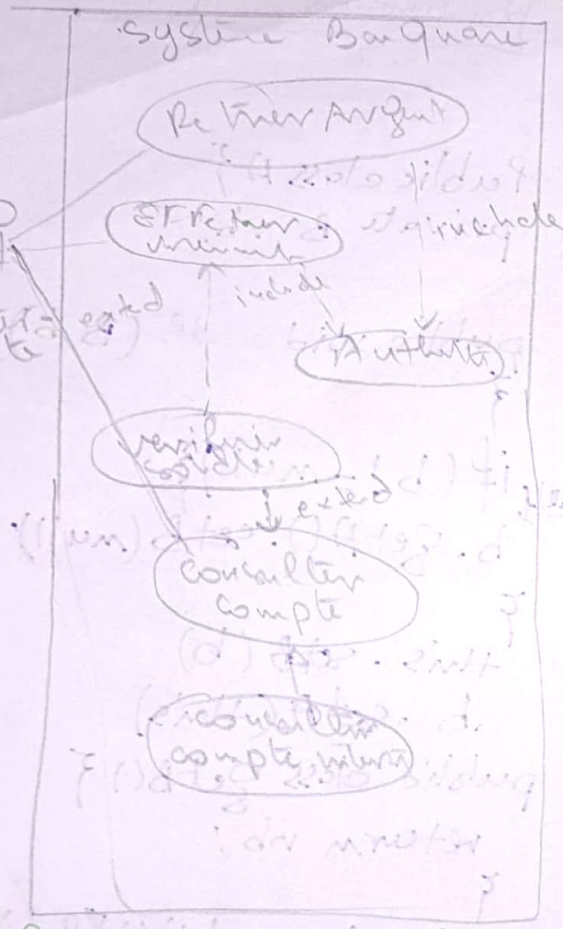
A inclut dans B : lorsque A sollicite B sollicitation forcement

A etend B : lorsque A peut etre appelé en cour execution

Cas particulier
Mère
Fille

: exemple :

Dans un système bancaire, Internaute Retire Argent doit Authentifier, il effectuer virement et consulter son Compte, il peut verifier code de plus il peut consulter depuis internet



Classe :

est un concept abstrait representé des elements variés

(généralement pluriel ISI)

Nom

Attribut

Methode

Relation Entre les classe (Verbe)

① Heritage

② Association

decrit les connexion entre leur instance

③ Classe intermidiaire

Association ne pouvant posséde une propriété, cela introduire "classe intermediaire"

④ Aggregation et composition

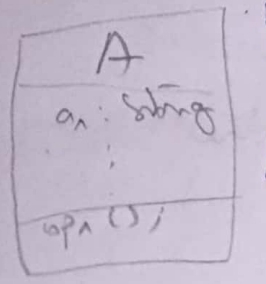
Aggregation : A ◇—— B, A ◇——

Composition : ◆——B, A ◆——

X Implémentation

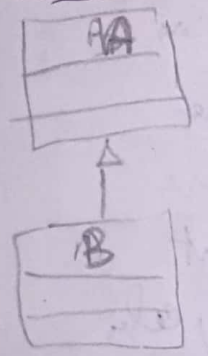
etape 3 cycle de vie

① classe :




```
public class A {
    String a1;
    public void op1() {
    }
}
```

② Héritage :



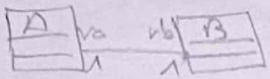

```
public class A {
}
public class B extends A
{
}
```

③ Association

A - bidirectionnel (↔)

B - unidirectionnel (→)

bidirectionnelle 1-1




```
Public class A {
 private B rb; // crée nouveau type class B
public void addB (B b)
{ //Methode pour ajouter B
if (b != null) { // si b connecte déjà autre A
 b.getA().setB(null);
 // set autre doit connecter
{
 this.setB(b)
  b.setA(this)
public class getB() {
  return rb;
 {
public class setA() (B b) {
    this.rb = b;
  }
}
```



```
Public class B {
 private A ra;
public void addA (A a)
{ if (a!= null)
if (a != null) {
if (a.getB1 = null)
 a.getB().setA() (null)
}
public class getA() {
    return ra;
  }
public class setB(A a) {
   this.ra = a;
}
```

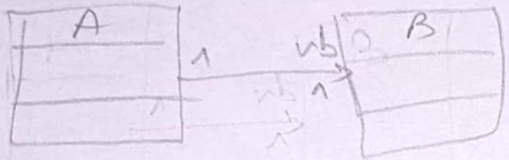
bidirectionnel 1-N

```
Public class A{
private ArrayList<B> nb;
public A(){
nb = new ArrayList<B>();}
Public ArrayList<B> getArray(){
    return (nb);
    }
Public remove (B b){
    nb.remove(B);}
Public void addB (B b){
if(!nb.Contains(b)){
if(b.getA()!=null){
    b.getA().remove(b);
b.setA(this);
nb.add(b);
}
}
}
```



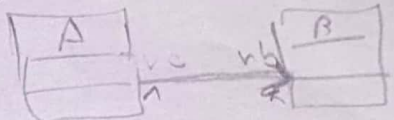
```
Public class B(){ //créer instance B (déjà créé dans list)
private A ra;
public class B(){}
}
```

Unidirectionnelle: 1-1.




```
public class A{
  private B nb;
  public void addB(B b){
    if(b!=null){
      this.nb=b;
    }
}}
Public class B{
  // Class B ne connait existance A
}
```


Unidirectionnelle 1-N




```
Public class A{
private ArrayList<B> nb;
public A(){
nb = new ArrayList<>();}
public void addB(B b){
  if(!nb.contains(b)){
    nb.add(b);
  }
}
Public class B{
  // B connait pas existance A
}
```

Agrégation et Composition

les deux implementer comme Association (1-N)