



REPUBLIQUE DE GUINEE

Travail-Justice-Solidarité

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

INSTITUT SUPERIEUR DE
TECHNOLOGIE DE MAMOU

DEPARTEMENT :GENIE INFORMATIQUE

Mr Ibrahima TOURE Chef de
Département Génie Informatique
Tel: 666 25 31 70/621 22 34 63
Email:astourep@gmail.com

Département Génie
Informatique

Programme de génie logiciel

Chapitre I: le logiciel

- ❖ Définition d'un logiciel
- ❖ Les participants et rôle de Génie logiciel
- ❖ Les Grandes parties du développement
- ❖ Production d'un logiciel
- ❖ Code source, exécutable
- ❖ Les participants
- ❖ Le cycle de développement

Chapitre II: les projets de développement

- ❖ Les différents types de projets
- ❖ Participer au projet de développement
- ❖ Grandes Phases du cycle de développement

Programme de génie logiciel

- **Chapitre III: Les grands Etapes du cycle de développement**
 - ❖ Analyse des données
 - ❖ Le chiffrage des données
 - ❖ La conception
 - ❖ Choix de l'architecture des données
 - ❖ Les Modules de conception
 - ❖ Exemples d'architecture
- **Chapitre IV :le développement**
 - ❖ Environnement de développement
 - ❖ Le cycle de développement
 - ❖ Les Tests
 - ❖ L'agilité
 - ❖ Synthèse des Informations

Programme de génie logiciel

- **Chapitre V:le Scrum**
 - ❖ Agilité
 - ❖ Équipe de développement
 - ❖ Des itérations
 - ❖ Grands principes de Scrum
 - ❖ Le Back-log(Analyse)
 - ❖ Chiffrer le Back-log
 - ❖ Les besoins non fonctionnels
 - ❖ Les Sprint
 - ❖ Les Tâches

Programme de génie logiciel

Chapitre VI: Cycle de vie d'un logiciel

- ❖ Avancement et itération du projet
- ❖ Rôle de Scrum
- ❖ Anomalies et Bug
- ❖ Testeur
- ❖ Environnement de test

Chapitre VII: les cycles d'un logiciel

- ❖ Les modèles
- ❖ Cycle en spirale
- ❖ Principes du modèle
- ❖ Avantage
- ❖ Inconvénient
- ❖ Modèle e Y

Programme de génie logiciel

- ❖ Shama du modèle
- ❖ Principe du modèle
- ❑ **Chapitre VIII:** les projets
- ❖ Exercices

Cours de Génie Logiciel

□ **Objectif du Cours** est de comprendre une vision claire du cycle de vie du logiciel, de faire apparaître les enjeux liés à chacune des phases du cycle de vie et de donner des méthodes, techniques et outils pour répondre aux enjeux.

Objectifs Général du cours de Génie Logiciel

- La Qualité de logiciel est un résultat direct du processus utilisé lors de la création.
- Comprendre et connaître l'utilité de ce qui suit :
 - ❖ Modèles de conception du logiciel;
 - ❖ Cycle de vie de logiciel et leur implémentations ;

Objectifs Général du cours de Génie Logiciel

- ❖ Expliquer les phases et étapes de production d'un logiciel.

LE GENIE LOGICIEL

- Définition du développement : est un ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi.

GENIE LOGICIEL

Les Participants et Rôle du Génie logiciel

- Utilisateur
- Développeur
- Propriétaire

Les grandes Parties du développement

Le développement d'un logiciel comprend trois grandes parties:

- La première partie :présentation de l'environnement de travail.
- Deuxième partie :Analyse et la conception.
- Troisième partie :le fonctionnement et l'interface de l'application.

Les grandes Parties du développement

- La première partie :présentation de l'environnement de travail.**

- Présentation de l'environnement de travail;
- Structure de l'environnement de travail (Organigramme);
- Expression du besoin
- Les spécifications

- Deuxième partie :Analyse et la conception.**

- Analyse**

- 1.Phase: Etude de l'existant
 - 2.Phase: Problématique
 - 3.Phase: Cahier des Charges
 - 4.Phase: Solutions proposées

Les grandes Parties du développement

- **Conception**
 - Le modèle conceptuel de communication
 - Les modèles conceptuel des données ou diagramme de classe
 - Les règles de gestion
 - Dictionnaire des données
 - Modèle logique des données
 - Modèle physique des données

Les grandes Parties du développement

- Les acteurs: une unité humaine ou matériel intervenant dans le système d'information.ils se divise en deux catégories ,externe et interne ,selon leur appartennances.
- **Les flux**
- **Flux interne** :toutes activités gérées au sein d'une entreprise est appelé flux interne.
- **Flux externe**: toutes activités gérées à l'extérieur de l'entreprise est appelé flux externe.

FLUX INTERNE

- Exemple: soit le flux interne du département Génie Informatique.

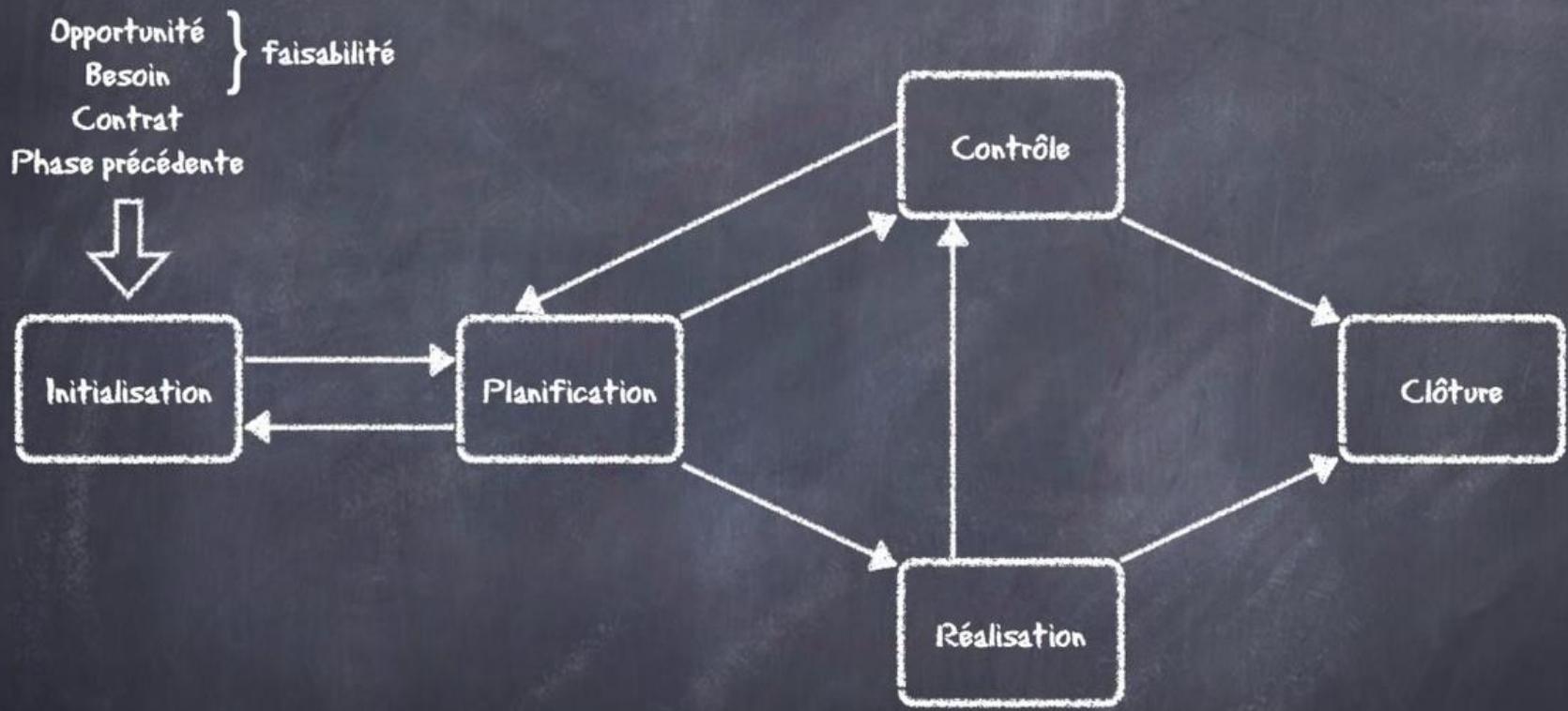
FLUX INTERNE

- Soit le flux externe du département Génie Informatique.

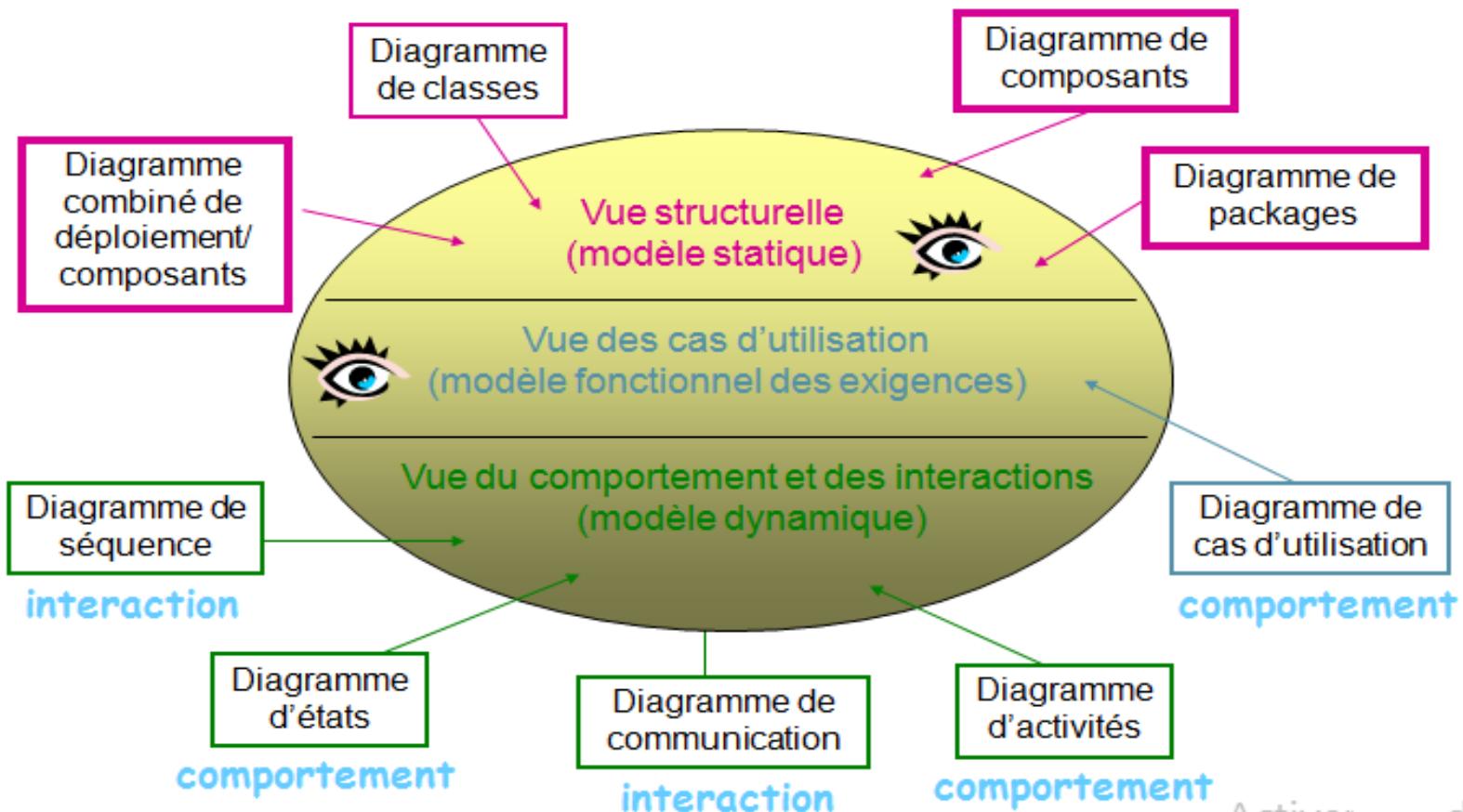
Les grandes Parties du développement

- Troisième partie : le fonctionnement et l'interface de l'application.**
- Sécurité des utilisateurs
- Sécurité des données
- Aide pour l'utilisateur
- Maintenance du logiciel

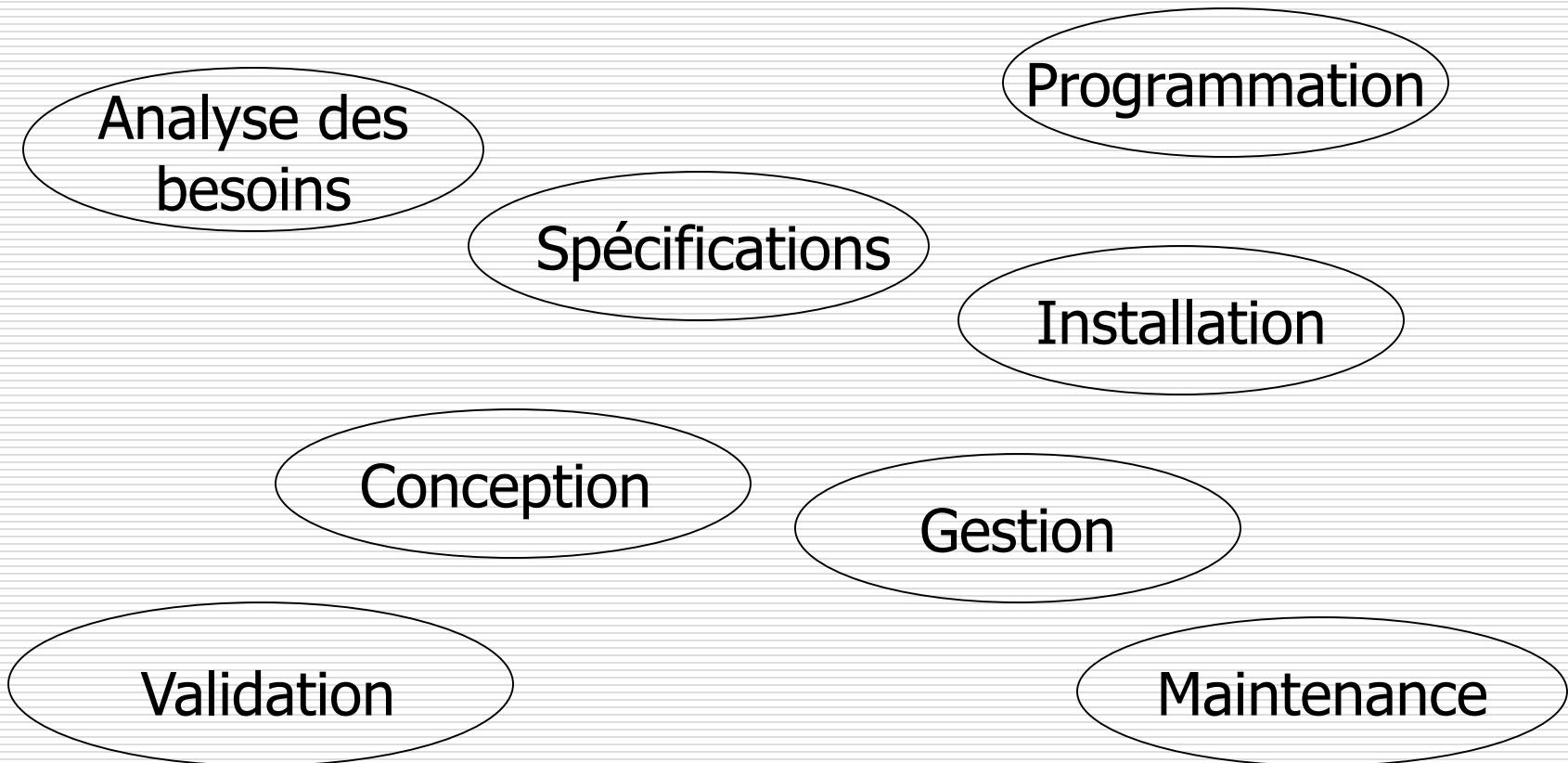
Projet Informatique



vues d'un système



Création de logiciel



Analyse des besoins

- Fonctionnalités attendues du système
- Point de vue métier, de l'utilisateur
- Aider à formuler le besoin
- Produit un document textuel, avec schémas, intelligible par le client
- Conduit à la définition du cahier des charges => les "spécifications"

Implémentation

- Traduction dans un langage de programmation
- Mise au point
- Documentation
- Production des données (saisies, conversions, récupérations)

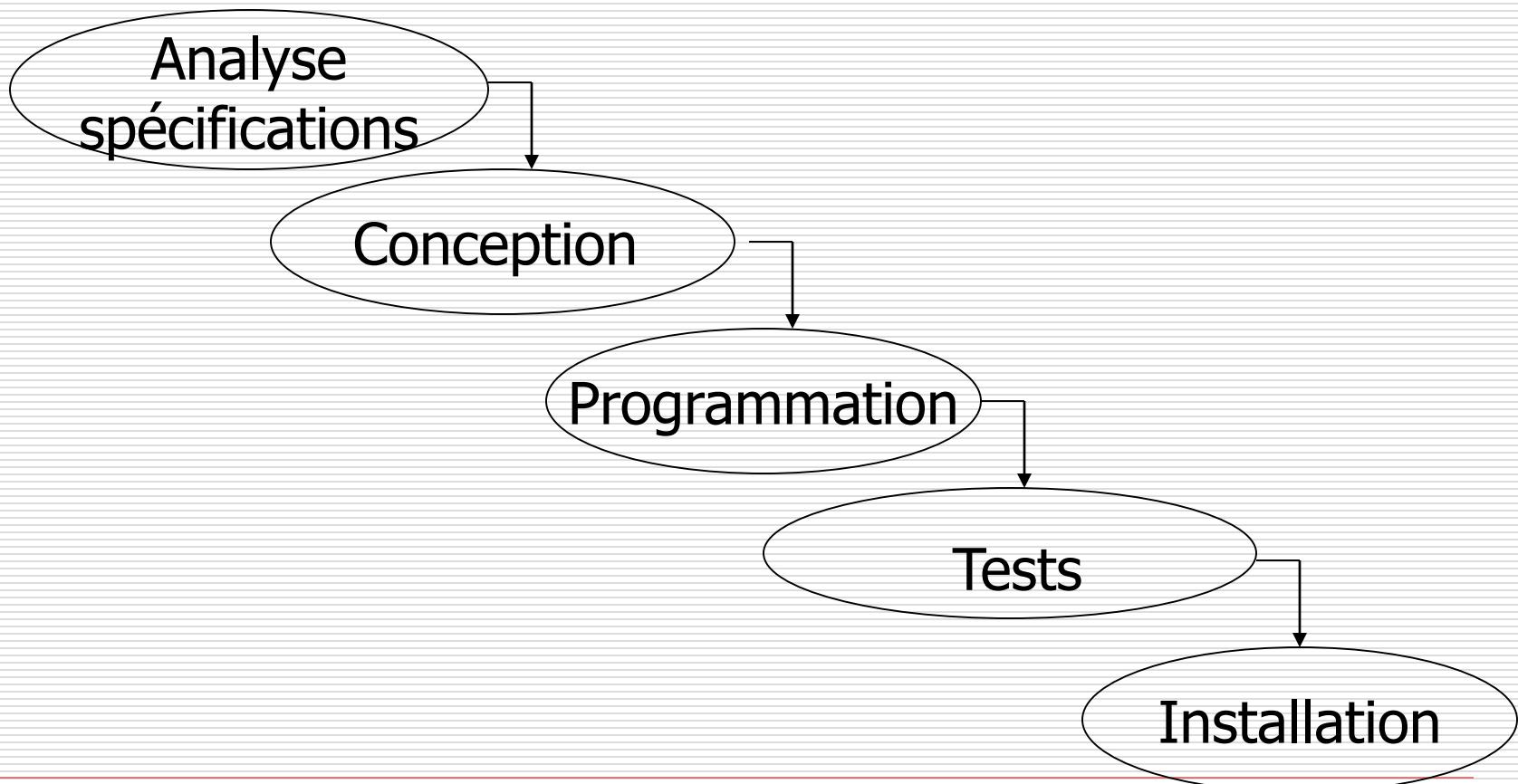
Documentation

- Les manuels d'utilisation et d'exploitation doivent être prêts au plus tard lors de l'installation.
- Selon la complexité du logiciel :
 - Guide d'installation
 - Guide de l'utilisateur
 - Guide de l'opérateur
 - Manuel de référence
 - Documentations des composants logiciels

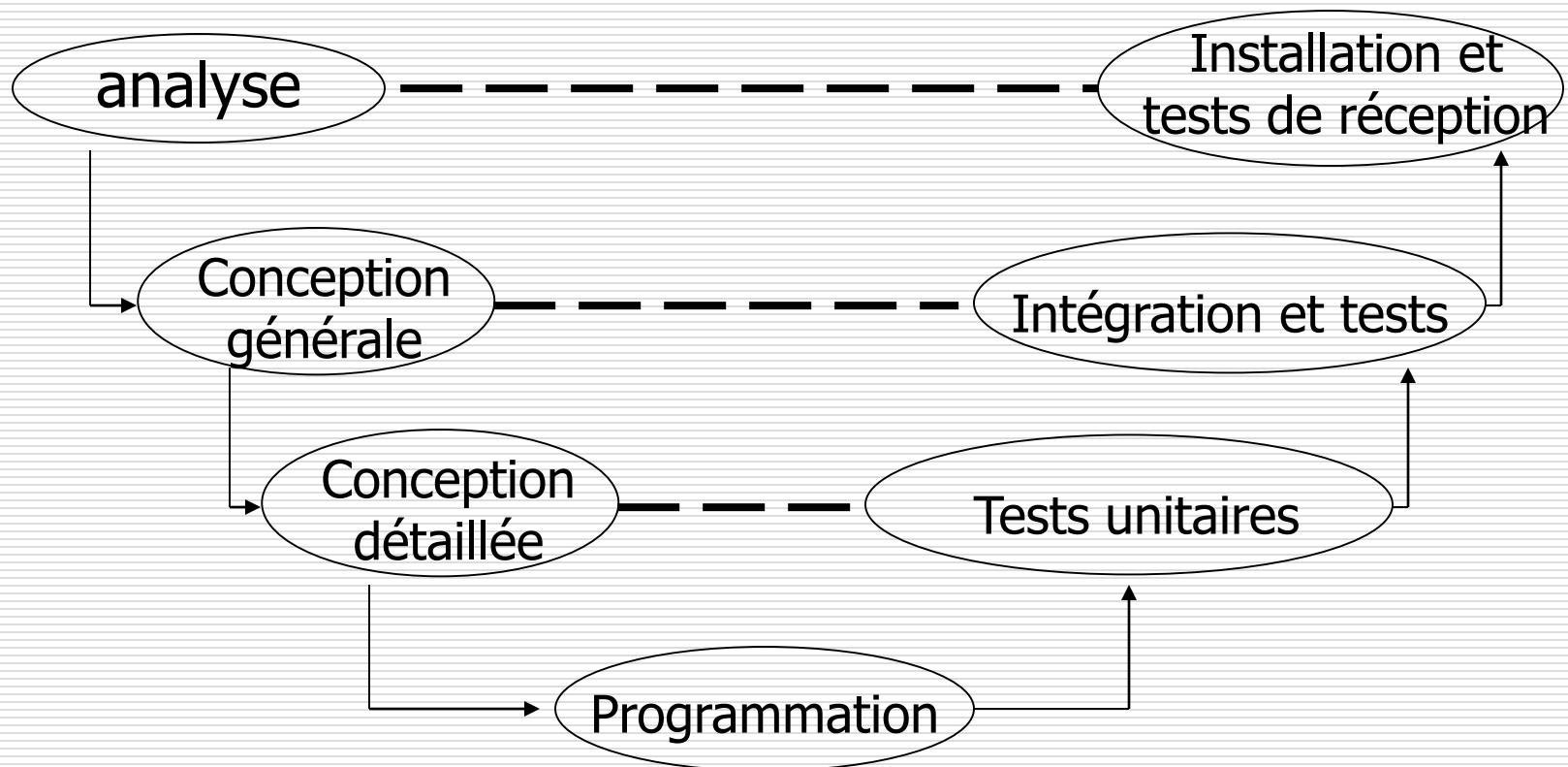
Maintenance et évolution

- Corrections des erreurs
- Prise en compte de nouveaux cas d'utilisation
- Ajout de nouvelles fonctionnalités
- Dans chaque cas, un "*cycle de développement*"

Organiser le développement ("cycle" de l'analyse à l'installation): **la cascade**

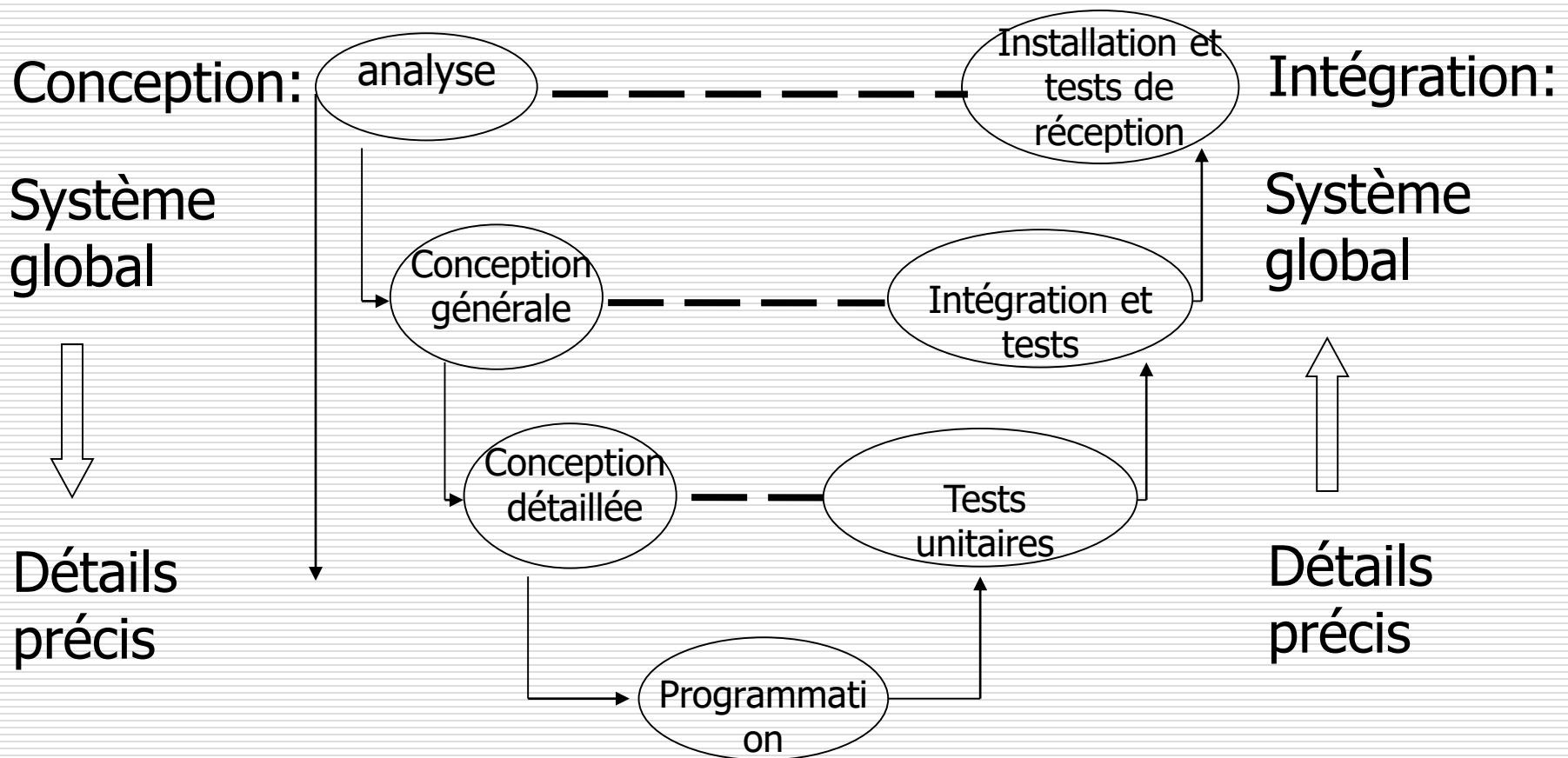


Le modèle en V

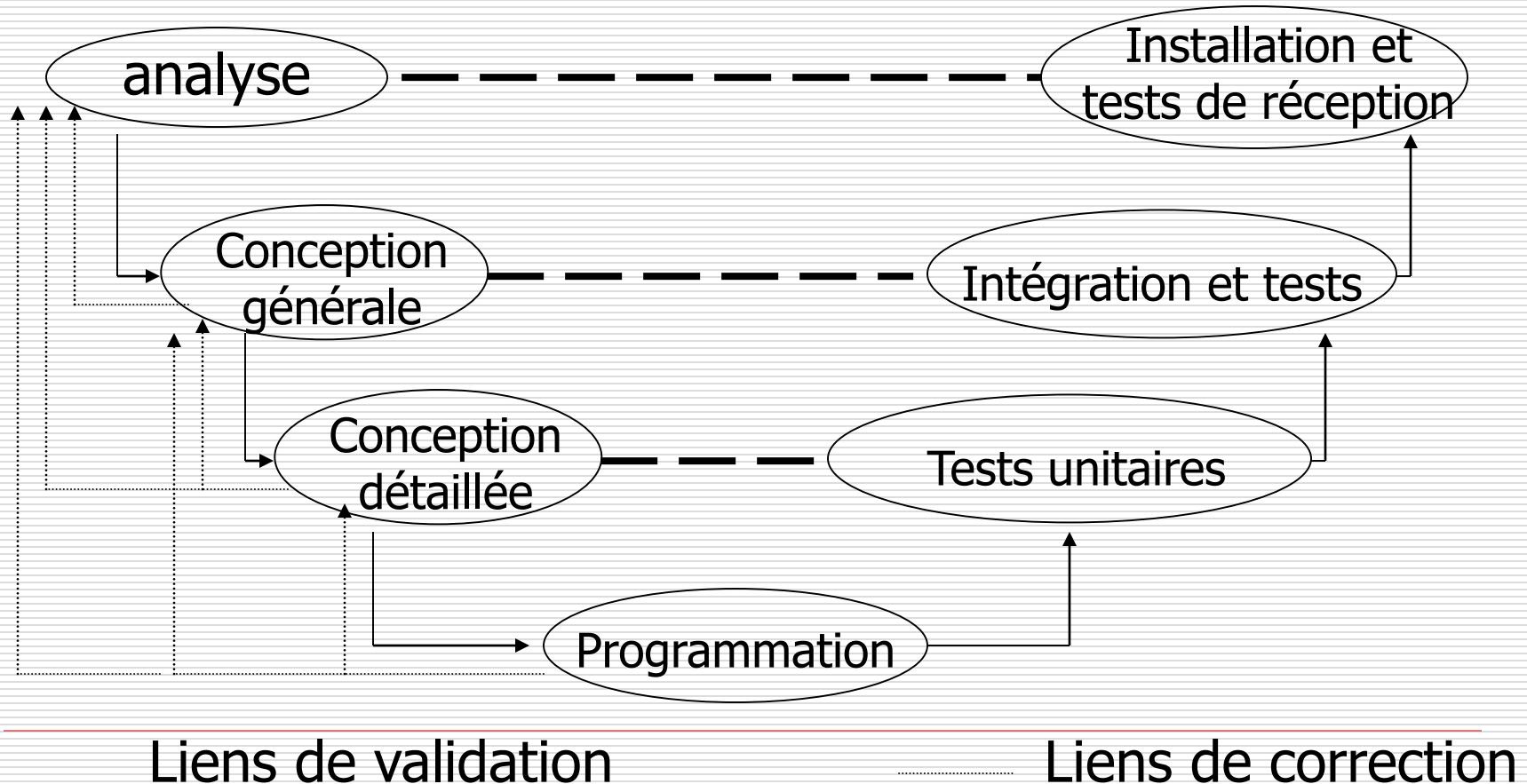


— — Liens de validation

Conception-programmation-intégration



Développement itératif : maquettes, prototypes...



Projet de développement

Exercices

- Une société d'assurance décide de proposer un cahier des charges pour la gestion du personnel de la société à un cabinet de développement pour la réalisation du projet.
- Chaque travailleur de cette société est identifié par un matricule (constituer de trois chiffres et une lettre) et caractérisé par deux attributs : identité du travailleur (nom et prénom) et sa fonction dans l'entreprise, les clients sont identifiés par leur identité, les fournisseurs de cette société sont identifiés que par leur numéro de téléphone.
- II-1-Identifier, les participants de ce projet.
- II-2-Etablir le modèle conceptuel de communication.
- II-3-Le modèle conceptuel données.
- II-4-Expliquer pourquoi sécuriser une application et quels sont les types de sécurité.

PRODUCTION DE LOGICIEL

Qu'est-ce qu'un logiciel ?

Application (Store), SiteWeb, Jeux,
Micro-Contrôleur, etc.

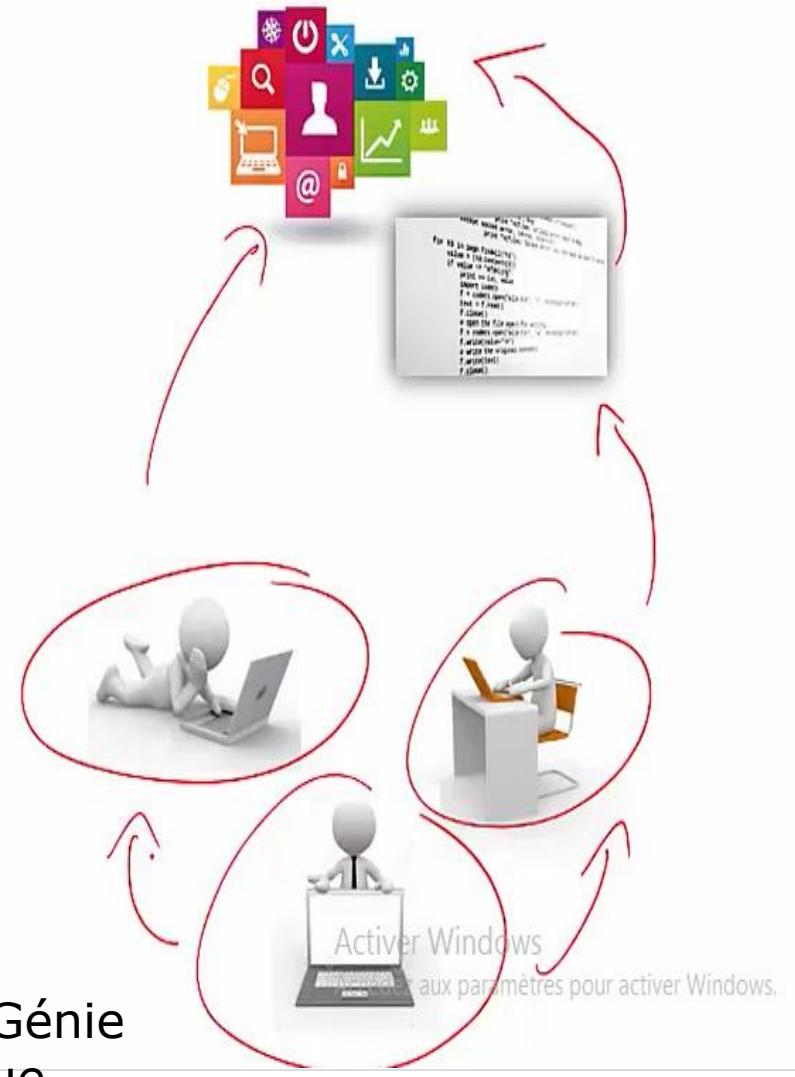
Produire => Code Source, Exécutable

Participants (rôles)

L'utilisateur

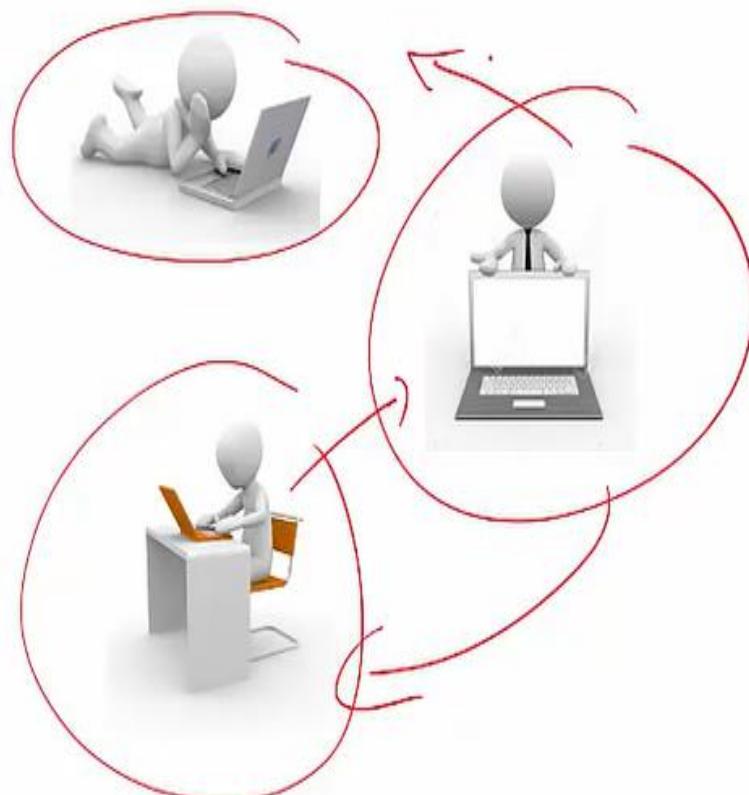
Le développeur

Le propriétaire



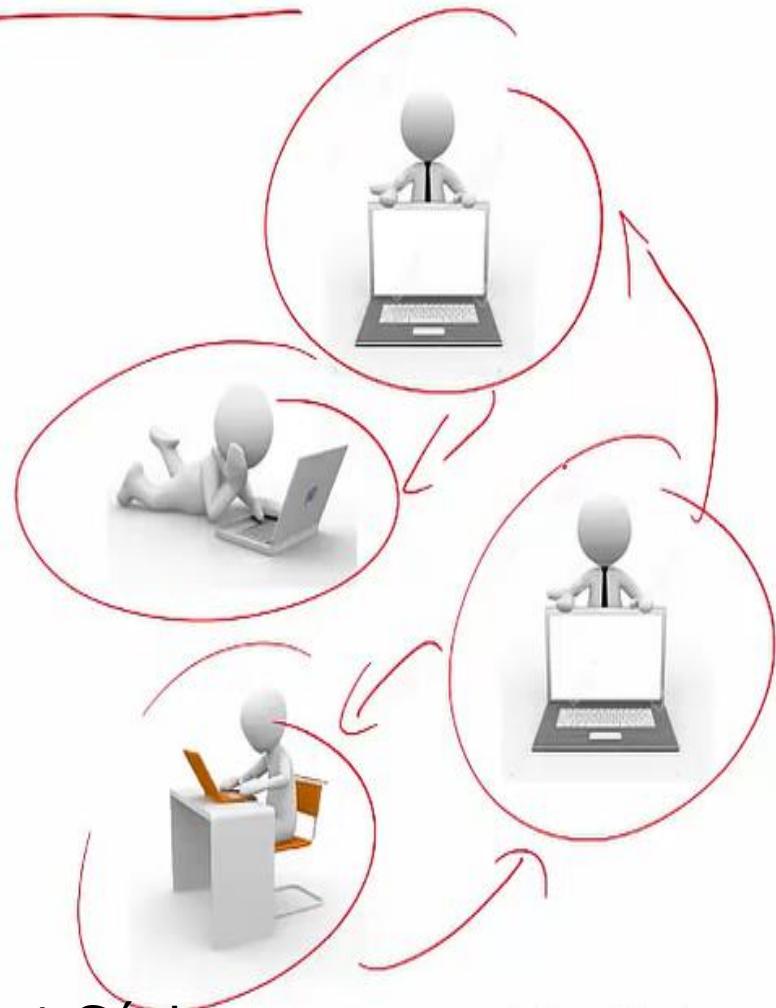
PROJET LOGICIEL - COMMENCEMENT

1. L'utilisateur a besoin d'un logiciel
2. Le propriétaire demande au développeur de réaliser le logiciel
3. Le développeur réalise le logiciel qui correspond au besoin de l'utilisateur



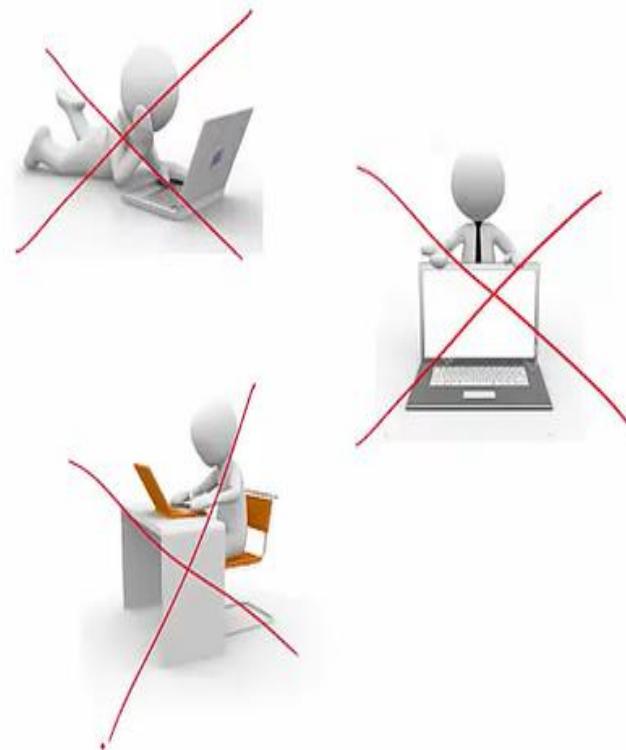
PROJET LOGICIEL – EXPLOITATION

4. Le propriétaire déploie le logiciel afin que celui-ci soit utilisé.
5. L'utilisateur utilise le logiciel, il rencontre des anomalies ou aimerait voir des évolutions
6. Le propriétaire demande au développeur de corriger les anomalies ou de développer les évolutions
7. Le développeur corrige les anomalies et réalise les évolutions

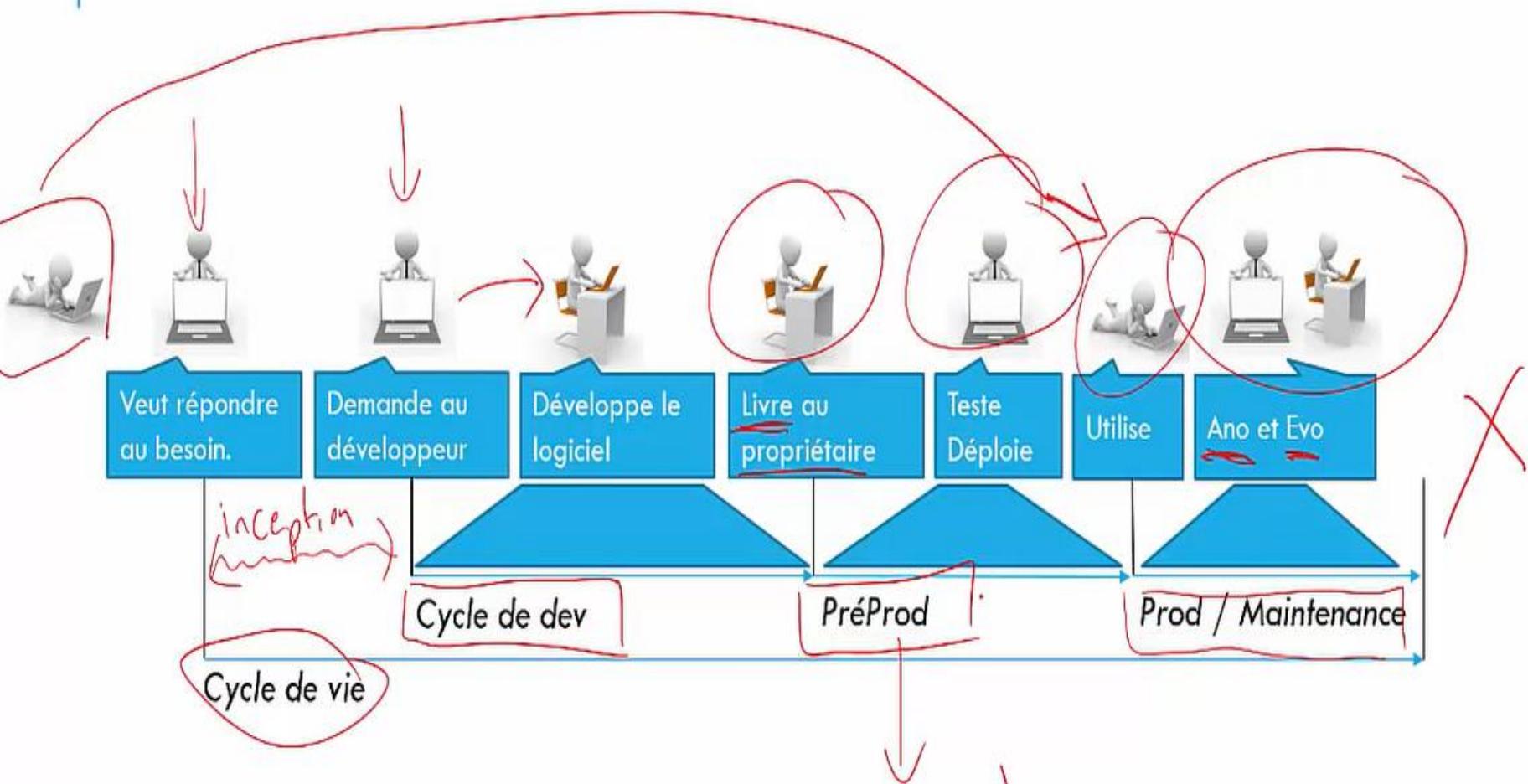


PROJET LOGICIEL – FIN

8. L'**utilisateur** n'utilise plus le logiciel,
9. Ou le **propriétaire** arrête le logiciel
10. le **développeur** ne sert plus à rien,



CYCLE DE VIE – CYCLE DE DÉVELOPPEMENT



DIFFÉRENTS TYPES DE PROJETS

Projet de mise en production

- Objectif : développer et assurer la mise en production
- Démarré dès l'expression du besoin
- Termine lorsque l'utilisateur peut utiliser le logiciel ++

Cycle de vie

Projet de développement

- Objectif : livrer la première version utilisable par l'utilisateur
- Démarré à partir de l'expression du besoin par le propriétaire
- Termine lors de la livraison par le développeur

Cycle de dev



TMA (Pas vraiment un projet) \Rightarrow Evo

- Objectif : assurer la maintenance d'une application mise en production
- Démarré lorsque l'utilisateur peut utiliser le logiciel
- Termine lorsque le logiciel n'évolue plus

Tierce Maintenance
Activer Windows

Accédez aux paramètres pour activer Windo...
A application

PROJET DE DÉVELOPPEMENT (ET TMA – EVO)

L'utilisateur a un besoin (même s'il ne le sait pas)

Le propriétaire exprime clairement ce besoin

- Cahier des charges

Le développeur spécifie ce qu'il est capable de produire

- Réponse au cahier des charges (Spec)

Le propriétaire accepte la proposition du développeur

- Cycle de dev

Une fois terminé, le développeur livre le logiciel au propriétaire

- Livraison



Expression du besoin

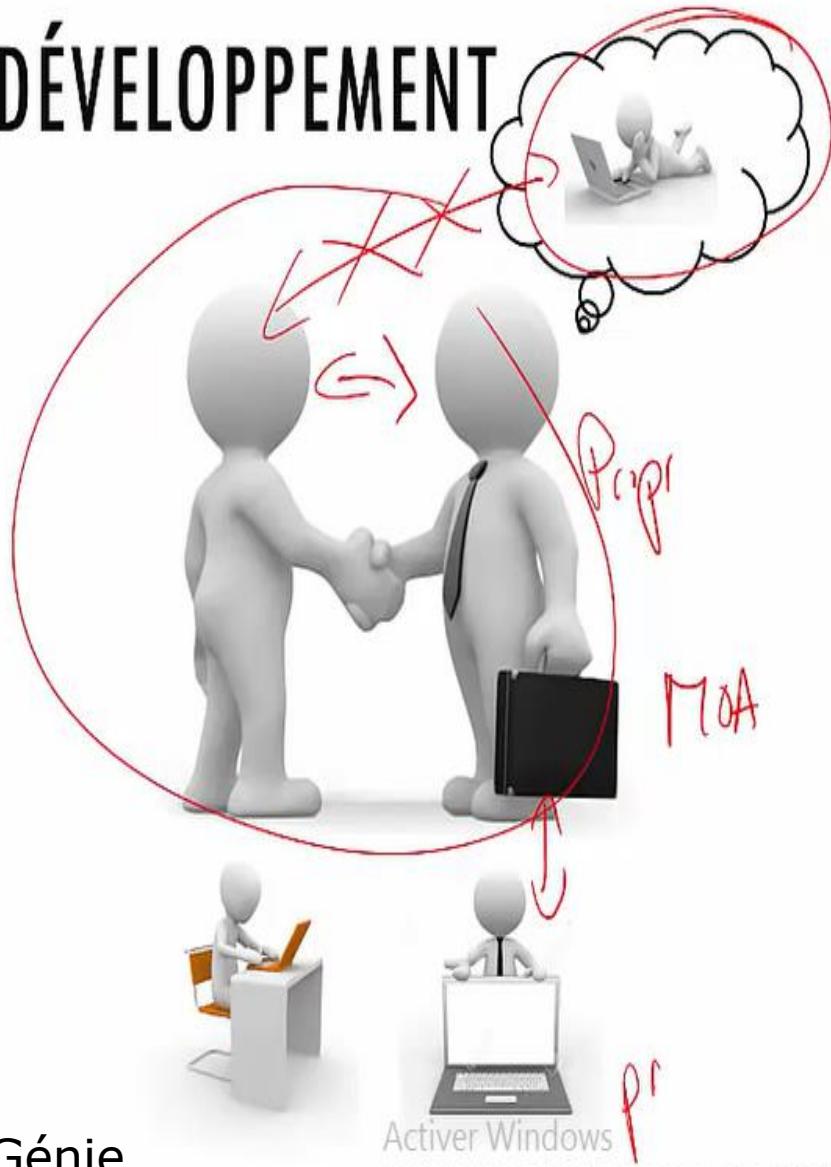
PARTICIPANTS AU PROJET DE DÉVELOPPEMENT

MOA (Maîtrise d'Ouvrage) : Celui qui possède le résultat du projet, et qui paye
=> Le propriétaire

MOE (Maîtrise d'Œuvre) : Celui qui réalise le logiciel => Le développeur

L'Utilisateur n'apparaît pas réellement dans le projet (il est représenté par la MOA)

- ~~Client, un terme trop ambiguë~~
- ~~Le client du propriétaire est l'utilisateur~~
 - ~~Le client du développeur est le propriétaire~~



PROJET DE DÉVELOPPEMENT : EXEMPLE

Assister → Exposi

Les enseignants et les chercheurs ont besoin d'un site web pour organiser la fête de la science

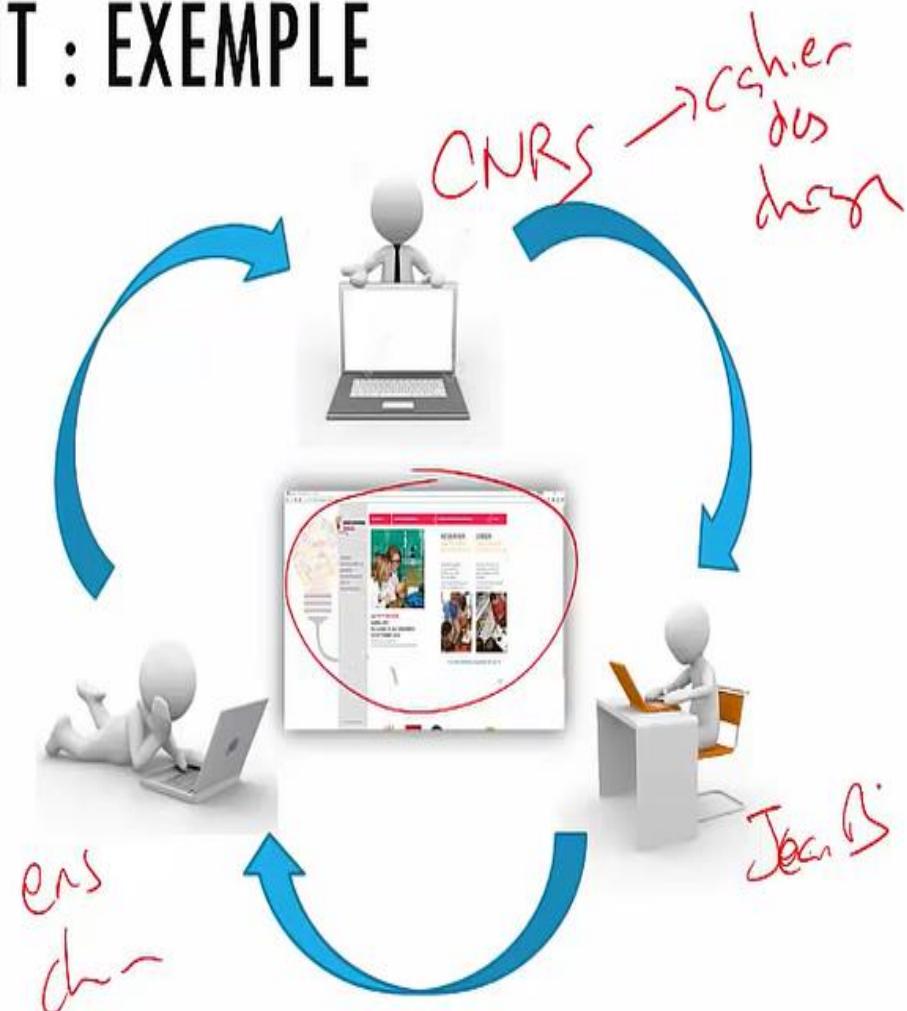
- Utilisateurs = Enseignants et Chercheurs

Le CNRS veut répondre à ce besoin en demandant à plusieurs sociétés de développer ce site web

- Propriétaire = le CNRS

La société JeanB réalise le site web qui est utilisé par les enseignants et les chercheurs

- Développeur = JeanB



(Prop) cdc → MOE

LES 3 GRANDES PHASES DU CYCLE DE DEV

Analyse (Analysis)

- La MOE spécifie ce qu'elle est capable de développer et chiffre le coût



Design (Conception)

- La MOE organise son développement, c'est-à-dire précise les gros modules à développer et précise les tâches



Design

Développement (Dev)

- La MOE développe les composants définis dans la conception et *in fine* livre le résultat



Dev

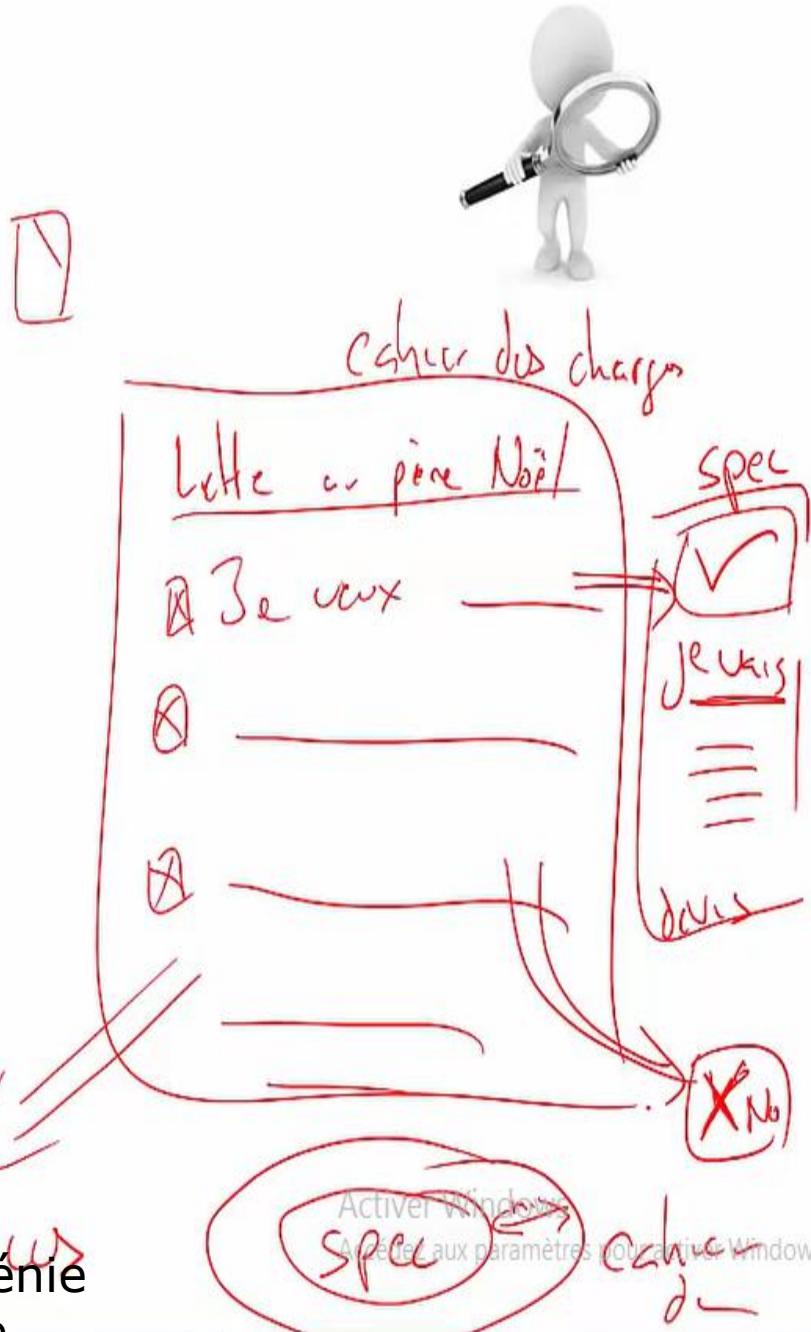
L'ANALYSE

Qui : La MOE

Entrée : Le cahier des charges

Objectif : spécifier ce que la MOE est capable de fournir

Sortie : Une spécification de ce qui sera développé





LE CHIFFRAGE

Le chiffrage se fait en jours-homme dans les projets de développement

- 1 jh = 1 homme qui travaille pendant un jours
- 20 jh = 1 mh (mois-homme)
- 12 mh = 1 ah (an-homme)

2jh 1h - 2j
2h - 1j

Idéalement, l'analyse devrait fournir les éléments permettant de chiffrer le coût

Pour autant, trop souvent le chiffrage est réalisé avant l'analyse!

Chiffrer un projet nécessite de l'expérience

Quelques éléments:

- Développer un écran d'un site web par exemple
 - 1,5 à 3 jours-homme pour les écrans simples
- Nombre de ligne de code développé en moyenne par développeur = 150
 - Une classe d'une complexité moyenne nécessite donc au moins un jours

Taille des projets :

- Petit : max 2 mh
- Moyen : entre 2 mh et 1,5 ah
- Gros : plus de 1,5 ah

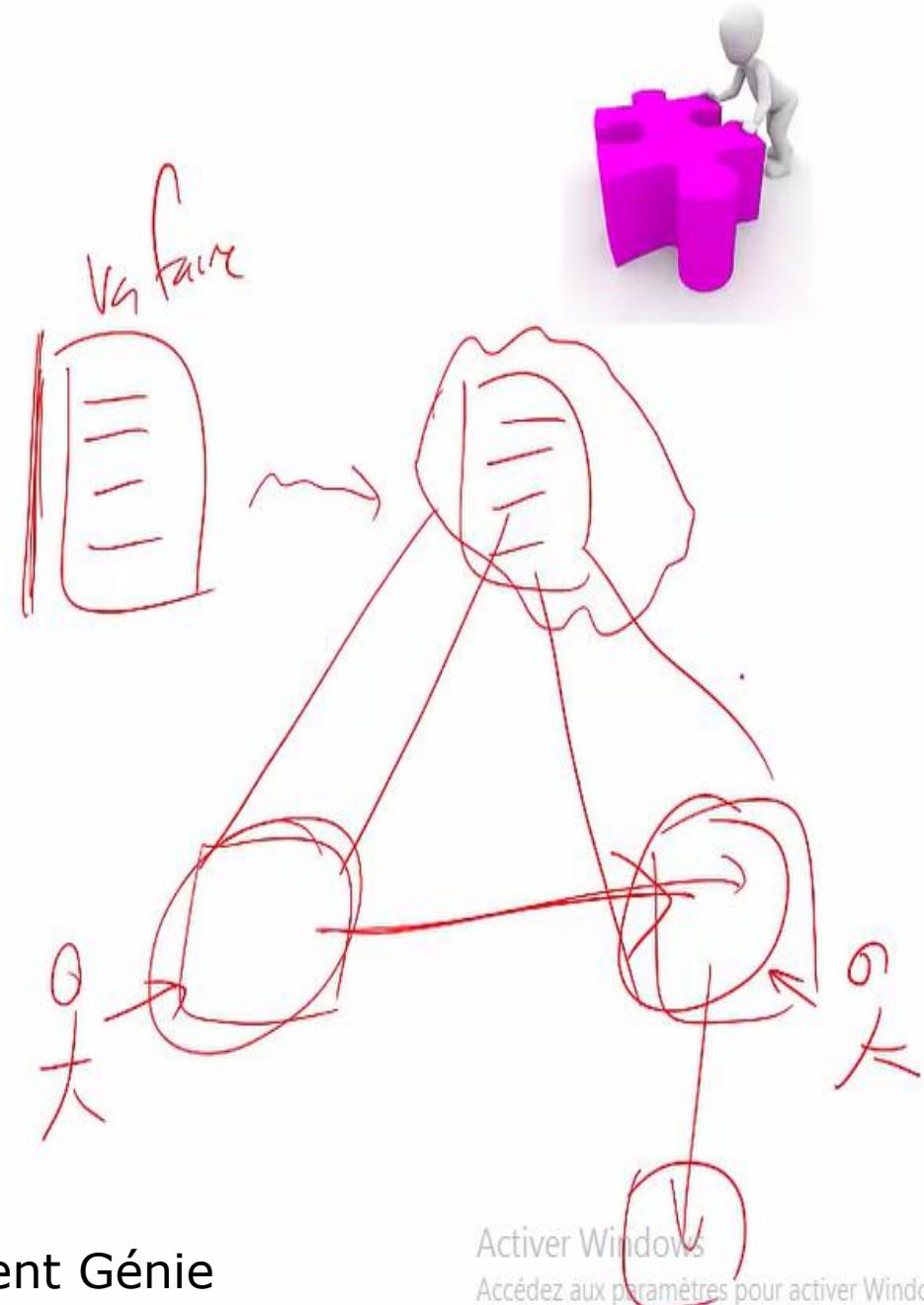
LA CONCEPTION

Qui : La MOE

Entrée : La spécification réalisée lors de l'analyse

Objectif : décomposer le logiciel en gros modules, préciser les tâches permettant la construction de ces modules

Sortie : Une spécification des composants, de leur manière de communiquer, une liste des tâches.



Architecture

L'architecture d'un logiciel est la structure des structures (modules) d'un système

Elle inclut

- Les composants logiciels
- Les propriétés externes visibles de ces composants
- Les relations entre ces composants

L'ARCHITECTURE



Le choix d'une architecture facilite la mise en place d'une conception

- 3 tiers, En couche, Par Service, Par entrepôt de données, etc.

La définition des modules et de leurs interactions

- Interface, protocole, etc.

La projection d'une architecture sur un socle logiciel permet l'identification des tâches

L'identification des tâches permet de mesurer le travail à réaliser

- (RAF : Reste A Faire)

Cela permet aussi d'organiser le travail dans une équipe

Et de maîtriser les délais et les risques.

LE MODÈLE 3-TIERS



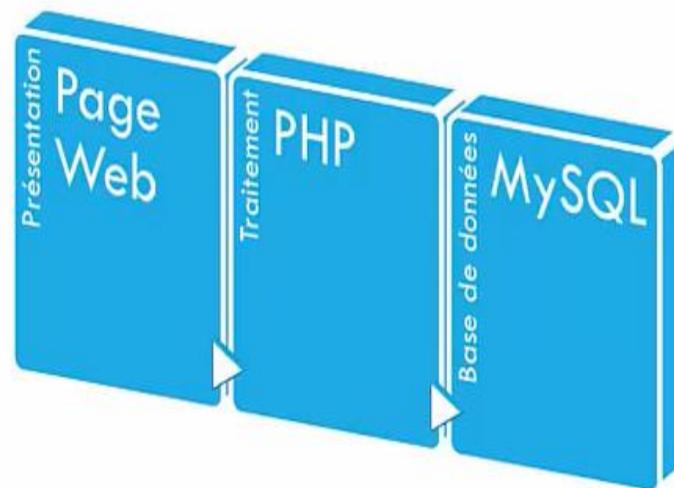
Trois modules en couche

- Base de données
- Traitement
- Présentation

Focus mis sur la protection des données,
et pas sur les performances

Tâches

- Module : BD, Traitement, Présentation
- Interfaces :
 - Traitement <-> BD,
 - Présentation <-> Traitement



LE DÉVELOPPEMENT



Qui : La MOE

Entrée : La définition des modules et la liste des tâches

Objectif : coder les modules et livrer le logiciel

Sortie : le logiciel

```
print "number)" + output"
except socket.error, (errno, strerror):
    print "ncfiles: Socket error (%s) for host %s (%s)" % (errno,
for h3 in page.findAll("h3"):
    value = (h3.contents[0])
    if value != "Afdeling":
        print >> txt, value
        import codecs
        f = codecs.open("alle.txt", "r", encoding="utf-8")
        text = f.read()
        f.close()
        # open the file again for writing
        f = codecs.open("alle.txt", "w", encoding="utf-8")
        f.write(value + "\n")
        # write the original contents
        f.write(text)
        f.close()
```

L'ENVIRONNEMENT DE DÉVELOPPEMENT



Le logiciel est composé d'artefacts logiciels (Fichiers)

Il faut savoir qui peut les éditer, les modifier, les tester, etc.

L'environnement de développement défini ce cadre

Edition Collaborative

- Les artefacts sont-ils partagés en écriture?
- Quelles sont les versions? Comment les retrouver?

Répartition des tâches

- L'affectation des tâches est-elle faite par une personne ou par l'équipe?
- Comment suivre l'avancement?

LES TESTS : CYCLE EN V

Tester le logiciel pendant tout le cycle de développement

Selon les phases:

- Analyse : Test de validation (E2E)
- Conception : Test d'intégration
- Dev : Test Unitaire

Rédaction des tests != faire passer les tests



L'ITÉRATION : CYCLE EN SPIRALE

Les phases Analyse, Conception, Codage doivent être réalisées les unes après les autres (waterfall)

Pour autant, on peut aussi faire des itérations ne prenant en compte qu'une partie du cahier des charges

Les nouvelles itérations nécessiteront des aménagements sur ce qui a déjà été réalisé

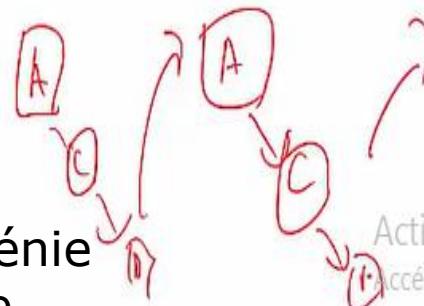


L'ITÉRATION : CYCLE EN SPIRALE

Les phases Analyse, Conception, Codage doivent être réalisées les unes après les autres (waterfall)

Pour autant, on peut aussi faire des itérations ne prenant en compte qu'une partie du cahier des charges

Les nouvelles itérations nécessiteront des aménagements sur ce qui a déjà été réalisé



L'AGILITÉ

Le cahier des charges contient les besoins exprimés par le propriétaire pour l'utilisateur

Si ces besoins changent au cours du projet, l'impact peut être important pour le logiciel

L'agilité considère comme hypothèse de base que les besoins peuvent changer et qu'il faudra y faire face!



SYNTHESE

Cycle de vie

- cycle de dev,
- pré-prod,
- prod (maintenance)

Propriétaire, Développeur Utilisateur

Cycle de dev

- Analyse, Conception, Dev

Itération (Cycle en spiral)

Test (Cycle en V)

Agilité



L'AGILITÉ

Les besoins peuvent changer et il faudra s'y faire !

Le propriétaire est responsable des changements

Les développeurs travaillent pour réaliser les évolutions du besoin



3 GRANDS PRINCIPES DE SCRUM

L'Equipe

- Composée des développeurs (dont un scrum master) et du propriétaire (Product Owner)
- Elle est responsable du développement

Des Itérations

- L'équipe livre périodiquement les versions du logiciel
- La progression du projet est calculée sur les itérations

Basé sur le temps

- Les itérations sont bornés dans le temps
- Toutes les réunions sont bornés dans le temps
- Le coût est borné dans le temps !





L'ANALYSE = LE BACK LOG

User Story

- Un scénario utilisable par l'utilisateur

En tant que <utilisateur> je veux <utiliser>
afin de <objectif>

- Ex: « En tant que laboratoire, je souhaite ajouter
un nouvel atelier, afin de permettre aux
enseignants de s'y inscrire »

Le Back Log contient toutes les user story que
l'équipe aimeraient pouvoir faire

- Expression / Spécification des besoins

Le PO (Product Owner) précise sa priorité

- Souhait d'ordre de réalisation

id	User Story	Prio
#1	En tant que ...	N°5
#2	En tant que ...	N°1
#3	En tant que ...	N°3
#4	En tant que ...	N°2
...



CHIFFRER LE BACK LOG

Chaque US (User Story) a une difficulté

- Coût de développement de l'US

Le chiffrage est abstrait et défini par comparaison

- Coût : 1, 2, 3, 5, 8, 13, 21, 34, etc
- Si US#1 = 2 alors US#2 = 3 (Scrum Master)

La somme des coûts des USs donne le coût du Back Log

- Coût abstrait !

id	User Story	Prio	Coût
#1	En tant que ...	N°5	3
#2	En tant que ...	N°1	2
#3	En tant que ...	N°3	5
#4	En tant que ...	N°2	3
...
			=410



ET LES BESOINS NON FONCTIONNELS?

Le BackLog est très orienté User Story

- Besoins fonctionnels (Démo)

On peut quand même y ajouter des User Story non fonctionnelles

- Temps de réponse, Tolérance aux pannes, etc.

Pour autant, l'idéal est d'intégrer ces besoins dans les User Story visibles





LA CONCEPTION = LES SPRINT

Sprint = période de temps pendant laquelle l'équipe travaille à réaliser un ensemble fini de US

La séparation du BackLog en plusieurs sprint et la première étape de la conception

Un sprint a un coût estimé

id	User Story	Prio	Coût
#1	En tant que ...	N°5	13
#2	En tant que ...	N°1	8
#3	En tant que ...	N°3	21
#4	En tant que ...	N°2	8
...
			=410



Sprint #1



Sprint #2



LA CONCEPTION = LES TÂCHES

Choix de l'architecture

- Impact sur les sprints suivants

Définition des composants

- Uniquement pour les US du sprint

Lister les tâches à faire

- Inclure les tests, l'intégration, documentation, qualité, et autres aspects techniques

Se répartir les tâches

- Répartition spontanée (Scrum Master)



LE DÉVELOPPEMENT = SUIVI DES TÂCHES



La réunion du matin (Daily Meeting)

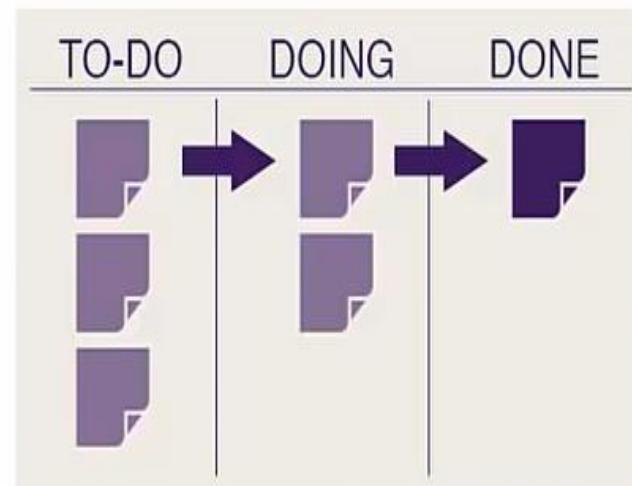
- Qui fait quoi? Qui a besoin de quoi?

Le développement

- Travail collaboratif

Bilan

- Ce qui a été fait, les points durs, l'amélioration



AVANCEMENT DU PROJET ET ITÉRATION

A la fin d'un sprint, on sait quels USs ont été réalisées complètement

- Vélocité = Somme des scores des USs complétés

La vélocité est un bon indicateur de la capacité de l'équipe

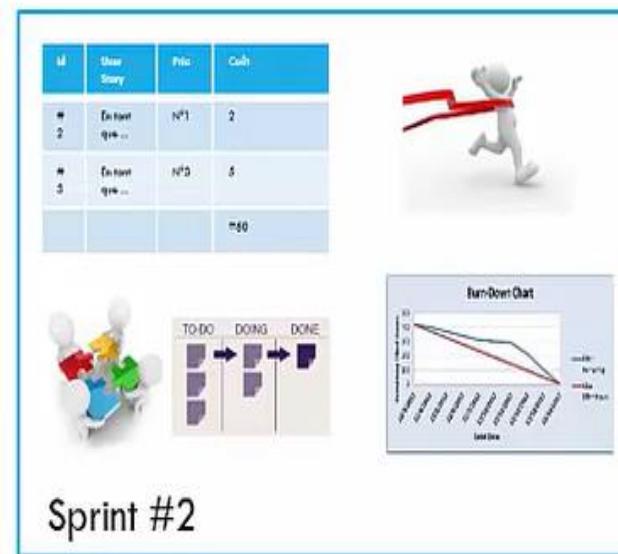
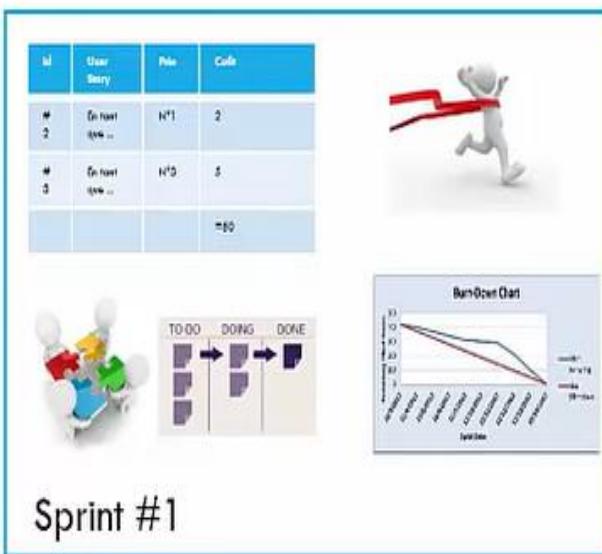
- Capacité = Score que l'équipe devrait pouvoir réaliser

=> Burn down chart



SCRUM : VUE GLOBALE

Id	User Story	Prio	Coût
#1	En tant que ...	N°5	3
#2	En tant que ...	N°1	2
#3	En tant que ...	N°3	5
#4	En tant que ...	N°2	3
...
			=410



...

SCRUM RÔLES

1 PO (Product Owner)

- Représente l'utilisateur / Possède le logiciel
- Responsable des USs
- Fixe les priorités des USs



1 Scrum Master

- Maître du temps (Time Box)
- Anime l'harmonisation des coûts des U.S.
- Anime l'affectation spontanée des tâches



N Développeurs

- Responsable de l'intégralité des développements



SCRUM DANS LA VRAIE VIE

Back Log

- Privilégiez les User Story assez conséquentes, proposant une réelle valeur ajoutée
- Détaillez au maximum les User Story, en utilisant des maquettes par exemple

Sprint

- Un sprint 0 peut vous aider à mettre en place l'architecture, le socle logiciel et l'environnement de développement
- Planifiez 3 sprints dès le début, et n'hésitez pas à replanifier
- Ajouter des tâches de test E2E qui permettent la validation d'un User Story

Tâches

- Lister les petites tâches réalisables par un développeur dans la journée
- Privilégiez le travail d'équipe au travail de spécialiste / mercenaire

Avancement

- Attendez 2 ou 3 sprints avant de tirer des conclusions sur votre avancement
- Avancez à un rythme de croisière

SCRUM Like

- N'hésitez pas à vous faire votre SCRUM

SYNTHÈSE

Méthode Agile

- Analyse = Back Log
- Conception = Sprint + Tâches
- Développement = Suivi des Tâches

Avantages

- Équipe Responsable
- Itération
- Time Box

Manques, Limites

- Tests, Jalons, Chiffrage Intrinsèque



ANOMALIES ET BUG

Tout logiciel a des anomalies (Ano)

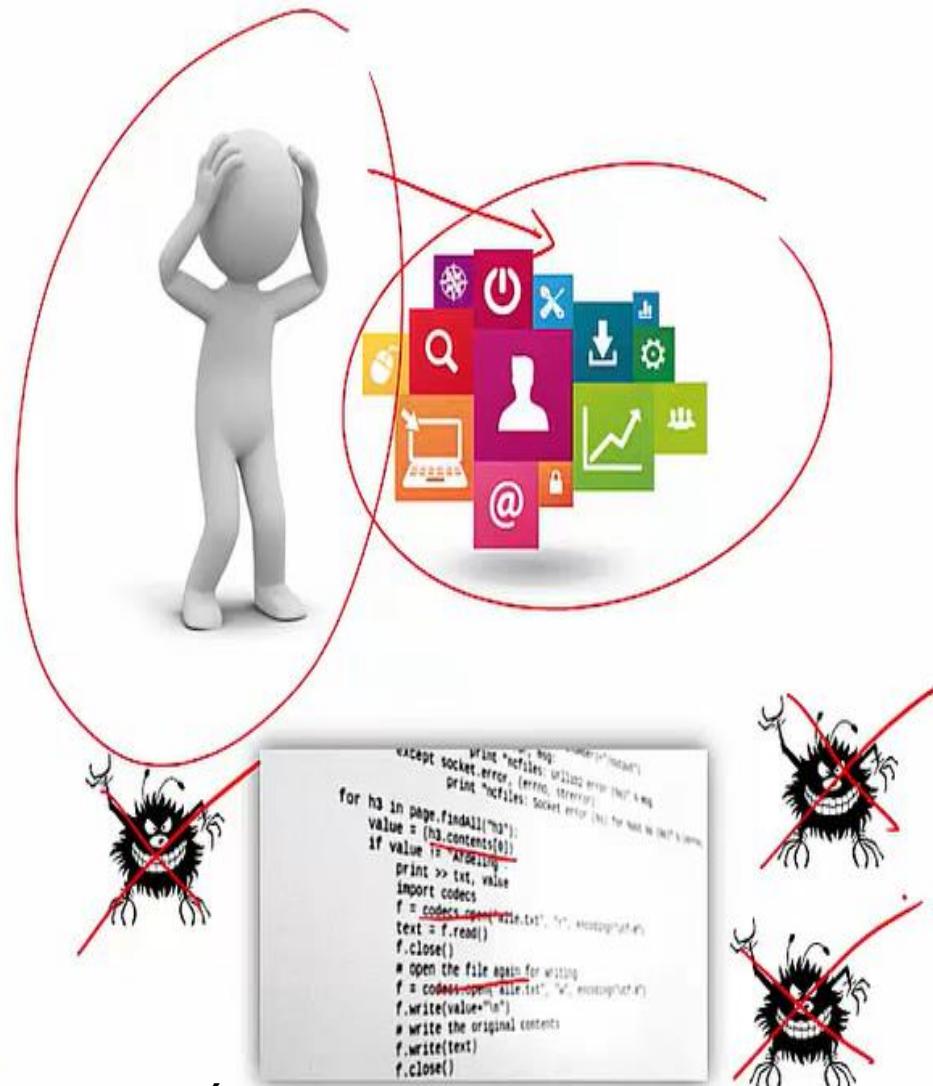
Il ne fonctionne pas comme il devrait

Il plante, rame, est moche, etc.

Causées par des bugs

Erreur de codage

Hypothèse de base : si Ano alors Bug



TEST – TESTEUR

Un test efficace trouve une anomalie
avant que l'utilisateur ne la trouve !

- Tester, c'est donc chercher des anomalies

Un bon testeur

il trouve des anomalies.

Un bon programmeur

il fixe les anomalies !



COMMENT TESTER?

1. Mettre l'objet dans un état *initial*
2. Lui faire exécuter des traitements
 1. Traitement —
 2. Traitement —
 3. Traitement —
 4.
3. Vérifier le résultat obtenu

Si RO != RA alors Bug

Résultat Attendu

Département Génie
Informatique

Définir un test nécessite

- De préciser l'état initial de l'objet à tester
- De préciser la suite des traitements à effectuer
- De préciser l'état attendu

Résultat Attendu



3 TYPES DE TEST : VALID., INT., UNIT.

Tests de Validation

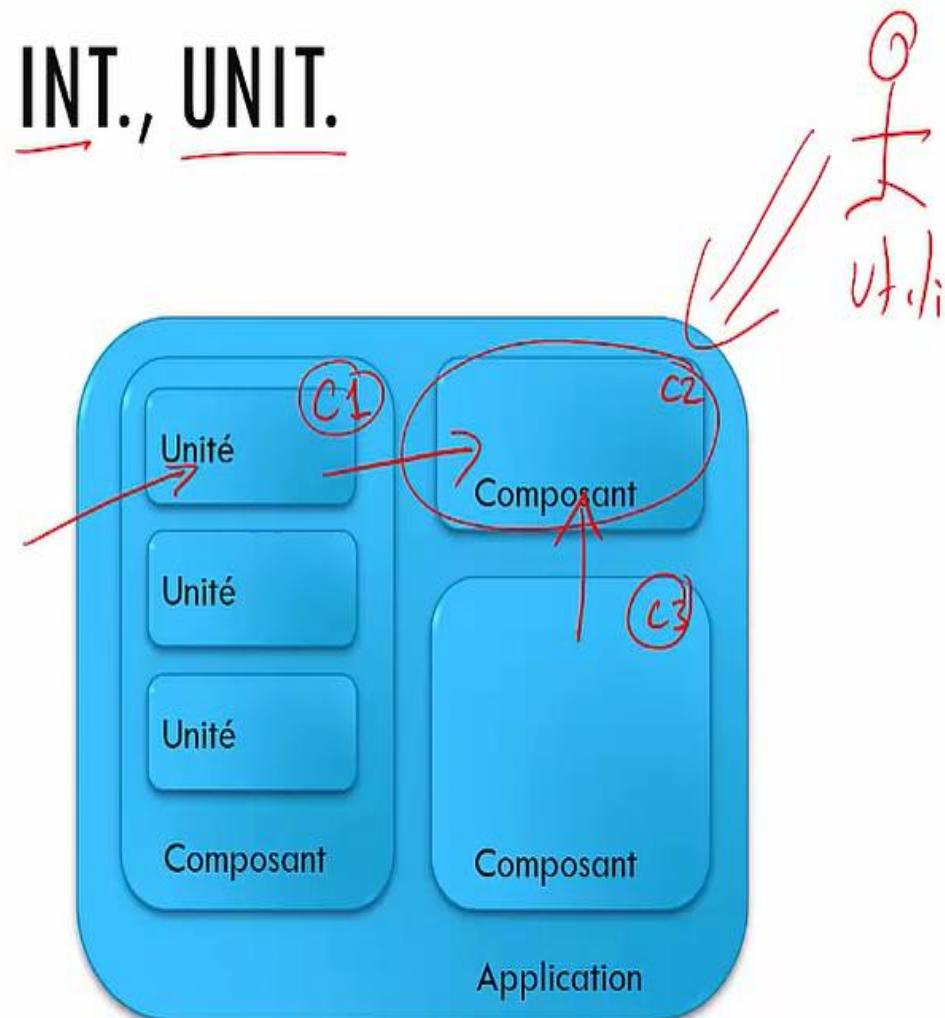
- Objet: L'application
- Pour: Vérifier l'absence de bug au niveau utilisateur

Test d'intégration

- Objet: Un composant de l'application
- Pour: Vérifier que le composant va bien s'intégrer avec les autres

Test Unitaire

- Objet: Une unité de code
- Pour: Vérifier que le bout de code ne contient pas de bugs



TEST UNITAIRE – FONCTION (1/2)

```
public static double div(double numerator, double denominator)
```

RA

$10\% \rightarrow +\infty$

$-10\% \rightarrow -\infty$

Principes

Tester les cas aux limites

Tester quand même les cas simples

Ne pas trop tester

```
@Test  
public void testDivInfinity() {  
    double posInf = MathEx.div(10, 0);  
    assertEquals(Double.POSITIVE_INFINITY, posInf, 0.1);  
    RA  
    double negInf = MathEx.div(-10, 0);  
    assertEquals(Double.NEGATIVE_INFINITY, negInf, 0.1);  
}  
  
@Test  
public void testSimpleCases() {  
    double pos = MathEx.div(10, 2);  
    assertEquals(5, pos, 0.1);  
    10/2  
    double neg = MathEx.div(-10, 2);  
    assertEquals(-5, neg, 0.1);  
  
    double dbl = MathEx.div(10, 3);  
    assertEquals(3.33, dbl, 0.1);  
}
```

Activer Windows

Accédez aux paramètres pour activer Windows

TEST UNITAIRE – FONCTION (2/2)

`public static int[] sort(int[] array)`

Principes

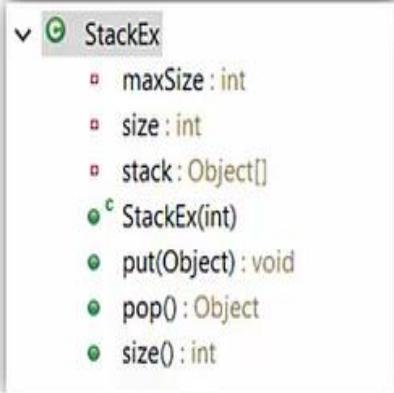
Coder un oracle

Coder un générateur aléatoire

Exécuter plusieurs fois le test

```
public boolean isSorted(int[] array) {  
    for (int i = 0; i < array.length - 1; i++) {  
        if (array[i] > array[i + 1])  
            return false;  
    }  
    return true;  
  
}  
  
public int[] generateArray() {  
    int size = (int) (Math.random() * 10) % 10;  
    int[] array = new int[size];  
    for (int i = 0; i < size; i++) {  
        int value = (int) (Math.random() * 100) % 100;  
        array[i] = value;  
    }  
    return array;  
}  
  
@Test  
public void testSort() {  
    int loop = 10;  
    for (int i = 0; i < loop; i++) {  
        int[] input = generateArray();  
        assertTrue(isSorted(sortEx.sort(input)));  
    }  
}
```

TEST UNITAIRE - OBJET



Principes

Tester chaque méthodes

Tester les effets de bord

```
@Test  
public void putMore() {  
    for (int i = 0 ; i <= maxSize ; i++) {  
        stack.put(i);  
    }  
    assertEquals(maxSize, stack.size());  
}  
  
@Test  
public void popMore() {  
    Object o = stack.pop();  
    assertNull(o);  
}  
  
@Test  
public void putAndPop() {  
    stack.put("1");  
    assertEquals("1", stack.pop());  
}
```

TEST D'INTÉGRATION – SERVICE REST

API call:

```
api.openweathermap.org/data/2.5/weather?q={city name}  
api.openweathermap.org/data/2.5/weather?q={city name},{country code}
```

Principes

Tester les appels nécessaires

Vérifier les cas d'erreur

Vérifier le format de retour (ex: Json)

```
@Test  
public void noAPIToken() {  
    try {  
        String url = "http://api.openweathermap.org/data/2.5/weather?q=London";  
  
        HttpClient httpClient = HttpClientBuilder.create().build();  
        HttpGet getRequest = new HttpGet(url);  
        HttpResponse response = httpClient.execute(getRequest);  
  
        assertEquals("HTTP/1.1 401 Unauthorized", response.getStatusLine().toString());  
  
    } catch (Exception ex) {  
        fail("exception");  
    }  
}
```

TEST DE VALIDATION

Vérifier un scénario de site web

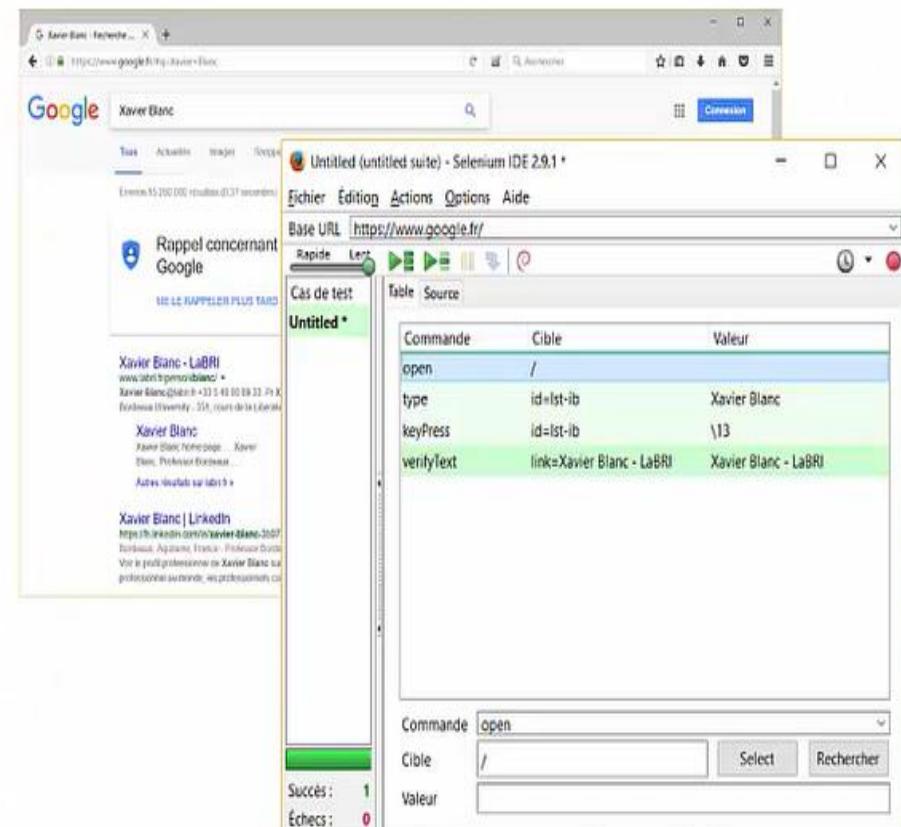
Principes

Scripter le scénario de site web

Préciser le résultat attendu

Jouer le scénario

Ex: Selenium



QUOI TESTER



Unitaire

- Le code complexe (traitements, effets, ...)
- Le code critique (business)
- Pas les DTO !!!
- Couverture ?

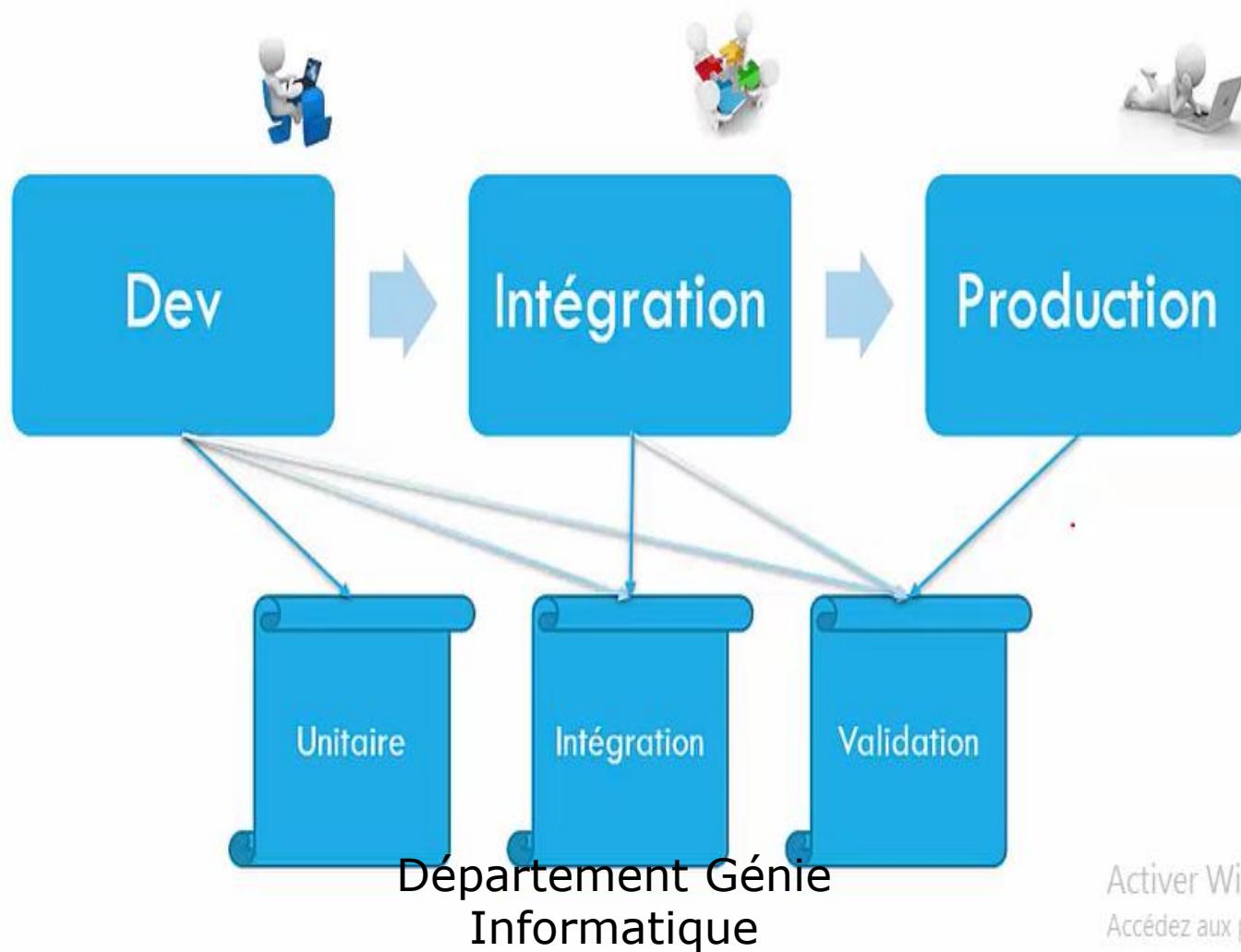
Intégration

- Les interactions complexes
- Les composants critiques
- Les appelants / appelés

Validation

- Les scénarios classiques

ENVIRONNEMENTS DE TEST



SYNTHESE

→ Tester diminue le nombre de bug

Unitaire > Intégration > Validation

Etat Initial, Traitement, Vérification

Tester les éléments complexes

Définir les tests avant de coder



Contact de base de donnée

Entre d'une part :

Département Génie Informatique

Mamou

BP : 2174, Mamou, République de
Guinée

Tél. : +224 664 74 48 67 / 622 57 57 30

Email : astourep@gmail.com

Ci-après dénommée « **Prestataire** »

Et d'autre part :

.....

Quartier :

BP :

Tél. :

Email :

Ci-après dénommée « **Client** »

Contact de base de donnée

IL A PREALABLEMENT ETE EXPOSE CE QUI SUIT :

E/se, représentée par son 1^{er} responsable, souhaite réaliser un système de gestion intégré (SGI) pour son institution comme défini dans le présent contrat.

Le Département Génie Informatique, représentée par le Chef de Département, ayant tous pouvoirs prévus à cet effet a accepté de réaliser ce système, et le présent contrat a pour objet d'en préciser les termes.

□ ARTICLE 1 : OBJET

E/SE confie au prestataire la conception et réalisation du système GP et sa documentation d'exploitation pour la gestion de ses activités.

Contact de base de donnée

- **ARTICLE 2 :LES TACHES POSSIBLES SUR LE LOGICIEL :**
- **Edition des documents :**
 - Ce module permet l'édition de tous les documents nécessaires au contrôles et suivi des activités de E/SE.
- **Enregistrement :**
 - Ce module permet de saisir et de sauvegarder toutes les informations concernant une opération dans le système,
- **Consultation :**

Ce module permet d'interroger la base pour tout type d'informations gérées dont on souhaite consulter
- **MAJ :**

Ce module permet d'effectuer toutes les mises à jour des informations enregistrées dans la base de données,

Contact de base de donnée

Sécurité:

Permet d'effectuer les tâches de sécurisation du système :

Utilisateurs : permet de créer et de définir les priviléges d'accès au système pour chaque utilisateur

Sauvegarde/Restauration : ce module permet de faire des backups réguliers et des restauration des données en cas de crash du système,

Journal : c'est un fichier qui permet de retracer toutes les activités effectuées par un utilisateur dans le système.

Contact de base de donnée

ARTICLE 3 - OBLIGATIONS DU CLIENT

Note : il est impératif de rappeler que le bénéficiaire doit assurer au prestataire toutes facilités pour l'exécution de sa prestation en particulier en ce qui concerne l'accès aux locaux et la mise à disposition d'un certain nombre de moyens, voire apporter son concours au prestataire dans l'exécution de sa prestation.

Contact de base de donnée

- Le client doit assurer au prestataire toute facilité pour l'exécution de sa prestation.
 - Le client devra laisser libre accès au logiciel conçu, étant précisé que l'intervention sur le logiciel par le prestataire ne pourra être effectuée qu'à l'adresse de l'installation, à l'exclusion de tous autres sites (non définis dans ce présent contrat).
-

-
- Il est expressément convenu entre les parties que dans le cas où le client serait dans l'obligation de changer de lieu d'installation de ses logiciels dans un périmètre dépassant celui des interventions prévues au contrat, il devrait en avertir le prestataire au préalable pour mise au point d'un nouvel avenant au contrat.
-

-
- De manière générale, le client devra apporter tout son concours au prestataire dans l'exécution de sa prestation et s'engage à collaborer afin de permettre au mieux la réalisation des prestations dues.
-

-
- Le client s'oblige à fournir au prestataire les coordonnées d'un interlocuteur technique désigné, joignable téléphoniquement et par email.
-

Département Génie
Informatique

Projet informatique

Exercices

Une société d'assurance décide de proposer un cahier des charges pour la gestion du personnel de la siccité.

Chaque travailleur de cette société est identifié par un matricule (constituer de trois chiffres et une lettre) et caractérisé par deux attributs : identité du travailleur (nom et prénom) et sa fonction dans l'entreprise.

- 1-quel sont les participants de ce projet ?
 - 2-cette application doit avoir la possibilité d'édition la liste du personnel
 - 3-Avoir la possibilité de d'ajouter un personnel à l'aide d'un formulaire
 - 4-Avoir la possibilité de rechercher un travailleur par son numéro matricule
 - 5-Avoir la possibilité de supprimer un travailleur dans la base
 - 6-Assurer la sécurité de la base de donnée.
-

Projet informatique

Exercices

Le département Génie Informatique à besoin d'une application pour la gestion des effectifs du département.

Ce département est composé de quatre promotions :

Promotion14=1

Promotion13=2

Promotion12=3

Promotion11=4

Travail demander :

- 1-Enregistrer chaque étudiant avec son matricule,nom,prenom.
- 2-Editer la liste des étudiants par promotion .
- 3-Editer les fiches de notes par promotion.
- 4-Sécuriser la base des données .