Université Cadi Ayyad
ENSA de Safi
Département Informatique Réseaux et Télécommunications

# Génie Logiciel

Kamal BARAKA
**baraka.kamal@yahoo.fr**

Le contenu de ce support de cours a été influencé par les lectures citées à la fin de ce support.

# Définition : Logiciel

- Ensemble de
  - Programmes informatiques,
  - Système d'exploitation,
  - Etc.
- Logiciel = ensemble de programmes qui coopèrent pour réaliser un objectif bien défini.
- Programme = ensemble d'instructions exécutées pour accomplir une tâche bien précise.

# Définition : Génie Logiciel

- Ensemble des activités de conception
  - Développement,
  - De l'exploitation
  - Et de la maintenance des logiciels,

  - Application de l'ingénierie au logiciel

# Pourquoi c’est si important ?

## Aujourd’hui, le logiciel contrôle le monde !

- Processus métiers (administrative etc.)
- Gouvernement
- L’industrie(Usines, chaines de fabrication)
- Transports
- Défense, finance, santé…
- Edition, médias…
- Les nouvelles technologies du web, commerce électronique…
- Et bien plus!!

## Des enjeux aussi bien économiques que politiques

# Pour c’est si important ?

Les conséquences en cas de problèmes peuvent être très lourdes!

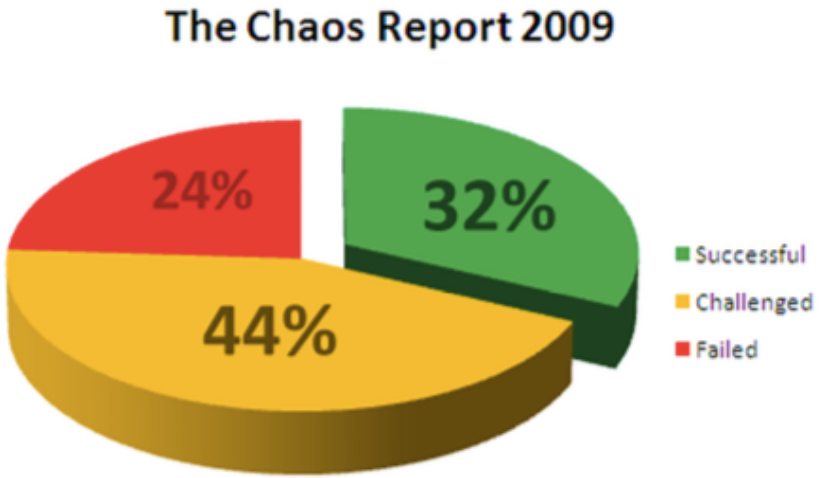
- Therac 25,'85‘87: 6 patients irradies, 2morts
- Syst. Bagages Aeroport Denver,'95:16 mois, 3.2Mds$
- Ariane 5vol 88/501,'96: 40s de vol,destr., 850M$
- Mars Climate Orbiter & Mars Polar Lander,'99: destr.

# Le talon d'Achille du GL

- Le coût
- Le temps
- La qualité́

# Quelques chiffres !

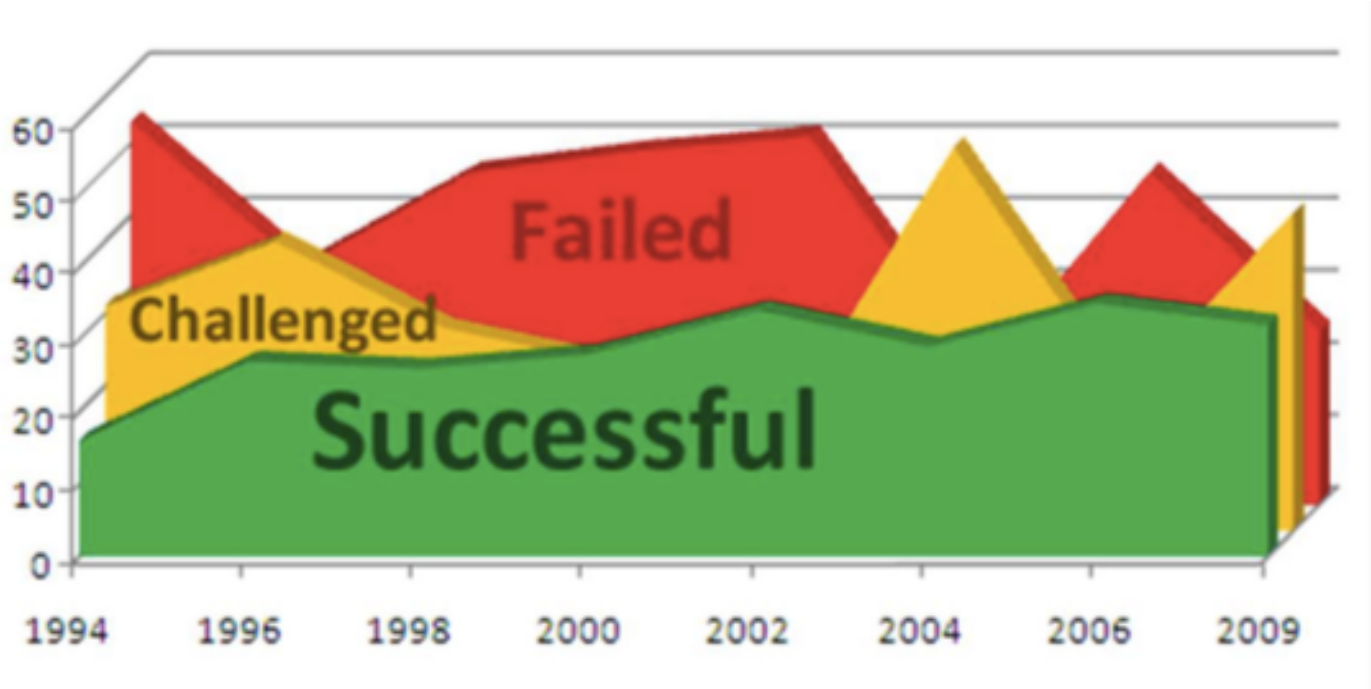
## Source: The Standish Group




- Successful means on-time, on-budget, and with all features and functions as defined in the initial scope;
- challenged means late, over budget, and/or with less features and functions than defined in the initial scope;
- failed means cancelled prior to completion, or delivered but never used.

# Quelques chiffres !

- Le taux de succès en constante hausse, mais le taux d’ échec reste aléatoire



# Evolution du logiciel

- La livraison n'est pas une fin en soi :
  - Après sa livraison un logiciel peut être modifié.
  - Chaque modification transforme un peu plus le logiciel
  - Le logiciel évolue en fonction de l'évolution du matériel.
- Augmentation du nombre de lignes de code est dû à la complexité croissante des systèmes conçus.

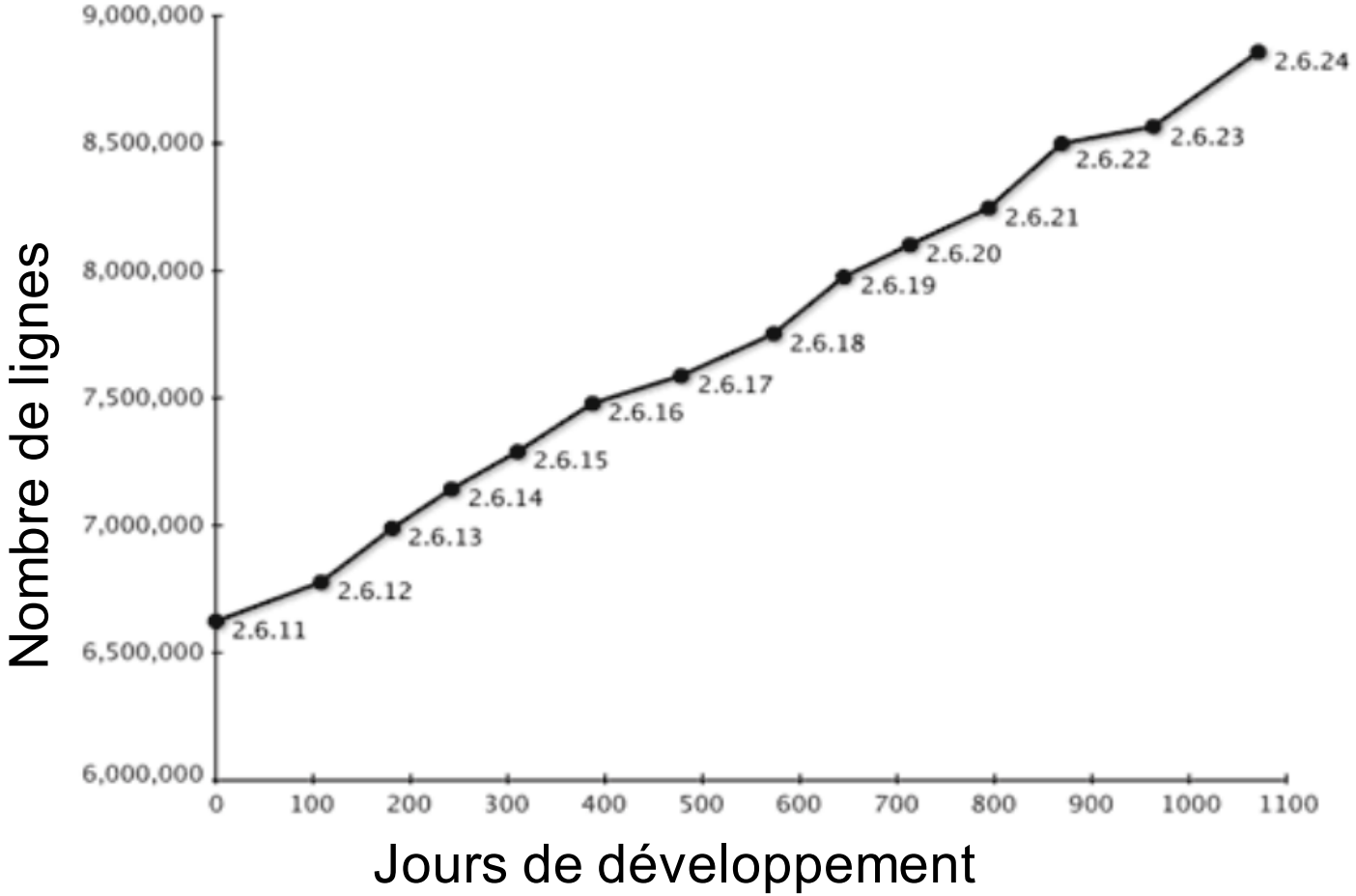
# Exemple d'évolution d'un logiciel

- Windows

| Sortie | Produit | Equipe dév. | Equipe test | Lignes de code ($\times 10^6$) |
|---|---|---|---|---|
| 07/93 | NT 1.0 (version 3.1) | 200 | 140 | 4/5 |
| 09/94 | NT 2.0 (version 3.5) | 300 | 230 | 7/8 |
| 05/95 | NT 3.0 (version 3.51) | 450 | 325 | 9/10 |
| 07/96 | NT 4.0 (version 4.0) | 800 | 700 | 11/12 |
| 12/99 | NT 5.0 (Windows 2000) | 1.400 | 1.700 | 29 |
| 10/01 | NT 5.1 (Windows XP) | 1.800 | 2.200 | 40 |
| 04/03 | NT 5.2 (Win. Serv. 2003) | 2.000 | 2.400 | 50 |

# Evolution d'un logiciel

- Noyau Linux



# Coût de développement

| Tâche | Proportion |
|---|---|
| Conception | 1/3 |
| Programmation | 1/6 |
| Tests des composants | 1/4 |
| Tests du système | 1/4 |

# Bogue (Errata)

- Il y a entre 0,5 et 3 erreurs de programmation toutes les 1000 lignes de code [Bell Labs].
- Cela entraîne les conséquences suivantes :
  - 2 projets sur 3 sont livrés en retard ou ont dépassé le coût initialement estimé.
  - 2 projets sur 3 révèlent d'importants défauts ou montrent une instabilité durant leur première année d'exploitation.
  - Coût estimé des bogues :
    - 60 milliards de $ par an (aux états unis)

# Coût de maintenance

- Il peut atteindre 70% du coût total

| Tâche | Proportion |
|---|---|
| Modifications demandées par le client | 2/5 |
| Correction d'erreurs | 1/5 |
| Modifications des formats (données, fichiers, ...) | 1/6 |
| Changement matériel | 1/20 |
| Modification de la documentation | 1/20 |
| Amélioration des performances | 1/20 |
| Divers | 1/12 |

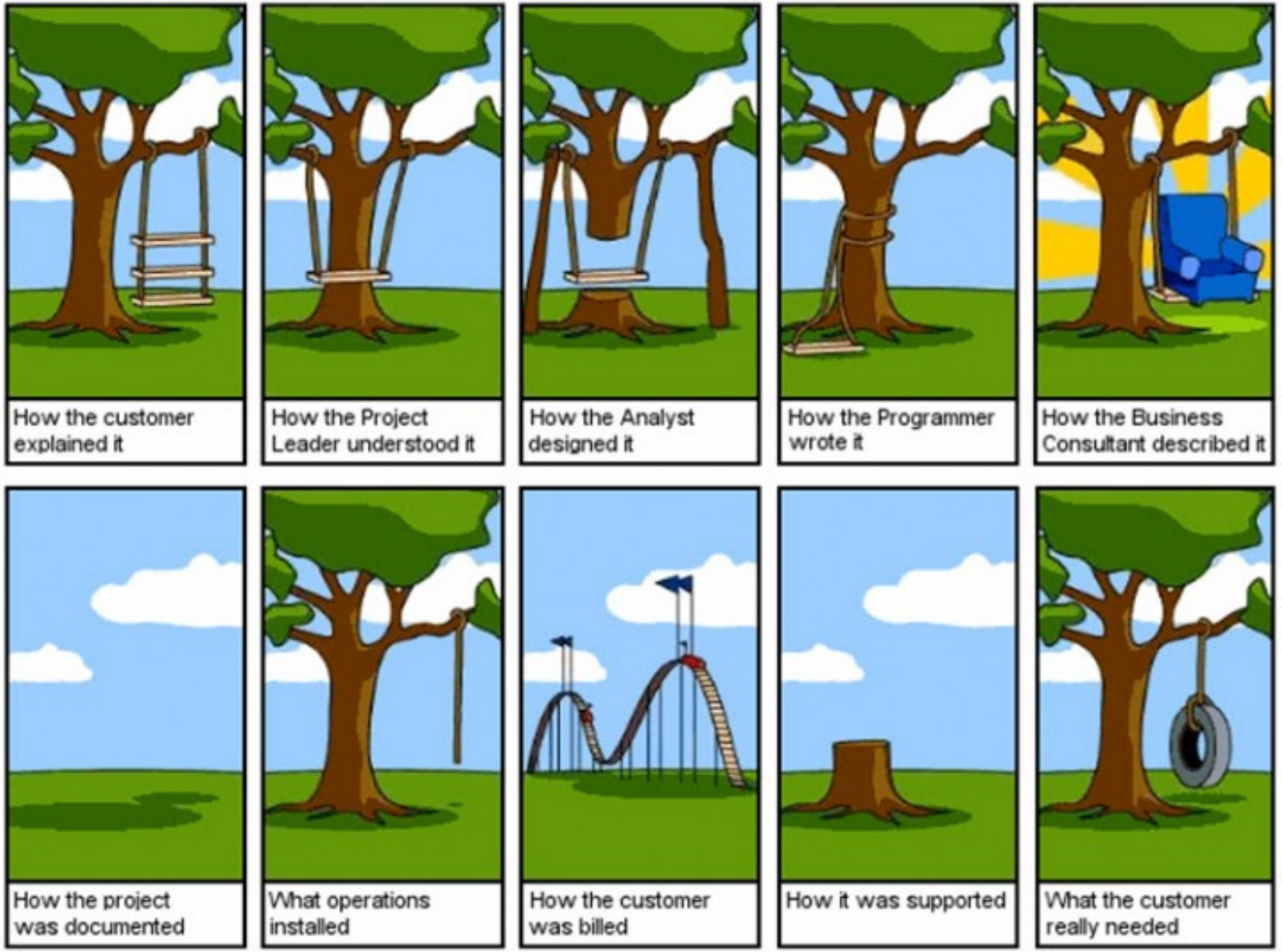
# Facteurs d'échec d'un projet

- Fonctionnalités non prévues (non souhaitées)
- Développement imprécis
- Problèmes d'intégration
- Réécriture du code source
- Absence de documentation motivant les décisions prises lors de la conception
- Etc.

# Risque d'abandon

- 30% des projets informatiques sont annulés avant la mise en production.
- 90% des projets informatiques sortent en retard.
- Le risque d'abandon ou de retard important augmente de plus en plus rapidement à mesure que la taille globale du projet croît.
- Coût estimé des abandons :
  - des milliards de dollars par an (aux états unis)

# Problème de qualité du logiciel

- 

# Dépassement de budget prévu

- Modifications effectuées après la livraison d'un logiciel coûtent cher, et sont à l'origine de nouveaux défauts.
- Les adaptations sont bien souvent une nécessité du fait de l'évolution des produits et des attentes des utilisateurs.
- 50% des projets informatiques dépassent le budget prévu :
  - Maintenance corrective : 20%
  - Maintenance adaptative : 25%
  - Maintenance évolutive ou perfective : 55%

# Sonde Mariner 1 (1962)

# Sonde Mariner 1 (1962)

- Coût: 18,5 millions de dollars.
- catastrophe: Mariner 1 est la première sonde du programme mariner, lancée le 22 juillet 1962 pour mission de survol de vénus mais cette dernière a détourné un peu sur le trajectoire de sa destination pour causer sa destruction après 293 secondes de son décollage.
- Cause: Le programmeur a mal transcrit une formule manuscrite en informatique.

# Sonde Mariner 1 (1962)

- Détails: La défaillance provient d'une erreur de transcription manuelle dans la spécification du programme de guidage. Le rédacteur a oublié la barre souscrite dans la formule .
- Le manque de cette barre a causé une mal interprétation des valeurs (variation du temps) et lors des corrections induites qui ont été erronés.
- Le programme de contrôle ne lisse plus les valeurs de la vitesse et réagit brutalement à des variations mineures
- La fusée a perdu sa trajectoire , ce qui obligea l'officier de sécurité de commander sa destruction après 297 secondes.

# Destruction d'Ariane 5 (1996)

- Cause directe : Conversion entier/flottant non autorisée.
- Impact :
  - Explosion 30s après le décollage
  - 1 année de retard pour le programme Ariane 5
- Origine :
  - Reprise du code spécifié pour Ariane 4
  - Absence de tests de validation pré-vol
  - Désactivation de la gestion des exceptions
  - Conservation de code inutile

# Perte de Mars Climate Orbiter (1999)

- Cause directe : Confusion entre pieds et mètres.
- Impact :
  - Destruction de la sonde à l'entrée de l'atmosphère martienne
  - Coût : 120 millions de $

# Bogue de l'an (2000)


- Cause directe : Codage de la date sur 2 caractères.
- Impact :
  - Très nombreuses mesures préventives et correctives
  - Coût : estimé à 500 milliards de F
- Origine :
  - Arbitrage économie mémoire / durée de vie
  - Mauvaise perception de la durée de vie des logiciels

# Echec du missile Patriot (1991)

# Echec du missile Patriot (1991)

- Coût: 28 soldats morts, 100 blessés.
- Catastrophe: Au cour de la première guerre de golfe , un système Américain des missiles Patriot en Arabie Saoudite n'a pas réussi à suivre et à intercepter un missile Scud Irakien entrant.
- Le missile a détruit une caserne de l'arme Américaine.
- Cause: un calcul inexact du temps depuis le démarrage en raison d'erreurs arithmétique des ordinateurs.

# Echec du missile Patriot (1991)

- Détails: la durée a calculer c'est le temps en deuxièmes seconde mesuré par l'horloge interne du système multiplier par 1/10 pour produire le temps en seconde.
- L'erreur est que le registre utilisé dans le Patriot est de taille 24 bit ce qui produit au lieu de stoker la valeur binaire:
  0,0001100110011001100110011001100
  le système a stocké la valeur:
  0,00011001100110011001100

# Echec du missile Patriot (1991)

- Ce qui implique une différence de 0,0000000000000000000011001100 et si on la transforme en seconde après un simple calcul (0,000000095 × 100 × 60 × 60 × 10 = 0,34) on trouve qu'il ya un décalage horaire de 0,34 seconde.
- Alors cela a été assez loin pour que le Patriot puisse suivre et intercepter le Scud.

# Division du Pentium (1994)

# Division du Pentium (1994)

- Coût: 475 millions de dollars, la crédibilité des entreprises.
- Catastrophe: Le bug de la division du Pentium est un bug informatique ayant affecté le microprocesseur Pentium du fabricant Intel peu après son lancement en 1994 : une erreur était introduite lors de certaines opérations de division.
- Cause: l'initialisation incomplète d'une table de valeurs servant à la division, plus rapide.

# Division du Pentium (1994)

- Détails: en octobre 1994 le professeur Thomas Nicely dévoile un disfonctionnement de l'unité de calcul en virgule flottant du pentium. Il s'est rendu compte que certaines opérations de division renvoient toujours des valeurs erronée sur ce processeur. Ces erreurs ont été rapidement confirmé par d'autres personnes.
- Ce bug est appelé « bug FDIV du Pentium » où FDIV est l'instruction de division en virgule flottante des microprocesseurs x86.

# Division du Pentium (1994)

- L'erreur provenait de l'initialisation incomplète (dans le silicium) d'une table de valeurs servant à un nouvel algorithme de division, plus rapide.
- La présence de ce problème convient dans l'exemple suivant :

4 195 835,0 / 3 145 727,0 = 1,333 820 449 136 241 002 (valeur correcte),

4 195 835,0 / 3 145 727,0 = 1,333 739 068 902 037 589 (valeur retournée par le processeur).

# Traitement du cancer (2000)

- Coût: 8 morts, 20 blessés graves
- Catastrophe: Le logiciel de radiothérapie par us système internationale multi data a mal calculé les dosages approprié et en exposant les patients a des niveaux nocives et parfois mortelles de rayonnement.
- Les médecins qui sont responsables de vérifier les calculs sont condamnés .

# Traitement du cancer (2000)

- Cause :
  - Les doses de rayonnement calculées par le logiciel sont basé sur l'ordre dans lequel les données ont été saisie
  - parfois il y a saisie des doubles doses ce qui conduit a une erreur dans le calcul suivant.

# Les difficultés liées au GL

## La complexité intrinsèque d'un projet

- L’ ingénierie du logiciel est un métier récent en comparaison avec d’autres métiers
    - le fameux parallèle avec le bâtiment

## La nature du produit informatique

- De l'information ! copiable, modifiable, malléable, bref « soft »

# Les difficultés liées au GL

## Les difficultés liées à la nature du logiciel

- Un logiciel ne s'use pas, sa fiabilité ne dépend que de sa conception
- Mais, pour rester utilisé un logiciel doit évoluer
- Pas de direction clairement exprimée,
- Changements fréquents,
- Contradictions des besoins,...

# Les difficultés liées au GL

## Difficultés liées aux personnes

- ne savent pas toujours ce qu'elles veulent, ou ne savent pas bien l'exprimer
- communication difficile entre personnes de métiers différents ( jargons )
- l'informaticien est souvent perçu comme introverti, peu solidaire du groupe (...ça change...)
- beaucoup d ’autodidactes qui croient savoir...

# Les difficultés liées au GL

## Les difficultés technologiques

- courte durée de vie du matériel,
- beaucoup de méthodes, de langages
- évolution des outils de développement,...

# L’ échec d’un projet informatique

## Cinq raisons majeures :

- Engagements irréalistes
- Gestion et conduite de projet inadéquates
- Manque de contrôle
  - pas de planification de la part des développeurs
  - connaissances insuffisantes en gestion de projet
- Technologies inappropriées (méthodes, outils, langages)
- Validation et vérification insuffisantes

# Conduite de projet

# Production de logiciel

- Application ( Store ), site Web, Jeux, Microcontrôleur, etc.
  - Production => Code Source, Exécutable


- Participants ( Rôles )
  - Utilisateur
  - Développeur
  - 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

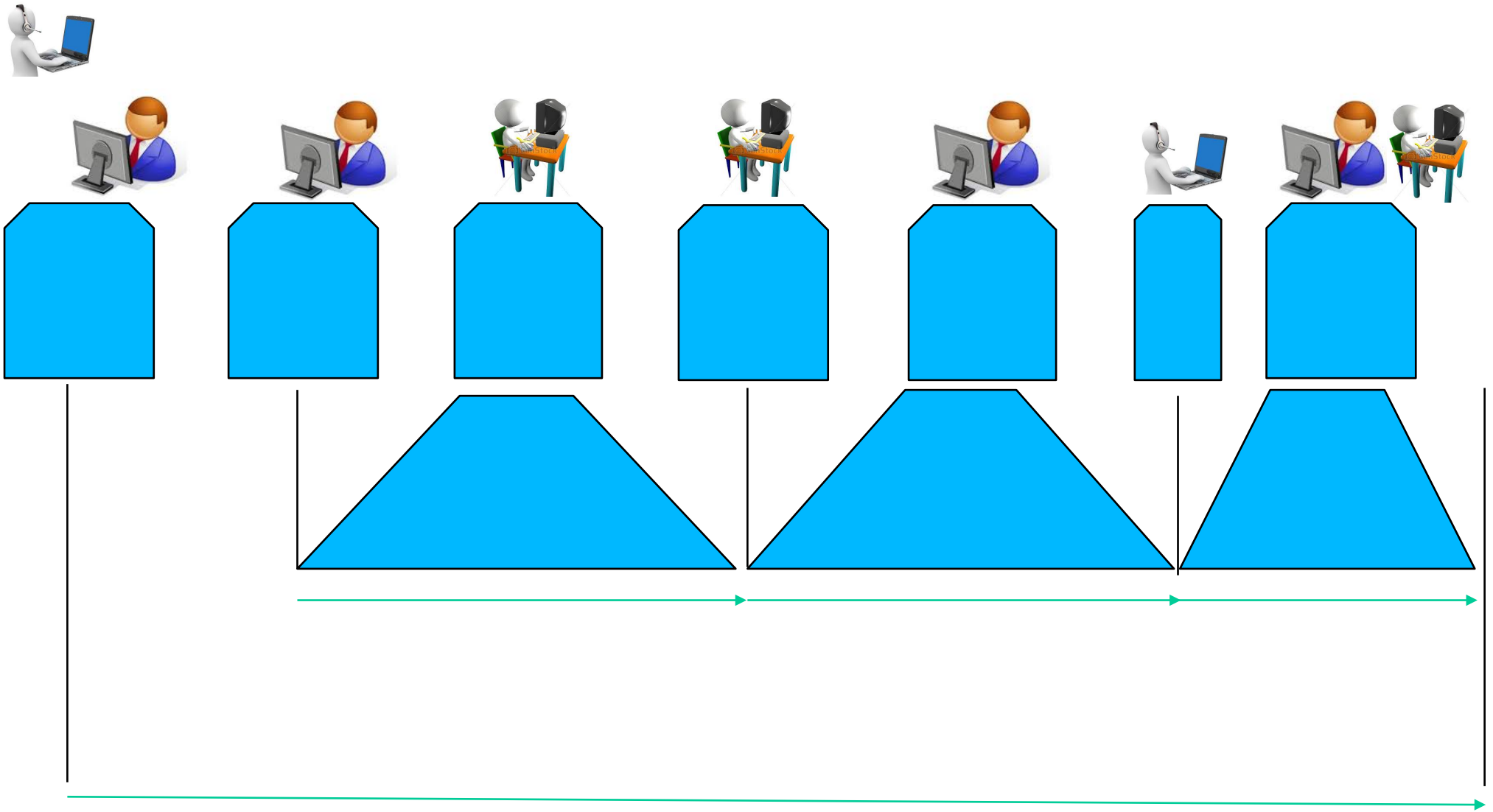
# Projet logiciel : Fin

8- L’**utilisateur** n’utilise plus le logiciel

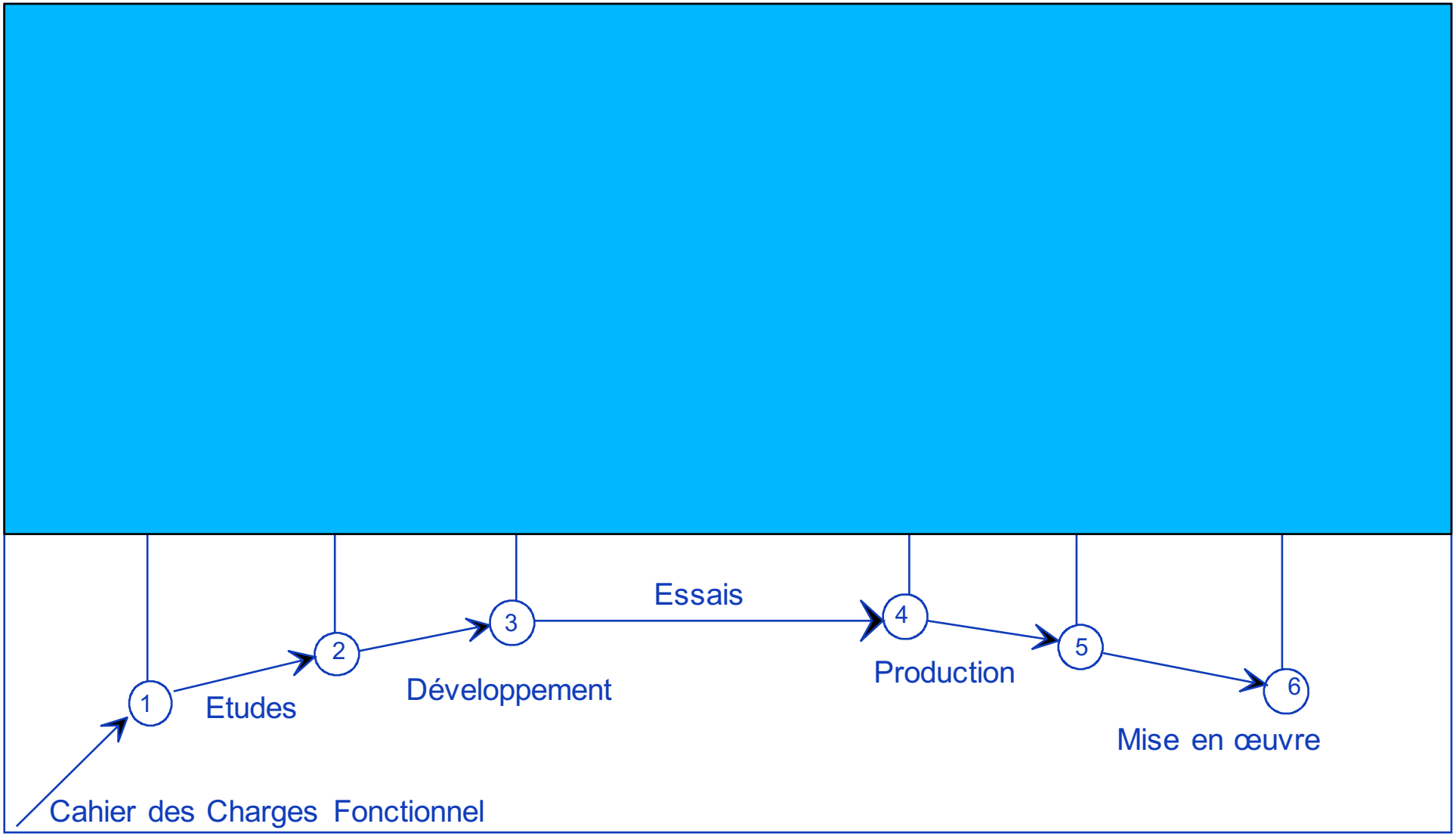
9- Ou le **propriétaire** arrête le logiciel

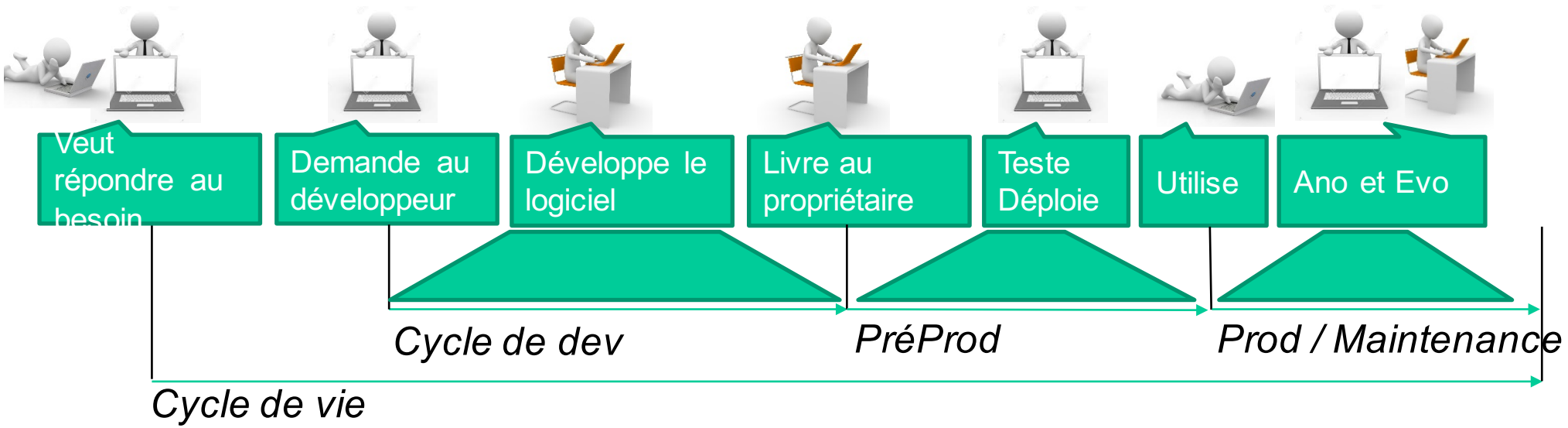
10- Le **développeur** ne sert plus à rien.

# Cycle de vie – Cycle de développement



# Plan de développement de projet...




Veut répondre au besoin
Demande au développeur
Développe le logiciel
Livre au propriétaire
Teste Déploie
Utilise
Ano et Evo
Cycle de dev
PréProd
Prod / Maintenance
Cycle de vie

# Différents types de projet

## Projet de mise en production

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

## Projet de développement

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

## TMA ( Pas vraiment un projet )

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

# Projet de développement (TMA – EVO)

L'utilisateur à 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 (Spéc )**

Le propriétaire accepte la proposition du développeur

**Cycle de Dév**

Une fois terminé, le développeur livre le logiciel au propriétaire

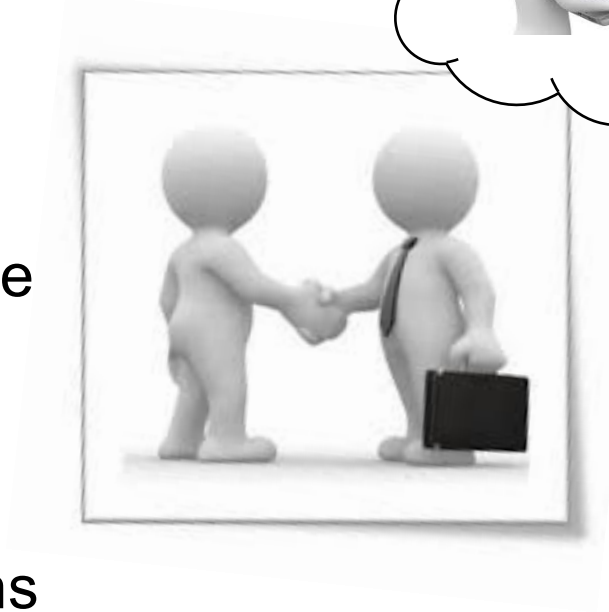
**Livraison**

# Participants au projet de développement

MAO ( Maitrise d'Ouvrage) : celui qui possède

le résultat du projet, et qui paye

=> **Le propriétaire**



MOE ( Maitrise d'Œuvre ) : celui qui réalise

le logiciel

=> **Le développeur**

L'utilisateur n'apparaît pas réellement dans

le projet ( 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

Les enseignants chercheurs ont besoin d'un site web pour organiser une manifestation scientifique

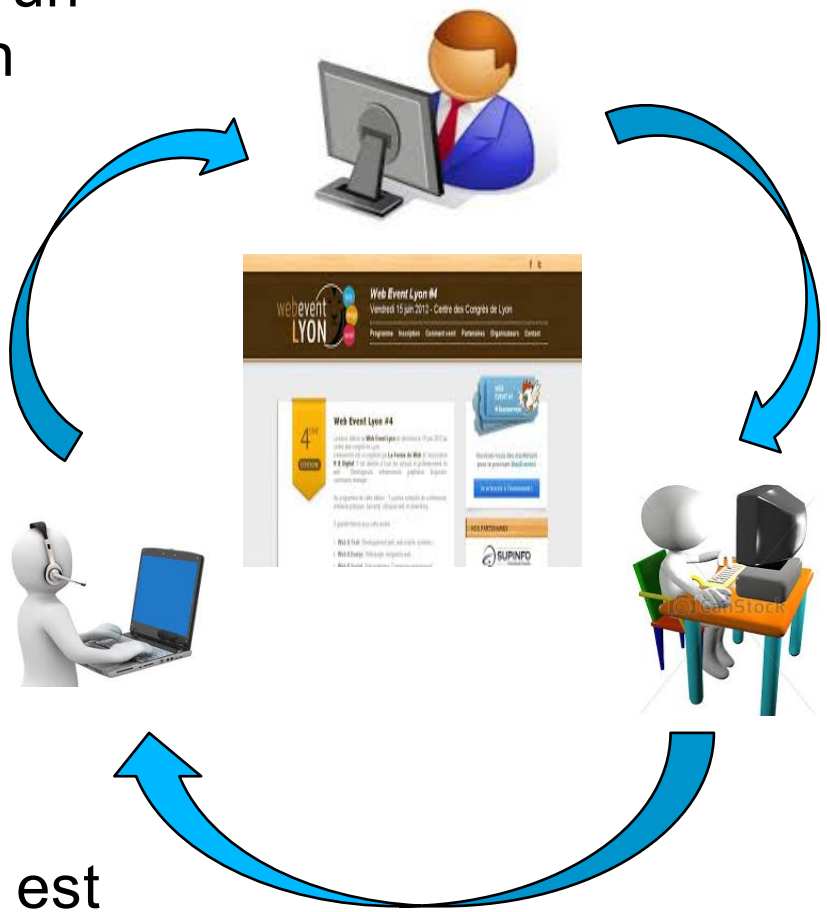
**=> Utilisateurs : Enseignants**

L' ENSAS veut répondre à ce besoin en demandant à plusieurs sociétés de développer ce site web

=> **Propriétaire : ENSAS**

La société SARL-X réalise le site web qui est utilisé par les enseignants chercheurs

=> **Développeur : SARL-X**



# Les 3 grandes phases du cycle de Dév

**Analyse**

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

**Conception**

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

**Développement**

La MOE développe les composants définis dans la conception et livre le résultat

# 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 développer

# LE CHIFRAGE

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

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

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

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 écran simple
  - Nombre de ligne de code développé en moyenne par développeur = 150
  - Une classe d'une complexité moyenne nécessite donc au moins un jour

**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 taches permettant la construction de ces modules

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

# 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 leurs interactions

- Interface, Protocole, etc.

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

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

- RAF : Reste A Faire

Cela permet aussi d'organiser le travail dans une équipe

Et de maitriser les délais et les risques

# LE MODELE 3-TIERS

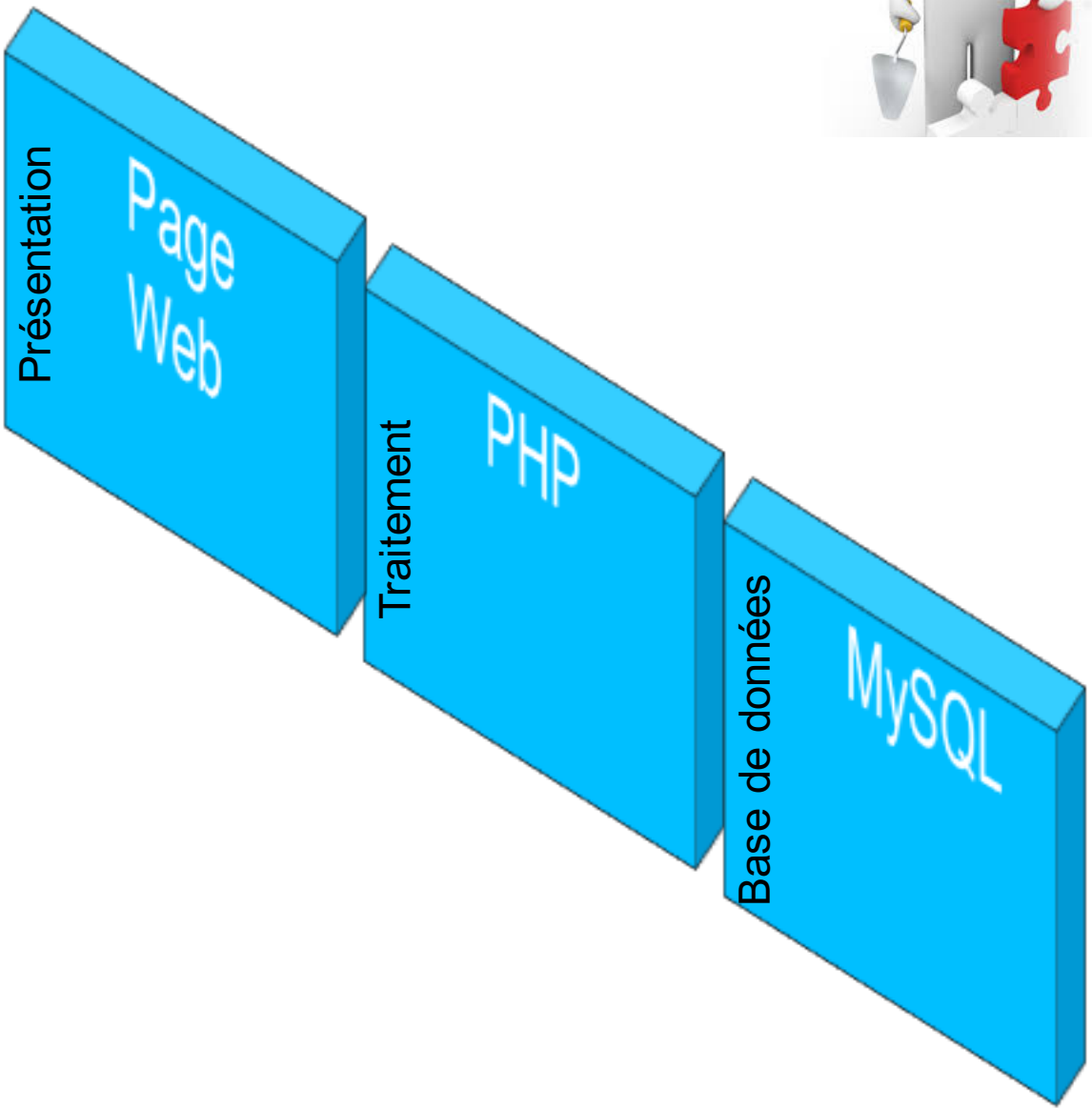
Trois modules en couche

- Base de données
- Traitement
- Présentation

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

Tâches

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



# LA CONCEPTION

Qui : la MOE

Entrée : La définition des modules et la liste des tâches

Objectif : coder les modules et livrer le logiciel

Sortie : Le logiciel

# 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?

# Le CYCLE EN CASCADE

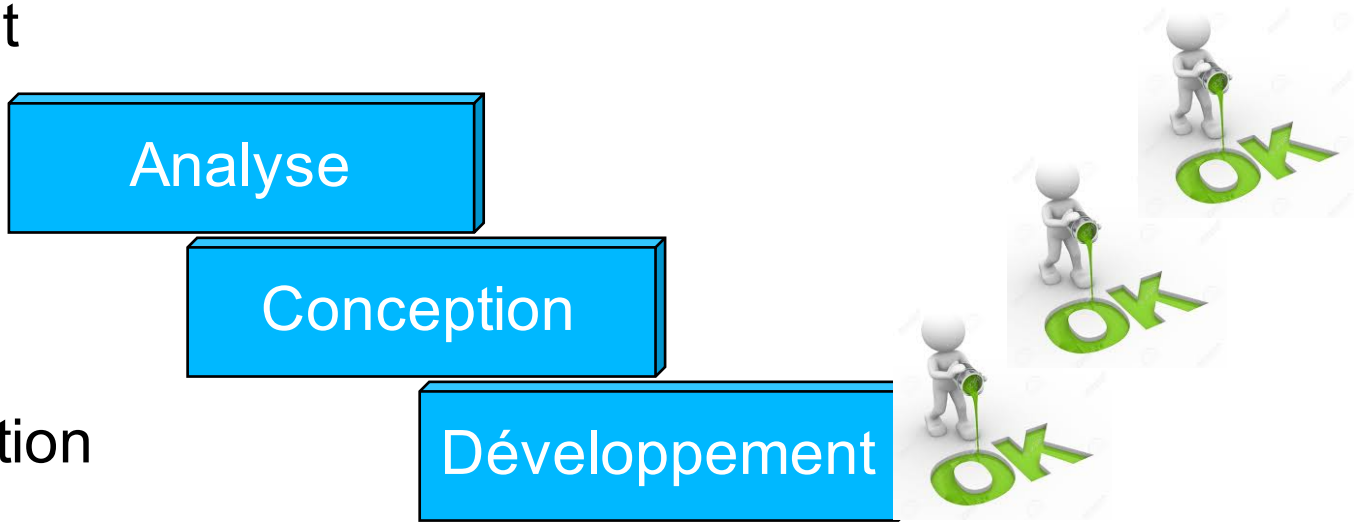
Analyse

Conception

Développement

# Les tests : CYCLE EN V

Tester le logiciel pendant tout le cycle de développement




Selon les phases :

- Analyse : test de validation

- Conception : test d'intégration

- Dev : test unitaire

Rédaction des tests != faire passer les tests

# L’ITETRATION : CYCLE EN SPIRALE

Les phases d’Analyse, Conception et Codage doivent être réalisées les unes après les autres

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

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

# L’AGILITE

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 et Utilisateur

Cycle de dev

- Analyse, conception et Dev

Itération (Cycle en spiral)

Test (Cycle en V)

Agilité

**Conduite de Projets –(Réponse au cahier des charges)**

Le cahier des charges du CNRS

Lisez le cahier des charges du CNRS et considérez que vous êtes une jeune start-up dont le business consiste à réaliser des projets de développement.

Quelles sont les parties du cahier des charges que votre start-up est capable de réaliser ?

Quelles sont les parties du cahier des charges que votre start-up n'est pas capable (ou ne souhaite pas) réaliser ?

Proposez, à l'aire d'un outil de dessin (PowerPoint, Paint, utilisez ce que vous voulez), une maquette de ce que vous comptez réaliser.

Chiffrez en jh, le coût de réalisation de cette maquette (on parlera de coût brut de réalisation) Proposez un prix de vente de votre réalisation ainsi qu'une date de livraison.

**A faire:** La maquette, le chiffrage brut et le prix de vente.