

# COURS DE CONDUITE DE PROJETS INFORMATIQUES

UNB/ESI/ISI/L3 – 2023/2024

## « PROBLÉMATIQUE DE CONDUITE DES PROJETS DE GÉNIE LOGICIEL »

Comment réussir la conduite de projet de génie logiciel ?

3<sup>ème</sup> partie

# AGENDA DE CETTE 3<sup>ÈME</sup> PARTIE

1. Modèles de développement des méthodes agiles
2. Exposés & Regard sur vos projets de fin de cycle de formation
3. Quelques techniques d'estimation des coûts et des délais
4. Techniques de planification et des durées
5. En conclusion ...



# 1

## MODÈLES DE CYCLE DE VIE DES MÉTHODES AGILES

Prof. THIOMBIANO



# CYCLE DE VIE DES MÉTHODES AGILES

- Cycle de vie itératif (évolutif mais avec notion d'itérations)
  - Le projet est décomposé en itérations
  - On applique un cycle de type (Planifier – Développer – Contrôler – Ajuster) sur chaque itération
- Chaque itération est courte
  - **Ne dépassant pas les huit semaines le plus souvent.**
- L'idée est de livrer au plus tôt quelque chose qui puisse être testé par le client.
- Chaque itération est testée par le client et les retours sont pris en charge pour la prochaine itération



# CYCLE DE VIE DES MÉTHODES AGILES

---

- Cycle de vie semi-itératif
  - Deux premières phases « classiques » d'expression des besoins et de conception de la solution.
- Troisième grande phase de construction du produit **par des itérations courtes**
- Plusieurs méthodes connues reposent sur ces deux derniers types de cycle
  - **LES METHODES AGILES**



# MODÈLE DE DÉVELOPPEMENT

---

- Difficile d'avoir aujourd'hui une démarche unique: il faut construire le découpage temporel en fonction des caractéristiques de l'entreprise et du projet.
- D'où l'usage des découpages temporels génériques ou modèle de développement ou modèles de cycle de vie.
- Pour le développement itératif, on parle de modèle de développement évolutif ;



# PROBLÉMATIQUE DES MÉTHODES DITES AGILES

---

## Comment prendre en compte le facteur humain dans le management de projet ?

En effet, l'agilité s'adresse à des êtres humains qui fabriquent des produits destinés à d'autres êtres humains.

Les chefs de projet d'ingénierie logicielle ont souvent cherché à concilier le besoin de définir le plus précisément possible l'application à développer et l'évolution continue, parfois contradictoire, des besoins utilisateurs



# Les méthodes agiles remettent le client au cœur du projet.

**L'on n'est plus collé exclusivement aux  
lignes des annexes des contrats qui  
définissent clairement les  
spécifications fonctionnelles à réaliser  
à la lettre.**





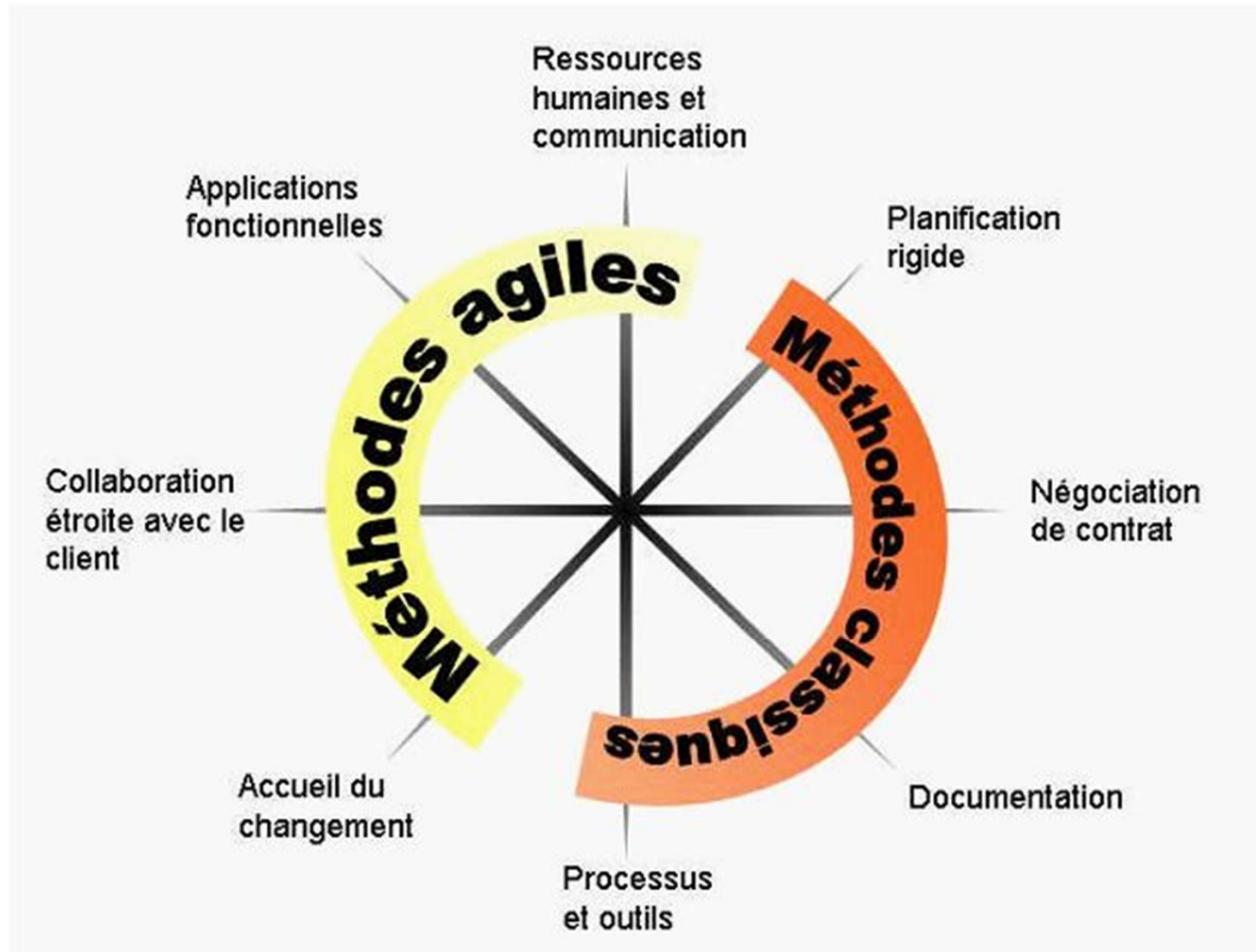
# MÉTHODES DITES AGILES

---

- Sont des méthodes de développement et de gestion de projet, qui se caractérisent par leur pragmatisme et leur adaptabilité au contexte.
- Sont nées en réaction aux méthodes dites classiques et jugées trop lourdes et limitatives et pour pallier aux risques d'échecs récurrents bien connus.



# MÉTHODES AGILES VS MÉTHODES CLASSIQUES



# METHODES AGILES

---

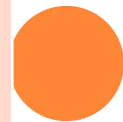
Les phases sont simplifiées  
afin d'en raccourcir la  
durée;

Les développeurs et les  
utilisateurs sont présents  
tout au long du processus. ●

# LES 12 PRINCIPES DES MÉTHODES AGILES

N°	Principes	Commentaires
1	Satisfaire en livrant tôt et régulièrement	C'est notre quotidien
2	Accepter les changements tardifs comme un avantage	C'est récurrent
3	Livrer fréquemment une application opérationnelle	Le logiciel doit marcher
4	Collaborer quotidiennement	Pas évident
5	Faire confiance et supporter	Pas évident
6	Transmettre les informations face à face	L'utilisateur n'aime pas tout écrire
7	Mesurer l'avancement avec le logiciel réalisé	Pas évident: mauvaise organisation du projet

Prof. THIOMBIANO



# LES 12 PRINCIPES DES MÉTHODES AGILES

N°	Principes	Commentaires
8	Avancer à un rythme de développement durable	Oui. Le logiciel est évolutif
9	Porter une attention continue à l'excellence technique	C'est notre passion
10	Faire simple	Pas évident
11	S'auto-organiser	C'est le quotidien
12	Réfléchir à ses pratique et les ajuster régulièrement	Pas évident

Prof. THIOMBIANO

Nous faisons de l'AGILE, mais très difficilement et pas toujours dans la rigueur qu'il faut.

**L'important: UN LOGICIEL QUI FONCTIONNE !!!**

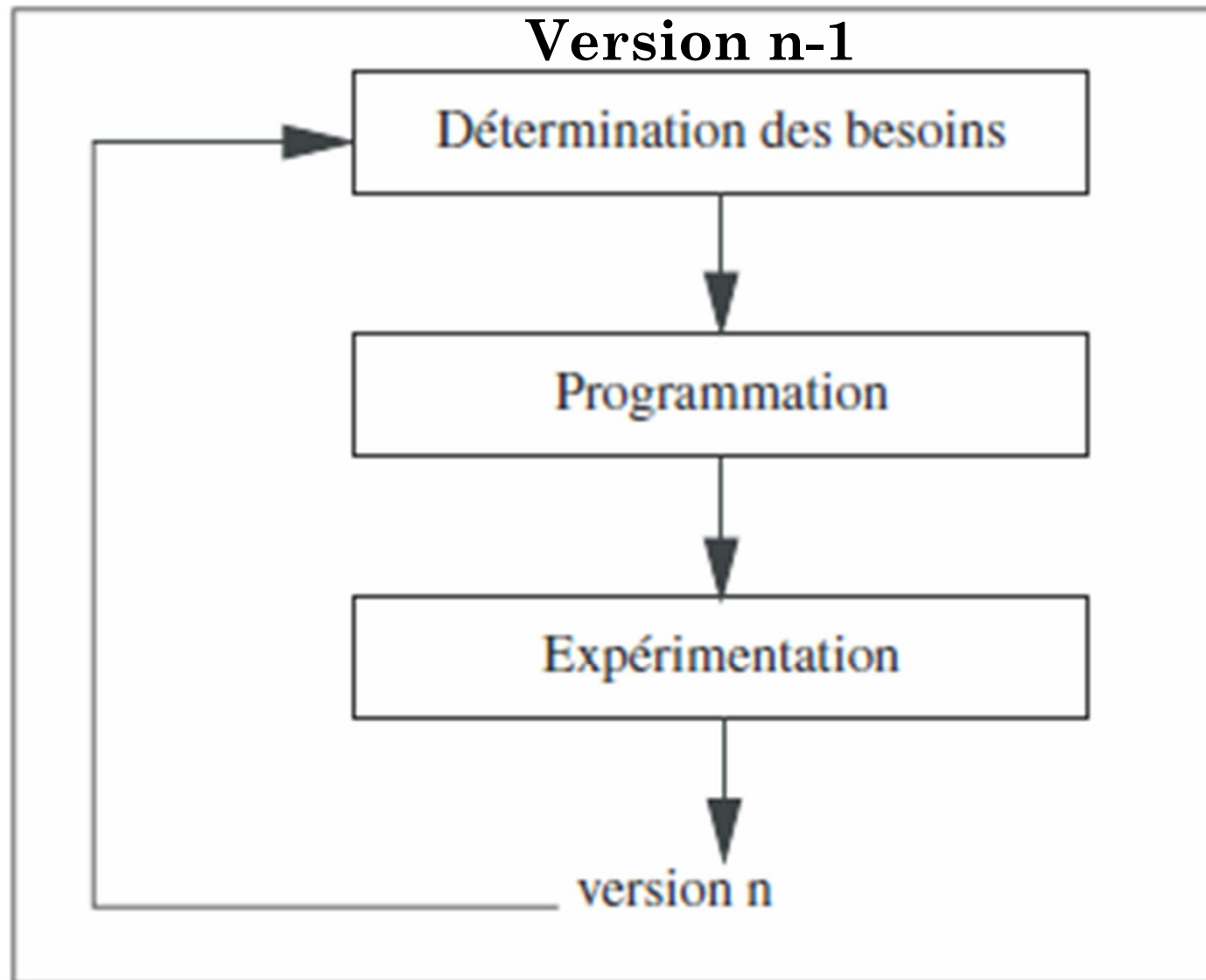
# MD: DÉVELOPPEMENT ÉVOLUTIF

---

- Construction progressive du système de façon participative.
- Principe: les besoins ne peuvent s'exprimer qu'après une expérimentation, même sur un système rudimentaire ou incomplet.
- On s'arrête lorsque le client juge le système satisfaisant.
- NB/ Tout cycle itératif n'est pas forcément incrémental. La 1<sup>ère</sup> itération peut livrer un système complet qui est ensuite affiné et ajusté dans les itérations suivantes.



# MD: DÉVELOPPEMENT ITÉRATIF

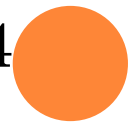


# MÉTHODES DITES AGILES

---

## ○ La méthode SANDROSE

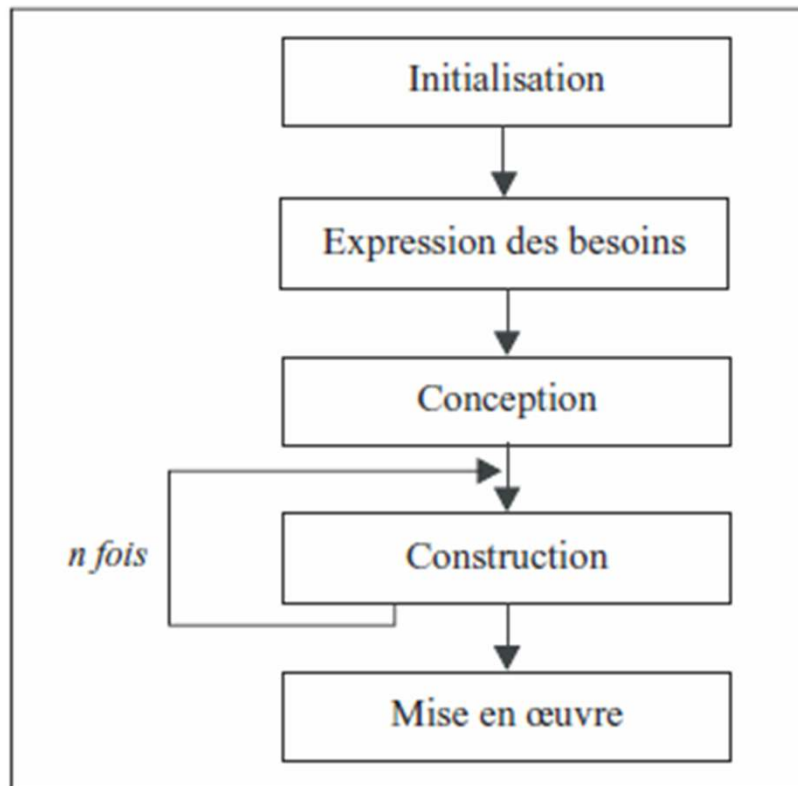
- L'objectif est la réussite de projets informatiques, une réussite du point de vue clients comme de celui des informaticiens
- Méthode qui a été utilisée avec succès au Burkina Faso pour la construction du SI informatisé de son Ministère des Finances (SIGASPE ou paie; CIE ou trésor et compta; CID ou budget)
- Englobe: la conduite de projet, la réalisation de logiciels, le déploiement et l'exploitation.
- Est qualifié d'extrêmement AGILE = n-AGILE
- Suit les recommandations de la norme ISO 15504



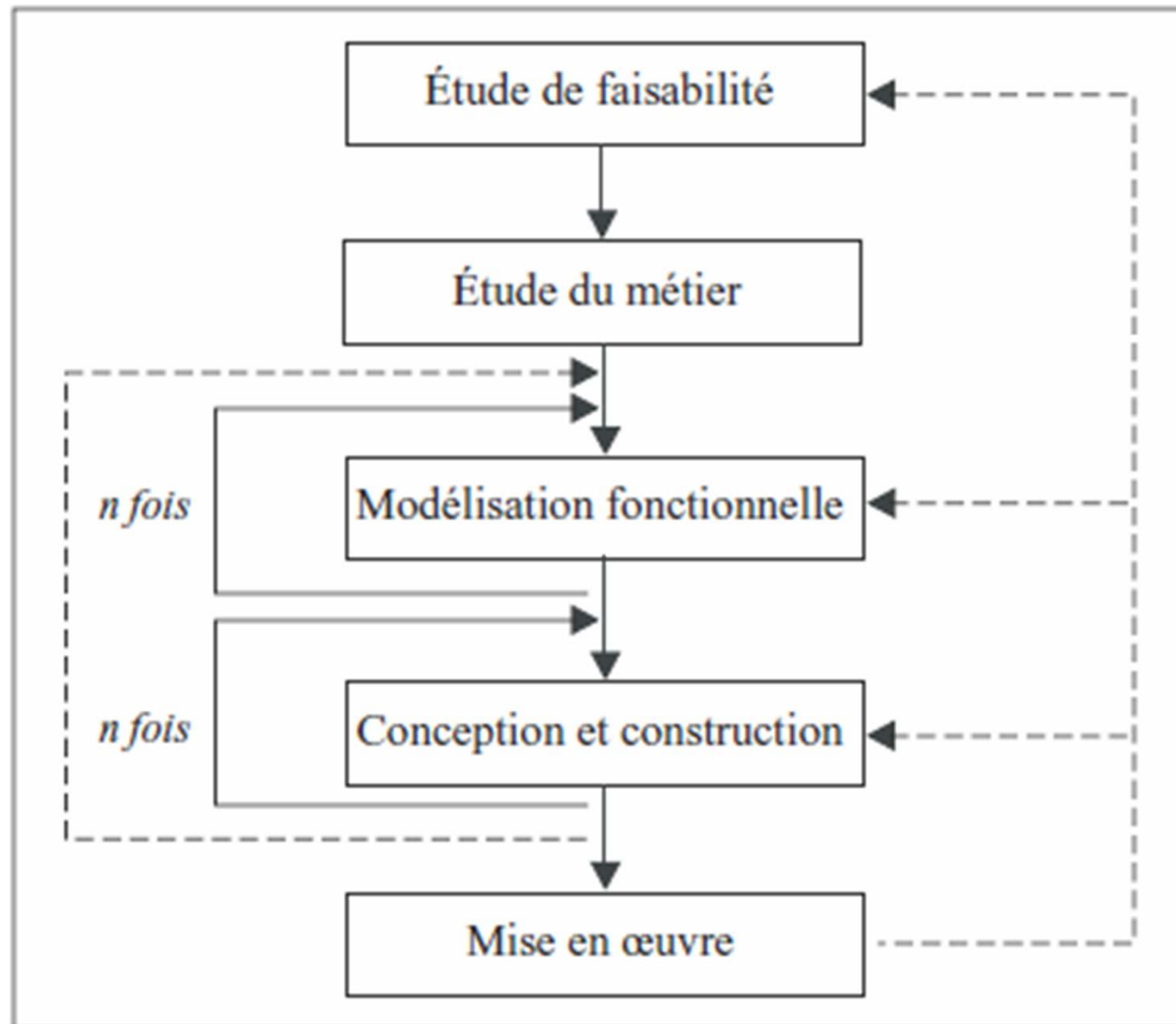


# MD: RAD

- RAD = Rapid Application Development  
(Développement rapide des applications)



# MD: DSDM



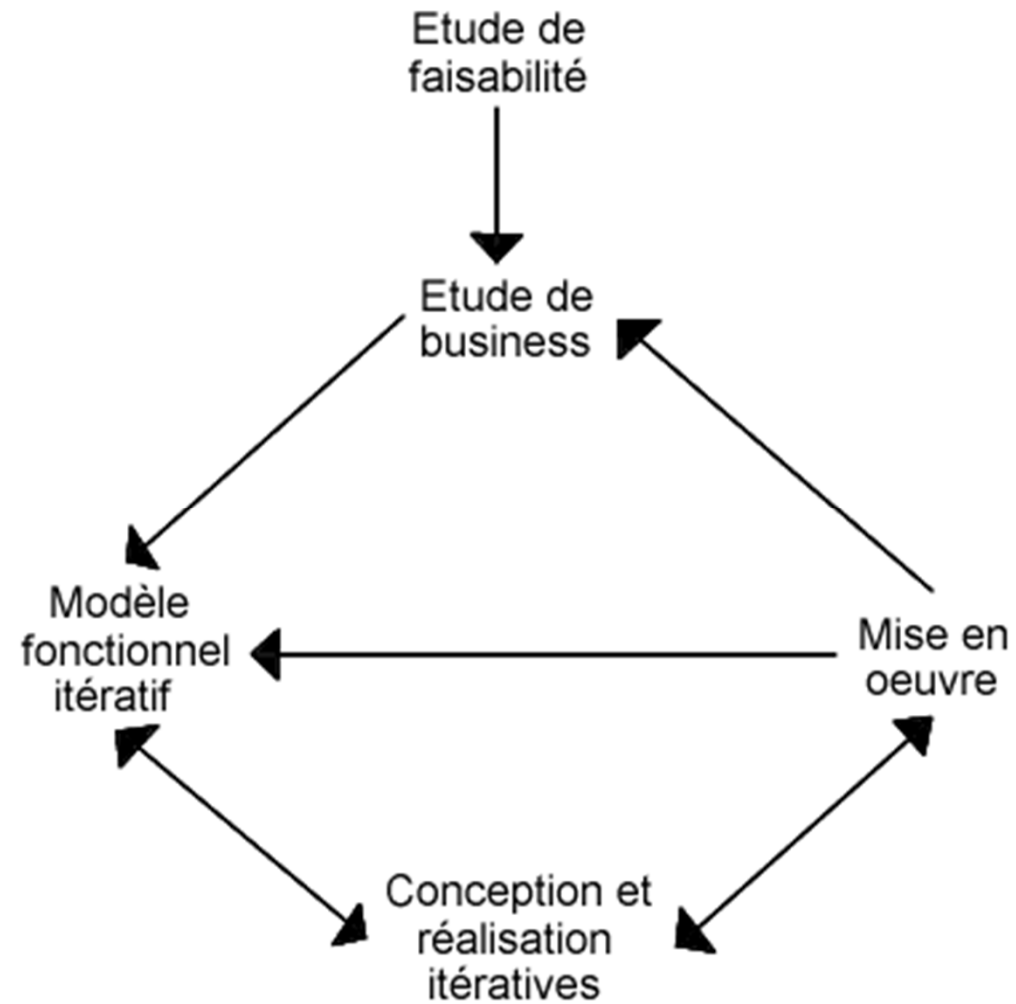
# DSDM: DYNAMIC SYSTEM DEVELOPMENT METHOD

---

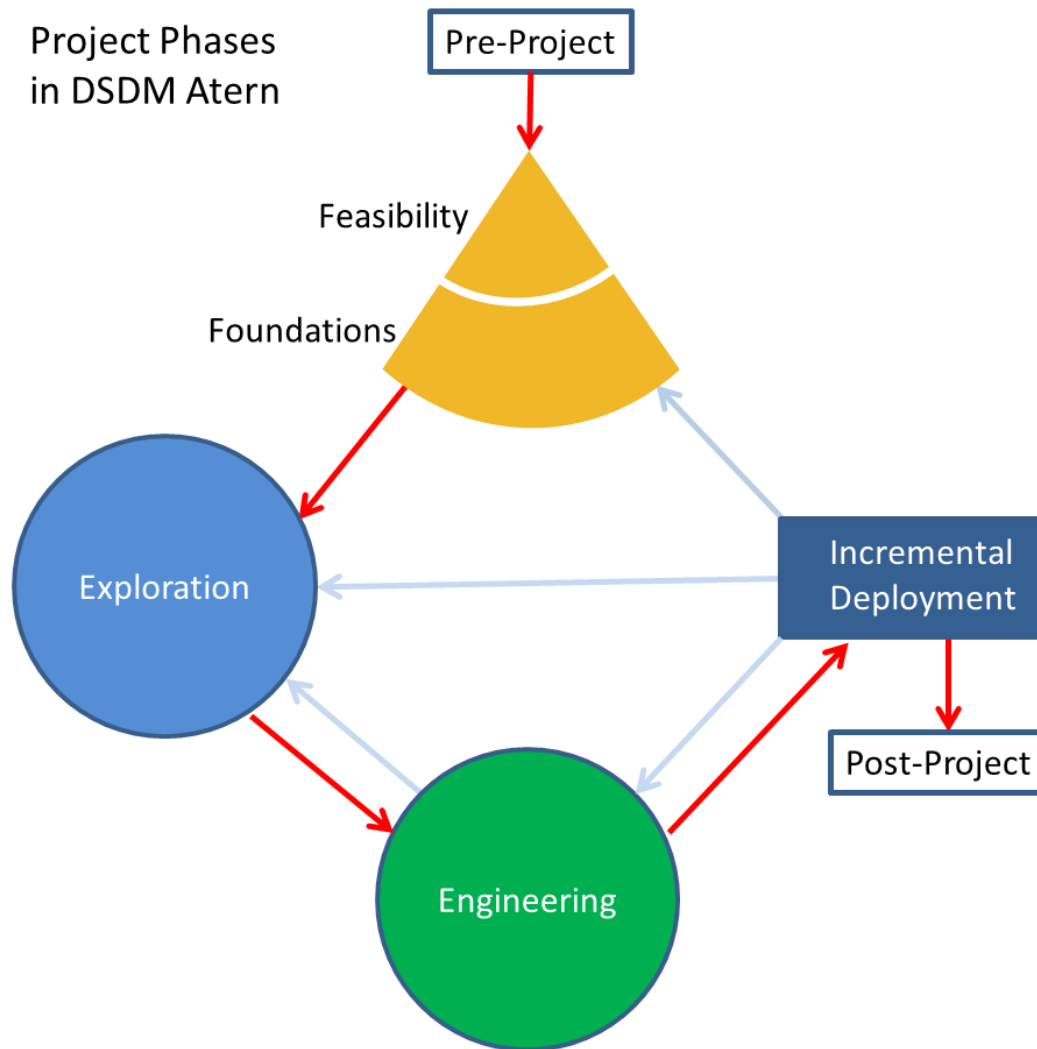
- Méthode de développement des systèmes dynamiques
- En 1994, elle agissait comme une méthode RAD
- Elle a évolué et a consolidé ce côté itératif et incrémental qui englobe le principe de développement agile avec la participation continue utilisateur / client.



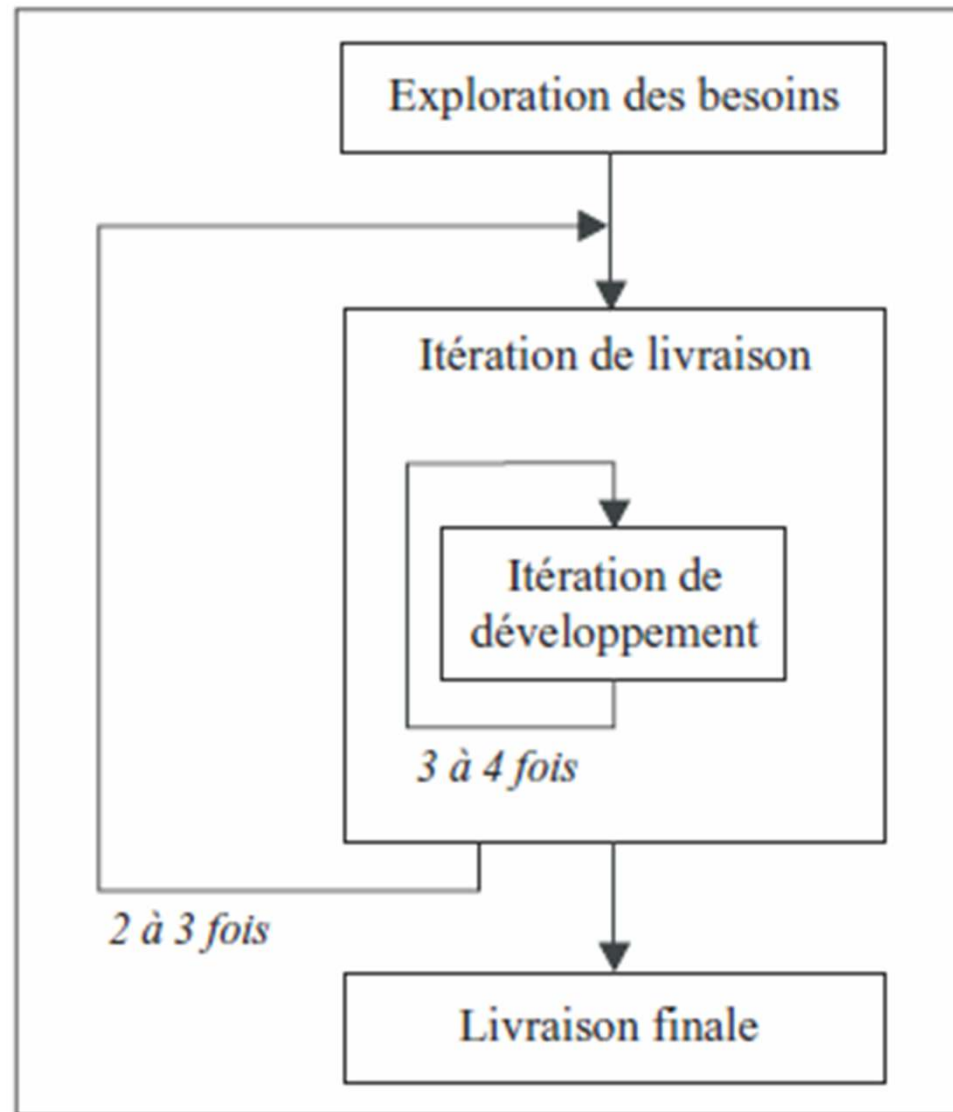
# MD: DSDM



# MD: DSDM



# MD: XP



# QUE DIRE DES NOUVELLES MÉTHODES ?

---

Les nouvelles méthodes telles que **ITIL** (gestion de la production) et **COBIT** (gouvernance), et **surtout les méthodes agiles**, ont pour but de décliner le projet sous forme de petits livrables qu'on peut déployer rapidement: accélère la mise en œuvre, adapte l'application à l'évolution des besoins métier tout au long de la phase de mise en place.



# 2

## REGARD SUR VOS PROJETS DE FIN DE CYCLE DE FORMATION

Prof. THIOMBIANO





UP,  
2TUP,  
SCRUM

...



# OBSERVATIONS SUR LA MÉTHODE CHOISIE PAR LES ÉTUDIANTS ...

---

**Les rapports** matérialisent : usage de la **méthode UP**  
(unified process) ou **2TUP** (Two Track Unified Process) ou  
... et du **langage de modélisation UML**.

Finalelement, dans tous le document, on constate  
simplement que c'est le formalisme UML qui est employé  
par les méthodes et le langage même n'est pas utilisé  
dans la solution (réalisation).

Quel commentaire vous pouvez faire?



# OBSERVATIONS SUR LA MÉTHODE CHOISIE PAR LES ÉTUDIANTS ...

---

- ❑ Rappel: **METHODE = DEMARCHE + FORMALISME**
- ❑ **Les rapports** matérialisent : usage de la **méthode UP**  
(unified process) ou **2TUP** (Two Track Unified Process)  
ou ... et du **langage de modélisation UML**.
- ❑ Finalelement, dans tous le document, on constate quoi ?
  - ❑ La démarche est quelque fois occultée
  - ❑ Le formalisme est utilisé, mais pas jusqu'au bout. En exemple:  
le langage n'est pas utilisé dans la solution (réalisation).

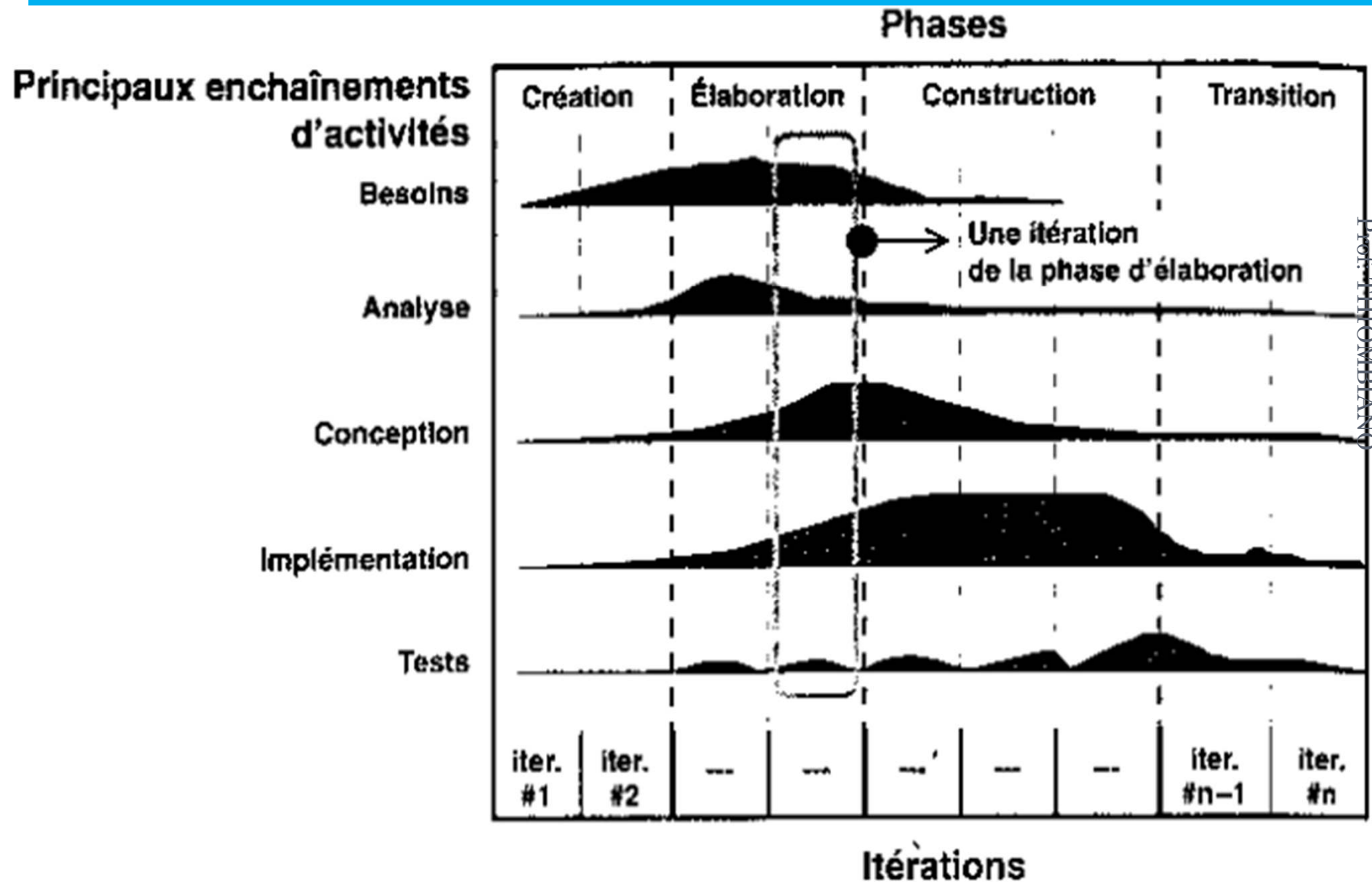


# CYCLE DE VIE DE UP

- Le processus unifié (Unified Process ou UP) est basé sur des composants. Il utilise UML et est basé sur les cas d'utilisation, l'architecture et le développement incrémental
- UP est une méthode générique de développement de logiciel développée par les concepteurs d'UML
  - Générique signifie qu'il est nécessaire d'adapter UP au contexte du projet, de l'équipe, du domaine et/ou de l'organisation.
  - Il existe donc un certain nombre de méthodes issues de UP comme par exemple RUP (Rational Unified Process), 2TUP (Two Track Unified Process)
  - Il existe d'autres approches (qui ne sont en général pas complètement antinomique), comme par ex. les méthodes « agile », beaucoup moins orientées modèle, comme XP (eXtreme Programming)



# CYCLE DE VIE DE UP



# CYCLE DE VIE DE UP : CREATION

---

- Traduit une idée en vision de produit fini et présente une étude de rentabilité pour ce produit
  - Que va faire le système pour les utilisateurs ?
  - A quoi peut ressembler l'architecture d'un tel système ?
  - Quels sont l'organisation et les coûts du développement de ce produit ?
- On fait apparaître les principaux cas d'utilisation.
- L'architecture est provisoire, identification des risques majeurs et planification de la phase d'élaboration.



# CYCLE DE VIE DE UP : ELABORATION

---

- Permet de préciser la plupart des cas d'utilisation et de concevoir l'architecture du système. L'architecture doit être exprimée sous forme de vue de chacun des modèles.
- Emergence d'une architecture de référence.
- A l'issue de cette phase, le chef de projet doit être en mesure de prévoir les activités et d'estimer les ressources nécessaires à l'achèvement du projet.

# CYCLE DE VIE DE UP : CONSTRUCTION

---

- Moment où l'on construit le produit. L'architecture de référence se métamorphose en produit complet, elle est maintenant stable. Le produit contient tous les cas d'utilisation que les chefs de projet, en accord avec les utilisateurs, ont décidé de mettre au point pour cette version.
- Celle ci doit encore avoir des anomalies qui peuvent être en partie résolue lors de la phase de transition.



# CYCLE DE VIE DE UP : TRANSITION

---

- Le produit est en version bêta. Un groupe d'utilisateurs essaye le produit et détecte les anomalies et défauts.
- Cette phase suppose des activités comme la fabrication, la formation des utilisateurs clients, la mise en œuvre d'un service d'assistance et la correction des anomalies constatées.(où le report de leur correction à la version suivante)



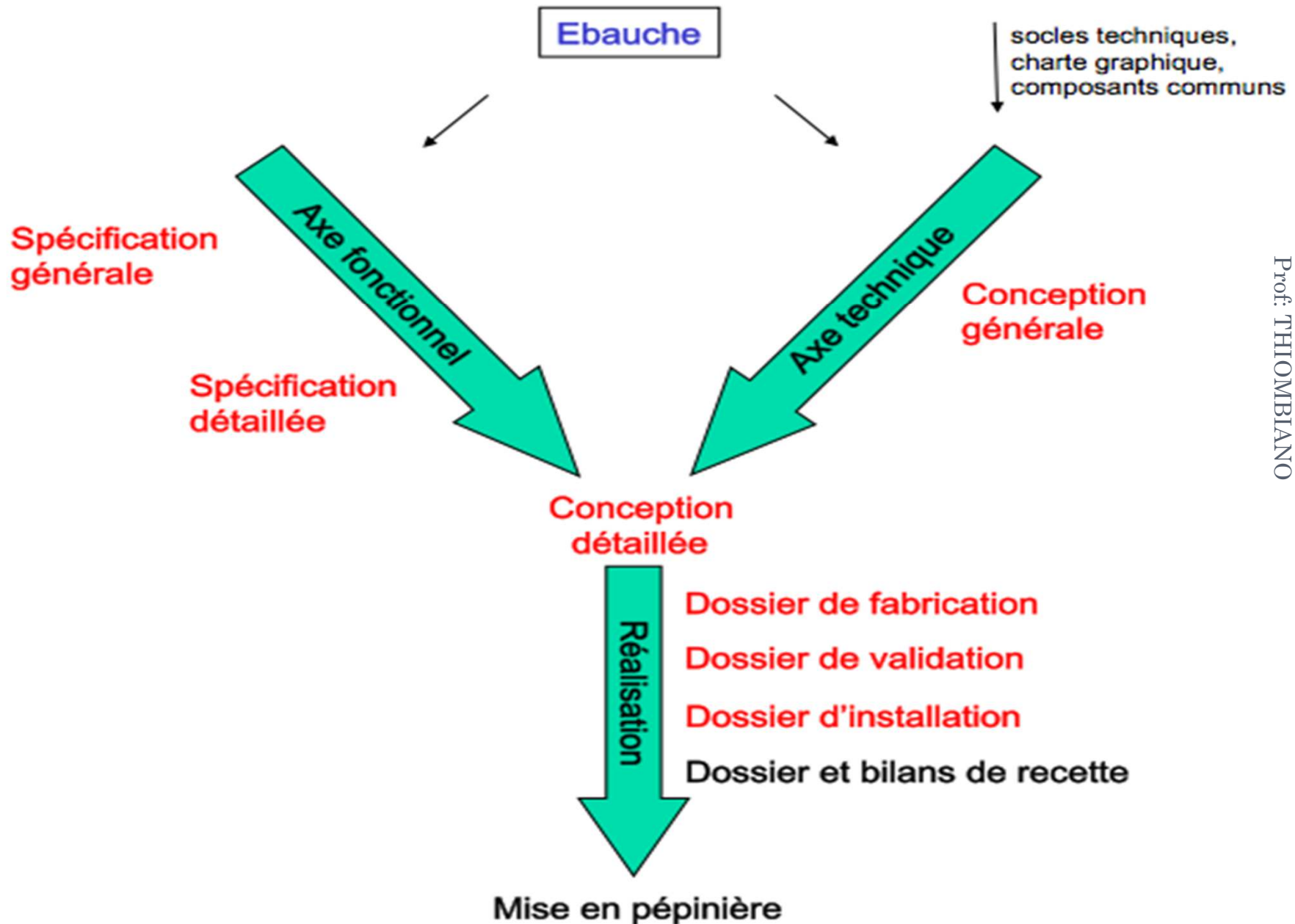
# CYCLE DE VIE DE UP : TRANSITION

---

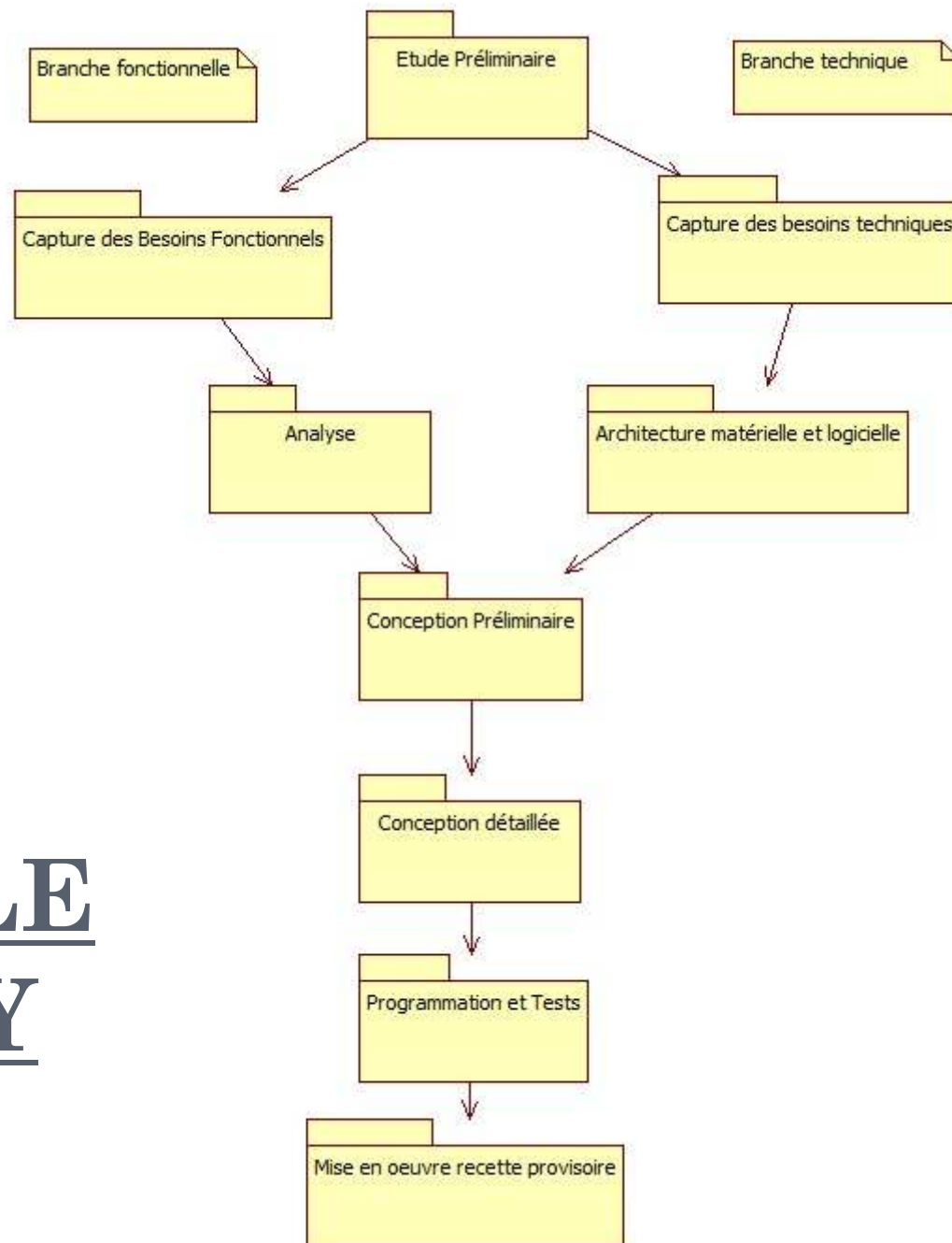
- Le processus unifié
  - utilise le langage UML
  - est piloté par les cas d'utilisation
  - Centré sur l'architecture
  - Itératif et incrémental



# REGARD SUR LA MÉTHODE 2TUP



# CYCLE EN Y



MD: 2TUP

---

# Déroulons un projet

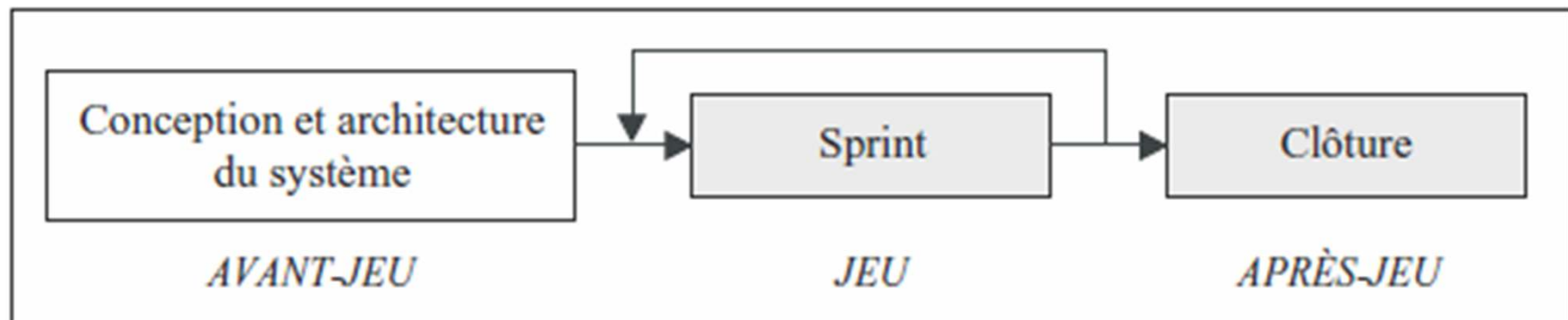
Prof. THIOMBIANO



# MD: SCRUM

- ❑ SCRUM a été présentée en 1990 par Jeff McKenna et Ken Schwaber, et est appliquée
- ❑ C'est un cadre de travail permettant de répondre aux problèmes complexes et changeants.
- ❑ Elle favorise le développement productif et créatif de produits de grande valeur.

Prof. THIOMBIANO



# MD: SCRUM – LES 3 PILIERS

---

- ❑ **Transparence:** Elle requiert la définition d'un standard commun pour les aspects importants du processus afin que les observateurs partagent la même compréhension de ce qui est observé. Exemple: langage de description des processus, notion de « terminé », etc.
- ❑ **Inspection:** il faut fréquemment inspecter les artéfacts scrum et l'état d'avancement par rapport à un objectif de sprint afin de dégager les écarts indésirables.
- ❑ **Adaptation:** il s'agit des ajustements à apporter dès que des dérives sont constatés.



# MD: SCRUM – LES ELEMENTS

---

## ❑ L'équipe Scrum et les rôles associés qui sont:

- ❑ **Product owner:** le propriétaire du produit. C'est l'expert métier, qui joue le rôle de client et des utilisateurs. Il définit les fonctionnalités du carnet du projet (Product Backlog) et les priorise.
- ❑ **Scrum master:** le maître Scrum. Il joue le rôle de facilitateur et gardien de la bonne application, et veille donc à la mise en œuvre de l'agilité tout en agissant sur le processus de développement. On peut dire que c'est le chef de projet.
- ❑ **Development team:** L'équipe de développement. Elle est chargée de la réalisation du produit, et composée de 3 à 9 développeurs.





# MD: SCRUM – LES ELEMENTS

---

## ❑ Les événements

### ❑ Le Sprint

### ❑ La réunion de planification de sprint: Sprint Planning Meeting

### ❑ La mêlée: Daily Scrum

### ❑ La revue: Sprint Review Meeting

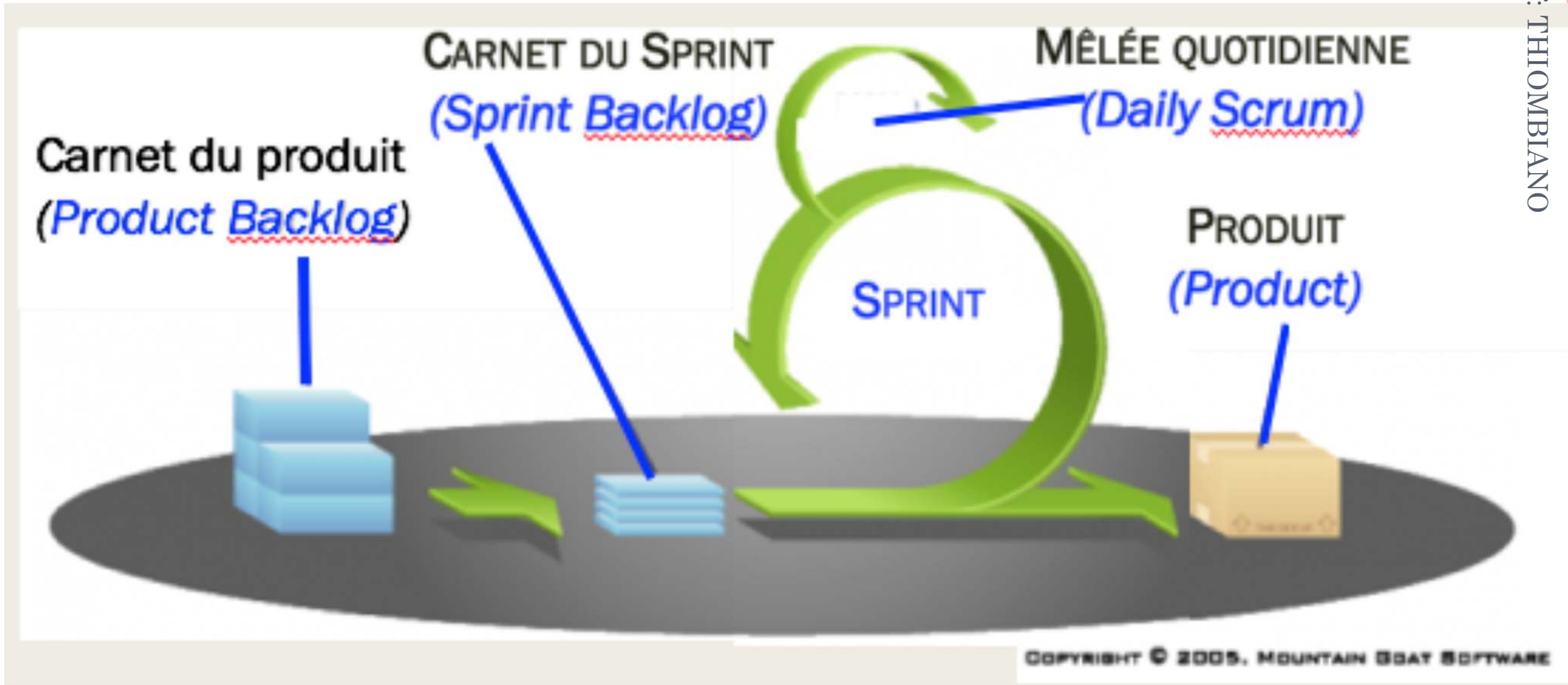
### ❑ La rétrospective du Sprint: Sprint Retrospective

❑ NB/ Les événements sont limités dans le temps. Chaque événement est une occasion pour inspecter et adapter.

# MD: SCRUM – QUE DIRE DU SPRINT ?

- **Un Sprint** est défini pour réaliser un objectif de l'activité de développement liée à la réalisation du produit attendu. Il a une durée bien limitée (moins d'un mois).
- Il contient et est constitué de la **planification du Sprint** (*Sprint Planning*), des **mêlées quotidiennes** (*Daily Scrums*), des **activités de développement**, de la **revue du Sprint** (*Sprint Review*) et de la **rétrospective du Sprint** (*Sprint Retrospective*).
- Le sprint a un objectif fixe auquel est associée une liste d'éléments du *Product backlog*, ce but est sans changements qui le remettent en cause. Les objectifs de qualités sont maintenus. Son périmètre peut être renégocié par la *product owner* et la *development team*.
- Les Sprints amènent de la *prévisibilité* en forçant une *inspection* et *adaptation* du progrès vers l'atteinte d'un objectif au moins mensuellement.
- Les sprints se **déroulent de façon séquentielle**.





# MD: SCRUM – L'ÉLÉMENT ARTÉFACTS

---

Les artéfacts de Scrum sont conçus pour maximiser la transparence d'informations essentielles.

Les différents artéacts sont:

- ❑ **Product Backlog:** c'est le carnet du produit
- ❑ **Sprint Backlog:** c'est le carnet du sprint
- ❑ **Increment:** c'est l'incrément
- ❑ **User Story:** ce sont les fonctionnalités
- ❑ **Sprint Burndown Chart:** c'est le diagramme de progression



# MD: SCRUM – EN SYNTHÈSE

- Scrum fonctionne en tant que conteneur pour d'autres techniques, méthodologies et pratiques.
- Ses éléments (*rôles, événements, artéfacts et règles*) sont immuables.
- Scrum ne fonctionne que dans son intégralité. Même si elle peut être utilisée partiellement (*ce n'est pas alors Scrum*).
- La notion de « **terminé** » doit être bien perçue par tous les membres de l'équipe Scrum.
- La **Transparence** des artéfacts est essentielles dans Scrum. Elle permet d'optimiser la valeur du travail réalisé et contrôler les risques.
- La transparence ne se produit pas du jour au lendemain, elle correspond plutôt à **un cheminement**.



**Voyons son  
application en  
entreprise !**



## VOS PROJETS PENDANT LE STAGE ?

---

**Que faites vous  
quand vous êtes en  
stage ? Générez-  
vous des itérations  
qui donnent des  
extrants ?**



## EN RAPPEL: PROJET AVANT-GARDE ....

---

- Avant de démarrer le processus proprement dit du processus de développement d'un logiciel, il y a le projet avant-garde.
- Définition de tous les contours du projet dont les trois éléments de suivi du projet:
  - La dérive des délais,
  - La dérive des dépenses (en fonction du budget initial),
  - et de la dérive du périmètre fonctionnel au regard d'une cible définie en amont. => c'est notre problème. On ne fait pas attention à ce point et on se plaint toujours des délais et des coûts.
- **Au terme de cette étude, le projet est lancé.**





# PROCESSUS DE DÉVELOPPEMENT: ETAPES STANDARD

N°	Etapes	Commentaires
1	Objectif ou lettre de mission ou cahier de charges	L'informaticien reçoit une mission
2	Etude de faisabilité	Le rôle de l'informaticien est d'aider à la décision: le rêve est-il réalisable
3	Analyse des besoins et spécifications	Dégager les fonctionnalités que doit posséder le logiciel : cahier de charges détaillées, spécifications fonctionnelles, maquettes au besoin
4	Organisation du projet	Il faut analyser les coûts, planifie, et définir l'assurance qualité du logiciel
5	Conception	Déterminer les solutions techniques: architectures, interfaces, ...

# PROCESSUS DE DÉVELOPPEMENT: ETAPES STANDARD

N°	Etapes	Commentaires
6	Implémentation	Rédiger le code source du logiciel
7	Intégration	Composition progressive des modules, assemblage de composants testés séparément
8	Tests	Essayer le logiciel. Il y a plusieurs niveaux de tests: unitaires, intégration, ...
9	Déploiement	Livraison, installation, formation
10	Maintenance	Corrections et évolutions

# MODÈLE DU PROCESSUS DE DÉVELOPPEMENT

---

- **Objectifs:** organiser les différentes activités et guider le développeur dans ses activités techniques
- **Types:**
  - des modèles linéaires (en cascades et variantes): maîtriser les exigences et moins de modification. Exemple: MERISE (1978-1979); SADT (Ross, 1977)
  - et des modèles non-linéaires (incrémentaux, prototypage, en spirale): le changement est une constate et construire par itération. Exemple: processus unifié, en Y, méthodes agiles, ...



# UN MOT SUR LES ACTEURS DU PROJET

---

- L'identification et le choix des acteurs est capital.
- Ne pas avancer sans certains acteurs: réussir donc l'avant-garde du projet.
- Trouver une personne physique ou morale neutre pour piloter ou mettre à table les acteurs pour faire avancer le projet.
- L'identification des ressources humaines ou compétences pour la réalisation des étapes du processus de développement ne doit être négligée. Par exemple demander un expert d'audit-qualité pour donner de l'assurance au logiciel, un expert conseil pour accompagner surtout dans les choix technologique, la gestion des changements, les solutions de déploiement ou de transfert du système, etc.

## ET LA GESTION DES CHANGEMENTS ?

---

**Consiste à accompagner les utilisateurs dans la prise en main du nouvel outil**

**La problématique principale étant la peur du changement, du contrôle, de ne pas pouvoir produire les résultats.**



# 3

Prof. THIOMBIANO

## QUELQUES TECHNIQUES D'ESTIMATION DES COÛTS ET DES DELAIS



# NOTION DE CHARGES ET DURÉE

- La charge représente la quantité de travail nécessaire, indépendamment du nombre de personne qui vont réaliser ce travail: elle permet d'obtenir un coût prévisionnel.
- La charge s'exprime en jour/personne ou mois/personne ou encore année/personne. Un mois/personne représente l'équivalent du travail d'une personne pendant un mois, en général 20 jours.
- Exemple de calcul:
  - projet de 60mois/personne = travail d'une personne sur 60 mois.
  - Si coût du mois/personne = 50 KF, alors le projet =  $50 \times 60 = 3 \text{ MF}$ .

# NOTION DE CHARGES ET DURÉE

- Exemple de classification des projets selon leur taille:


Taille	Type de projet
Inf à 6 mois/personne	Très petit projet
6 à 12 mois/personne	Petit projet
12à 30 mois/personne	Projet moyen
30 à 100 mois/personne	Grand projet
Sup 100 mois/personne	Très grand projet

- La durée sera fonction du nombre de personnes répondant aux tâches à réaliser. Ainsi 60mois/personnes signifie qu'avec 3 personnes on fera moins d'un an.



# ESTIMATION EMPIRIQUE: PRINCIPE

---

- Empirique: basé sur l'observation, ... pas d'outils ...
  - La charge de développement est en jour / homme (J.H)
  - Elle permet ENSUITE d'estimer le coût de réalisation et le délai en tenant compte des ressources disponibles
  - Prendre chaque tâche ou lot:
    - Estimer le nombre de jours de travail et le nombre de personnes qu'il faut sans dépasser les ressources humaines affectées au projet et qualifiées pour la tâche.
    - Appliquer le coût unitaire selon les salaires journalier pour la qualité de ces types de tâches.
    - Agréger sur l'ensemble des lots.
- 

# ESTIMATION EMPIRIQUE: EXEMPLE

---

- Un informaticien sait qu'il lui faut  $\frac{1}{2}$  journée pour réaliser un écran de mise à jour d'une table Oracle. Il compte le nombre de tables (30) et en déduit que le codage des écrans de MAJ prendra 15 J.H. Soit une **charge** de 15 J.H
- Ce type de codage est fait par des Analyste Programmeurs (AP) qui coûtent 300 Euros par jour. Le **coût** de développement des écrans sera donc de  $15 * 300 = 4.500$  Euros.
- Si on dispose de 3 AP compétents sur ce domaine, le **délai** de réalisation des écrans de MAJ sera de 5 Jours et le coût de développement sera toujours de 4.500.

## EXERCICE PERSONNEL ...

TRAVAUX

**Réaliser l'estimation  
empirique des délais  
et coût de votre projet**

Prof. THIOMBIANO



# LES METHODES D'ESTIMATION

---

- Il en existe plusieurs
- Des non méthodes : exemple la méthode dite de Parkinson.
- Le jugement d'expert
- L'estimation par analogie
- L'estimation ascendante
- L'estimation paramétrique
- L'estimation probabiliste



# LE JUGEMENT D'EXPERT

---

- Des experts (ceux qui ont eu l'habitude de ce type de projet), donnent des estimations par leur vécu et leurs observations.
- Exemple: la méthode DELPHI



# L'ESTIMATION PAR ANALOGIE

---

- Estimations basées sur les consommations ou coûts de projets similaires pour lesquels on a conservé une mémoire.
- On obtient donc par analogie un ordre de grandeur du coût du projet: on a réalisé le même déploiement dans une société similaire à 500MF et à la formation telle que exigée par le client dans une autre société à 25MF; et il y a eu des charges diverses de 5MF. Alors on estime notre projet à 550MF ou à 525MF ...

# L'ESTIMATION ASCENDANTE

---

- Estimer la charge d'éléments de travail (lot de travail, activité, tâche, etc.) et de totaliser ces estimations élémentaires.
- Il faut avoir réaliser la SFT suffisamment fine.



# L'ESTIMATION PARAMÉTRIQUE

---

- Basé sur un modèle mathématique, issu d'analyses statistiques sur la relation entre un ou plusieurs paramètres, qui sont des variables caractéristiques du projet, et la charge prévisible.
- Analogue à la technique des unités d'œuvre utilisée en comptabilité analytique pour répartir les coûts généraux de façon simple ...
- Repérer quelques éléments simples et calculer l'effort. Exemple: le modèle COCOMO, la méthode des points de fonction



# EXEMPLE D'ESTIMATION PAR ANALOGIE

## Méthode de répartition proportionnelle:

- Basée sur les observations de projets antérieurs subdivisé en phase.
- Exemple de RUP

Phase	Charge	Durée
Opportunité	5%	10%
Elaboration	20%	30%
Construction	65%	50%
Transition	10%	10%



# EXEMPLE D'ESTIMATION PAR ANALOGIE

## Méthode des ratios:

- Mise en relation de deux activités.
- Souvent utilisé pour estimer des charges complémentaires au développement
- Exemple

DR. THIOMBIANO

Activités	Ratio
Encadrement du projet: <ul style="list-style-type: none"><li>- À la phase de réalisation</li><li>- Aux autres phases</li></ul>	20% de la charge de réalisation 10% de la charge de la phase
Recette	20% de la charge de réalisation
Documentation utilisateur	5% de la charge de réalisation

# ESTIMATION PARAMÉTRIQUE: LE MODÈLE COCOMO

---

- COCOMO: Constructive Cost Model ou modèle de construction des coûts.
- Proposé en 1981 par B.W. Boehm
- Deux hypothèses:
  - L'informaticien chevronné sait facilement donner une évaluation de la taille du logiciel à développer
  - Il faut généralement le même effort pour écrire un nombre donné de lignes de programme
- L'estimation est basée sur des coefficients de corrélation entre la taille du logiciel et la charge consommée à partir d'observations ...

# ESTIMATION PARAMÉTRIQUE: LE MODÈLE COCOMO

- Le paramètre utilisé = l'instruction source ou nombre de lignes de programme en langage source, en dehors d'éventuels commentaires.
- Charges en mois/personne =  $a(kisl)^b$
- Délais normal en mois =  $c(\text{charge en mois/personne})^d$
- Légende:
  - Kils = nombre de milliers d'instruction sources livrées
  - A, b, c et d = prennent des valeurs différentes selon la catégorie de projet.
- Taille moyenne de l'équipe =  $\text{charge} / \text{délais}$



# ESTIMATION PARAMÉTRIQUE: LE MODÈLE COCOMO

Type	Description	Charge mois/pers	Délais en mois
<b>Simple</b>	moins de 50.000 instruction, spécifications stables, petite équipe	<b>2,4(kisl)<sup>1,05</sup></b>	<b>2,5(charge)<sup>0,38</sup></b>
<b>Moyen</b>	Entre 50.000 et 300.000 instructions	<b>3(kisl)<sup>1,12</sup></b>	<b>2,5(charge)<sup>0,32</sup></b>
<b>Complexe</b>	Plus de 300.000 instructions, équipe nombreuse, domaine nouveau	<b>3,6(kisl)<sup>1,2</sup></b>	<b>2,5(charge)<sup>0,32</sup></b>

# ESTIMATION PARAMÉTRIQUE: LE MODÈLE COCOMO

- **Exemple: Soit par exemple un projet simple visant à développer un logiciel estimé à 40.000 instructions sources**
  - Charge =  $2,4 \times (40)^{1,05} = 116$  mois-personne (arrondi)
  - Délai normal =  $2,5 \times (116)^{0,38} = 12$  mois (arrondi)
  - Taille moyenne de l'équipe =  $116 / 12 = 9$  personnes
- NB/ Cette méthode a évolué pour intégrer d'autres caractéristiques devant permettre d'être plus précis: voir COCOMO II qui est inspiré de la méthode des POINTS DE FONCTION.

# COCOMO SIMPLE

- mode organique : HM = 2,4 (KLSL) 1.05
- semi-détaché : HM = 3.0 (KLSL) 1.12
- détaché : HM = 3,6 (KLSL) 1.20
  
- mode organique : TDEV = 2.5 (HM) 0.38
- semi-détaché: TDEV = 2.5 (HM) 0.35
- détaché : TDEV = 2.5 (HM) 0.32

EFFORT

DUREE

$$N = HM / TDEV$$

HM : Hommes-Mois (152heures)

KLSL : Kilo de Ligne de Source Livrées

Prof. CHIOMBIANO



# COCOMO INTERMÉDIAIRE

---

- **Quinze facteurs correctifs sont introduits**
  - valués de VeryLow à XtraHigh
- **Pour le projet :**
  - fiabilité requise du logiciel
  - taille de la base de donnée
  - complexité du produit
- **Pour les contraintes de l'environnement :**
  - contraintes de temps d'exécution / place mémoire
  - stabilité de la machine virtuelle
  - système de développement interactif ou non





# COCOMO INTERMÉDIAIRE

---

## ○ Pour le personnel :

- aptitude à l'analyse
- expérience du domaine
- expérience de la machine virtuelle
- aptitude à la programmation
- expérience du langage

## ○ Pour les méthodes :

- méthode de programmation moderne
- outils logiciels
- durée du développement



# COCOMO DÉTAILLÉ

---

- **Les facteurs correctifs dépendent de la taille (KLSL)**
- **Une répartition de l'effort sur les phases de développement est réalisée**



# LA MÉTHODE DES POINTS DE FONCTION : HISTORIQUE

---

- Présentée pour la 1<sup>ère</sup> fois par Alan Albrecht d'IBM en 1979
- Difficultés au départ: l'interprétation et le dénombrement des unités d'œuvre
- En 1995, l'AFNOR a édicté une norme sur la mise en œuvre de la méthode.



# LA MÉTHODE DES POINTS DE FONCTION : PRINCIPE

---

- On identifie cinq (5) types avec trois (3) degrés de complexité.
- A chaque type et chaque degré est affecté un nombre de « points »: ce qui permet de calculer pour un projet, son poids en points de fonction.
- A partir du nombre de points de fonction, il y a des règles pour passer à une charge.



# LA MÉTHODE DES POINTS DE FONCTION

## :PRINCIPE

---

- Faire une estimation à partir d'une description externe du futur système, de ses « fonctions » ou « composants fonctionnels ».
- Faire un comptage au début du projet en faisant l'hypothèse des fonctions du futur système, et faire un comptable en fin de projet basé sur les fonctionnalités livrées: l'écart éventuel est le « changement d'envergure » du champ d'étude.

# LA MÉTHODE DES POINTS DE FONCTION

## :LES COMPOSANTS FONCTIONNELS

---

- Cinq (5) types de composants fonctionnels servent d'unités d'œuvre
  - Deux (2) = aspect statique d'un SI (GDI et GDE)
  - Trois (3) = aspect dynamique (ENT, SOR, INT)
- GDI (groupe de données interne)
  - C'est un groupe de données perçu par l'utilisateur comme liées.
  - Il est créé et mis à jour à l'intérieur du domaine d'étude: il est assimilable au formalisme entité/association.
  - Les entités reliées à d'autres par la relation de cardinalité (1,1) sont susceptibles d'être regroupées en un seul GDI si elles traduisent un même concept de gestion.

# LA MÉTHODE DES POINTS DE FONCTION

## :LES COMPOSANTS FONCTIONNELS

---

- GDE (Groupe de données Externes)
  - Groupe de données que l'utilisateur perçoit comme logiquement liées.
  - Le domaine d'étude ne fait que l'interroger.
  - Il est créé et mis à jour par un autre domaine.
  - Il est donc obligatoirement un GDI dans un autre domaine.



# LA MÉTHODE DES POINTS DE FONCTION

## :LES COMPOSANTS FONCTIONNELS

---

### ○ ENT(Entrée)

- Fonction élémentaire, significative pour l'utilisateur, qui permet d'introduire des données à l'intérieur du domaine. Entrée = écran de saisie, etc.
- A chaque GDI doit correspondre au moins une entrée permettant sa mise à jour. Chaque ENT permet de saisir au moins un certains nombres de champs appelés DE (données élémentaires).
- On compte une seule DE pour un champ répétitif.



# LA MÉTHODE DES POINTS DE FONCTION

## :LES COMPOSANTS FONCTIONNELS

---

- SOR (Sortie)

- Fonction élémentaire significative pour l'utilisateur, qui envoie des données vers l'extérieur du domaine et qui n'effectue aucune mise à jour l'intérieur du domaine.
- SOR = génération d'un état, écran de visualisation, etc.
- Les champs en sorties sont des DE.
- On compte une seule DE pour un champ répétitif. Si deux SOR ont la même logique de traitement et les mêmes DE, on compte une seule SOR.



# LA MÉTHODE DES POINTS DE FONCTION

## :LES COMPOSANTS FONCTIONNELS

---

- INT (Interrogation)

- Fonction élémentaire qui a pour résultat l'extraction des données.
- La demande d'interrogation peut être saisie ou provenir d'une autre application.



# LA MÉTHODE DES POINTS DE FONCTION

## :LA COMPLEXITÉ ET LE NOMBRE PF

- Un GDI ou un GDE est composé de DE (données élémentaires) qui correspondent aux propriétés d'une entité conceptuelle ou logique.
- On compte une DE par champ, y compris les clés étrangères.
- On peut identifier plusieurs sous-ensembles logiques de données (SLD) à l'intérieur d'un GDI/GDE. Dans un diagramme de classes UML, un SLD peut être assimilé à une entité spécialisée: on compte un SLD pour l'entité générale et un SLD pour le spécialisée.
- La complexité du GDI ou d'un GDE est fonction du nombre de DE et du nombre de SLD



# LA MÉTHODE DES POINTS DE FONCTION

## :LA COMPLEXITÉ ET LE NOMBRE PF

---

- Une ENT utilise, en lecture ou mise à jour, différents groupes logiques de données, internes ou externes (GDI ou GDE), que l'on appelle de façon générale des GDR (groupes de données référencées).
- Une SOR ou une INT utilise, en lecture uniquement, différents GDR.
- La complexité d'une ENT, d'une SOR ou d'une INT est fonction du nombre de DE et du nombre de GDR impliqués.

# LA MÉTHODE DES POINTS DE FONCTION

## :LA COMPLEXITÉ ET LE NOMBRE PF

GDI ou GDE		1 à 19 DE	20 à 50 DE	51 DE ou plus
	1 SLD	Faible	Faible	Moyenne
	2 à 5 SLD	Faible	Moyenne	Élevée
	6 SLD ou plus	Moyenne	Élevée	Élevée
ENT		1 à 4 DE	5 à 15 DE	16 DE ou plus
	0 ou 1 GDR	Faible	Faible	Moyenne
	2 GDR	Faible	Moyenne	Élevée
	3 GDR ou plus	Moyenne	Élevée	Élevée
SOR ou INT		1 à 5 DE	6 à 19 DE	20 DE ou plus
	0 ou 1 GDR	Faible	Faible	Moyenne
	2 à 3 GDR	Faible	Moyenne	Élevée
	4 GDR ou plus	Moyenne	Élevée	Élevée

Préf. THOMBIANO

**Evaluation de la complexité**

# LA MÉTHODE DES POINTS DE FONCTION

## :LA COMPLEXITÉ ET LE NOMBRE PF

Degré de complexité du composant	Faible	Moyenne	Élevée
Nombre de points de fonction du GDI	7	10	15
Nombre de points de fonction du GDE	5	7	10
Nombre de points de fonction de l'ENT	3	4	6
Nombre de points de fonction de la SOR	4	5	7
Nombre de points de fonction de l'INT	3	4	6

Prof. THOMBIANO

**Nombre de points de fonction selon la complexité**



# LA MÉTHODE DES POINTS DE FONCTION :L'AJUSTEMENT DE LA TAILLE

- Nombre de points de fonction brut (PFB) = la somme des points de fonction de chaque composant fonctionnel.
- La 2<sup>nd</sup>e étape est facultative = ajuster l'évaluation obtenue. On corrige le nombre de PFB en appréciant les spécificités du projet pouvant influencer sur l'effort: fonctionnalités techniques et ergonomiques.
- La méthode a identifié 14 appelées CGS (caractéristiques générales du système). Chacune reçoit une note entre 0 et 5 en fonction de l'influence qu'il exerce: on l'appelle la DI (degré d'influence).



# LA MÉTHODE DES POINTS DE FONCTION

## :L'AJUSTEMENT DE LA TAILLE

Prof. THIOMBIANO

- L'analyse des caractéristiques permet de calculer le degré d'influence total, le DIT, compris entre 0 et 70:
  - $DIT = \text{somme } (D_{ii}) \text{ avec } i = 1 \text{ à } 14$
- Le facteur d'ajustement FA permet d'ajuster le nombre de PFB de +/- 35%:
  - $FA = 0,65 + DIT/100$
- Nombre de PFA :
  - $PFA = FA \times PFB$
- NB/ On peut bien s'en tenir au PFB.





# LA MÉTHODE DES POINTS DE FONCTION :LA TRANSFORMATION DU NB DE PF EN CHARGE

---

- Le coefficient de transformation est variable selon l'environnement matériel et humain. Chaque entreprise établit sa base de projets pour déterminer ses propres coefficients.
- C'est là que s'applique le modèle COCOMO II: le coefficient est déterminé en fonction du nombre d'instructions sources livrées en prenant en compte le langage de programmation.



# LA MÉTHODE DES POINTS DE FONCTION :LA TRANSFORMATION DU NB DE PF EN CHARGE

- En général, la charge se calcule en convertissant directement le nombre de PF:
  - En fin d'étude préalable : 3jrs/PF; 2jrs/PF pour les petits projets; 4jrs/PF s'il s'agit d'un grand projet.
  - En fin d'étude détaillée: 1 à 2 jrs/PF selon l'environnement
- Exemple: en fin d'étude détaillée, un projet de 80PF « pèse » en général 160 jours/pers en environnement grand système et 80 en environnement client/serveur.



# 4

## TECHNIQUES DE PLANIFICATION ET DES DURÉES

Prof. THIOMBIANO



# LA PLANIFICATION OPÉRATIONNELLE

- Elle procède à l'ordonnancement (mise en séquence) des activités du projet.
- C'est une planification dite de délais.
- Consiste à identifier les liaisons logiques ou les relations d'ordre qui existent entre les différentes activités du projet. Les relations de dépendance ou d'antécédence, entre les tâches indiquent que l'exécution d'une tâche dépend ou doit débuter après ou avant celle d'une autre tâche
  - Fin-début : le prédécesseur doit obligatoirement terminé avant que le successeur ne démarre.
  - Début-début : les tâches s'exécutent simultanément.
  - Fin-fin : les tâches finissent en même temps.



# LE PLANIFICATION OPÉRATIONNELLE

- c'est véritablement au stade de la planification de délais que l'on apprécie la définition du projet comme une série de tâches bien définies, reliées entre elles, exécutées dans un temps et avec des ressources limités, pour converger vers un out put (extrait ou produit final attendu) préalablement déterminé.
- Le réseau logique dans lequel on modélise suit des techniques d'ordonnancement telles que la méthode PERT (Program Evaluation and Review Technic) dont le calcul du chemin critique. NB/ sera étudié dans un autre cours.
- Le diagramme de Gantt (via MS EXCEL, MS PROJECT, Gant Project, etc.)



# EXEMPLE RESEAU DE PERT

## ○ Exemple 1

- D est postérieure à la réalisation de A et B
- C doit succéder à D

Tâches	Antécédents
A	/
B	/
C	D
D	A,B

Prof. THOMBIANO

## ○ Exemple 2

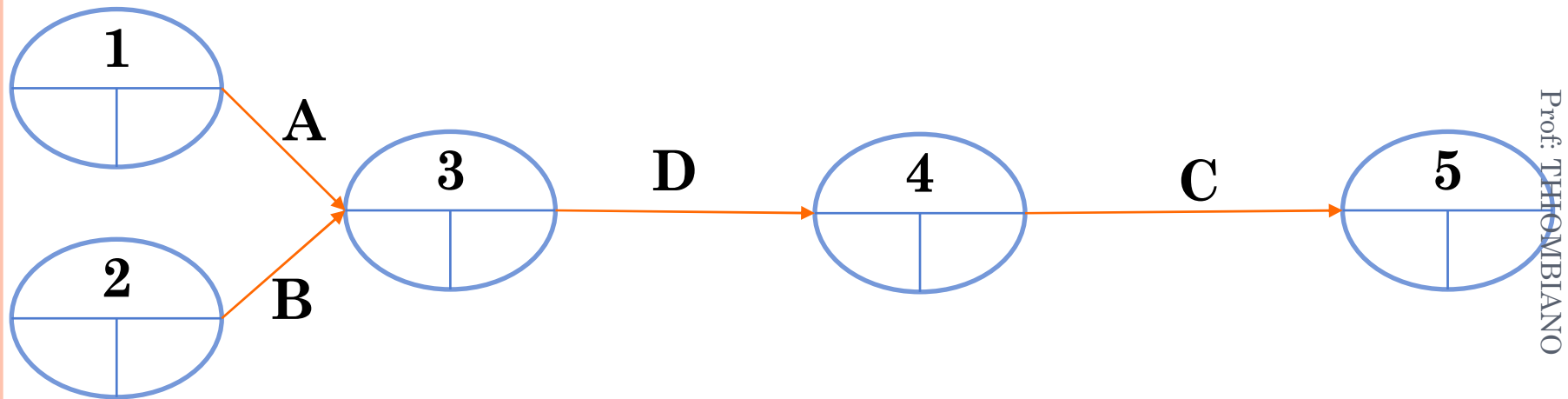
- A, B et C précèdent le début de D
- E succède à B

Tâches	Antécédents
A	/
B	/
C	/
D	A,B,C
E	B



# EXEMPLE RESEAU DE PERT

## ○ Exemple 1



Prof. THOMBIANO

Tâches	Antécédents
A	/
B	/
C	D
D	A,B



# EXEMPLE RÉSEAU DE PERT ET CHEMIN CRITIQUE

Tâches	Antécédents	Durée
A	/	3
B	A	1
C	A	5
D	C,I	6
E	B,D	4
F	C,I	2
G	E,F	9
H	/	5
I	H	8
J	H	2
K	I	3
L	J,J	7





**EXERCICE PERSONNEL ...**

**TRAVAUX**

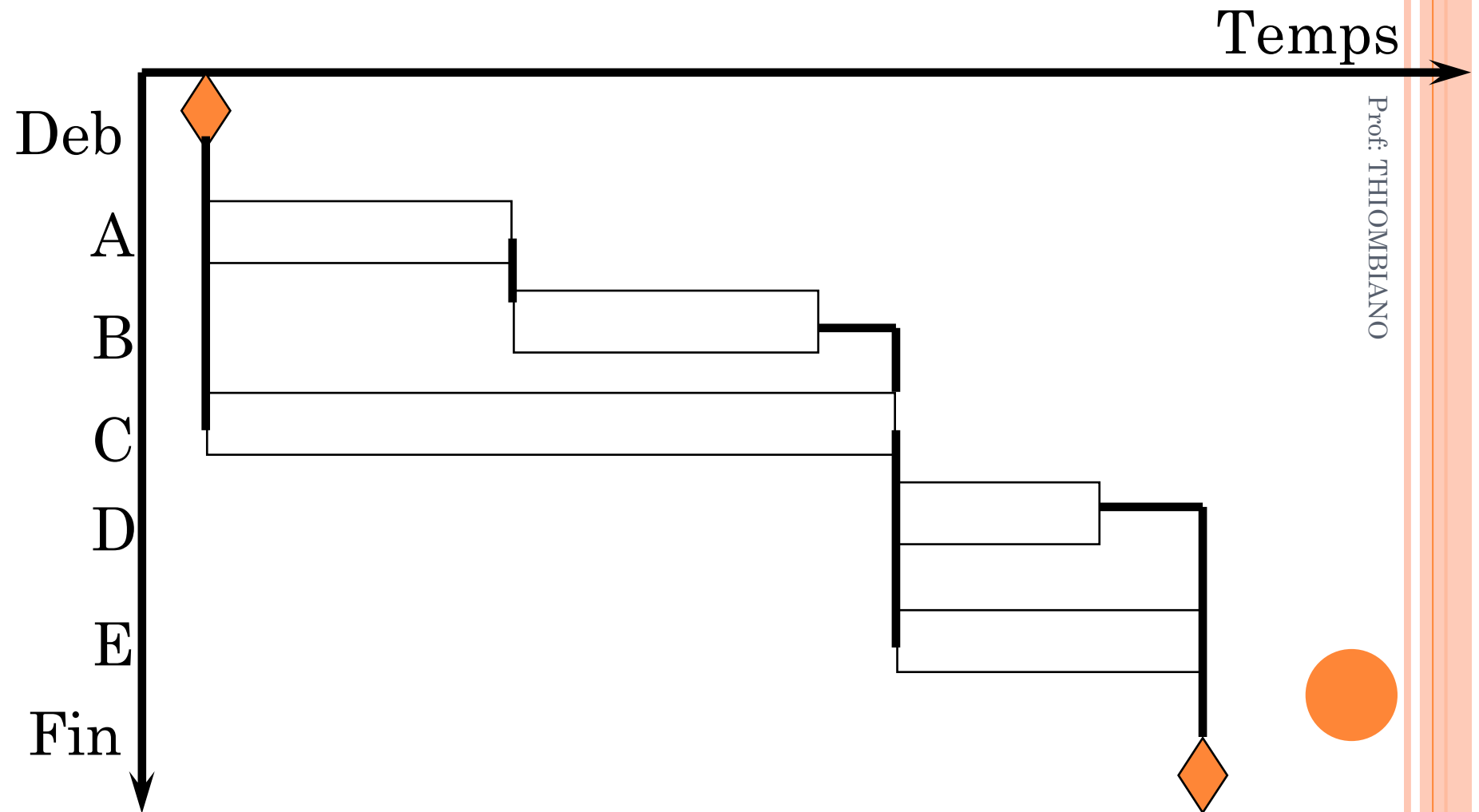
**Des exercices sur le  
réseau de PERT !!!**

Prof. THIOMBIANO

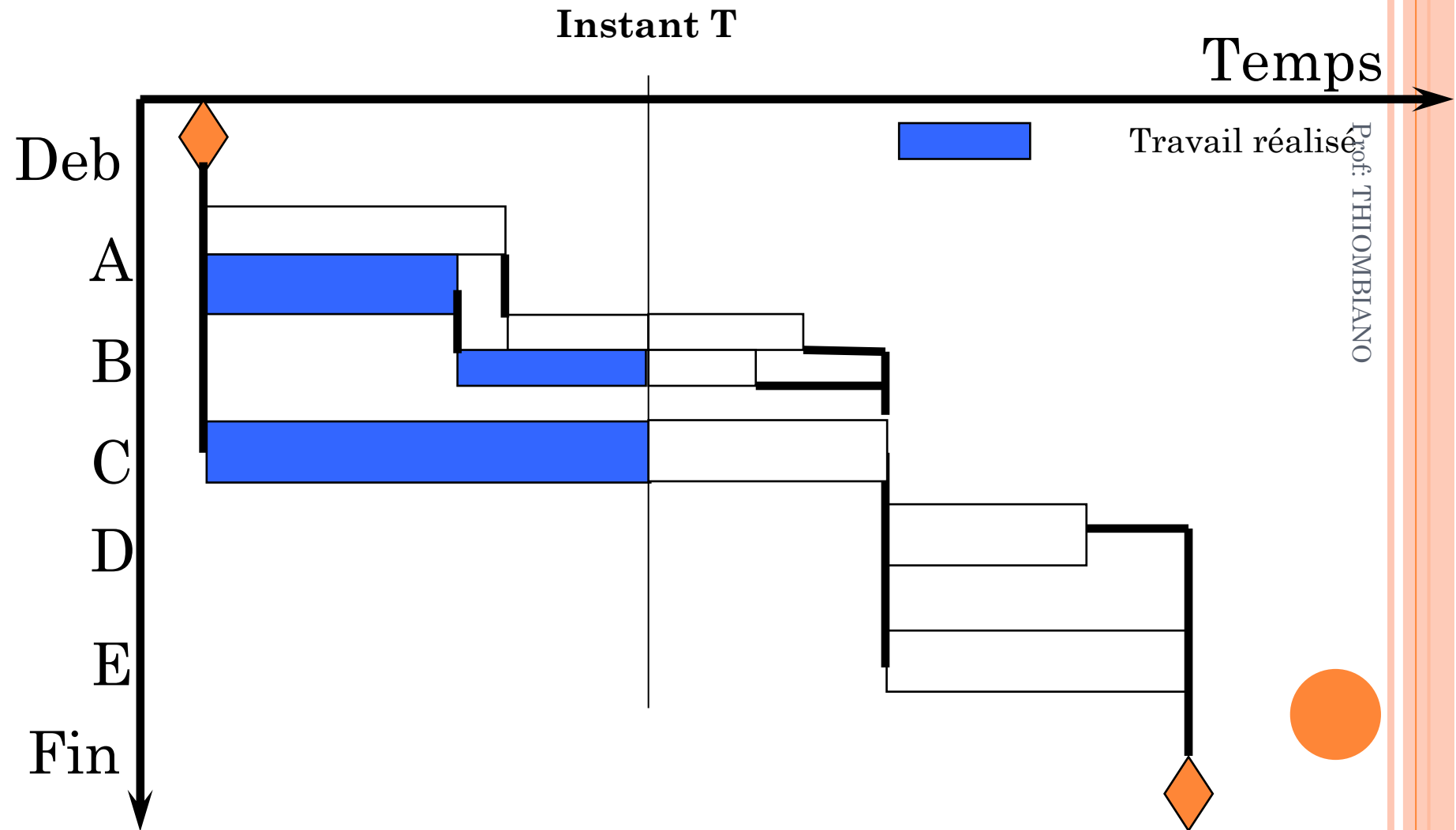
**Dans un autre cours ...**



# FORME DU DIAGRAMME DE GANT



# FORME DU DIAGRAMME DE GANT



TRAVAIL PERSONNEL ...

TRAVAUX

# Réaliser le diagramme de Gantt de votre projet

Prof. THIOMBIANO



# TRAVAUX DE GROUPE ...

## EXPOSE

**Problématique/arbres des problèmes et arbres des objectifs; formuler objectif global, objectif spécifique à résoudre, résultats attendus, acteurs ou équipe projet, analyse, conception et organisation des itérations, décrire deux à trois solution et justifier un choix de solution, quel prototype réalisé, etc. En somme: proposer un plan selon la méthode adoptée, et nourrir chaque point.**

Prof. THIOMBIANO



# 5

Prof. THIOMBIANO

## EN CONCLUSION



# AU CŒUR DE L'ESPRIT ...

Les informaticiens doivent  
pouvoir nous livrer des logiciels  
prêts à porter ...

**Quel processus de  
développement, pour ne pas  
échoué?**



# CONSTRAINTES DANS LES PROJETS DE DÉVELOPPEMENT

1. Nos entreprises ne sont pas organisées selon une fluidité de la circulation de l'information de bout en bout.
2. L'informatique est toujours vue comme une contrainte, comme un outil de contrôle, comme un gendarme.
3. Les processus ne sont pas toujours existants, ou ne sont pas fluides, ou encore ne sont pas intégrées.
4. Les utilisateurs ne se sentent pas responsables de la réussite des projets informatiques.
5. Les budgets ne sont pas définis et les projets souffrent du manque d'organisation.
6. Le logiciel n'est pas vu comme un produit, une richesse de l'entreprise, et donc n'est pas valorisé.
7. Etc.





# POUR RÉSUMER ...

Tout est une question de choisir la ou les méthodes, de les combiner et surtout de tenir compte de son environnement pour générer sa propre METHODE. Exemple de SANDROSE.

**Les METHODES se basent sur les normes (voir IEE, ...) ou encore les bonnes pratiques (voir ITIL)**



# POUR RÉSUMER ...

A chaque projet, sa matrice d'indicateurs de suivi à élaborer sur la base d'une norme telle que ISO 9001 qui définit des critères de conception, de développement, de production et d'installation de produits.



## LISTE DES QUESTIONS

- Le stagiaire académique a-t-il un droit sur les applications qu'il développe en entreprise ? A quelles conditions peut-il revendre le produit ou service réalisé ?
- Critères d'appréciation de la qualité d'un logiciel: pourquoi le logiciel 1 est-il meilleure au logiciel 2 ?
- Comment se gère les mises à jour du logiciel ?
- Notre avis sur le développement des logiciels au Burkina Faso: ne va-t-il pas disparaître avec les SSII?
- Y-a-t-il des développeurs au Burkina pour de gros projets ?



## LISTE DES QUESTIONS

- Quelle spécialisation pour la génération actuelle des informaticiens ?
- Quel délais est accordé au développeur? Quand déclare-t-on que le produit est fini ?
- Hormis la gestion des droits, comment gérer la sécurité des applications ?
- Comment faire pour avoir la compétence de développeur d'applications sur Android?
- Que dire de la consultation indépendante?
- Quelle différence entre les applications web et les sites web classiques ?
- Comment gérer le budget des projets ?



## RESSOURCES

- **SANDROSE**: méthode agile et un système intégré de gestion informatique des finances publiques  
<http://www.sandrose.org/>
- **TELIA INFORMATIQUE** : concrétisation top AGILE, Ouagadougou  
<http://www.teliainfo.com/>
- **RAD**  
<http://www.rad.fr>
- **Méthode AGILE**  
<http://www.aggil.com/methode/methode-agile.html>
- <http://www.commentcamarche.net/contents/477-methodes-agiles-rad-xp>
- <http://www.guidescomparatifs.com/telechargement.asp>
- [https://www.lamicrofinance.org/resource\\_centers/systemeinformation/faire\\_evolution\\_le\\_sig\\_de\\_votre\\_imf/choisir\\_un\\_logiciel\\_pour\\_votre\\_imf](https://www.lamicrofinance.org/resource_centers/systemeinformation/faire_evolution_le_sig_de_votre_imf/choisir_un_logiciel_pour_votre_imf)






# MERCI

**Contact :**

**M. THIOMBIANO Aristide Paalou**

**[athiombiano@gmail.com](mailto:athiombiano@gmail.com)**

**70 50 81 50**



Prof. THIOMBIANO