IFT159
Analyse et programmation
Introduction

Gabriel Girard

Département d'informatique
UNIVERSITÉ DE SHERBROOKE

19 août 2015

Introduction

UNIVERSITÉ DE SHERBROOKE

Introduction

Les ordinateurs

Premiers ordinateurs

- Début dans les années 40 ;
- Ordinateur = outil inanimé ;
- Exécute les instructions données ;
- Programme = suite d'instructions ;
- Programme écrit dans le langage de l'ordinateur.

Introduction

UNIVERSITÉ DE SHERBROOKE

Historique

Origine du mot *calcul*

Historique

Origine du mot *calcul*

- Du latin *calculi*

Historique

Origine du mot *calcul*

- Du latin *calculi*
- Qui veut dire **cailloux**

Historique

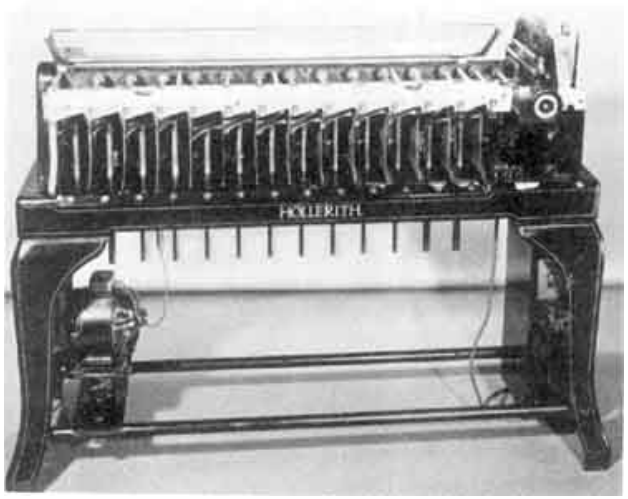
Premiers outils de calcul manuels

- Abaque (-500)
- Règle à calcul (1625)

Historique

Premiers calculateurs mécaniques

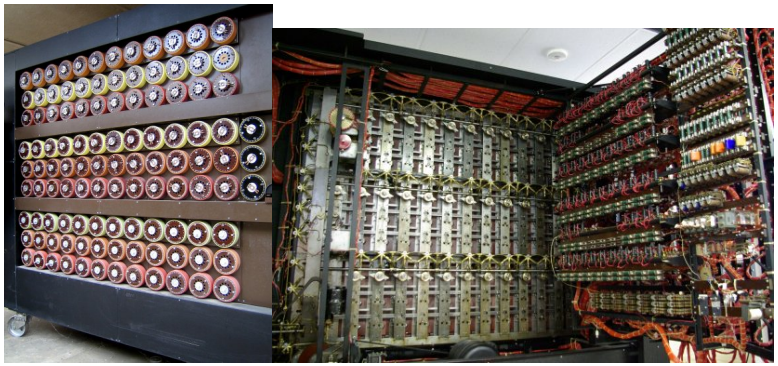
- Pascaline (1642)
- Babbage (1830)
- Hollerith (1884)

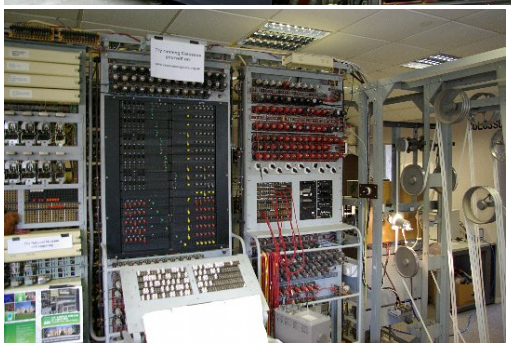


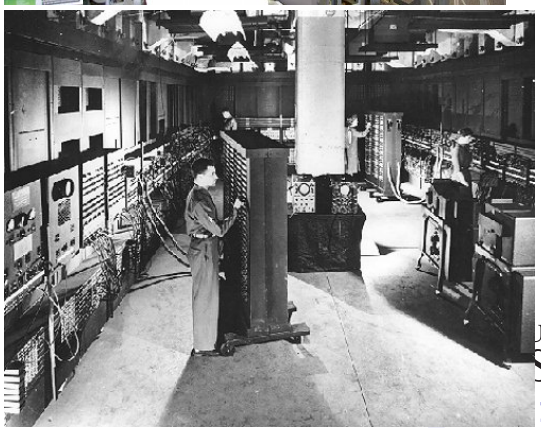
Historique

Premiers ordinateurs

- Turing/Bombe/Colossus (1937-40)
- Zuse (1938-41)
- ABC (1939)
- ENIAC (1946)
- Von Neumann/EDVAC (1945-49)







Historique

Quel était le poids du premier Ordinateur ?

1. 100 kilos (réfrigérateur)
2. 1000 kilos (Toyota Yaris)
3. 1800 kilos (minivan)
4. 15 000 kilos (niveleuse)
5. 27 000 kilos (baleine à bosse)

Historique

Évolution technologique

- Transistors (1947)
- Premier ordinateur à transistors (1956)
- Circuit intégré (TI 1958)
- Microprocesseur (1971)

Historique

Évolution technologique

- Avant 1939 $\rightarrow$ manque de technologie
- Avant 1975 $\rightarrow$ taille et consommation importante
- Après 1975 $\rightarrow$ début des micro-ordinateurs

Historique

Programmation

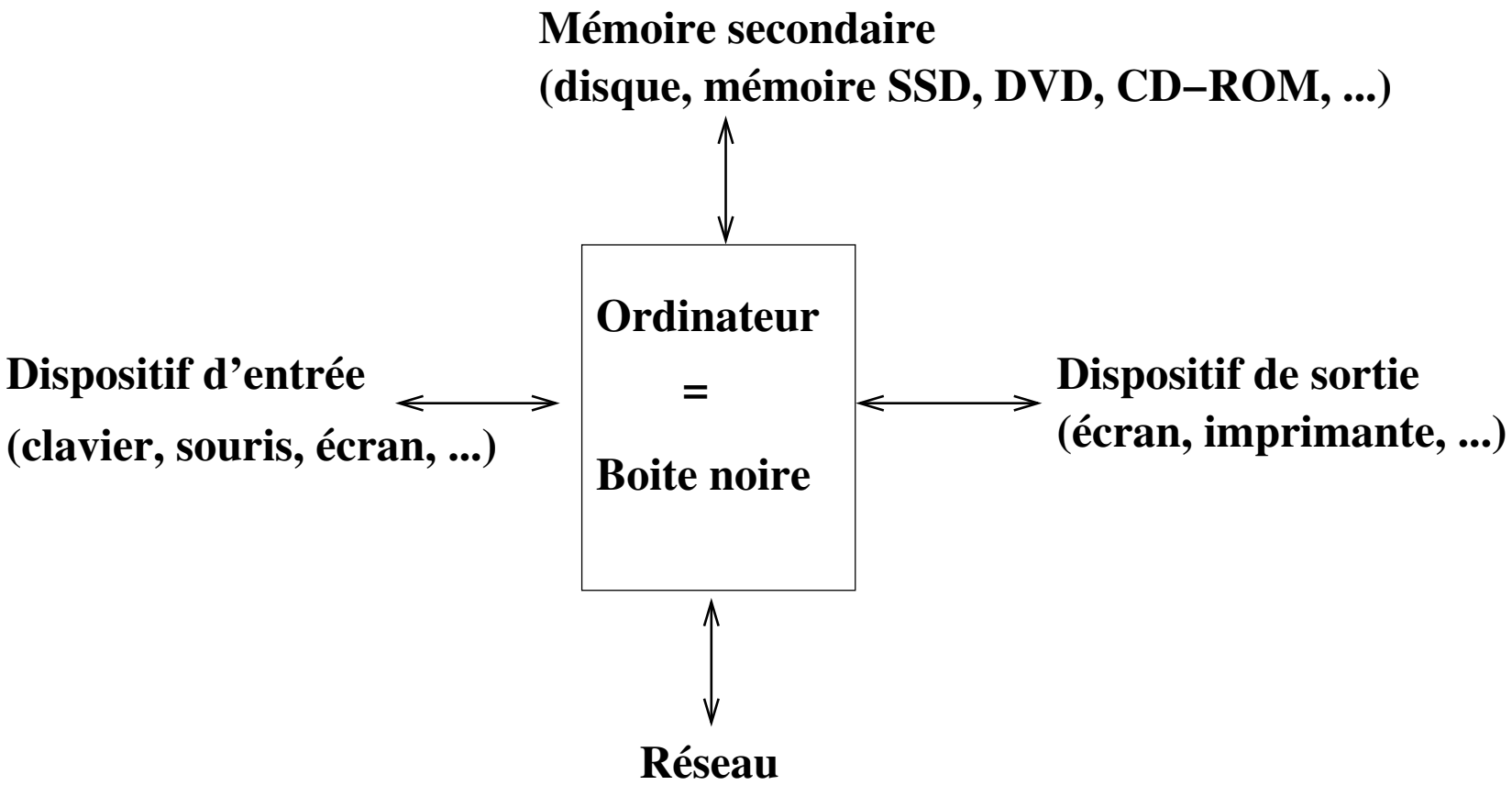
- Ada Augusta Lovelace (algorithme 1840)
- Assembleur (Wilkes 1950)
- Fortran (1957), Lisp (1958) et Cobol (1960)
- Algol (1958/1960) est ses successeurs
- Interface graphique (Engelbart 1968)
- Arpanet/TCP ($>$ 1970)
- WEB/Internet (Berners-Lee $>$ 1990)

Phrases célèbres de l'histoire

- «*I think there is a world market for maybe five computers*»
  Thomas Watson, Chairman of IBM, 1943
- «*Computers ine the future may weigh no more than 1.5 tons*»
  Popular Mechanics, 1949
- «*There is no reason anyone would want a computer in their home*»
  Ken Olson. président fondateur, DEC
- «*I see little commercial potential for the internet for the next 10 years !*»
  Bill gates, 1994

Introduction

Environnement matériel

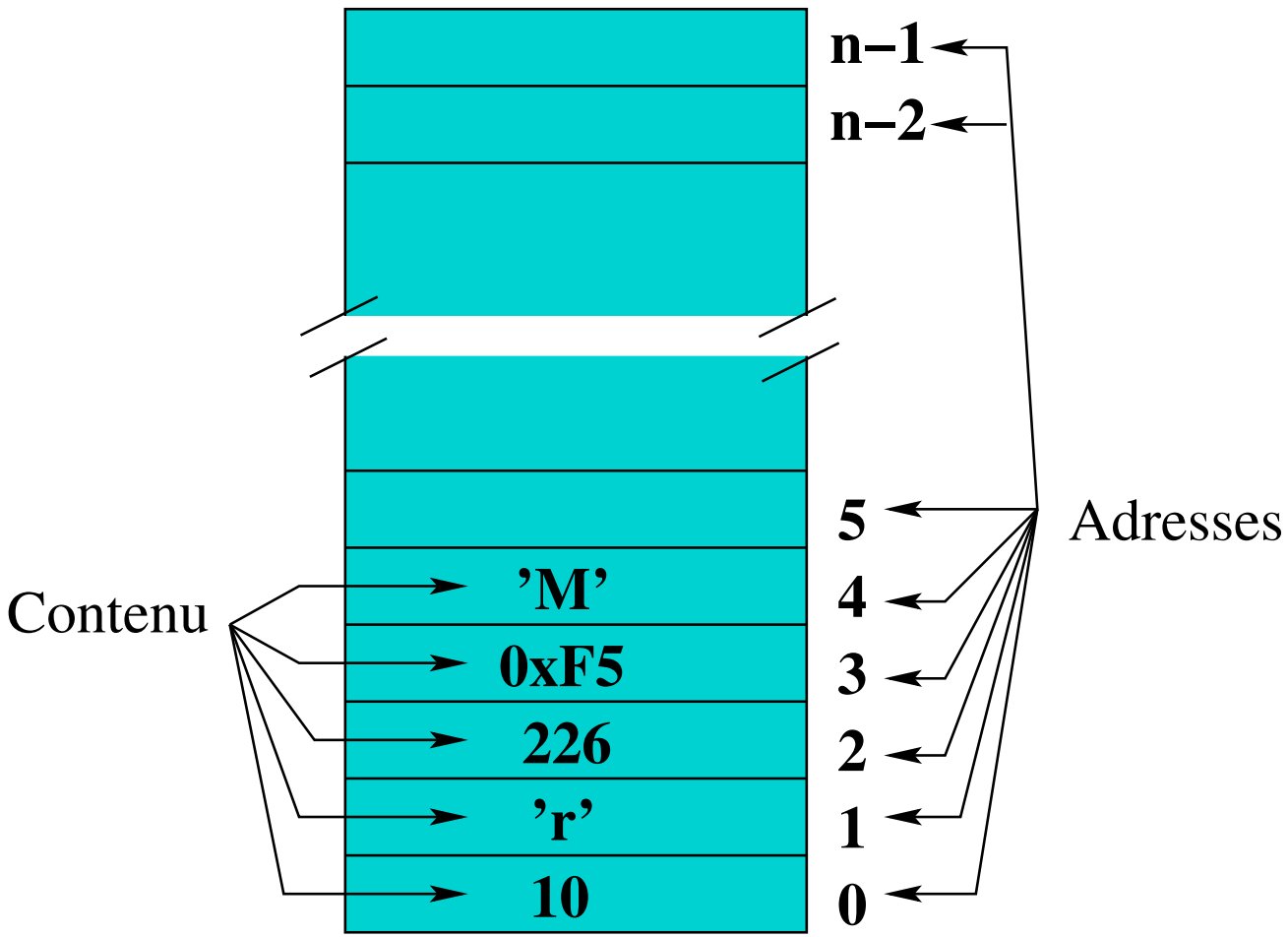


Ordinateur et ses périphériques

Mémoire centrale

- Séquence ordonnée de cellules (0 à N-1) ;
- N est la taille exprimée en Meg ou Gig ;
- Numéro de la cellule = adresse ;
- Cellule = 1 octet = 8 bits ;
- valeur occupe une ou plusieurs cellules ;

Mémoire centrale



UCT

- Unité de contrôle ;
- Unités fonctionnelles (UAL, ...) ;
- Registres (généraux, PC, PS) ;
- ift209, gei201, gei301 ;

Unités d'entrées et de sorties

- Clavier, souris, crayon, guichet, ...
- Écran, imprimante, ...

Mémoire secondaire

- Disque souple ;
- Disque rigide ;
- Carte de mémoire (flash, ...) ;
- Bande magnétique ;
- Disque compact (CD, DVD, Blu-ray) ;
- Concept de fichiers

Réseau

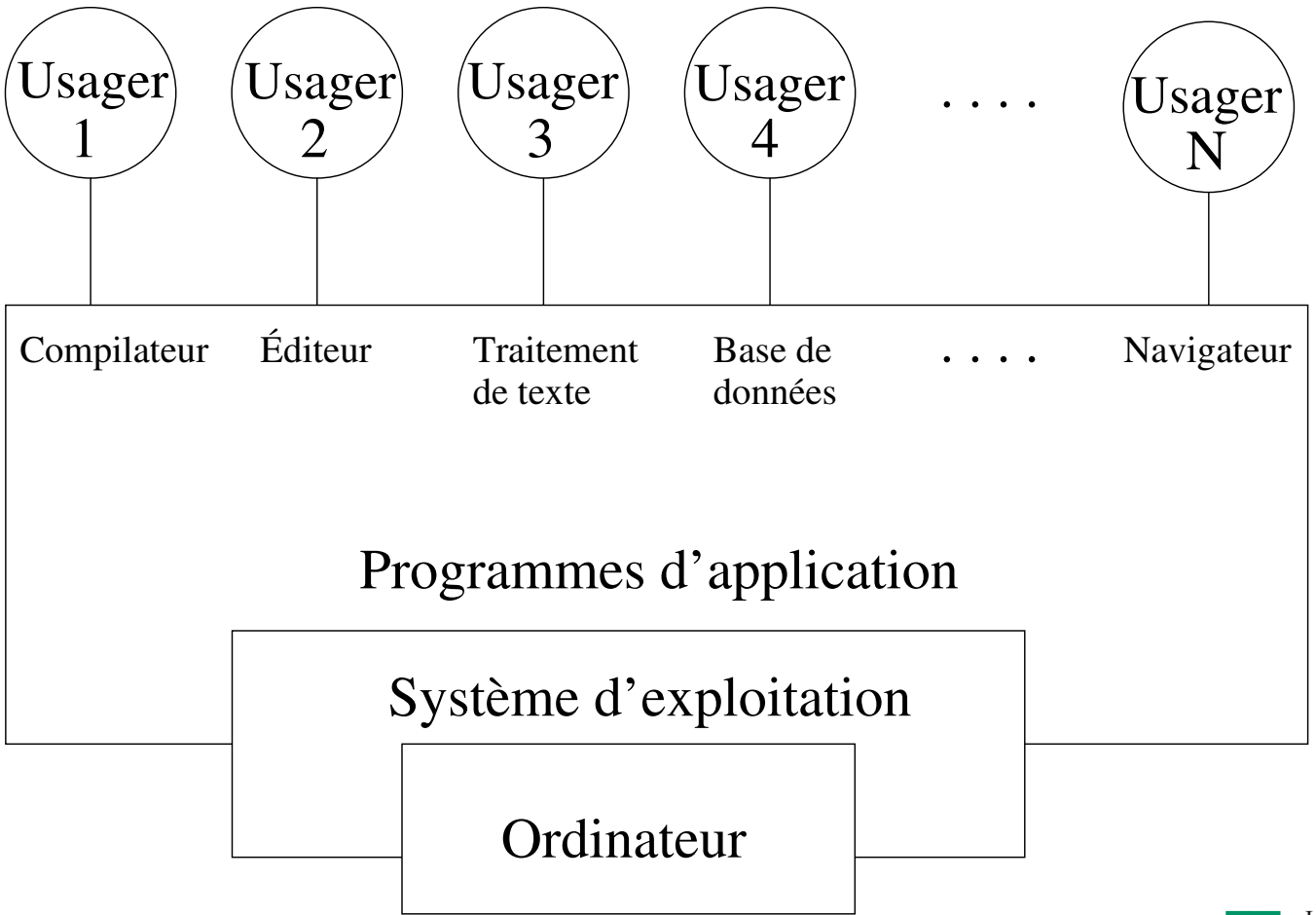
- Local (Ethernet, sans-fil) ;
- Modem ;
- Ift585 ;

Introduction

Environnement Logiciel

- Système d'exploitation ;
- Applications ;
- Outils de développement (langage, compilateurs, ...) ;

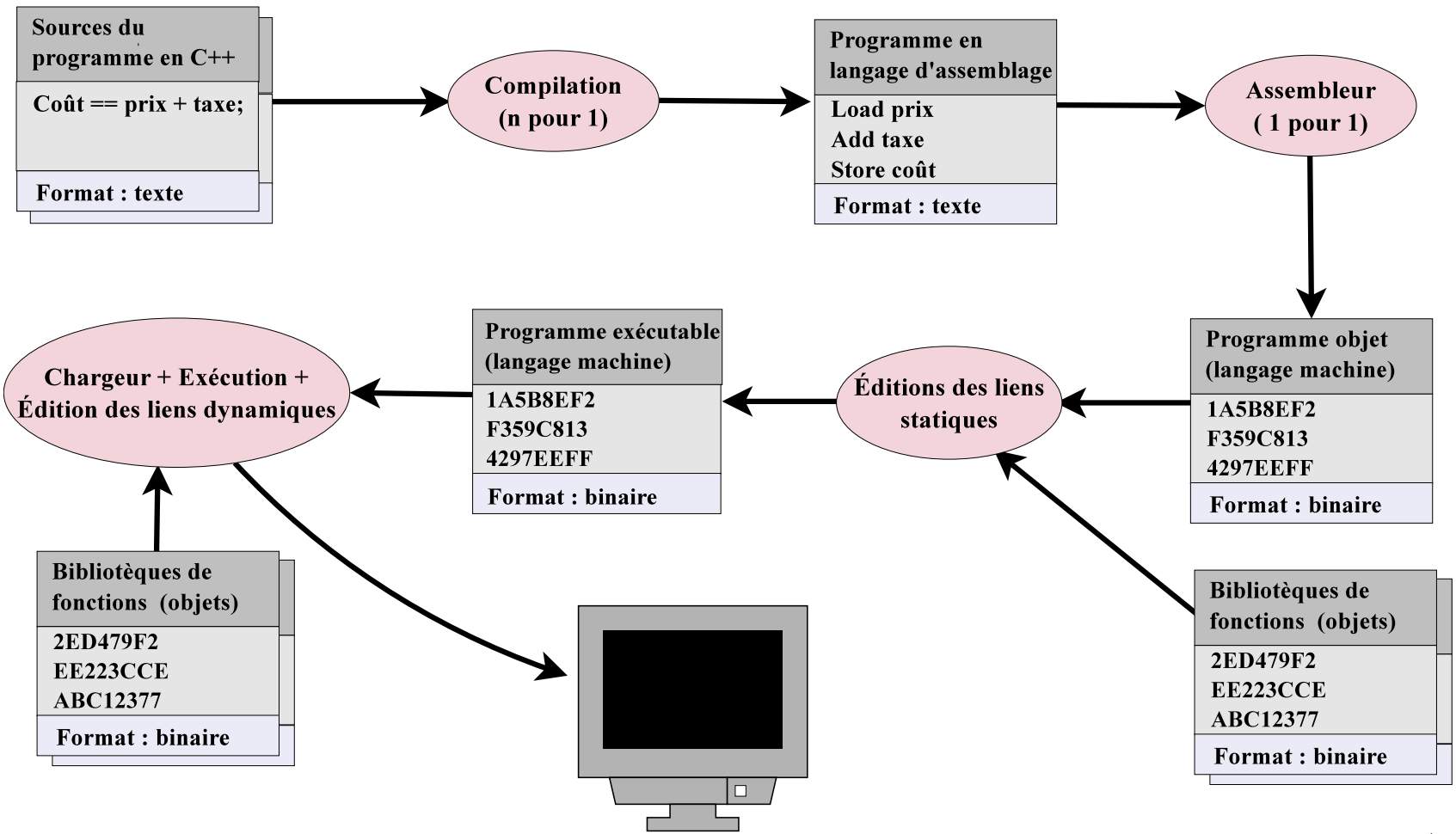
Environnement Logiciel



Environnement de développement

- Langage de programmation ;
- Compilateur ;
- Éditeurs, outil de mise au point, éditeur de liens, ... ;
- Environnement de développement intégré ;

Environnement de développement

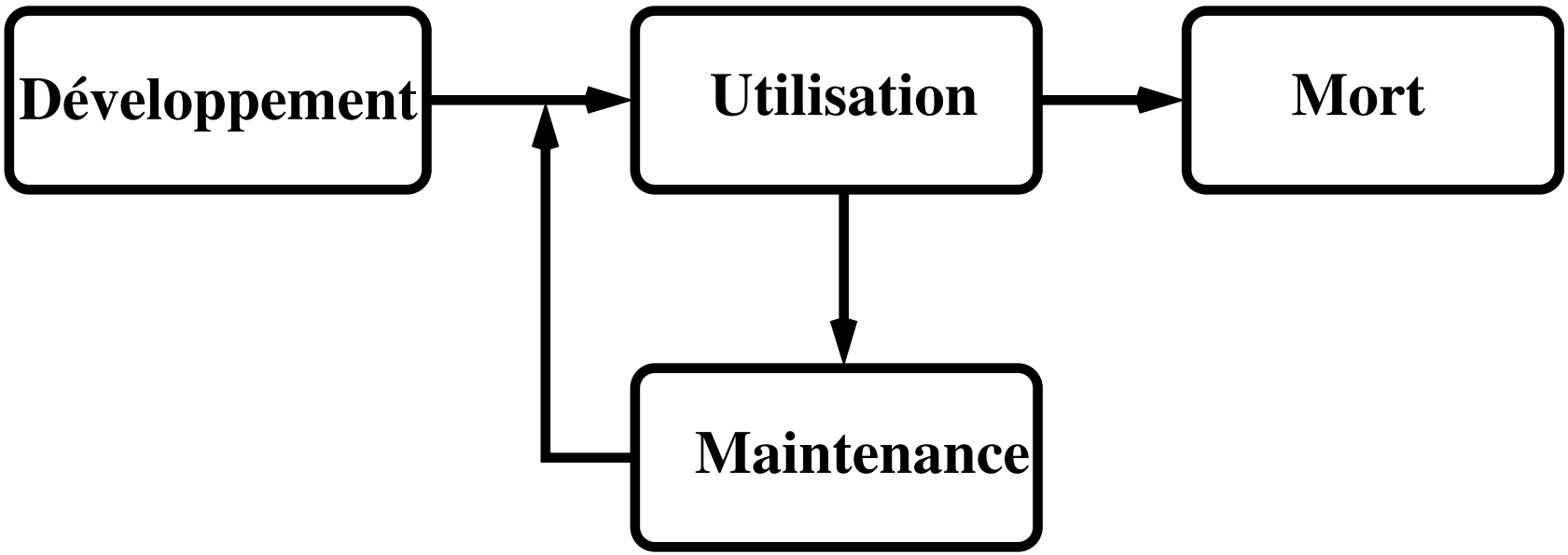


Introduction

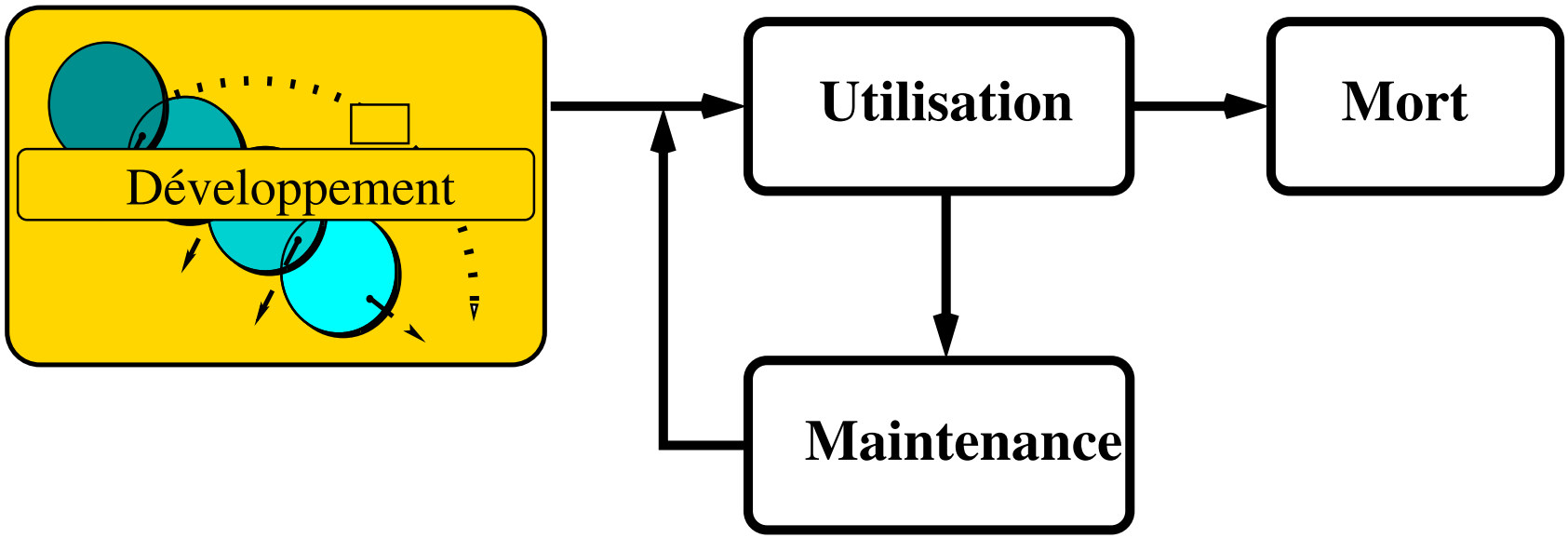
Développement = résolution de problèmes

- But ultime → programmes ;
- Étapes préliminaires :
  - Comprendre le problème (résolution de problèmes) ;
  - Analyser le problème ;
  - Concevoir une solution générale.

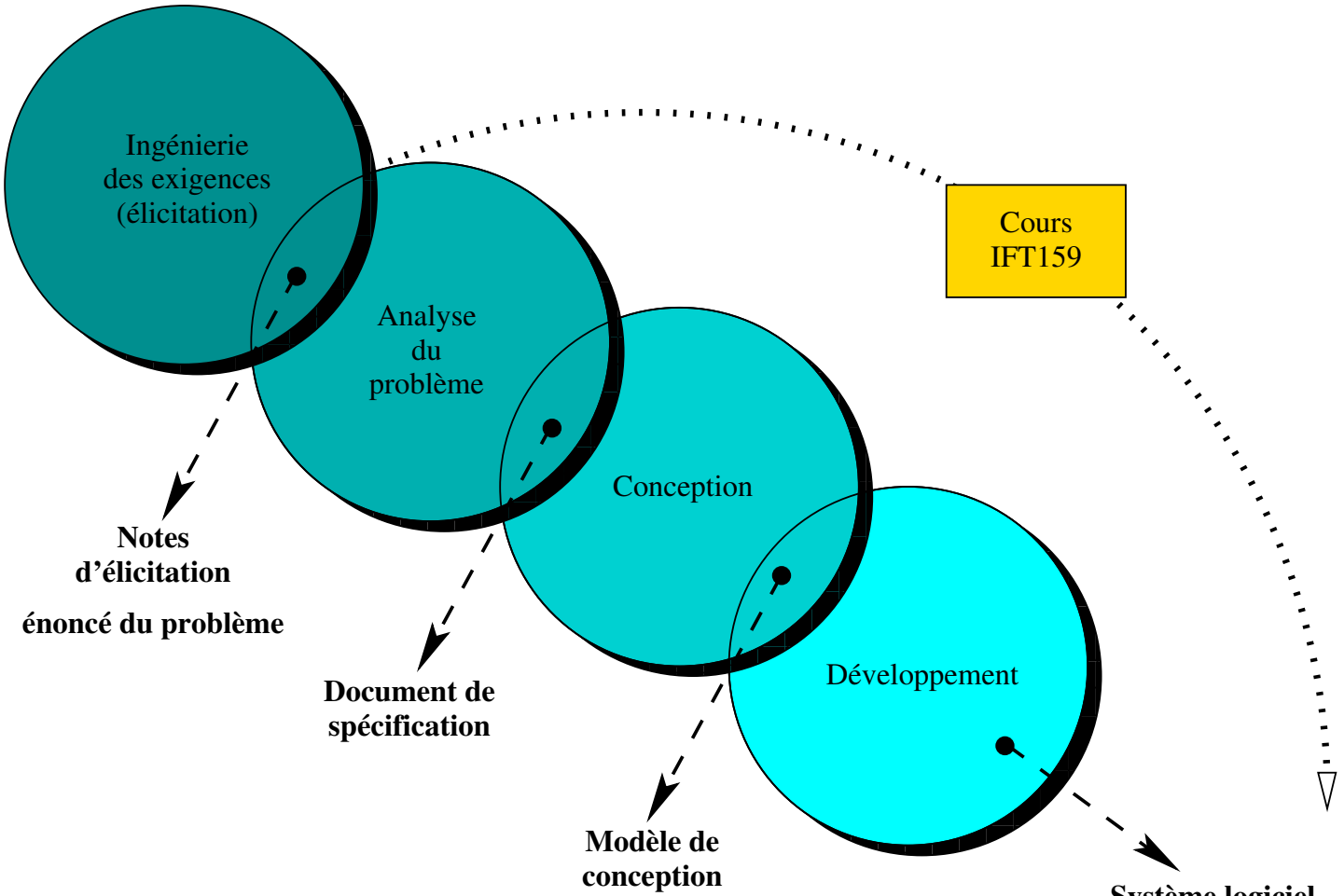
Cycle de vie d'un logiciel



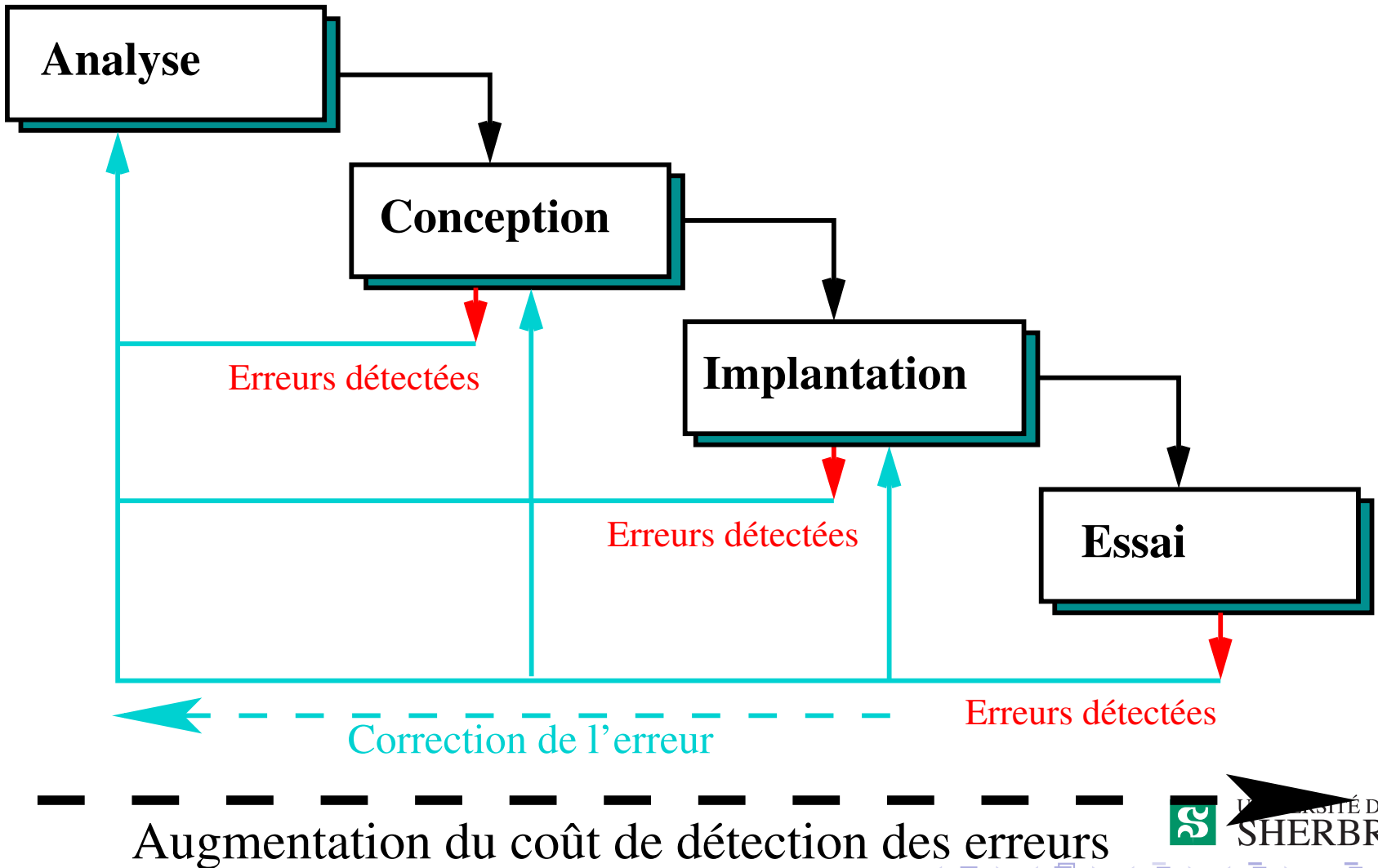
Cycle de vie d'un logiciel : Le développement



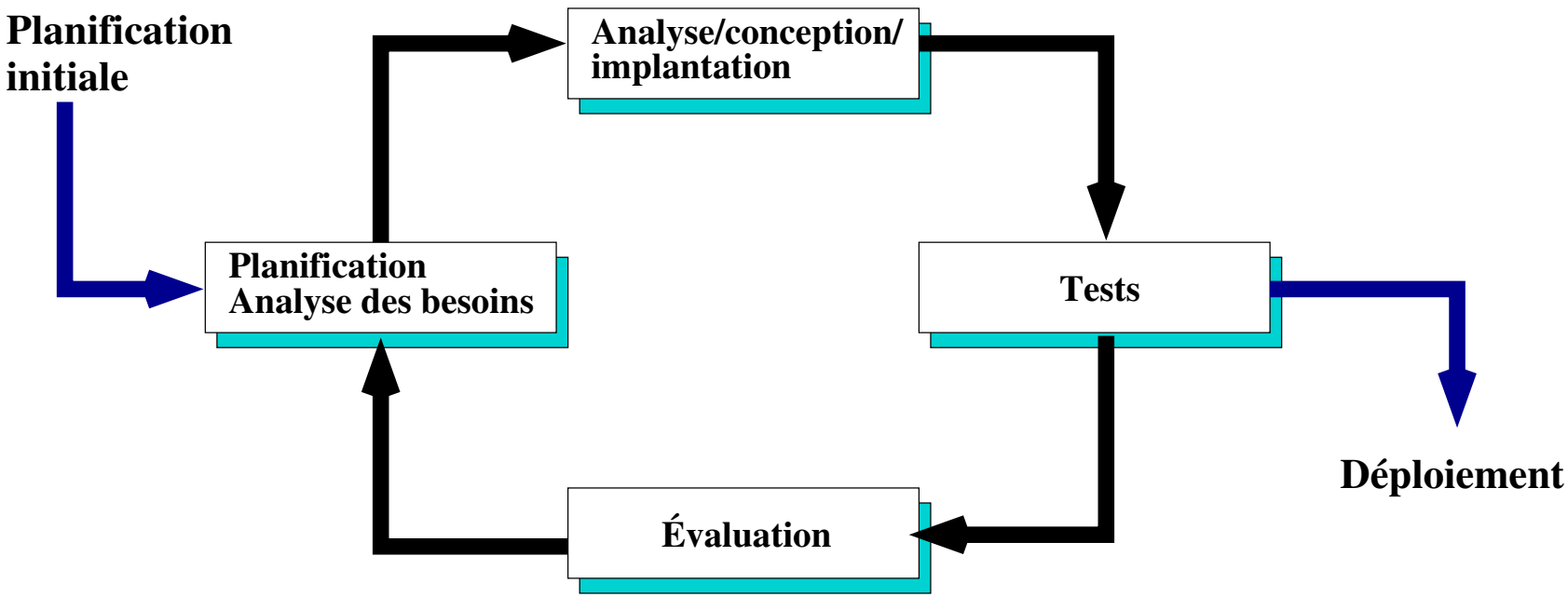
Phases de développement (version simplifiée)



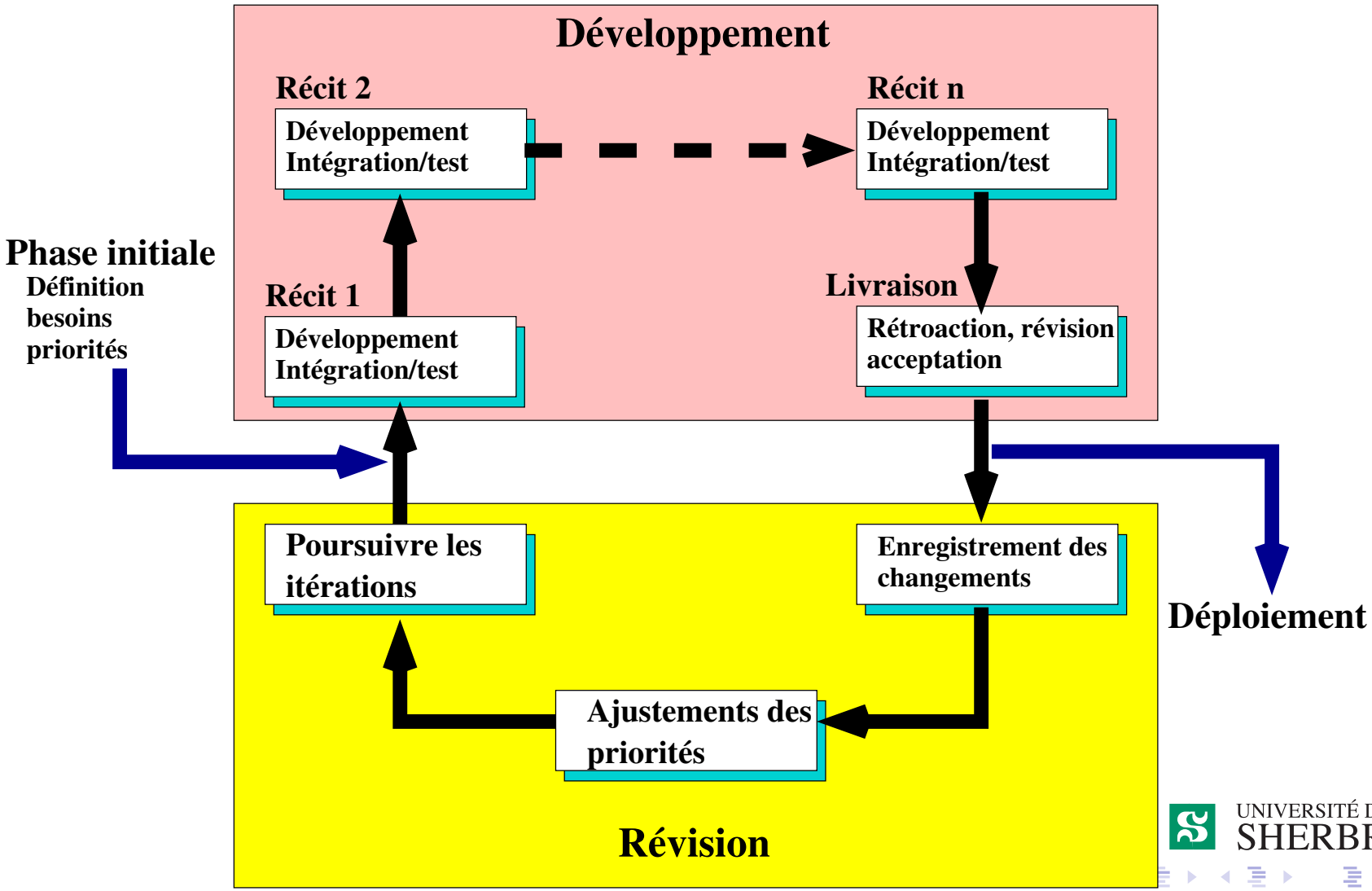
Modèle de la chute d'eau (version simplifiée)



Modèle de développement Agile



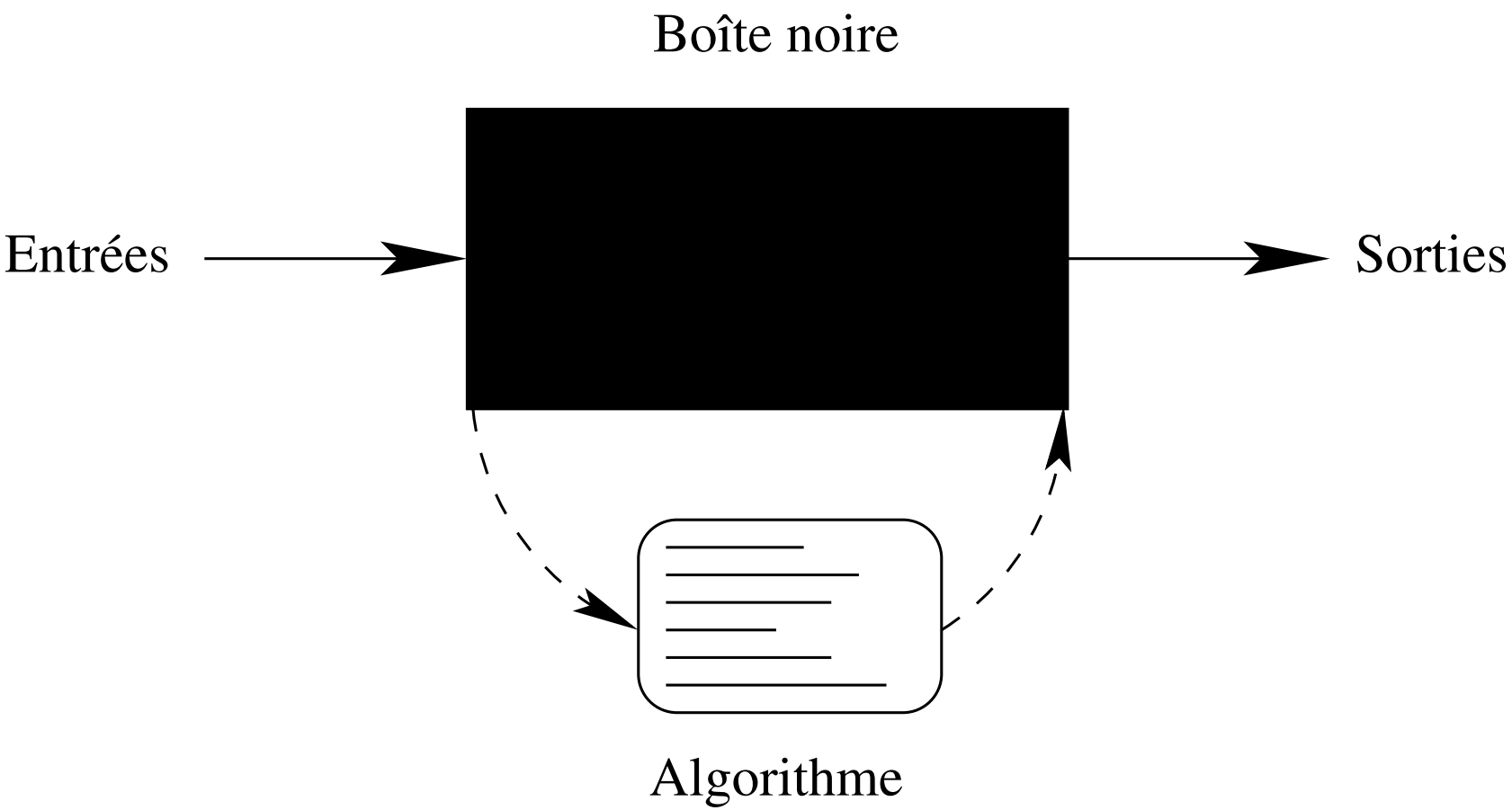
Modèle de développement Agile



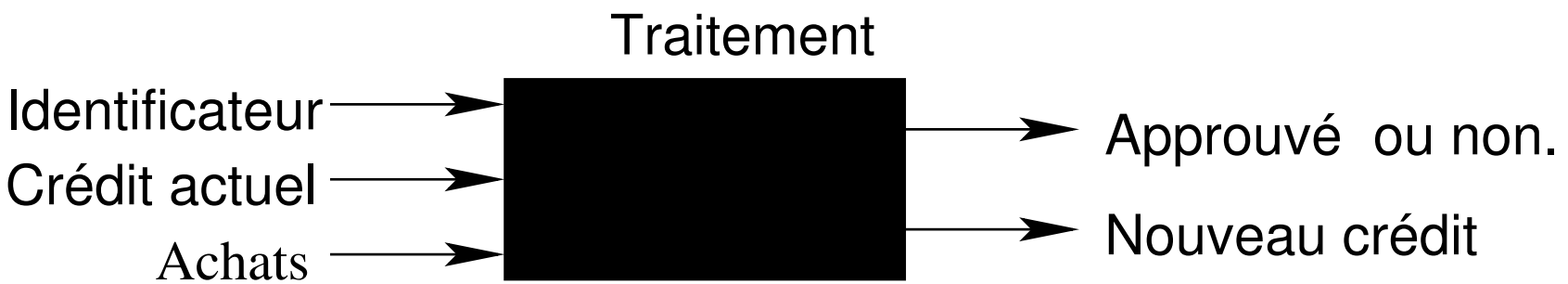
Analyse

- Déterminer les attentes de l'utilisateur ;
- Comprendre et clarifier le problème ;
- Déterminer les tenants et aboutissants ;
- IMPORTANT : séparer l'essentiel du détail, c'est-à-dire procéder par abstraction ;
- On travaille sans se préoccuper du «comment» ;

Analyse — boîte noire



Analyse — boîte noire



Analyse — méthodologie

- Comprendre le domaine du problème ;
- Définir les fonctions du logiciel ;
- Représenter le comportement du logiciel ;
- Hiérarchiser le modèle d'analyse contenant l'information, la fonction et le comportement du logiciel ;
- Aller de l'information essentielle vers les détails du système.

Questions pertinentes pour faire l'analyse

1. Quelles sont les données initiales ?
   $\Longrightarrow$ Entrées

Questions pertinentes pour faire l'analyse

1. Quelles sont les données initiales ?
   $\Longrightarrow$ Entrées
2. Par quel biais le programme les obtient-elles et à quel ensemble appartiennent-elles ?
   $\Longrightarrow$ média et types

Questions pertinentes pour faire l'analyse

1. Quelles sont les données initiales ?
   $\Longrightarrow$ Entrées
2. Par quel biais le programme les obtient-elles et à quel ensemble appartiennent-elles ?
   $\Longrightarrow$ média et types
3. Lesquelles sont à considérer comme des constantes ?
   $\Longrightarrow$ constantes

Questions pertinentes pour faire l'analyse

1. Quelles sont les données initiales ?
   $\Longrightarrow$ Entrées
2. Par quel biais le programme les obtient-elles et à quel ensemble appartiennent-elles ?
   $\Longrightarrow$ média et types
3. Lesquelles sont à considérer comme des constantes ?
   $\Longrightarrow$ constantes
4. Quels sont les résultats cherchés ?
   $\Longrightarrow$ Sorties

Questions pertinentes pour faire l'analyse

1. Quelles sont les données initiales ?
   $\Longrightarrow$ Entrées
2. Par quel biais le programme les obtient-elles et à quel ensemble appartiennent-elles ?
   $\Longrightarrow$ média et types
3. Lesquelles sont à considérer comme des constantes ?
   $\Longrightarrow$ constantes
4. Quels sont les résultats cherchés ?
   $\Longrightarrow$ Sorties
5. Que dois-en faire le programme et à quel ensemble appartiennent-ils ?
   $\Longrightarrow$ média et types

Questions pertinentes pour faire l'analyse

1. Quelles sont les données initiales ?
   $\Longrightarrow$ Entrées
2. Par quel biais le programme les obtient-elles et à quel ensemble appartiennent-elles ?
   $\Longrightarrow$ média et types
3. Lesquelles sont à considérer comme des constantes ?
   $\Longrightarrow$ constantes
4. Quels sont les résultats cherchés ?
   $\Longrightarrow$ Sorties
5. Que dois-en faire le programme et à quel ensemble appartiennent-ils ?
   $\Longrightarrow$ média et types
6. Quelles sont les relations entre les données initiales et les résultats recherchés ?
   $\Longrightarrow$ Relation entre les E/S, les formules

Objectifs de la conception

- Créer les « plans » du système

Objectifs de la conception

- Créer les « plans » du système
- Détailler son architecture (module/classe/composante *etc.*)

Objectifs de la conception

- Créer les « plans » du système
- Détailler son architecture (module/classe/composante *etc.*)
- Détailler le comportement de chaque élément

Objectifs de la conception

- Créer les « plans » du système
- Détailler son architecture (module/classe/composante *etc.*)
- Détailler le comportement de chaque élément
- Respecter les besoins explicités par l'analyse

Méthodologie de conception

- On part avec le problème principal

Méthodologie de conception

- On part avec le problème principal
- Décomposition du problème en sous-problèmes (modules)

Méthodologie de conception

- On part avec le problème principal
- Décomposition du problème en sous-problèmes (modules)
- On fait l'analyse et la conception des sous-problèmes (modules) afin de les diviser en sous-problèmes.

Méthodologie de conception

- On part avec le problème principal
- Décomposition du problème en sous-problèmes (modules)
- On fait l'analyse et la conception des sous-problèmes (modules) afin de les diviser en sous-problèmes.
- À chaque étape on ajoute des détails

Méthodologie de conception

- On part avec le problème principal
- Décomposition du problème en sous-problèmes (modules)
- On fait l'analyse et la conception des sous-problèmes (modules) afin de les diviser en sous-problèmes.
- À chaque étape on ajoute des détails
- Niveau de détails suffisants $\rightarrow$ production de la liste d'étapes à suivre (algorithme).

Questions pertinentes pour faire la conception

1. Quelles sont les étapes à réaliser (les calculs à faire) pour obtenir les résultats escomptés à partir des données du problème ?
   $\Longrightarrow$ Algorithme

Conception — Outil graphique

Nous verrons trois outils :

- Diagramme structurel
- Diagrammes UML

Réalisation

- Choix du langage en fonction des besoins et des ressources.

Réalisation

- Choix du langage en fonction des besoins et des ressources.
- Chaque module est codé dans le langage choisi.

Réalisation

- Choix du langage en fonction des besoins et des ressources.
- Chaque module est codé dans le langage choisi.
- La documentation et la lisibilité du code sont primordiaux.

Réalisation

- Choix du langage en fonction des besoins et des ressources.
- Chaque module est codé dans le langage choisi.
- La documentation et la lisibilité du code sont primordiaux.
- $\Longrightarrow$ La maintenance coûte très cher.

Essais — l'importance

- Coûts des erreurs annuellement ?

Essais — l'importance

- Coûts des erreurs annuellement ?

Essais — l'importance

- Coûts des erreurs annuellement ?

$ 59,5 milliards

$ 59,5 milliards en 2002 juste aux États-Unis

Essais — l'importance

- Coûts des erreurs annuellement ?

$ 59,5 milliards

$ 59,5 milliards en 2002 juste aux États-Unis

- Combien aurait-on pu économiser avec de meilleurs tests ?

Essais — l'importance

- Coûts des erreurs annuellement ?

$ 59,5 milliards

$ 59,5 milliards en 2002 juste aux États-Unis

- Combien aurait-on pu économiser avec de meilleurs tests ?

Essais — l'importance

- Coûts des erreurs annuellement ?

$ 59,5 milliards

$ 59,5 milliards en 2002 juste aux États-Unis

- Combien aurait-on pu économiser avec de meilleurs tests ?

$ 22,2 milliards

Le tier de ces coûts ($ 22,2 milliards) aurait pu être évité par une meilleure infra-structure de tests.

Essais — l'importance – Exemples

- Ariane 5 :

Essais — l'importance – Exemples

- Ariane 5 : coût $\equiv$ \$ 500 millions

Essais — l'importance – Exemples

- Ariane 5 : coût $\equiv$ \$ 500 millions
- Therac-25 - machine de radiothérapie :

Essais — l'importance – Exemples

- Ariane 5 : coût $\equiv$ \$ 500 millions
- Therac-25 - machine de radiothérapie : coût $\equiv$ 3 morts, 3 blessés graves

Essais — l'importance – Exemples

- Ariane 5 : coût ≡ $ 500 millions
- Therac-25 - machine de radiothérapie : coût ≡ 3 morts, 3 blessés graves
- Échec du missile Patriot :

Essais — l'importance – Exemples

- Ariane 5 : coût $\equiv$ \$ 500 millions
- Therac-25 - machine de radiothérapie : coût $\equiv$ 3 morts, 3 blessés graves
- Échec du missile Patriot : coût $\equiv$ 28 morts, 100 blessés

Essais — l'importance – Exemples

- Ariane 5 : coût $\equiv$ \$ 500 millions
- Therac-25 - machine de radiothérapie : coût $\equiv$ 3 morts, 3 blessés graves
- Échec du missile Patriot : coût $\equiv$ 28 morts, 100 blessés
- Écrasement du *Mars Climate Orbiter* :

Essais — l'importance – Exemples

- Ariane 5 : coût ≡ $ 500 millions
- Therac-25 - machine de radiothérapie : coût ≡ 3 morts, 3 blessés graves
- Échec du missile Patriot : coût ≡ 28 morts, 100 blessés
- Écrasement du *Mars Climate Orbiter* : coût ≡ $ 125 millions

Essais — l'importance – Exemples

- Ariane 5 : coût ≡ $ 500 millions
- Therac-25 - machine de radiothérapie : coût ≡ 3 morts, 3 blessés graves
- Échec du missile Patriot : coût ≡ 28 morts, 100 blessés
- Écrasement du *Mars Climate Orbiter* : coût ≡ $ 125 millions
- Bug de l'an 2000 :

Essais — l'importance – Exemples

- Ariane 5 : coût ≡ $ 500 millions
- Therac-25 - machine de radiothérapie : coût ≡ 3 morts, 3 blessés graves
- Échec du missile Patriot : coût ≡ 28 morts, 100 blessés
- Écrasement du *Mars Climate Orbiter* : coût ≡ $ 125 millions
- Bug de l'an 2000 : coût ≡ $ 500 milliards

Essais — l'importance – Exemples

- Ariane 5 : coût ≡ $ 500 millions
- Therac-25 - machine de radiothérapie : coût ≡ 3 morts, 3 blessés graves
- Échec du missile Patriot : coût ≡ 28 morts, 100 blessés
- Écrasement du *Mars Climate Orbiter* : coût ≡ $ 125 millions
- Bug de l'an 2000 : coût ≡ $ 500 milliards
- ... Il y a plein d'autres exemples...

Essais — l'importance

- La phase de tests est la partie la plus coûteuse de l'ingénierie logicielle.

Essais — l'importance

- La phase de tests est la partie la plus coûteuse de l'ingénierie logicielle.
- Détection tardive des erreurs $\Rightarrow$ hausse du coût exponentielle.

Essais — l'importance

- La phase de tests est la partie la plus coûteuse de l'ingénierie logicielle.
- Détection tardive des erreurs $\Rightarrow$ hausse du coût exponentielle.
- Établir une stratégie de tests (jeu de tests) au plus tôt.

Essais — l'importance

- La phase de tests est la partie la plus coûteuse de l'ingénierie logicielle.
- Détection tardive des erreurs ⇒ hausse du coût exponentielle.
- Établir une stratégie de tests (jeu de tests) au plus tôt.
- Des tests hasardeux ⇒ erreurs critiques.

Essais — l'importance

- La phase de tests est la partie la plus coûteuse de l'ingénierie logicielle.
- Détection tardive des erreurs $\Rightarrow$ hausse du coût exponentielle.
- Établir une stratégie de tests (jeu de tests) au plus tôt.
- Des tests hasardeux $\Rightarrow$ erreurs critiques.
- Il faut se poser deux questions pour chaque étape :

Essais — l'importance

- La phase de tests est la partie la plus coûteuse de l'ingénierie logicielle.
- Détection tardive des erreurs ⇒ hausse du coût exponentielle.
- Établir une stratégie de tests (jeu de tests) au plus tôt.
- Des tests hasardeux ⇒ erreurs critiques.
- Il faut se poser deux questions pour chaque étape :
  1. Construisons nous le bon produit ? (validation)

Essais — l'importance

- La phase de tests est la partie la plus coûteuse de l'ingénierie logicielle.
- Détection tardive des erreurs ⇒ hausse du coût exponentielle.
- Établir une stratégie de tests (jeu de tests) au plus tôt.
- Des tests hasardeux ⇒ erreurs critiques.
- Il faut se poser deux questions pour chaque étape :
  1. Construisons nous le bon produit ? (validation)
  2. Construisons nous le produit correctement ? (vérification)

Essais — l'importance

- La phase de tests est la partie la plus coûteuse de l'ingénierie logicielle.
- Détection tardive des erreurs ⇒ hausse du coût exponentielle.
- Établir une stratégie de tests (jeu de tests) au plus tôt.
- Des tests hasardeux ⇒ erreurs critiques.
- Il faut se poser deux questions pour chaque étape :
  1. Construisons nous le bon produit ? (validation)
  2. Construisons nous le produit correctement ? (vérification)

Essais — l'importance

- La phase de tests est la partie la plus coûteuse de l'ingénierie logicielle.
- Détection tardive des erreurs ⇒ hausse du coût exponentielle.
- Établir une stratégie de tests (jeu de tests) au plus tôt.
- Des tests hasardeux ⇒ erreurs critiques.
- Il faut se poser deux questions pour chaque étape :
  1. Construisons nous le bon produit ? (validation)
  2. Construisons nous le produit correctement ? (vérification)

Attention

Les tests découvrent des erreurs mais ne prouvent rien.

Essais — les catégories

- Tests unitaire $\Rightarrow$ valide l'implémentation.

Essais — les catégories

- Tests unitaire $\Rightarrow$ valide l'implémentation.
- Tests d'intégration $\Rightarrow$ valide le modèle de conception.

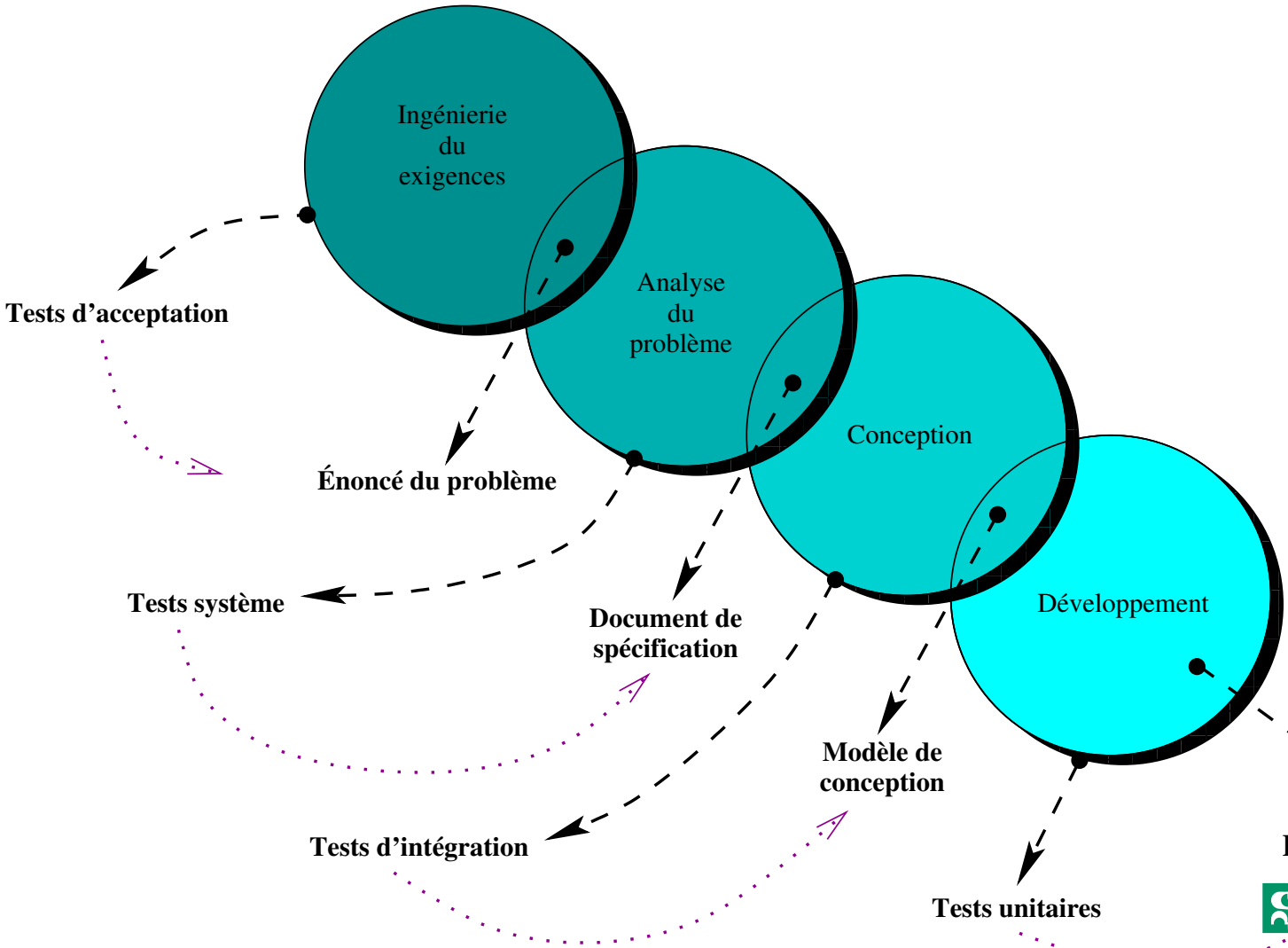
Essais — les catégories

- Tests unitaire $\Rightarrow$ valide l'implémentation.
- Tests d'intégration $\Rightarrow$ valide le modèle de conception.
- Tests systèmes $\Rightarrow$ valide le modèle d'analyse (spécification).

Essais — les catégories

- Tests unitaire ⇒ valide l'implémentation.
- Tests d'intégration ⇒ valide le modèle de conception.
- Tests systèmes ⇒ valide le modèle d'analyse (spécification).
- Tests d'acceptation ⇒ valide le produit (s'il répond aux besoins du client).

Cycle de tests



Prévisions des coûts

Définition de l'effort.

L'**effort** est le produit du nombre de personnes par le nombre d'heures disponible par personne.

Dans le cours, il faut prévoir

Prévisions des coûts

Définition de l'effort.

L'**effort** est le produit du nombre de personnes par le nombre d'heures disponible par personne.

Dans le cours, il faut prévoir

- Analyse entre 5 % et 15 % des efforts

Prévisions des coûts

Définition de l'effort.

L'**effort** est le produit du nombre de personnes par le nombre d'heures disponible par personne.

Dans le cours, il faut prévoir

- Analyse entre 5 % et 15 % des efforts
- Conception entre 20 % et 50 % des efforts

Prévisions des coûts

Définition de l'effort.

L'**effort** est le produit du nombre de personnes par le nombre d'heures disponible par personne.

Dans le cours, il faut prévoir

- Analyse entre 5 % et 15 % des efforts
- Conception entre 20 % et 50 % des efforts
- Implémentation entre 10 % et 30 % des efforts

Prévisions des coûts

Définition de l'effort.

L'**effort** est le produit du nombre de personnes par le nombre d'heures disponible par personne.

Dans le cours, il faut prévoir

- Analyse entre 5 % et 15 % des efforts
- Conception entre 20 % et 50 % des efforts
- Implémentation entre 10 % et 30 % des efforts
- Tests entre 20 % et 50 % des efforts

Planification des efforts

Format du document

Document disponible sur internet.

Planification des efforts

Format du document

- Nombre d'étudiant(e)s dans l'équipe

Document disponible sur internet.

Planification des efforts

Format du document

- Nombre d'étudiant(e)s dans l'équipe
- Temps prévu (en *heure*)

Document disponible sur internet.

Planification des efforts

Format du document

- Nombre d'étudiant(e)s dans l'équipe
- Temps prévu (en *heure*)
- Pour chaque activité décrire
  - Activité
  - Temps prévu (en *heure*)
  - Temps effectivement consacré (en *heure*)
  - Justification de la différence

Document disponible sur internet.

Planification des efforts

Format du document

- Nombre d'étudiant(e)s dans l'équipe
- Temps prévu (en *heure*)
- Pour chaque activité décrire
  - Activité
  - Temps prévu (en *heure*)
  - Temps effectivement consacré (en *heure*)
  - Justification de la différence
- Effort consommé total (en *heure*)

Document disponible sur internet.

Étude de cas
Consommation automobile

Entre deux pleins d'essence un automobiliste désire connaître sa consommation d'essence.

Loi de Murphy (suite)

- Laissées à elles-mêmes, les choses tendent à aller de mal en pis.
- Même si quelque chose ne peut pas aller mal, cela ira mal quand même.
- Si tout semble fonctionner correctement, alors vous avez manifestement oublié quelque chose.
- La nature nous réserve toujours des pannes.

Introduction

UNIVERSITÉ DE SHERBROOKE

Confidentialité et mauvais usage

- Les informaticiens peuvent avoir accès à des données sensibles.
  → Ne pas :
  - les accéder si cela n'est pas utile pour la tâche
  - les utiliser pour des gains personnels
  - les utiliser pour des activités illégales, non éthiques ou nuisibles à autrui
  - les modifier pour nos besoins personnels
- Doivent faire comme les médecins ou les avocats

Effraction informatique (*computer hacking*)

- Cette activité est illégale peu importe l'objectif visé
- Elle peut entraîner des poursuites judiciaires
- Créer/propager un virus peut être considéré comme une effraction informatique

Plagia ou piratage

- Ne pas utiliser les programmes des autres sans leur permission
- Ne pas modifier les programmes d'un autre et affirmer que c'est le nôtre
- Ne pas pirater de logiciels ou produits commerciaux

Mauvaise utilisation des ressources informatiques

- Les codes et droits d'accès sont des propriétés privées
- Ils vous sont attribués pour des besoins précis
- Ne pas les prêter, les louer ou les partager
- Ne pas les utiliser pour d'autres fins que celles prévues

En cas de doute....

On s'informe ! ! !

Conclusion

- Edger W. Dijkstra
  “If debugging is the process of removing bugs, then programming must be the process of putting them in.”
- Alan J. Perlis
  “There are two ways to write error-free programs ; only the third one works.”

Unités de mesure

Multiples de l'octet : préfixes SI et mésusages

Nom	Symbole	Valeur	Mésusage
kilooctet	ko	10^3	2^{10}
mégaoctet	Mo	10^6	2^{20}
gigaoctet	Go	10^9	2^{30}
téraoctet	To	10^{12}	2^{40}
pétaoctet	Po	10^{15}	
exaoctet	Eo	10^{18}	
zettaoctet	Zo	10^{21}	
yottaoctet	Yo	10^{24}	

Unités de mesure

Multiples de l'octet : préfixes binaires
(Normes établies par la Commission électrotechnique internationale)

Nom	Symbole	Valeur
kibioctet	Kio	2^{10}
mébioctet	Mio	2^{20}
gibioctet	Gio	2^{30}
tébioctet	Tio	2^{40}
pébioctet	Pio	2^{50}
exbioctet	Eio	2^{60}
zébioctet	Zio	2^{70}
yobioctet	Yio	2^{80}