
# Génie logiciel

# Modèle de développement

Louis-Edouard LAFONTANT

geodes

Université de Montréal

# Organisation du cours

StudiUM

Slides du cours et démos

Quiz et révisions

Remises des travaux

Ressources et liens rapides

Annonces et Questions-réponses

Communication direct avec les enseignants

Travail en équipe

Lien: https://discord.gg/vJQDrbtGEN


Plan de cours

Suivi de la semaine (calendrier + résumé)

Ressources et liens rapides

Lien: https://ceduni.github.io/udem-ift2255/aut2022.html

# Projet

Permettre à chacun de faire le suivi de ces déchets domestiques

- 3 remises
  - Remise 1: 10%
  - Remise 2: 15%
  - Remise 3: 25%
- Travail en équipe de 3 ou 4
- Langage de programmation: Java
- Rapport en HTML (web)
- Visite d’un expert (possible ☝ )

**Ceci ☝ est un vrai projet!**

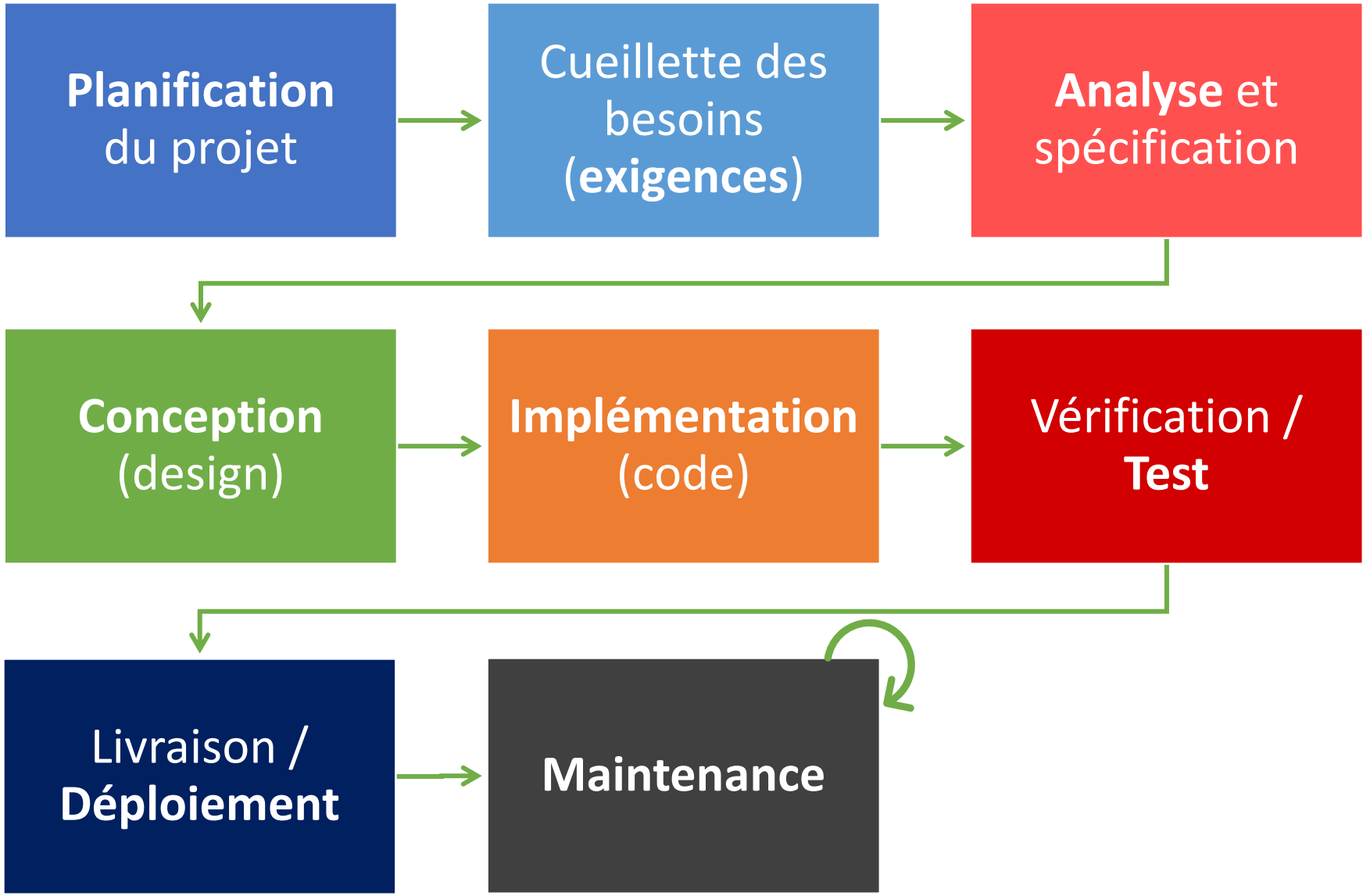
# Génie

**Rigueur et Garantie**

- Méthodologie
- Outils
- Observation
- Analyse
- Conception
- Éthique
- …

- Esprit d’équipe & Communication
- Imagination & Créativité

# Activités


**Planification** du projet
Cueillette des besoins (**exigences**)
**Analyse** et spécification
**Conception** (design)
**Implémentation** (code)
Vérification / **Test**
Livraison / **Déploiement**
**Maintenance**

# Facteurs décisifs

**Distance entre les acteurs**
**→ Communication**

- Fréquence (visite, réunion, échanges d'informations)
- Domaine (connaissances, capacité de comprendre l'information)

**Contraintes sur ressources**
**→ Cout**

- Matériel (physique, logiciel, personne)
- Temps (délai à respecter)
- Rareté (compétences, outil inexistant)

# Processus

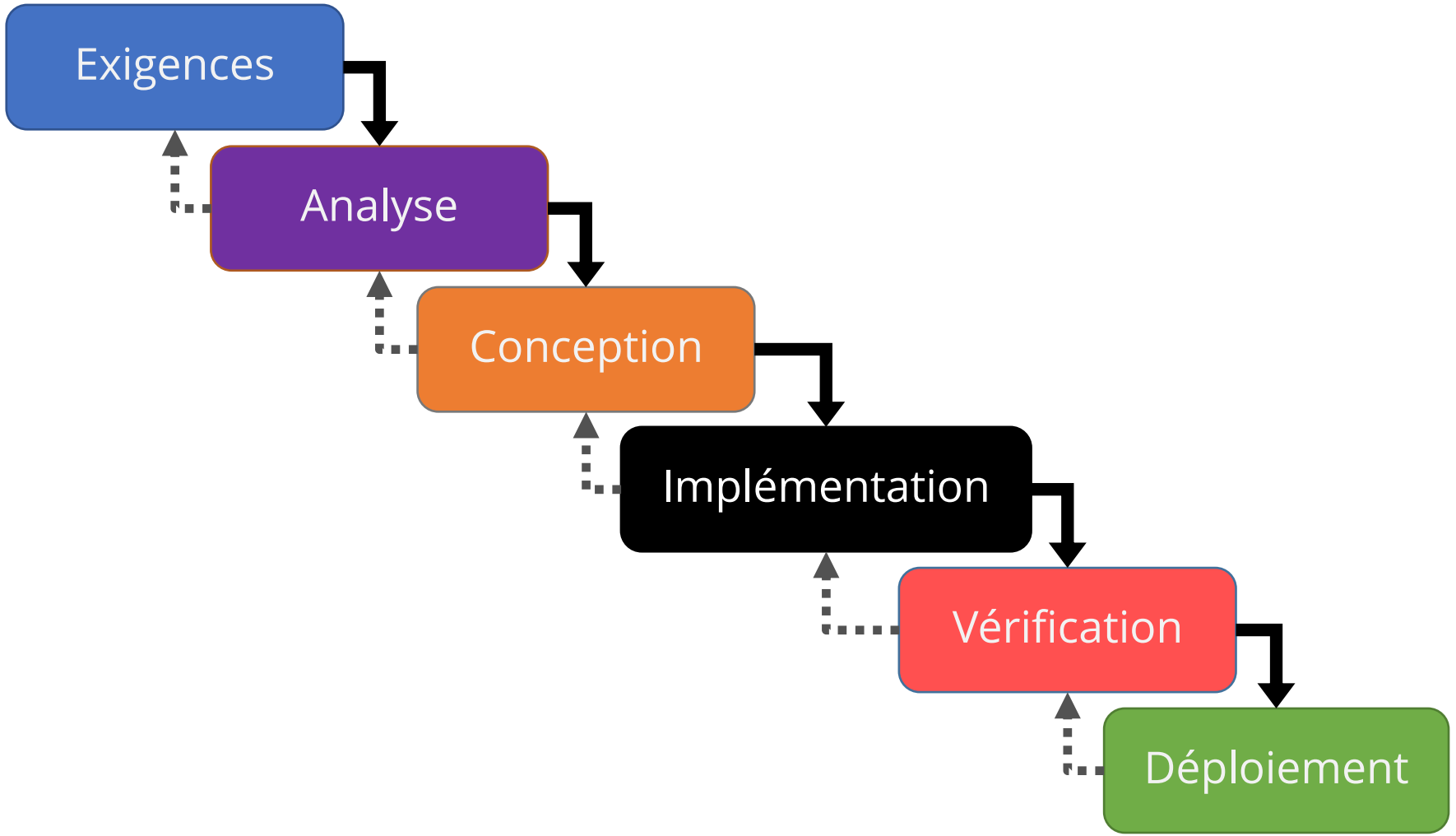
*La manière selon laquelle une organisation produit un logiciel*

- Méthodologie: Ensemble d'activités **ordonnées**
- Modèle de développement du logiciel
- Techniques et outils à utiliser
- Suivi du projet
- Rôles et individus nécessaires

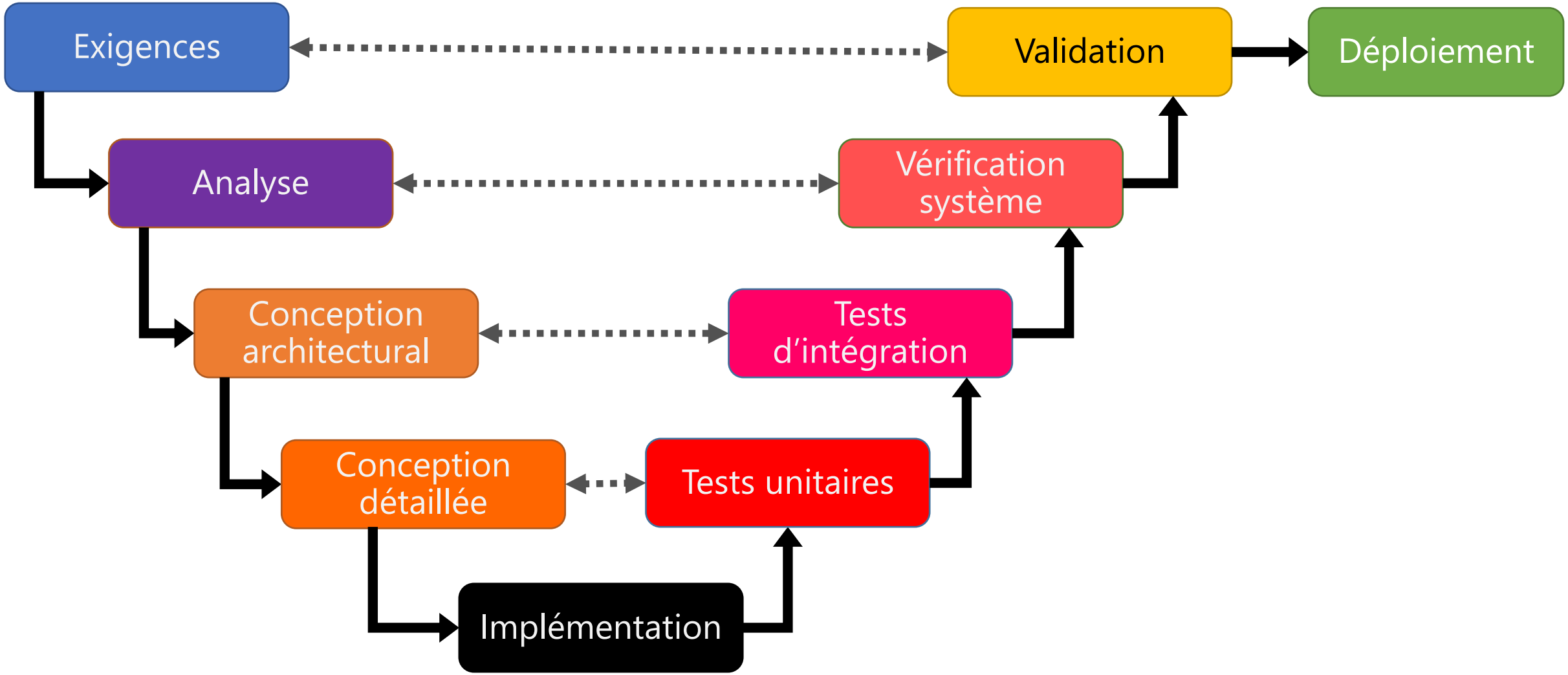
# Processus linéaire

Cascade → Modèle en V

# Cascade


Exigences
Analyse
Conception
Implémentation
Vérification
Déploiement

# Modèle en V


Exigences
Validation
Déploiement
Analyse
Vérification système
Conception architectural
Tests d'intégration
Conception détaillée
Tests unitaires
Implémentation

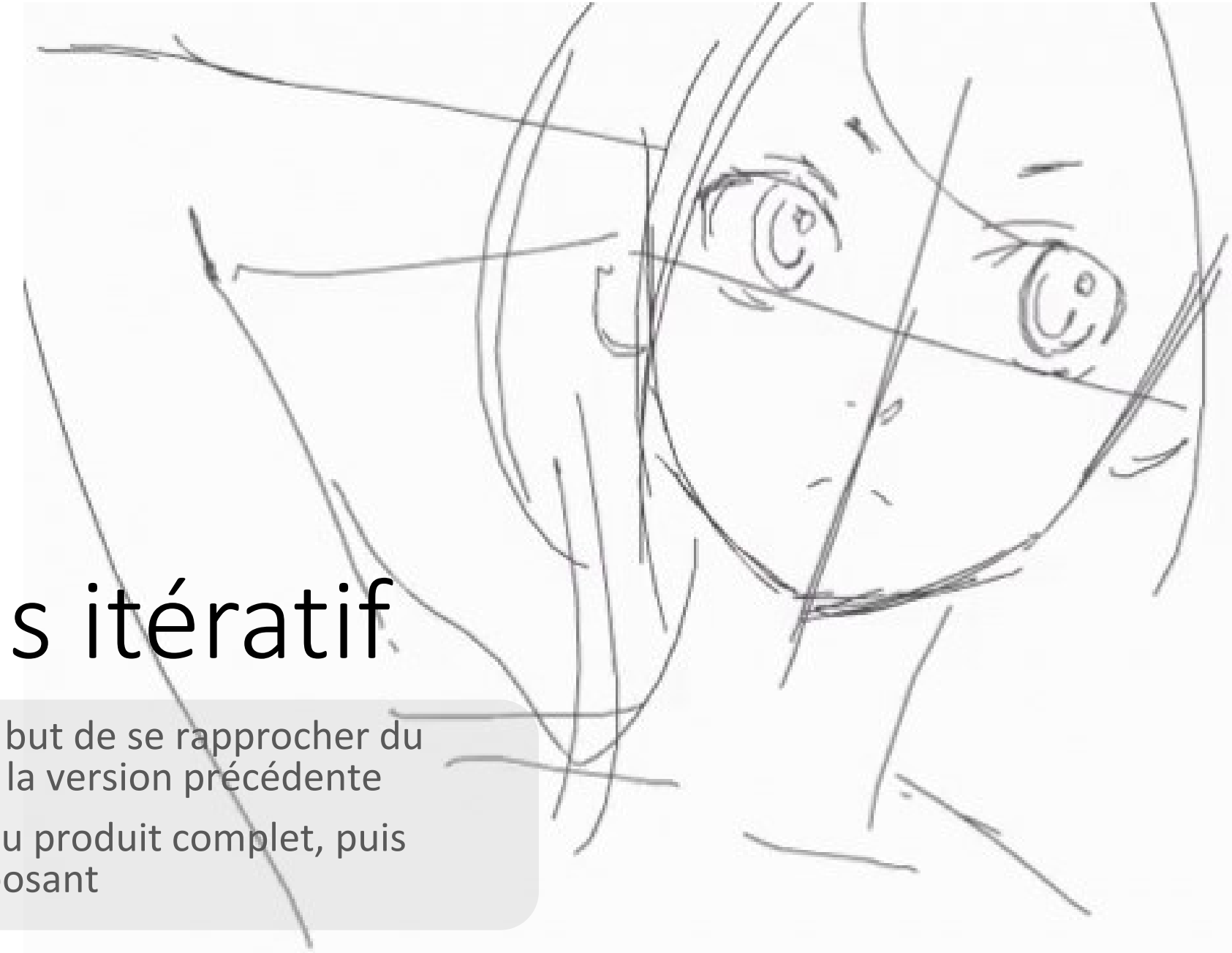
# Processus linéaire
## **Caractéristique**

### Avantages

- Simple et facile à suivre
- Axé sur la **documentation**
- Permets une conception bien pensée

### Inconvénient

- Purement **linéaire**
- Trop **rigide**
- Pas de **feedback** du client avant la livraison
- **Vérification** tardive

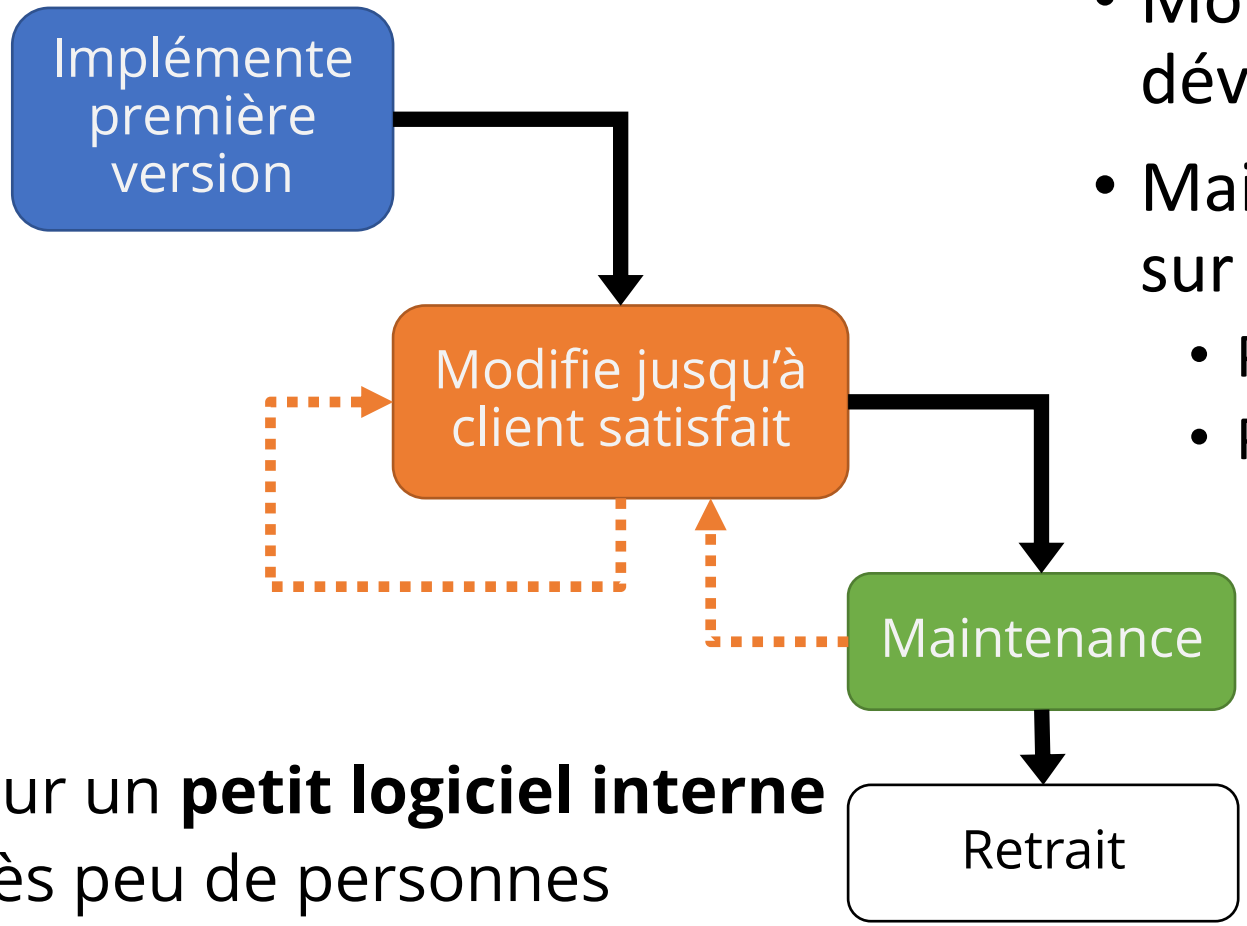


# Processus itératif

Chaque version a pour but de se rapprocher du système cible plus que la version précédente

Architecte la coquille du produit complet, puis améliore chaque composant

# Code-et-modifie (**pas recommandé**)

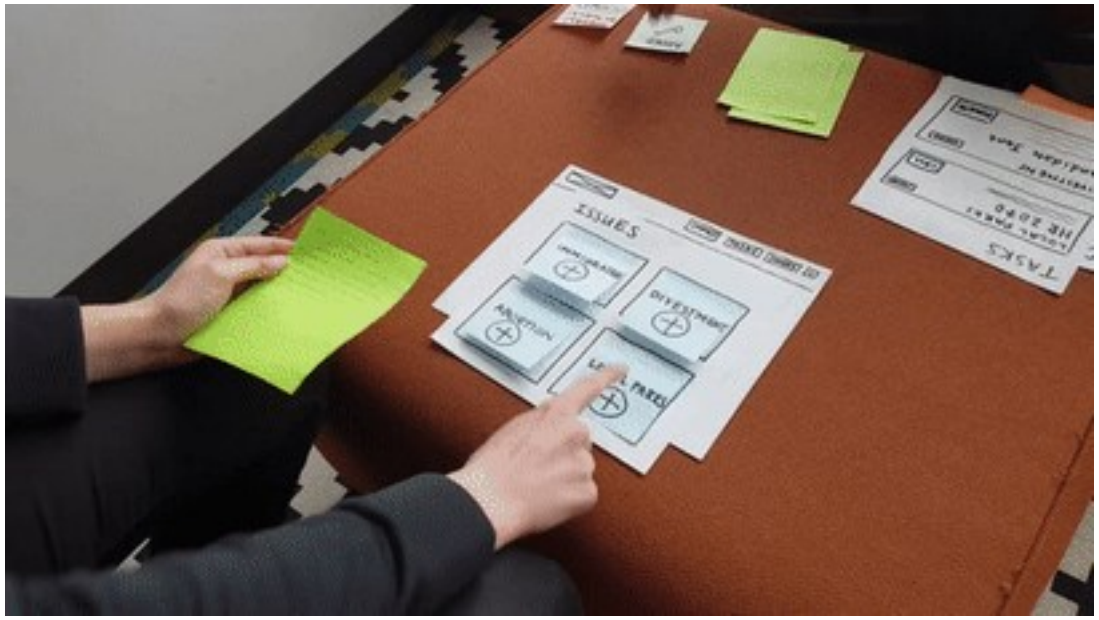



- Moyen le plus facile de développer un logiciel
- Mais aussi le plus cher sur le long terme
  - Pas de conception
  - Pas de spécification

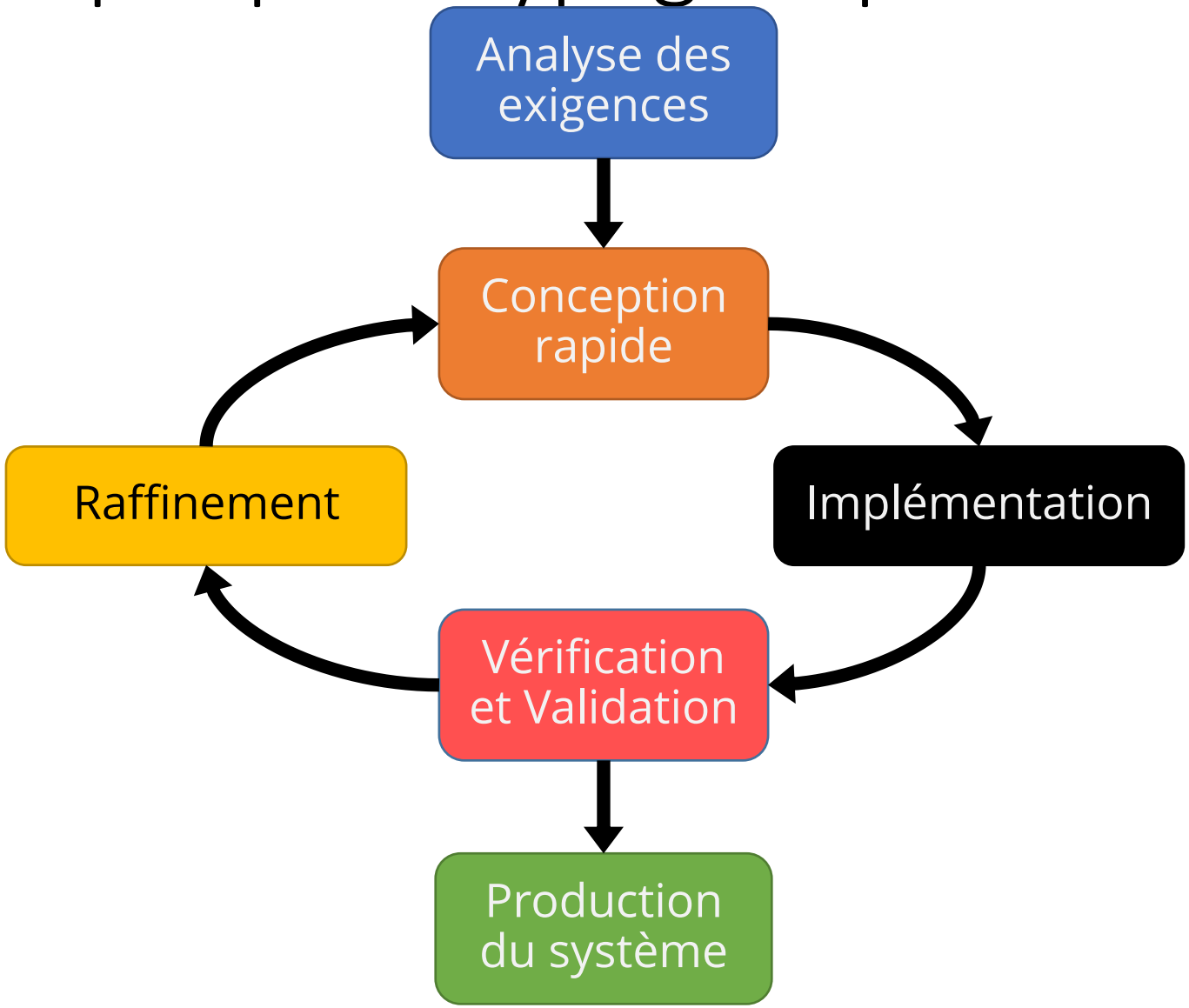
Pertinent pour un **petit logiciel interne** utilisé par très peu de personnes

# Processus par prototypage rapide

- Pertinent pour les projets où les exigences ne sont pas clairement définies
- **Prototype**
  - Jetable: Compréhension du client + Évaluation d'alternatives
  - Évolutif: réutilisé à chaque itération jusqu'au produit final



# Processus par prototypage rapide



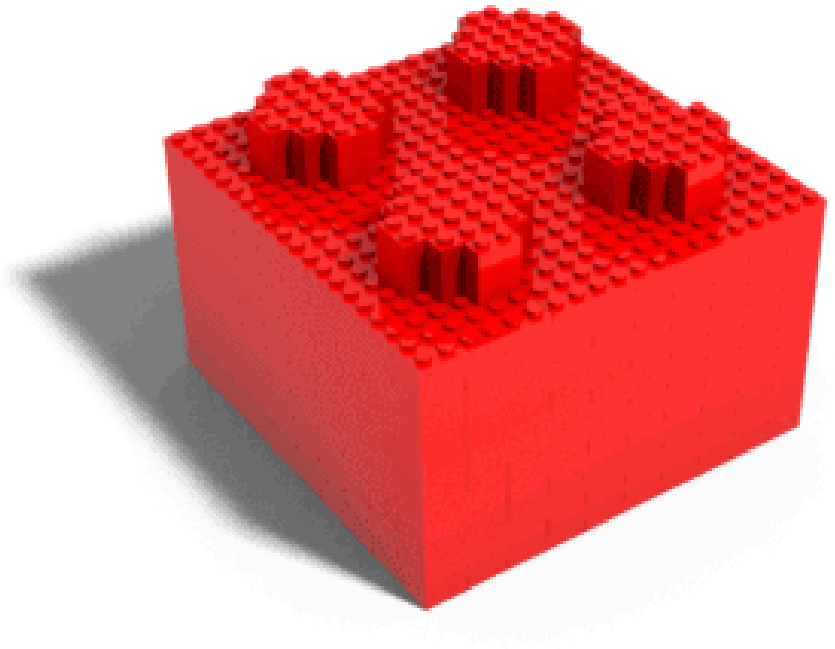
# Processus itératif
## **Caractéristique**

### Avantages

- **Réutilisation** de prototypes
- Produit visible très tôt
- Souci de **vérification** et validation du **client** anticipé
- Intégration facilitée, moins de raffinement

### Inconvénient

- **Retravaille** chaque itération
- Produit livré à la fin

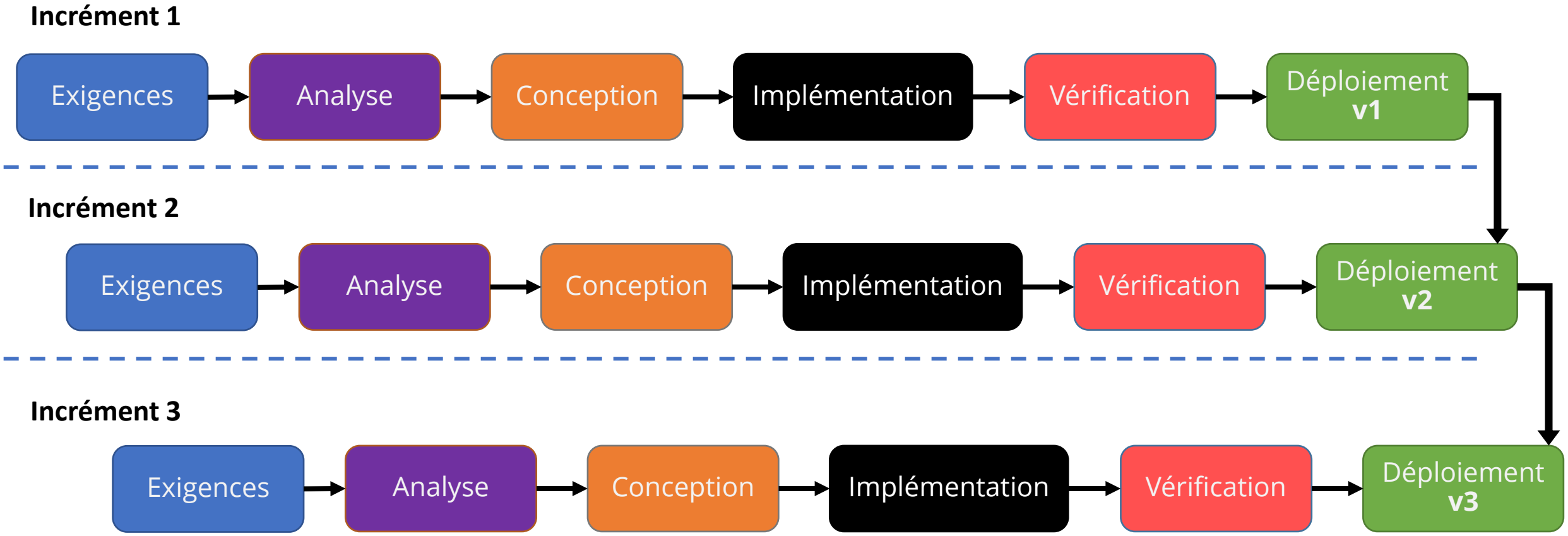



# Processus incrémental

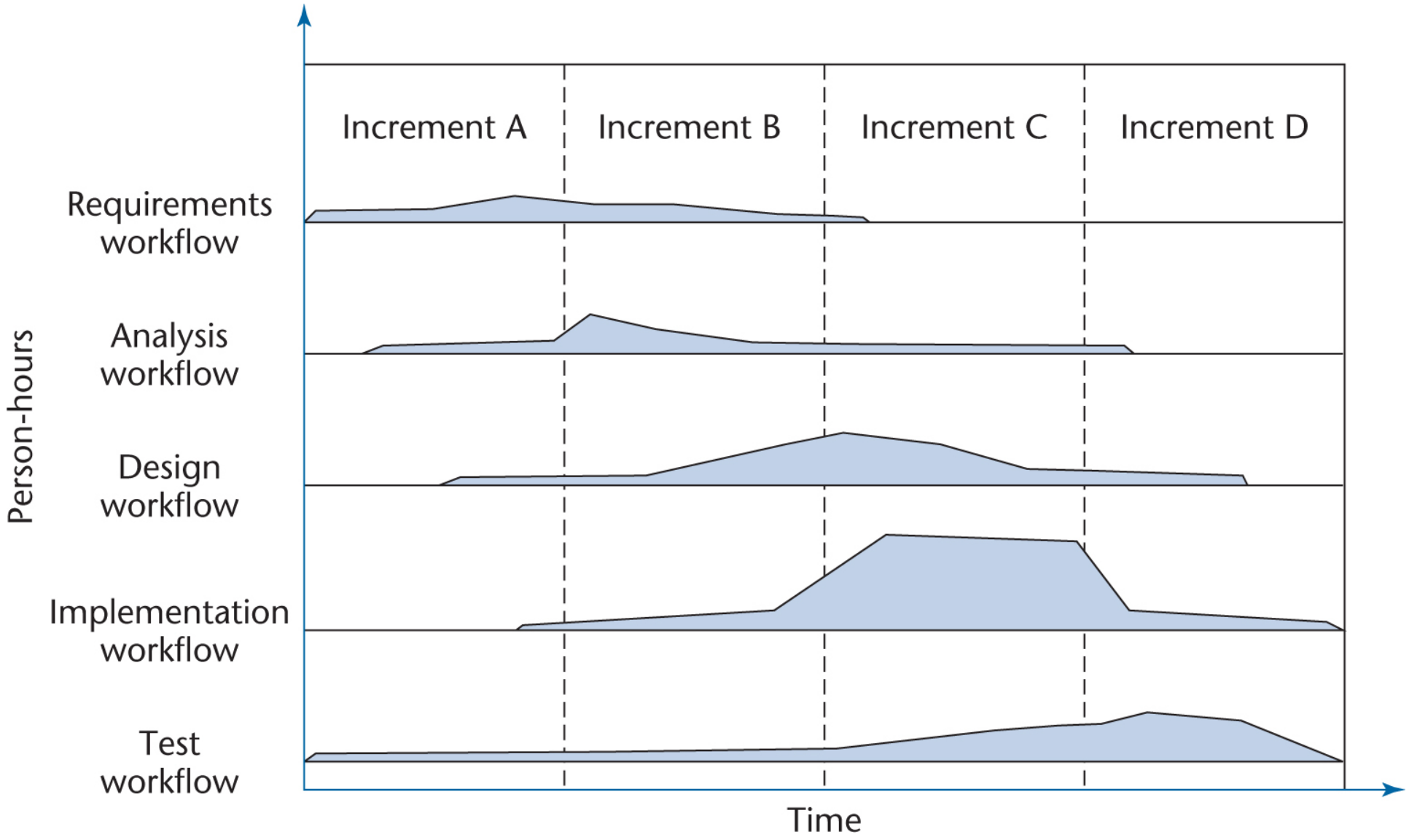
Pour gérer plus information, utiliser le **raffinement par étapes *(stepwise refinement)***

Chaque incrément abouti à une livraison

# Processus incrémental



# Processus incrémentaux



# Processus incrémental
## **Caractéristique**

### Avantages

- Développer par ordre de **priorité**
- **Livraison** de composants rapides
- Facilement mesurer le **progrès**

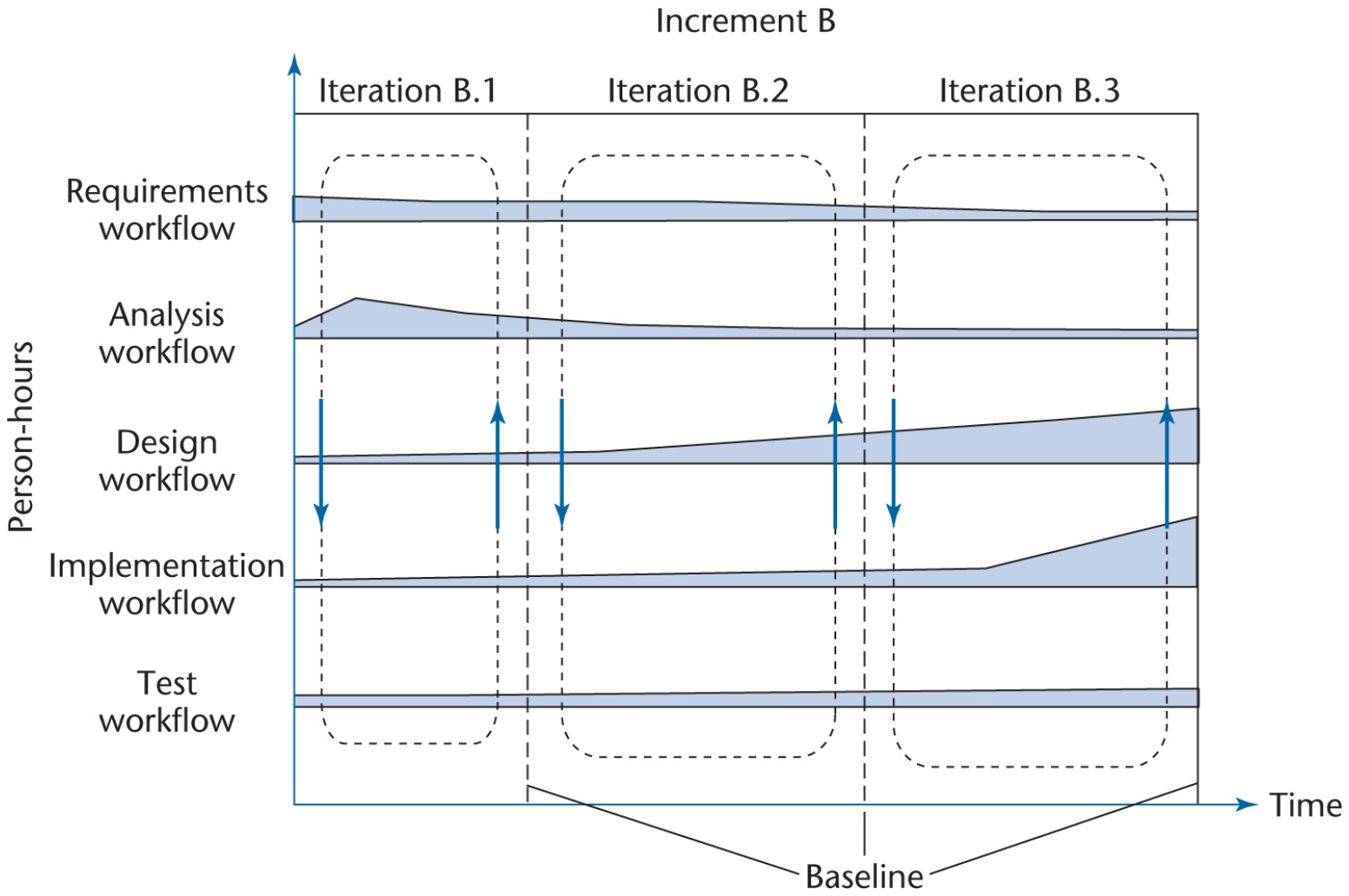
### Inconvénient

- Pas de **processus** visible et clair à suivre
- Tache d'**intégration** prend plus d'importance

# Itératif ou Incrémental?

Pourquoi pas les deux?

# Itération et incrémentation (I&I)



# Avantages du I&I

- Tous les flux d'activités *(workflow)* sont impliqués dans chaque incrément, mais certains vont **dominer** plus
- Plusieurs opportunités de **tester**, recevoir du feedback et s'ajuster
- **Robustesse** de **l'architecture** peut être déterminée **tôt** dans le développement
- **Livrables spécifiques** pour chaque incrément et chaque workflow
- On peut atténuer et résoudre les **risques** plus tôt

# Processus unifié

Jacobson, Booch, Rumbaugh 1999