Cycle de vie du développement logiciel

TCH099 - Projet intégrateur en informatique

Chargée de cours : Saida Khazri

Responsables: Abdelmoumène Toudeft, Iannick Gagnon

Le génie pour l'industrie

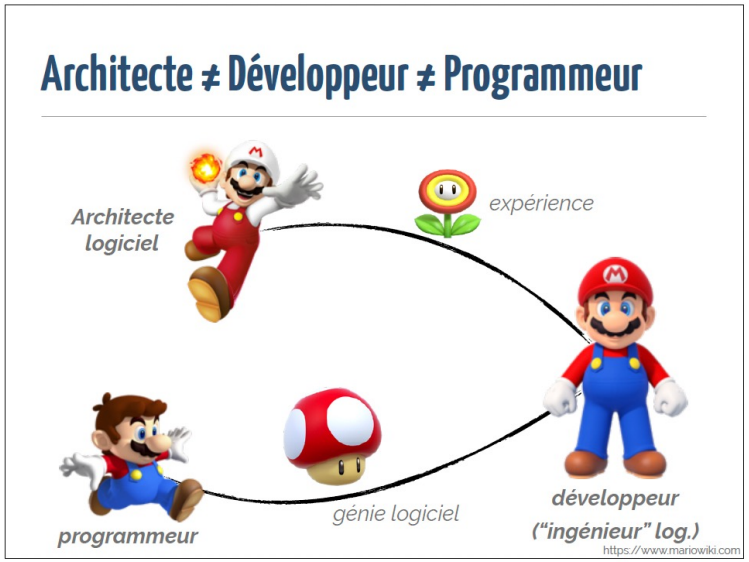
Références

- Introduction au génie logiciel, Professeur Hafedh Mili (INF5153)
- V-Model : https://medium.com/software-engineering-kmitl/v-model-3a71622b3d82

Plan du Cours

1. Qu'est-ce que le génie logiciel?
2. Phases de développement
3. Cycle de vie du développement logiciel (SDLC)
   - Méthodologies traditionnelles et agiles

Passage de programmeur à développeur

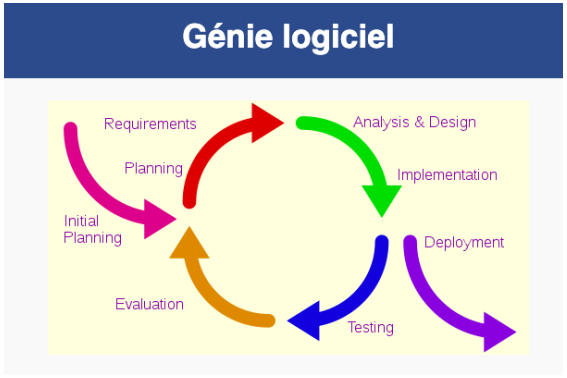


La différence entre un ***programmeur*** et un ***développeur*** est la **qualité des produits** développés. (William Flageol)

Génie logiciel

- Le **génie logiciel**, l'**ingénierie logicielle** ou l'**ingénierie du logiciel** (en anglais : *software engineering*) est une **science** de génie industriel qui étudie les méthodes de travail et les bonnes pratiques des ingénieurs qui développent des logiciels.

- **Le génie logiciel** s'intéresse en particulier aux **procédures systématiques** qui permettent d'arriver à ce que des logiciels de **grande taille** correspondent aux **attentes du client**, soient **fiables**, aient un **coût d'entretien réduit** et de **bonnes performances** tout en **respectant les délais et les coûts** de construction



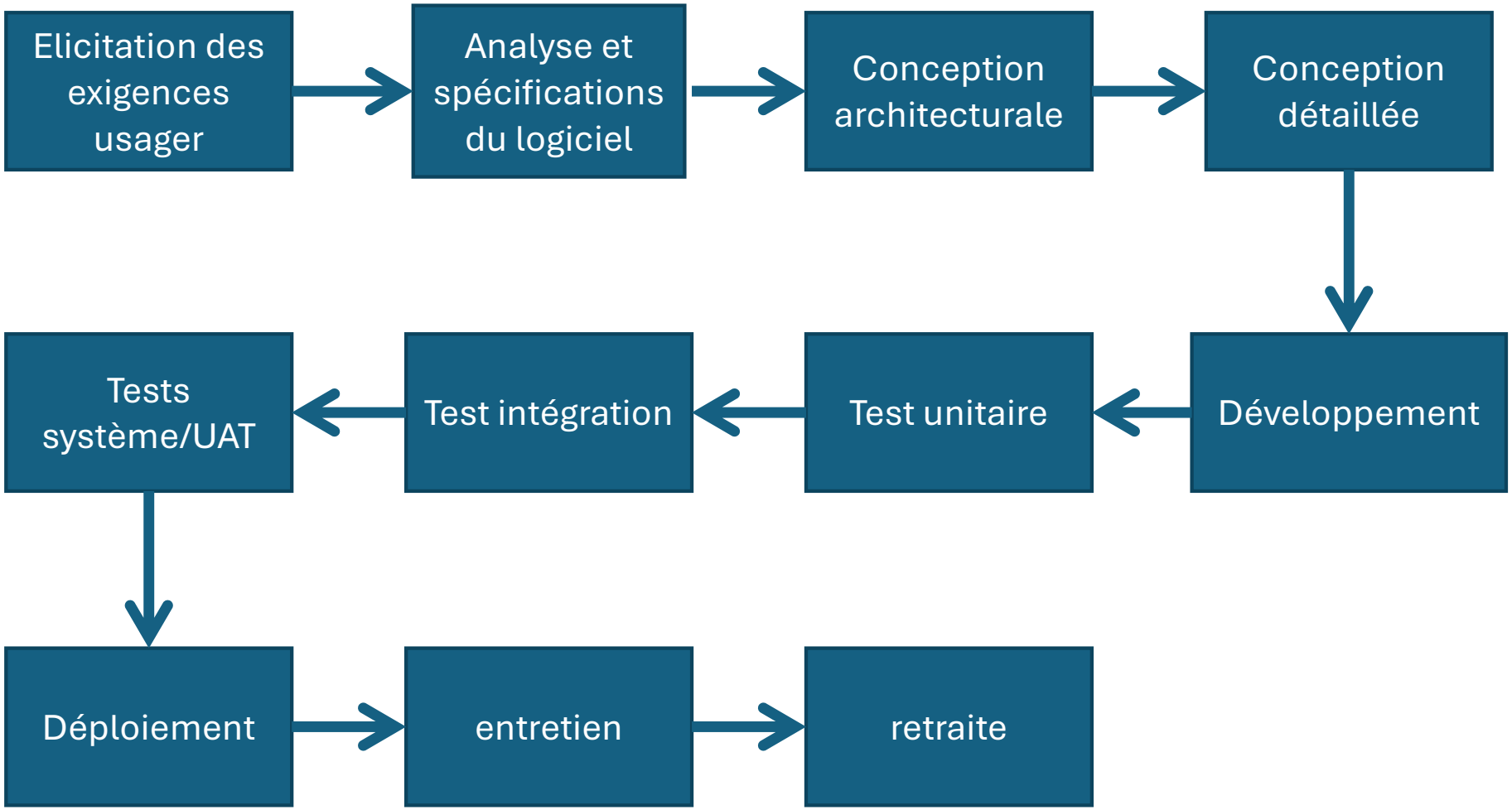
- Le **développement de logiciel** consiste à étudier, concevoir, construire, transformer, mettre au point, maintenir et améliorer des logiciels.

Activités de développement de logiciels

Comment mener un projet de développement logiciel?

- Deux types d'activités
  - Activités de base, consistant en la production de livrables:
    - Exigences, analyse, conception, implementation, test
  - Activités secondaires de support aux activités de base:
    - Gestion de configuration, mise en place de l'environnement de développement, gestion de projet

Activités de base



Elicitation des exigences

- Objectif: que doit faire le logiciel?

- On convient de distinguer deux types d'exigences:
  - Exigences fonctionnelles (quoi)
  - Exigences non-fonctionnelles (comment)

- La distinction n'est pas toujours claire (ou utile)
  - "le système doit produire un journal de transactions"
  - "le système doit sauvegarder les données en moins de 2K" … quand "le système" est un logiciel de compression de données

Exigences

- Lister les exigences potentielles / candidates
- Établir le contexte:
  - Modèle du domaine (entités)
  - Modèle du processus d'affaire
- Éliciter les exigences fonctionnelles (cas d'utilisation)
- Éliciter les exigences non-fonctionnelles (performance, fiabilité, réutilisabilité, etc.)

Analyse

- Traduire les *exigences* de l'usager en *spécifications* du logiciel
  - Vue *externe* du logiciel
    - Comportement externe du logiciel, en terme de fonctionnalités, et dépendances entre fonctionnalités (interfaces offertes)
    - Peu de préoccuptation pour la réalisation de ces fonctionnalités
  - La conception et l'implémentation se préoccuperont de *comment* réaliser les spécifications

Analyse et spécifications en OO

- Exemple: activités et livrables dans les *méthodes OO*
  - Produire un modèle de classes
    - Des classes qui correspondent aux descriptions d'entités du domaine qui seront emmagasinées par le système
  - Produire un modèle détaillé de cas d'utilisation
    - Reprendre les cas d'utilisation et traduire leur comportement sous forme (de diagrammes) d'intéractions entre objets
  - Modulariser les cas d'utilisation et les classes en des *modules d'analyse* (analysis packages)
  - Produire une première architecture *fonctionnelle*

Conception architecturale

- Objectif: fournir un principe d'organisation du logiciel en respectant les exigences non-fonctionnelles
- Concrètement: **conception d'un *moule* (architectural pattern) pour la *réalisation* de l'architecture fonctionnelle du logiciel**
- Le **moule dépend *presque exclusivement* des exigences non-fonctionnelles**

Conception détaillée

- “Optimisation locale” de la réalisation des composants de l’architecture
  - Implantation des connecteurs
  - Implantation interne des composants
- Patrons de conception

Conception en OO

- Conception architecturale:
  - Choisir style architectural
  - Identifier composantes
  - Spécifier interfaces des composantes
- Conception détaillée
  - Modèle de classe – conception
  - Modèle comportemental – conception
  - Modèle de use-case - conception

Implémentation

- Activités en OO
  - Implémentation de l'architecture
  - Implémentation de classe
  - Effectuer tests unitaires
  - Implémenter sous-systèmes
  - Intégration de sous-systèmes

Test

- Planifier les tests
- Concevoir les tests
- Implémenter les tests
- Effectuer tests d'intégration
- Effectuer tests systèmes
- Évaluer les tests

Maintenance

- **Corrective**: identifier et corriger des erreurs de, a) traitement (calcul), b) performance, ou c) implantation
- **Adaptive**: adapter le logiciel aux changements dans l'environnement, toutes fonctionnalités égales, a) changements dans le format des données, et b) changements dans l'environnement d'exécution.
- **Perfective**: améliorer la performance, ajouter des "features" (et bugs), et améliorer la maintenabilité du logiciel.

Cycle de vie du développement logiciel (SDLC)

- *SDLC signifie "Software Development Life Cycle" en anglais, ce qui se traduit en français par "Cycle de vie du développement logiciel".*
- Un processus de développement prescrit une organisation des *activités* de développement de logiciels
- *Une méthodologie qui guide les étapes du développement d'un logiciel depuis l'activité* "Elicitation des exigences usager"*à sa mise en production*

Différence entre les différents processus

- **La portée des activités**
  - Big bang versus par petits morceaux
- **L'intensité des activités**
  - Ad-nauseum versus "just enough"
- **Le timing des activités**
  - En amont vs. "just in time"

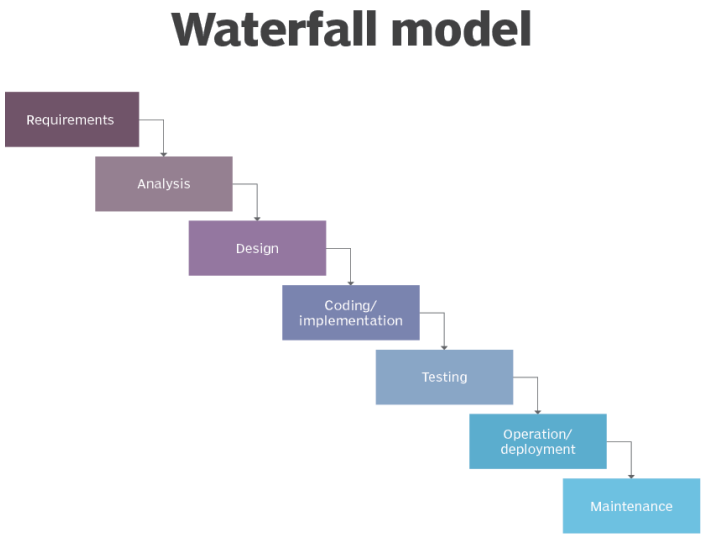
Critères de choix du processus de SDLC:

- **Complexité du projet :** Taille et complexité de l'application.
- **Risques techniques :** Identification des défis techniques.
- **Besoins du client :** Exigences spécifiques du client.
- **Flexibilité requise :** Niveau de flexibilité nécessaire.
- **Délais de livraison :** Contraintes temporelles du projet.
- **Ressources disponibles :** Compétences et ressources disponibles.
- **Historique de succès :** Succès passés avec des processus spécifiques.
- **Évolutivité :** Possibilité d'évolutions futures

Les différents cycles de vie du développement logiciel (SDLC)

- On retrouve les mêmes activités, quelle que soit la « méthodologie »
  - Cascade
  - Model-V
  - Spirale
  - Prototypage
  - ~~Cluster~~
  - ~~RUP~~
  - Agile/DevOps
  - ….

Modèle en Cascade (Waterfall)



Méthodologie linéaire et séquentielle

- Une structure bien définie
- Chaque phase (exigences, conception, développement, tests, déploiement) **est achevée avant de passer** à la suivante.

Équipes Isolées(Silos) :

- Équipes de développement, d'opérations et de sécurités, etc.

=> **Travaillent de manière isolée + Communication limité**

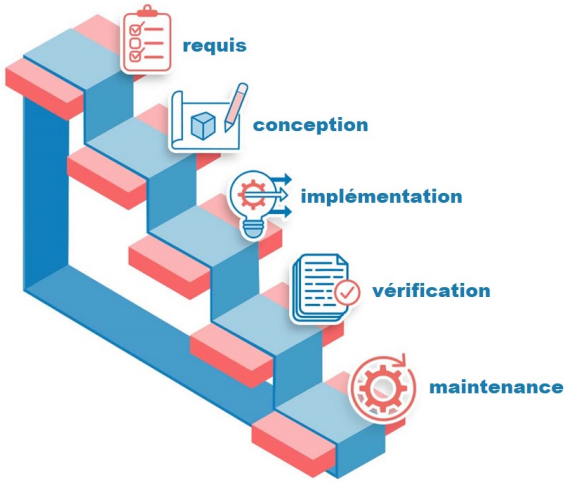
Cycles de développement longs :

- En raison de la nature séquentielle, les cycles de développement ont tendance à être longs,

Prolème: Difficulté pour la réponse rapide aux changements

Modèle en cascade ou *waterfall*

- Méthode de gestion linéaire (c.-à-d. **non itérative**) séquentielle qui repose sur :
  1. Une structure bien définie
  2. Une implication minimale des clients
  3. Une documentation robuste et détaillée



- Analogie du **projet de construction.**

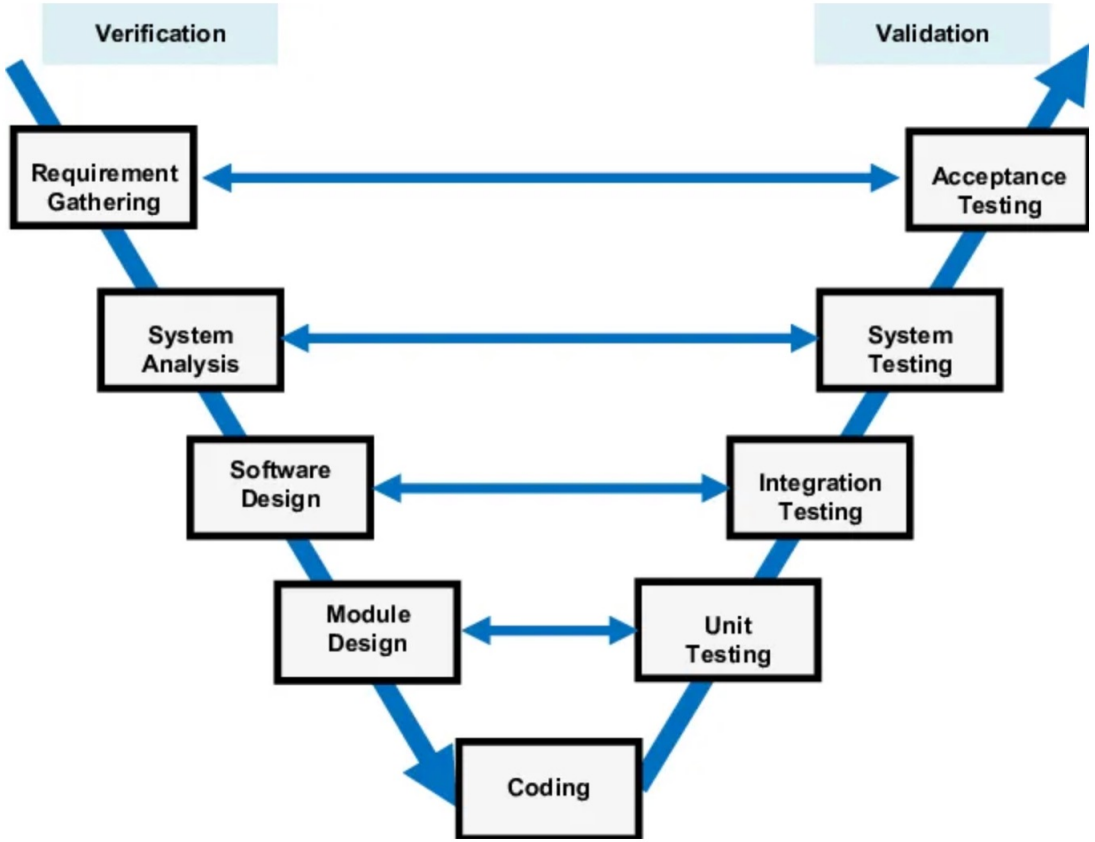
- À utiliser quand :
  - La majorité des requis sont connus à l'avance
  - Les dates limites sont strictes (controversé)
  - Vous avez le temps de faire la planification nécessaire

- À éviter lorsque les requis changent souvent.

Capsule vidéo : https://www.youtube.com/watch?v=5A5XCuWMG4o

PAR : Iannick gagnon

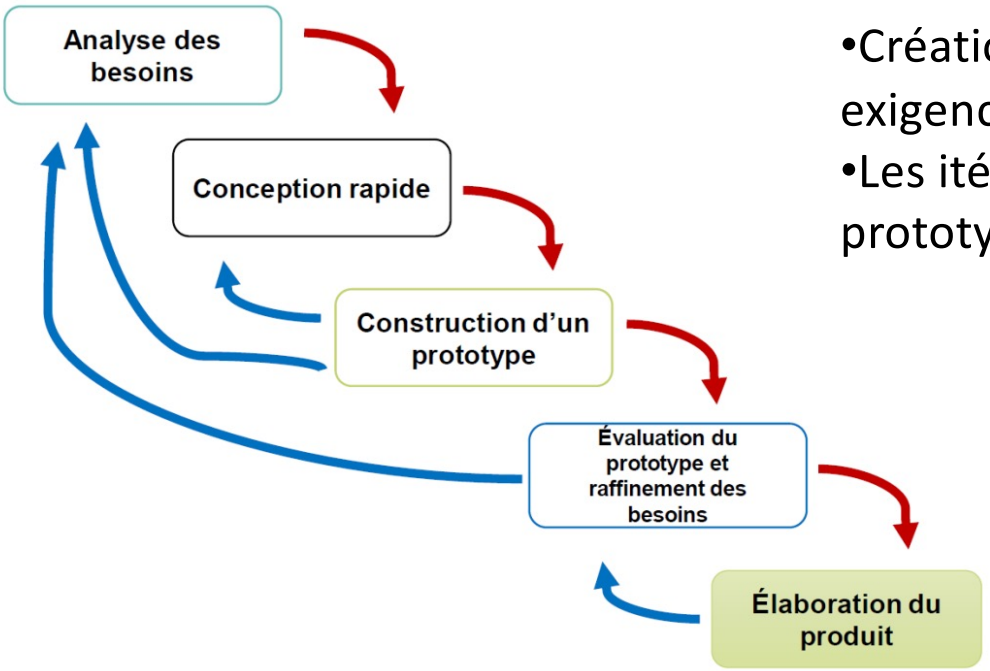
Modèle en V (V-Model) : Vérification et Validation



Chaque étape du développement de V-Model, il y aura une phase de test correspondante qui validera un tel processus.

https://medium.com/software-engineering-kmitl/v-model-3a71622b3d82

Modèle de prototypage



- Création de prototypes pour comprendre les exigences.
- Les itérations sont effectuées pour affiner le prototype

Modèle de prototypage

- Méthode de gestion basée sur la création d'un **prototype** qui est amélioré ou affiné de manière **itérative**.
- Contrairement au modèle en cascade, **les utilisateurs sont très impliqués**.

À utiliser quand :

- Les requis sont mal connus ou changent souvent
- Une implication importante du client est requise
- Les risques d'échecs sont élevés

À éviter quand :

- Les budgets sont fixes ou limités (les prototypes sont jetables)
- Le projet est simple ou petit
- Les délais sont serrés

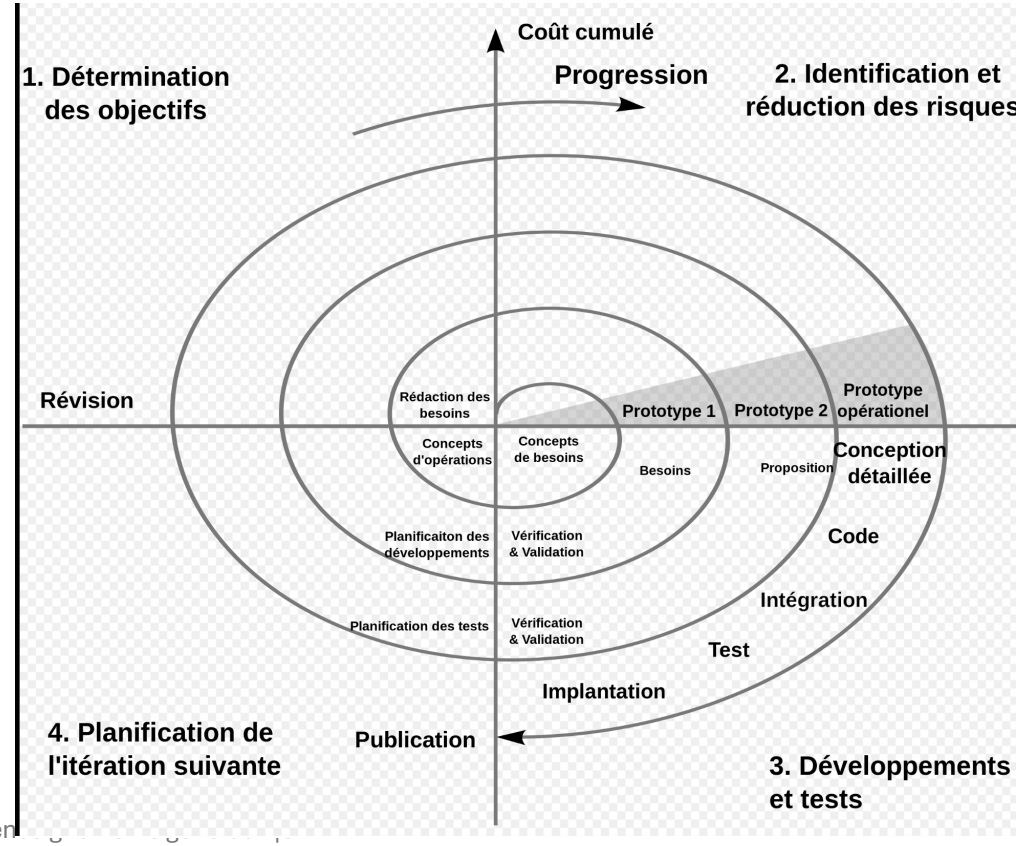
Capsule vidéo : https://www.youtube.com/watch?v=bAEnaGG8Otc

PAR : lannick gagnon

Modèle en Spirale

1. Le modèle en spirale combine le développement itératif avec la gestion des risques.

On distingue quatre phases dans le déroulement du cycle en spirale :
1. détermination des objectifs, des alternatives et des contraintes ;
2. analyse des risques, évaluation des alternatives ;
3. développement et vérification de la solution retenue ;
4. revue des résultats et vérification du cycle suivant.



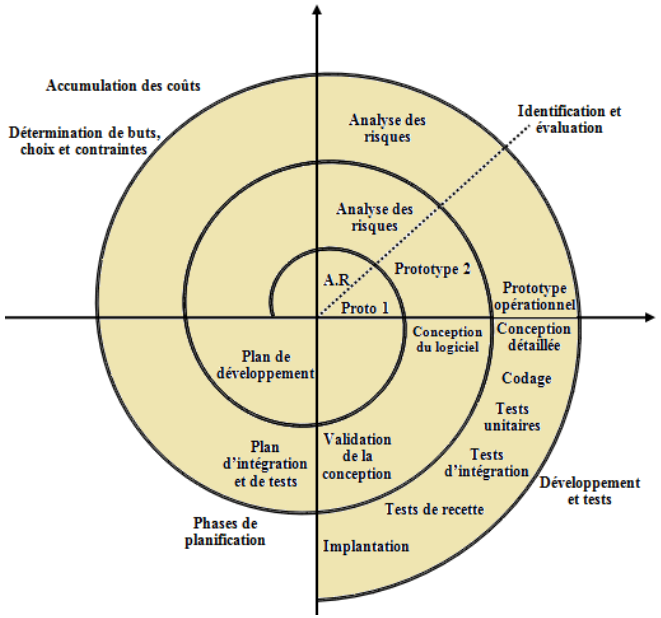
Modèle en spirale

À utiliser quand :

- Les requis sont mal connus ou changent souvent
- Une implication importante du client est requise
- Les risques d'échecs sont élevés

À éviter quand :

- Les budgets sont fixes ou limités
- Le projet est simple ou petit
- Les délais sont serrés

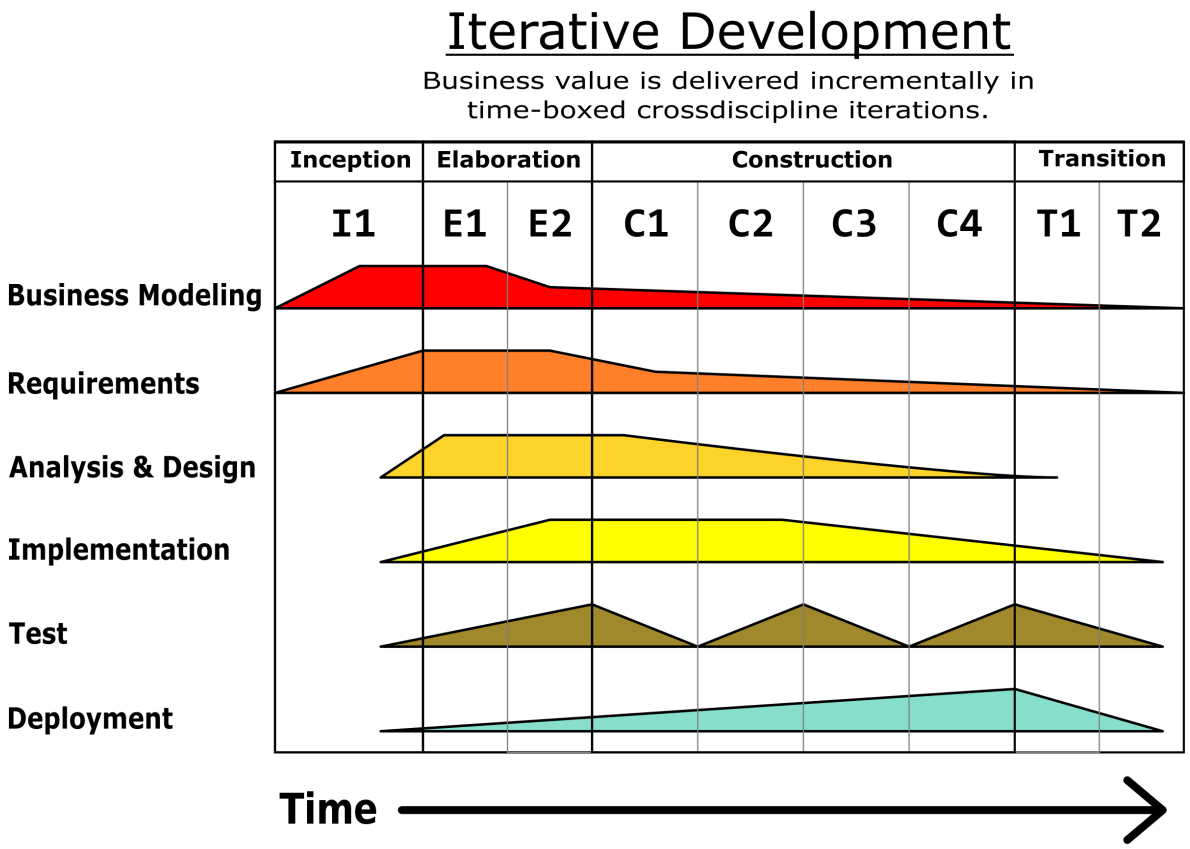


Capsule vidéo : https://www.youtube.com/watch?v=mp22SDTnsQQ

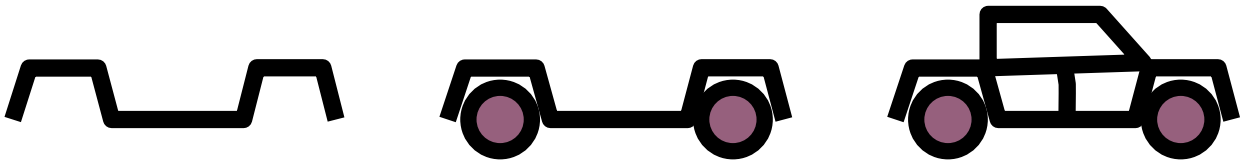
PAR : IANNICK GAGNON

Développement agile

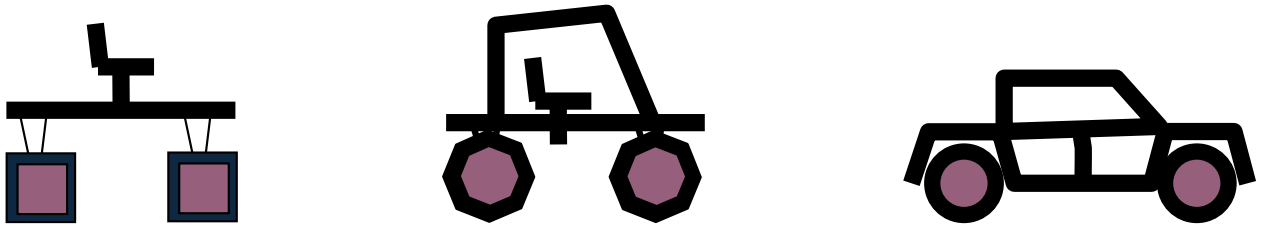
Approche itérative et incrémentale.



Itératif versus incrémental



Incrémental



Itératif

À retenir: Processus de développement

SDLC (Cycle de vie de développement Logiciel)

- Cascade (Développement linéaire et séquentiel)
- Spirale (Développement itératif et incrémentiel avec une gestion des risques systématique)
- Prototypage (Créer des versions simplifiées ou partielles d'une solution afin de tester)
- Agile (Approche itérative et incrémentale)

Questions