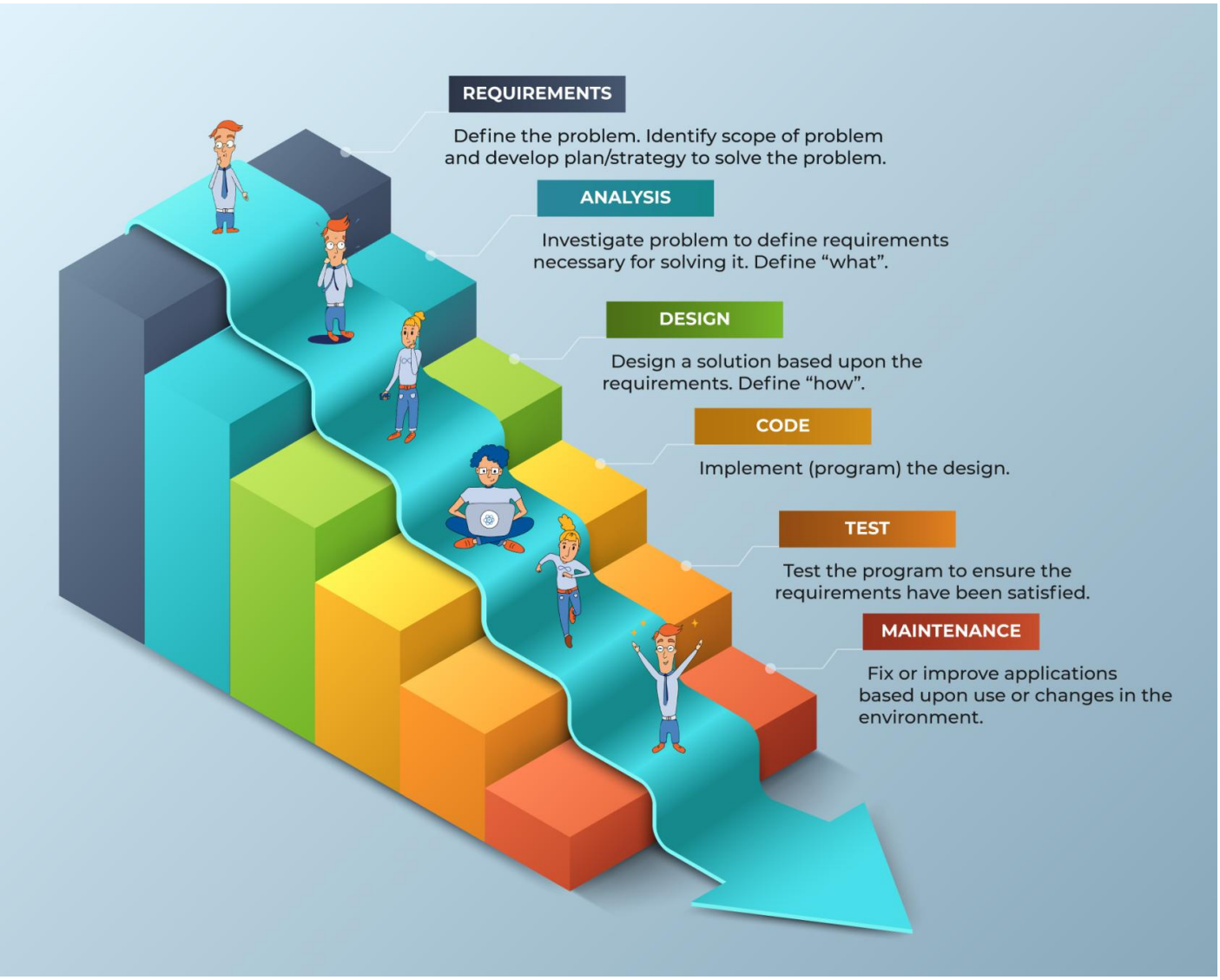
Chapitre 2 : Méthodologies de développement

- Waterfall
- Cycle en V
- Méthodologies Agiles
- SCRUM

Waterfall

Le modèle **Waterfall** organise les phases d'un projet séquentiellement. Chaque phase dépend des résultats de la précédente.



Les différentes phases peuvent varier, mais voici un exemple :

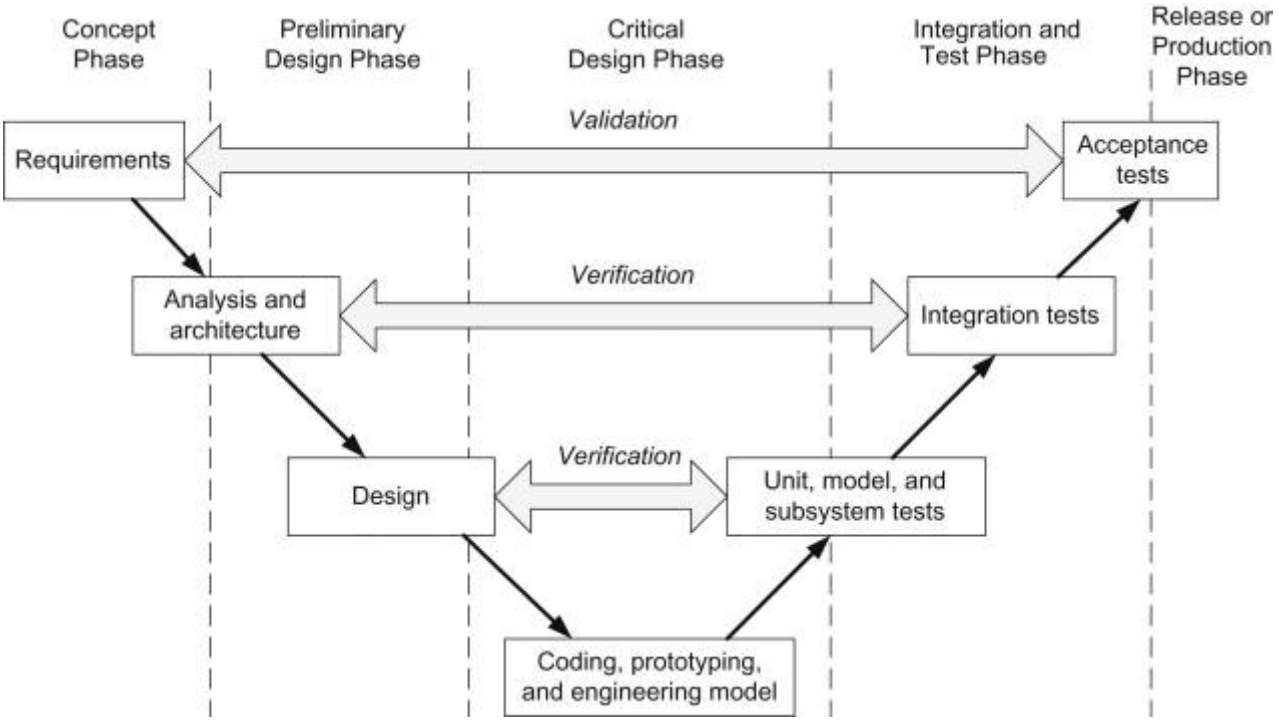
1. Les **requirements** où on va aller interviewer le client et écrire le cahier des charges
2. L’**analyse** où on va créer des schémas, définir les règles métiers et les tests d’acceptations
3. Le **design** qui va définir l’architecture technique du projet
4. Le **code** où on applique la structure définie par le design
5. Les **tests** où on vérifie la conformité du projet aux normes et à l’analyse initiale
6. La **maintenance** où on maintient le code réalisé pour résoudre d’éventuels bugs qui pourraient survenir

Avantages	Désavantages
Nécessite une réflexion globale	Incertitude élevée
Bien structurée et complète	Peut devenir lourde et rapidement obsolète
Adaptée aux projets bien définis dès le départ	Difficulté à gérer les changements de besoins
Simplifiée grâce à des étapes séquentielles	Manque de flexibilité
Une étape à la fois (clarté des jalons)	Effet tunnel : peu de visibilité pour le client jusqu'à la fin
Faible nombre de personnes impliquées simultanément	Utilisation rigide des ressources
Chacun intervient à une étape précise	Développeurs inactifs pendant l’analyse ; analystes inactifs pendant le dev
-	Risques postposés et concentrés en fin de projet

Cycle en V

Le modèle **Cycle en V** est une évolution du modèle **Waterfall** qui met l’accent sur la **validation** à chaque étape du cycle de développement. Chaque phase de spécification (côté gauche du "V") correspond à une phase de test ou de validation (côté droit du "V").

Il conserve une approche séquentielle, mais introduit une **correspondance directe entre les étapes de conception et celles de test**, améliorant la traçabilité et la qualité du logiciel.



Avantages	Désavantages
Nécessite une réflexion globale (planification rigoureuse)	Incertitude toujours présente dès le début
Bonne documentation	Utilisation rigide des ressources
Adaptée aux projets avec périmètre bien défini	Effet tunnel
Gestion de projet assez simple avec des jalons de validation clairs	Développeurs inactifs durant la phase d’analyse
Validation à chaque étape (meilleure maîtrise de la qualité)	Analystes inactifs pendant le développement

Méthodologies Agiles

Les méthodes Agiles, dont SCRUM, KANBAN, eXtreme Programming, et LEAN, sont basées sur le Manifeste Agile (4 valeurs et 12 principes). Elles prônent l'intégration continue, le TDD, les équipes transverses, le pair programming, les user stories, et surtout les livraisons fréquentes par itérations.

- **Approche Incrémentale** : "on développe les différentes parties les unes après les autres." On spécifie toutes les caractéristiques du produit précisément.
- **Approche Itérative** : "on développe au plus vite un produit minimum viable (MVP)" et "on améliore / corrige au fur et à mesure." On donne une vision globale du produit.
- **L'Agile combine les deux** : "Réaliser un produit fonctionnel minimal de manière itérative" et "Enrichir le produit de manière incrémentale."

Un MVP (Minimum Viable Product) est "la version d'un produit (par exemple un logiciel) qui permet de limiter le nombre de bugs avec un minimum d'effort."

SCRUM

Le framework SCRUM repose sur l'empirisme et trois piliers : **Transparence, Inspection, Adaptation**.

Rôles SCRUM :

- **Equipe (Dev Team)** : Max 10 membres, autonome, sans hiérarchie.
- **Product Owner** : "Possède la “vision” du produit", "Crée et priorise les items du backlog."
- **Scrum Master** : "Veille à la bonne application de la méthodologie SCRUM", "Protège l’équipe des événements extérieurs perturbateurs", "Aide à la collaboration entre l'équipe et le product owner."

Artéfacts SCRUM :

- **Product Backlog** : "Liste priorisée de tâches" dérivée de la Roadmap et des exigences du produit. "Accessible à toutes les parties prenantes." Priorisation continue par le Product Owner.
- **Product Backlog Item (PBI)** : Description du "quoi ?", généralement sous forme de user story, avec critères d'acceptation et estimation d'effort.
- **Sprint Backlog** : Créé lors du Sprint Planning, il est "un ensemble de PBI sélectionnés pour le sprint" et le "plan d’actions pour délivrer l’incrément". Il inclut le Sprint Goal, qui est l'objectif principal du sprint et est immuable.
- **Sprint Task** : Décomposition des PBI, plus petite unité de planification.
- **Sprint Burndown Chart** : "Montre visuellement le travail qui a été accompli et celui restant à accomplir pour le sprint en cours", permettant de prédire si les objectifs seront atteints.
- **Release Burndown Chart** : Similaire au Sprint Burndown Chart, mais pour la "release" en cours, prédisant le nombre de sprints nécessaires.

Événements SCRUM :

- **Sprint Planning** : "La cérémonie qui permet de construire le Sprint backlog", au début du sprint. Le Product Owner décrit le Sprint Goal et priorise, la Dev Team s'engage sur ce qui peut être fait.
- **Daily Meeting** : Quotidien (15 min), partage d'informations au sein de l'équipe (ce qui a été fait, ce qui sera fait, obstacles).
- **Sprint Review** : À la fin du sprint, inspection du résultat (démo des PBI "Done"), identification des réussites et problèmes, mise à jour du Product Backlog.
- **Sprint Retrospective** : À la toute fin du sprint, discussion sur le déroulement du sprint (personnes, processus, outils, DoD), avec pour résultat un plan d'amélioration.
- **Backlog Refinement** : Décision de l'équipe (2 à 4 fois par Sprint), pour s'assurer que les PBI sont clairs et respectent le DoR, transformer les Epics en User Stories.