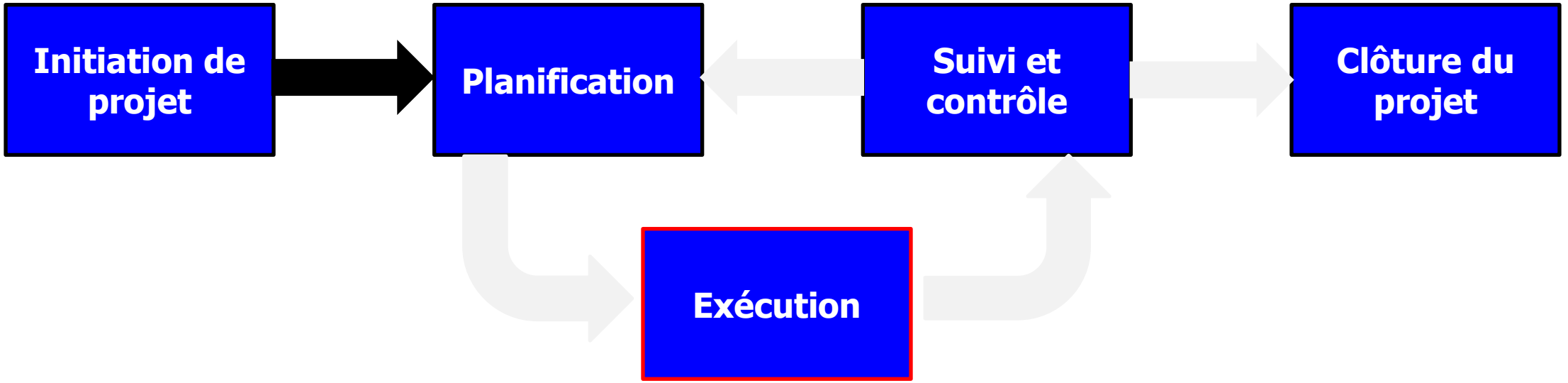
Chapitre 3

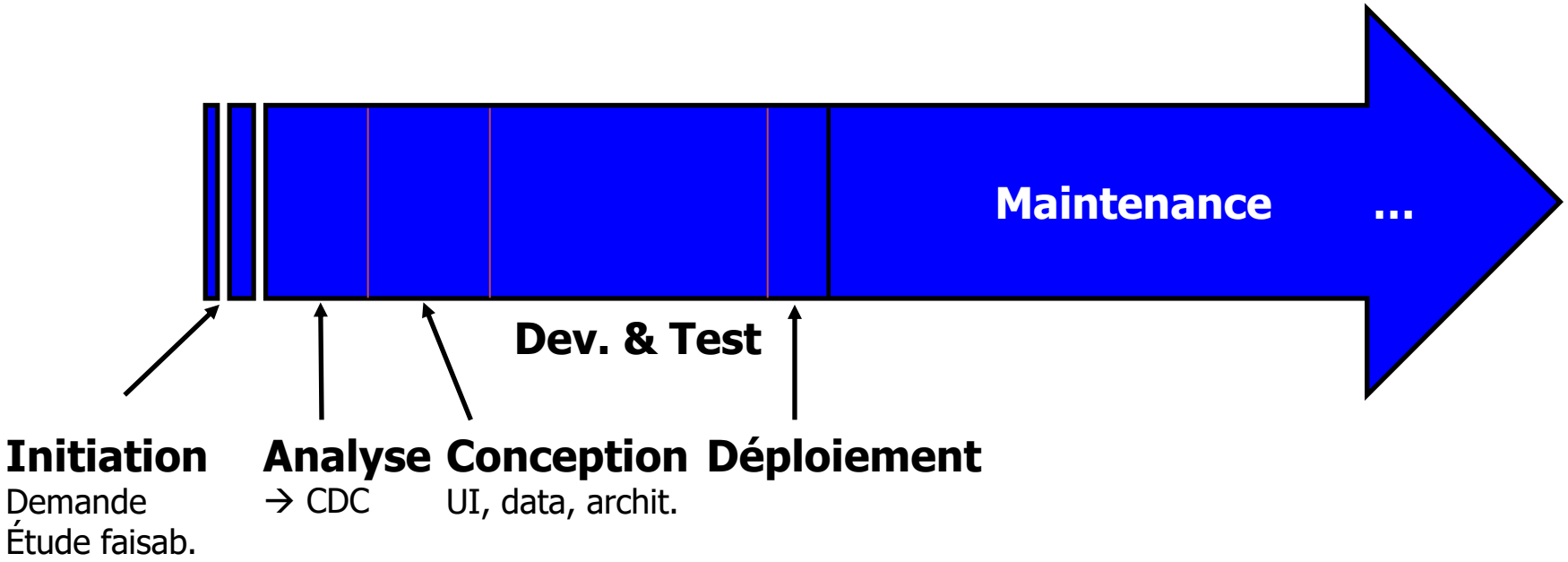
Méthodologies de développement

Vue d'ensemble



RAPPEL

Cycle de vie d'un logiciel ou système:



3.1 – Types de Méthodes

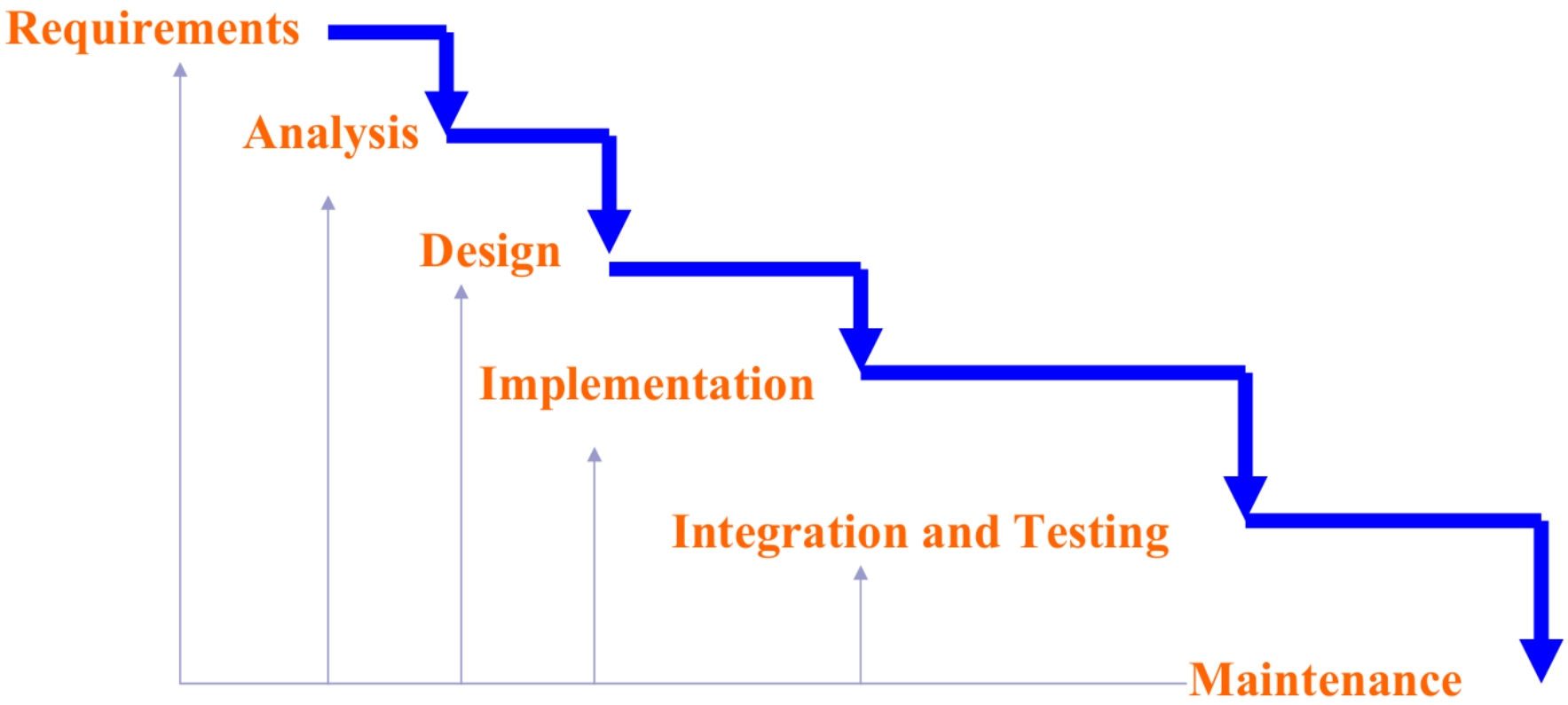
Les méthodes prédictives

- Vision à long terme
- Spécifications finalisées et figées en amont du développement
- Plan de développement (et délivrables) figés

Les processus adaptatifs

- Vision à long terme ET vérifications court terme
- Spécifications, plans et délais affinés par étapes (itératives et incrémentales)
- **Les utilisateurs sont au centre du processus**

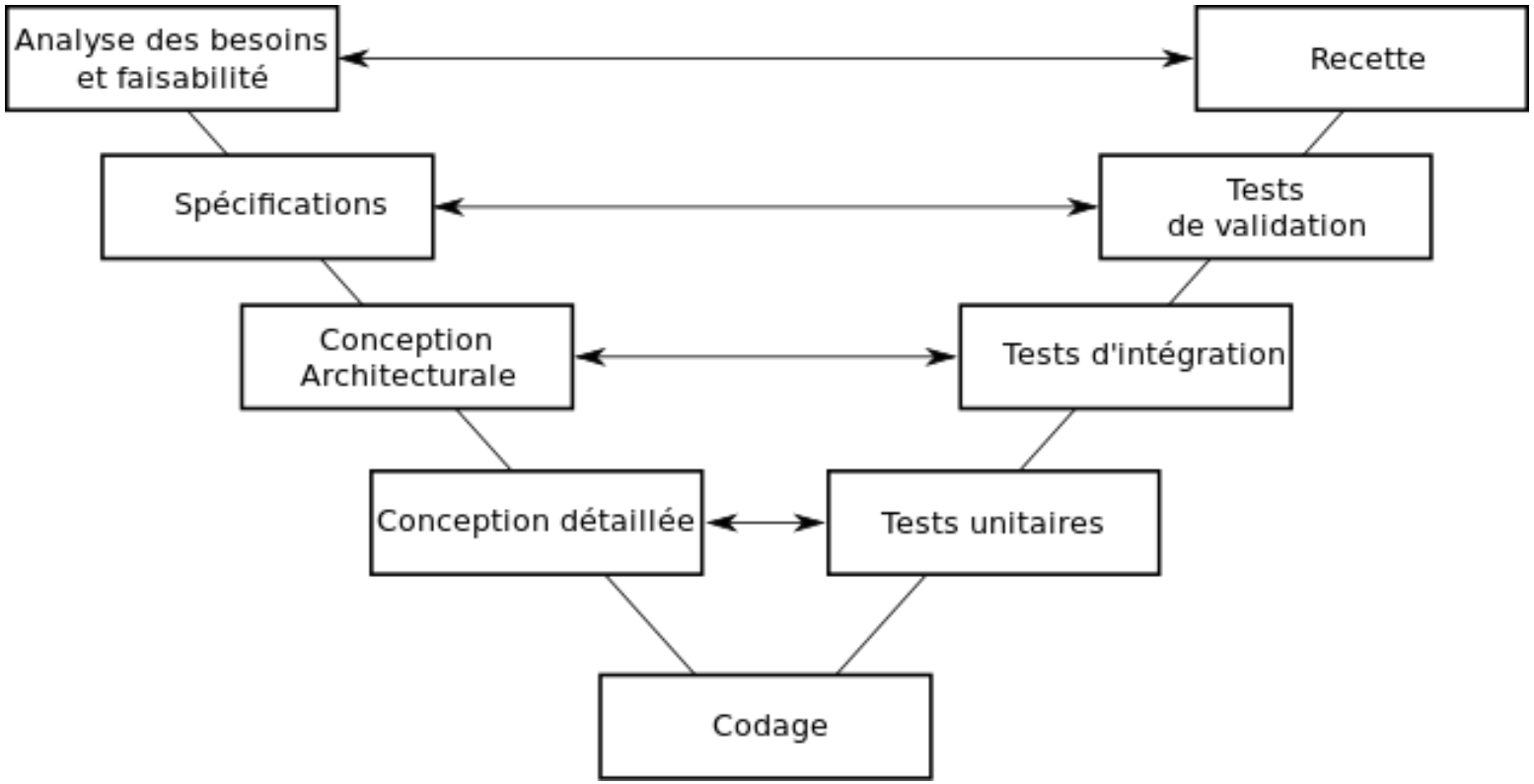
3.2 - Méthodes en cascade



3.2 - Méthodes en cascade

Évaluer les points forts (+) et les points faibles (-) du processus en cascade

3.3 - Méthodes en V

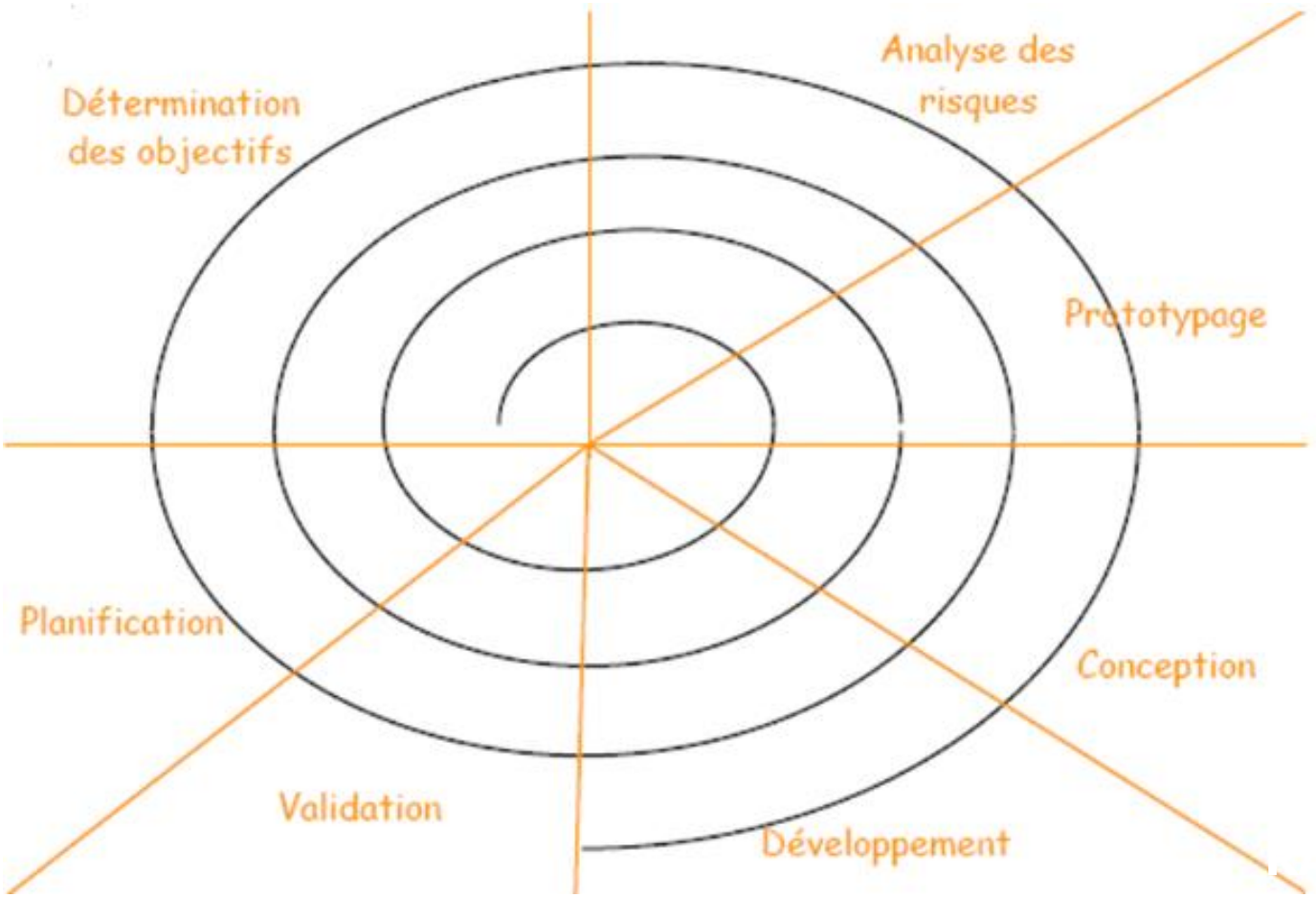


3.3 - Méthodes en V

Quelles améliorations apporte le processus en V, au processus en cascade?

Quelles limitations persistent?

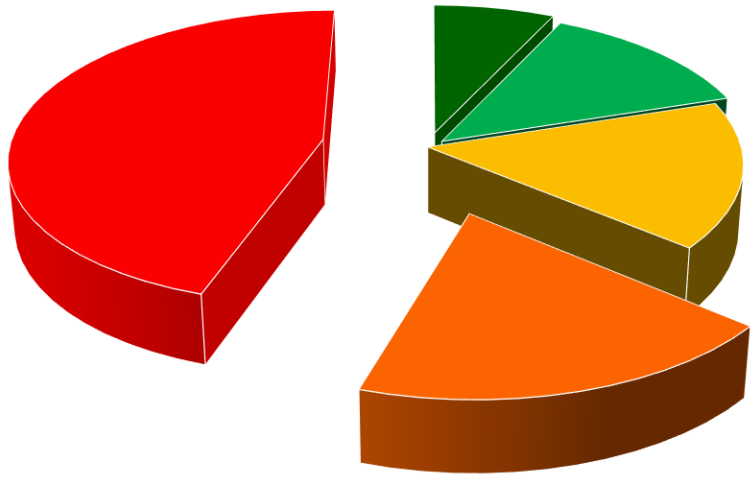
3.4 - Méthodes itératives



3.4 - Motivations

Capture des besoins !

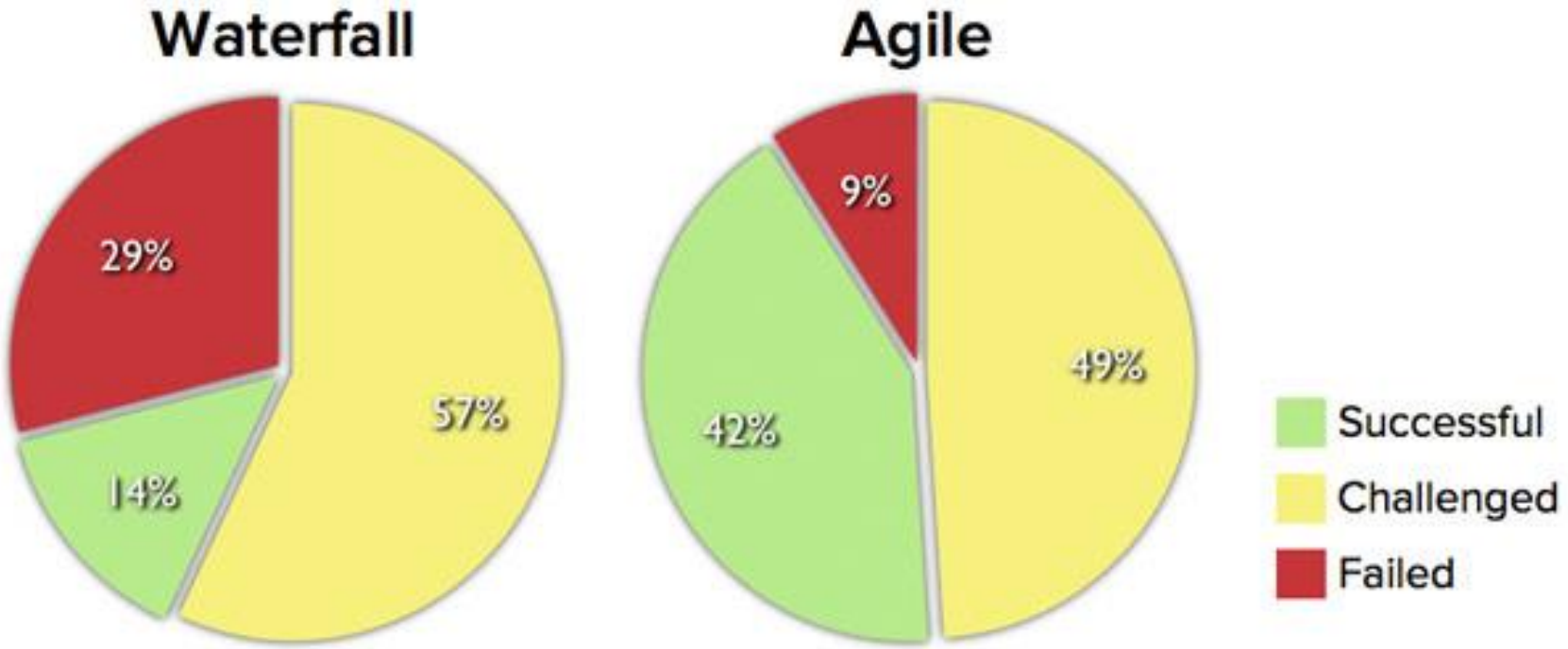
Taux d'utilisation des fonctionnalités développées par des projets avec des méthodes classiques



Toujours 7% Souvent 13% Parfois 16% Rarement 19% Jamais 45%

3.4 - Motivations

Coûts !!



Source: The CHAOS Manifesto, The Standish Group, 2012.

3.4 - Principes

Bien	Mieux
Processus et outils	Individus et interactions
Documentation exhaustive	Produit opérationnel
Négociation de contrat	Collaboration avec le client
Suivre un plan	Répondre aux changements

[Manifeste Agile]

3.4 - Caractéristiques des projets

Livraison rapide et continue de produits utilisables

Les produits à livrer utilisables donnent la mesure des progrès réalisés

Accepter les changements, même si le projet est avancé

Cycles de livraison courts

3.4 - Caractéristiques des équipes

Responsable des opérations et équipe de projet collaborent constamment

Équipes auto-organisées (habilitées)

Optimiser le recours aux conversations en personne

Favoriser un développement durable

3.5 - Exemples

Rapid application development (RAD, 1991)

Dynamic systems development method (DSDM, 1995)

Scrum (1996)

Extreme programming (XP, 1999)

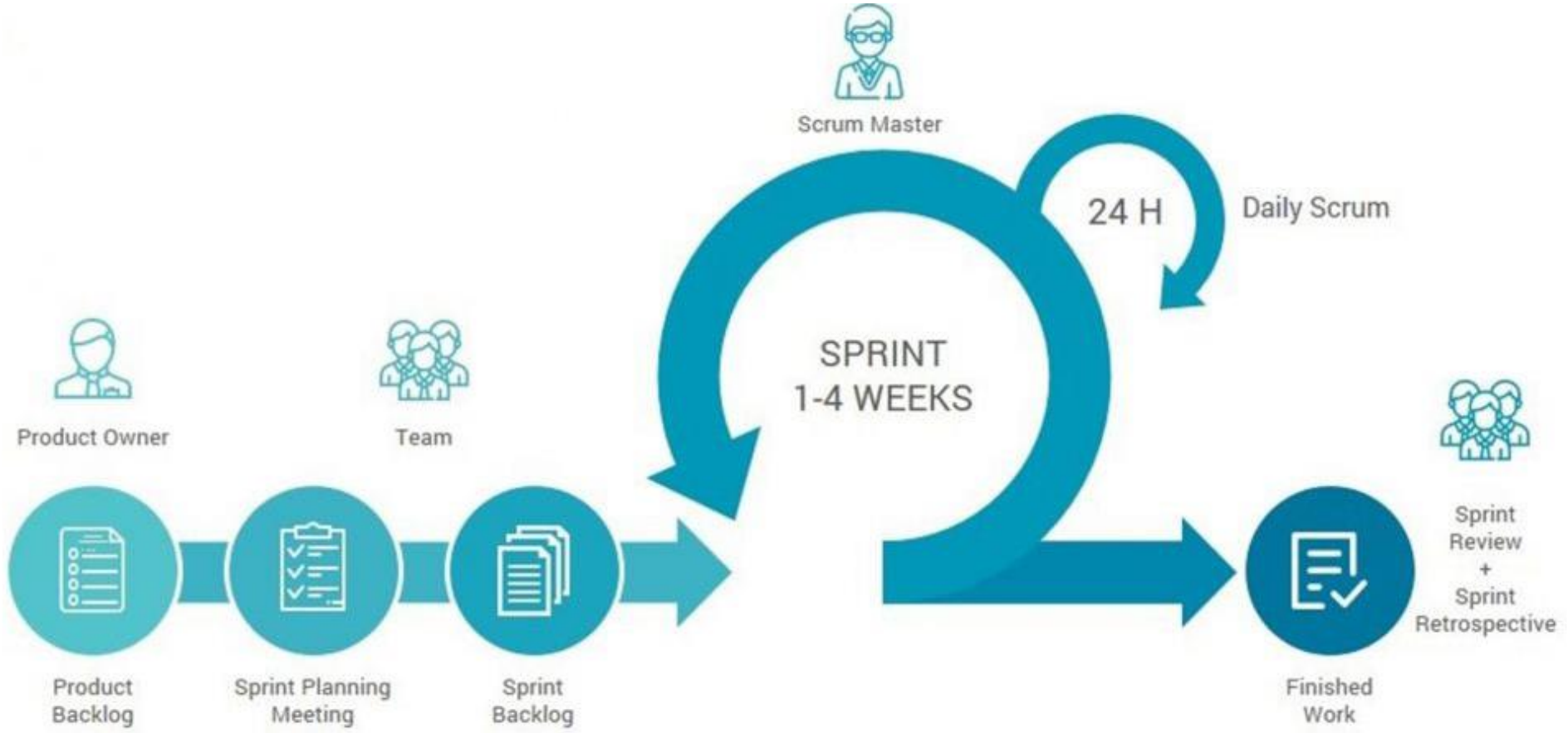
Adaptive software development (ASD, 2000)

Développement basé sur les fonctionnalités (FDD, 2003)

Behavior-driven development (BDD, 2003)

Crystal clear (2004)

3.6 - Scrum



https://www.intersim.ch

3.6 - Les Rôles : Product Owner

Définit les fonctionnalités du produit

Choisit la date et le contenu de la release

Responsable du retour sur investissement

Définit les priorités dans le *backlog* en fonction de la "valeur métier"

Ajuste les fonctionnalités et les priorités à chaque sprint si nécessaire

Accepte ou rejette les résultats

3.6 - Les Rôles : ScrumMaster

Assume la gestion du projet

Responsable de faire appliquer par l'équipe les valeurs et les pratiques de *Scrum*

Élimine les obstacles

S'assure que l'équipe est complètement fonctionnelle et productive

Facilite une coopération poussée entre tous les rôles et fonctions

Protège l'équipe des interférences extérieures

3.6 - Les Rôles : équipe Scrum

De 5 à 10 personnes

Regroupant tous les rôles (architecte, concepteur, développeur, spécialiste IHM, testeur, …)

A plein temps sur le projet, de préférence

L'équipe s'organise par elle-même

La composition de l'équipe ne doit pas changer pendant un *Sprint*

3.7 - Intégration continue

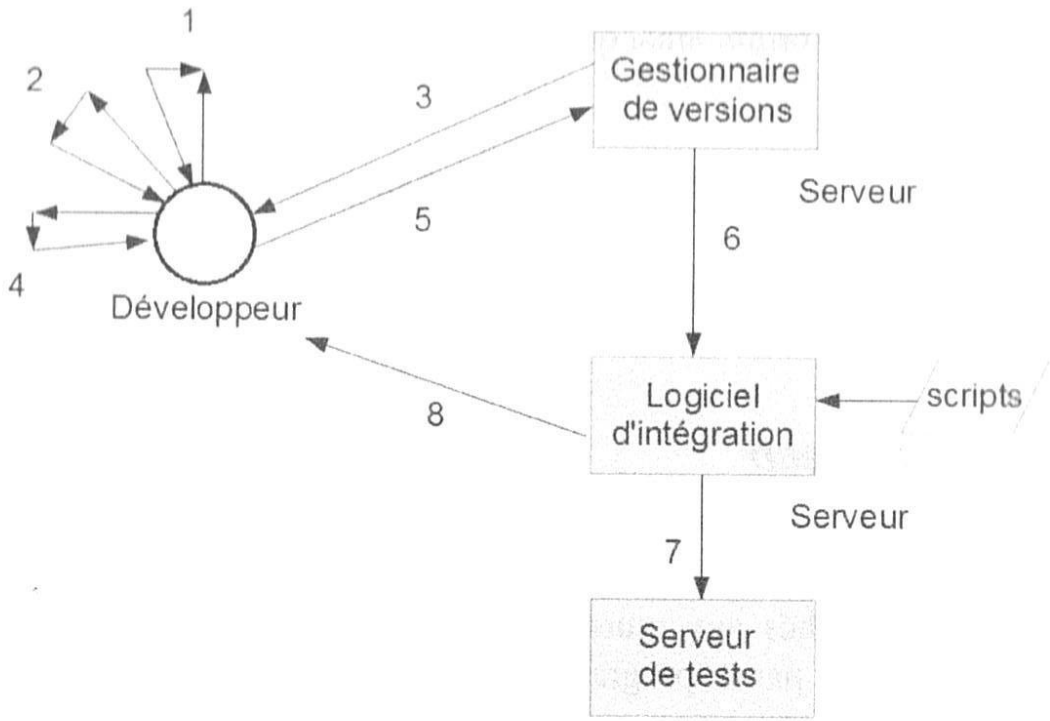


FIGURE 5.17 Un scénario d'intégration continue

1 code une fonctionnalité
2 test sur son PC (build)
3 récup dernière version
4 Fusion, corr., test
5 Publie modif (*commit*)
6 Intégration
7 déploiement & tests
8 feedback

Outils: *Git*, *Jenkins*, Sonar, emails, *Docker* …

Exercice

1) Lire le pdf Le Guide Scrum
2) En discuter en groupe
3) Proposer une adaptation pour votre cas particulier

   Le backlog ? Les sprints ? Les rôles ? Brundown chart ? Daily standup / scrum ?