Le cycle en V

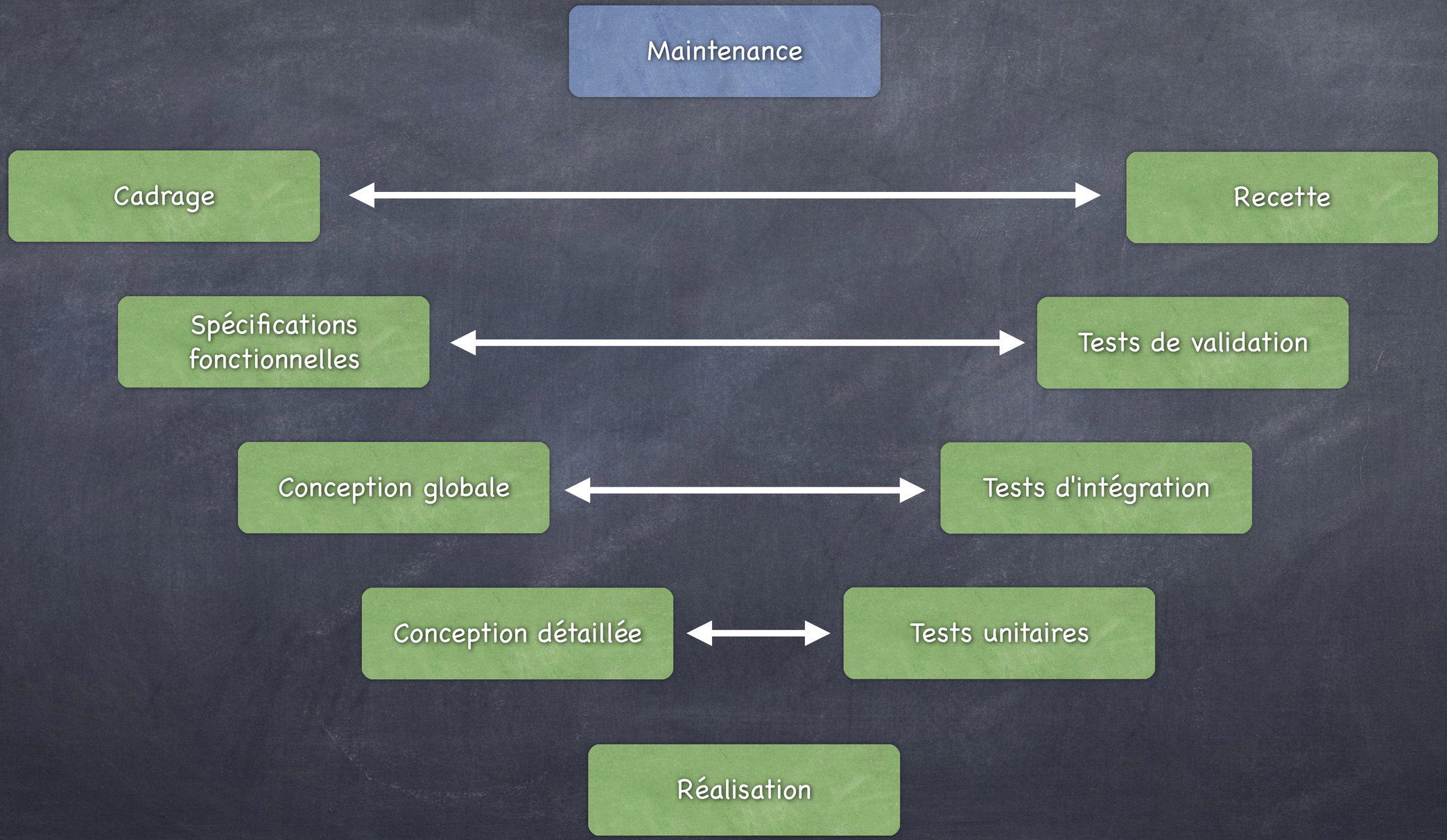
Objectifs

- ✓ Comprendre les étapes du cycle en V dans le développement logiciel.
- ✓ Identifier les avantages et inconvénients de ce modèle.
- ✓ Être capable d'appliquer chaque phase du cycle en V à un cas pratique.
- ✓ Développer une approche méthodique pour gérer les exigences, la conception, le développement et les tests.

Historique

- ✓ Formalisé dans les années 80 pour répondre aux critiques de la méthode Waterfall
- ✓ Adapté aux projets complexes nécessitant une forte traçabilité et une validation stricte à chaque étape

Les phases du cycle en V



Principe

- ✓ Chaque phase, hormis celle de l'implémentation, est associée à des tests
- ✓ Les tests sont conçus sur la première branche du V et exécutés sur la seconde

Cadrage

Objectif : Recueillir les besoins du client ou des utilisateurs finaux et les documenter clairement.

Activités :

- ✓ Réunions avec les parties prenantes pour comprendre les attentes et besoins.
- ✓ Rédaction d'un cahier des charges fonctionnel qui décrit ce que le système doit accomplir, sans entrer dans les détails techniques.
- ✓ Validation des exigences avec le client pour s'assurer que tout est bien compris.

Résultat : Un document de spécifications des besoins qui servira de référence pour la suite du projet.

Spécifications fonctionnelles

Objectif : Traduire les exigences du client en fonctionnalités détaillées, exprimant comment le système va répondre aux besoins.

Activités :

- ✓ Décomposition des besoins en fonctionnalités précises et mesurables.
- ✓ Documentation des interactions entre le système et les utilisateurs (interfaces utilisateur, flux de données).
- ✓ Établir des critères d'acceptation pour chaque fonctionnalité (comment tester que chaque besoin est satisfait).

Résultat : Un document de spécifications fonctionnelles, définissant les fonctionnalités exactes que le système doit fournir.

Conception globale

Objectif : Définir l'architecture globale du système,
c'est-à-dire comment les différents composants interagiront entre eux.

Activités :

- ✓ Définir les principales composantes du système
(base de données, interfaces, modules fonctionnels).
- ✓ Identification des dépendances entre ces composants.
- ✓ Choix des technologies, langages de programmation et outils qui seront utilisés.
- ✓ Rédaction d'un plan d'architecture avec des schémas et des diagrammes
(UML, par exemple).

Résultat : Un plan d'architecture générale qui sert de cadre pour le développement détaillé.

Conception détaillée

Objectif : Détailler la conception de chaque composant ou module du système.

Activités :

- ✓ Spécification technique de chaque composant
  (structure des bases de données, algorithmes à utiliser, protocoles de communication).
- ✓ Définir les interfaces entre les composants et les méthodes d'intégration.
- ✓ Élaboration des diagrammes détaillés
  (schémas de bases de données, organigrammes, etc.).
- ✓ Planification des tests unitaires pour chaque module.

Résultat : Un document de conception détaillée qui spécifie comment chaque fonctionnalité sera implémentée techniquement.

Réalisation / Implémentation

Objectif : Développer le système selon les spécifications et la conception définies.

Activités :

- ✓ Codage de chaque composant du système en suivant les instructions de la conception détaillée.
- ✓ Développement des interfaces utilisateur et des interactions avec les bases de données.
- ✓ Tests unitaires sur chaque composant développé *
  (vérifier que chaque module fonctionne correctement isolément).
- ✓ Intégration progressive des composants entre eux
  (assurer la compatibilité entre les modules).

Résultat : Un système complet, mais qui n'a pas encore été entièrement testé dans son ensemble.

Tests unitaires

Objectif : Vérifier que chaque module du système fonctionne individuellement.

Activités :

- ✓ Exécution de tests automatisés ou manuels pour chaque composant.
- ✓ Validation que chaque fonction ou module respecte les spécifications techniques définies dans la phase de conception détaillée.

Résultat : Rapport de test et du taux de couverture

Tests d'intégration

Objectif : Vérifier que les différents modules fonctionnent ensemble correctement après intégration.

Activités :

- ✓ Tester les interactions entre les modules (communication, échanges de données).
- ✓ Identifier et corriger les problèmes d'interfaçage ou d'incompatibilité entre les composants.
- ✓ Effectuer des tests de performance pour vérifier la stabilité du système intégré.

Résultat : Un système global intégré, où les modules interagissent correctement entre eux.
Un rapport de test module par module.

Tests de validation

Objectif : Vérifier que le système complet respecte les spécifications fonctionnelles définies au début du projet.

Activités :

- ✓ Exécuter des scénarios de tests basés sur les exigences fonctionnelles (vérifier que chaque fonctionnalité définie dans les spécifications fonctionne correctement).
- ✓ Tester le système dans des conditions proches de la réalité (charge de travail, données réelles).

Résultat : Un système validé relativement aux spécifications fonctionnelles.
Un rapport de test

Test d'acceptation

Objectif : Valider que le système correspond aux attentes du client et aux exigences des utilisateurs finaux.

Activités :

- ✓ Faire tester le système par le client ou des utilisateurs finaux. Éventuellement exécuter les tests de comportement automatisés
- ✓ Valider que les critères d'acceptation définis dans la phase des spécifications des besoins sont respectés.
- ✓ Identifier d'éventuels ajustements ou corrections de dernière minute avant la mise en production.

Résultat : Un système accepté par le client et prêt pour la mise en production.
Un rapport de test

Maintenance

Objectif : Corriger les défauts éventuels et faire évoluer le système en fonction des nouveaux besoins.

Activités :

- ✓ Maintenance corrective : Correction des bugs découverts après la mise en production.
- ✓ Maintenance évolutive : Ajout de nouvelles fonctionnalités en réponse aux nouveaux besoins des utilisateurs ou aux évolutions technologiques.
- ✓ Maintenance préventive : Optimisation des performances, gestion des risques futurs ou mise à jour des technologies.

Résultat : Un système stable, évolutif, et maintenu en bon état de fonctionnement tout au long de son cycle de vie.

Etude de cas

Un hôpital souhaite développer un système de gestion de patients qui devra suivre toutes les informations médicales des patients, y compris les consultations, traitements, hospitalisations, résultats d'examens, et rendez-vous.

Le système doit garantir la sécurité, la confidentialité des données, la conformité aux réglementations médicales (par exemple, RGPD), et une grande fiabilité étant donné l'importance de ces informations pour la prise de décision médicale.

A vous : cadrage

Imaginez les besoins exprimés par le client

Cadrage, quelques suggestions

- ✓ Stockage sécurisé des dossiers médicaux.
- ✓ Gestion des droits d'accès (accès différent selon le personnel).
- ✓ Système d'alerte pour les rendez-vous ou les traitements à suivre.
- ✓ Conformité aux réglementations sur la protection des données (ex. : RGPD).
- ✓ Interopérabilité avec les outils d'autres hôpitaux tout en garantissant le sécurité et la confidentialité

A vous : spécifications

Traduire les besoins en fonctionnalités détaillées, expliquant précisément ce que le système doit accomplir et comment il doit se comporter.

Spécifications, quelques pistes

- ✓ Définir les interfaces utilisateur pour les médecins et infirmiers.
- ✓ Décrire les fonctionnalités comme l'ajout, la modification et la consultation des dossiers médicaux.
- ✓ Définir les règles de sécurité (chiffrement des données, gestion des accès).
- ✓ Élaborer des critères d'acceptation pour chaque fonctionnalité.

A vous : conception globale

Définir l'architecture globale du système,
c'est-à-dire comment les différents composants interagiront.

Conception globale, quelques pistes

- ✓ Définir les principaux composants :
  - ✓ Base de données sécurisée pour les dossiers médicaux.
  - ✓ Interfaces utilisateur pour le personnel hospitalier.
  - ✓ Moteur d'alerte pour les rendez-vous et traitements.
- ✓ Schéma de la base de données
- ✓ Choix des technologies :
  langages de programmation, frameworks, bases de données, etc.
- ✓ Tests d'intégration pour vérifier le bon fonctionnement des modules entre eux

A vous : conception détaillée

Définir le détail de chaque module ou composant
du système à un niveau technique.

Conception globale, quelques pistes

- ✓ Spécification technique des fonctionnalités :
  gestion des utilisateurs,
  gestion des dossiers médicaux, notifications, etc.
- ✓ Définition des interfaces de programmation (API)
  et des interactions entre les modules.
- ✓ Définition des tests unitaires pour chaque composant.

Débat

Comparer cette méthode avec Waterfall, similitudes et différences

Quels sont les avantages et les inconvénients du cycle en V ?

À quel type de projet cette méthode est-elle particulièrement adaptée ?

Avantages et inconvénients

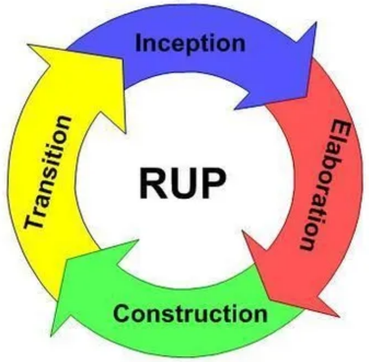
- ✓ Avantages
  - ✓ Traçabilité des exigences
  - ✓ Gestion des risques grâce aux tests systématiques pour valider chaque étape
- ✓ Inconvénients
  - ✓ Pas beaucoup plus souple que le modèle Waterfall
  - ✓ Peu adaptable quand les spécifications changent

Quand l'utiliser

- ✓ Projets critiques avec des exigences strictes
  ex : Défense, médical, Aérospatial...
- ✓ Projets avec des réglementations strictes
  ex : Industrie nucléaire, secteurs pharmaceutique ou financier...
- ✓ Projets nécessitant une grande traçabilité
  ex: Industrie automobile, architecture et génie civil...
- ✓ Projets nécessitant des tests rigoureux et une validation formelle
  ex: Informatique industrielle, secteur ferroviaire...

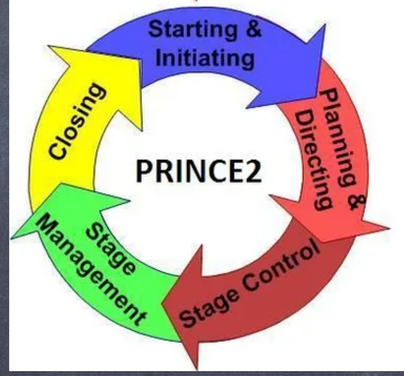
Autres méthodes de gestion de projet

PMI's Five Phases:
1. Initiating
2. Planning
3. Executing
4. Monitoring and Controlling
5. Closing



RUP's Four Phases:
1. Inception
2. Elaboration
3. Construction
4. Transition

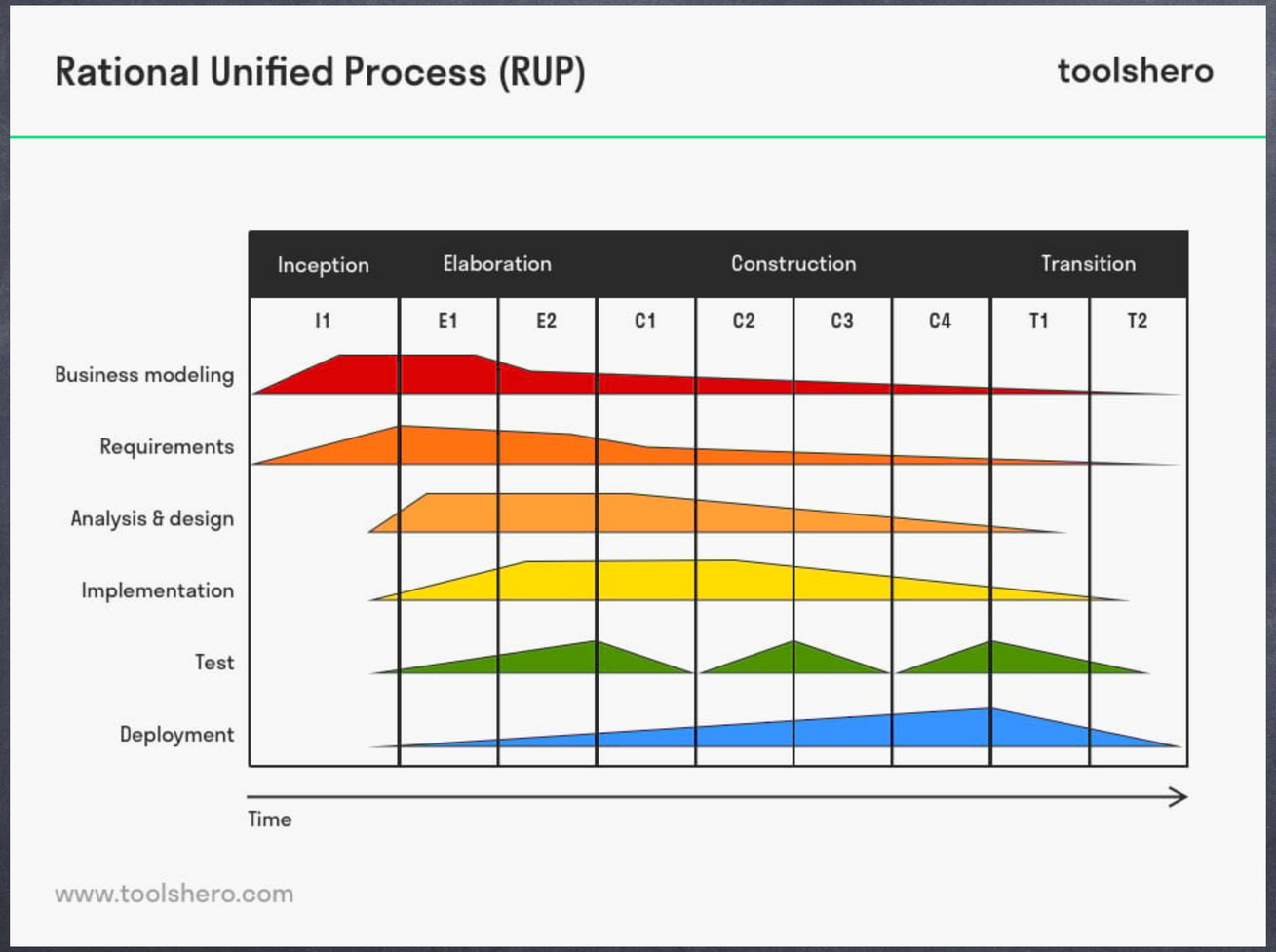
MSF's Five Phases:
1. Envisioning
2. Planning
3. Developing
4. Stabilizing
5. Deploying



Prince2's Five Phases:
1. Starting / Initiating
2. Planning / Directing
3. Stage Control
4. Stage Management
5. Closing

Les phases bouclent, on n'a pas attendu Agile pour faire de la gestion de projet itérative

RUP



RUP prend acte des limites
de l'approche linéaire et séquentielle

Les activités sont réparties
tout au long du cycle de vie du projet

En début de projet, l'analyse,
la spécification et la conception
sont prédominantes

En fin de projet, l'accent est mis
sur l'implémentation et le déploiement

Les tests sont présents tout au long du
cycle de vie