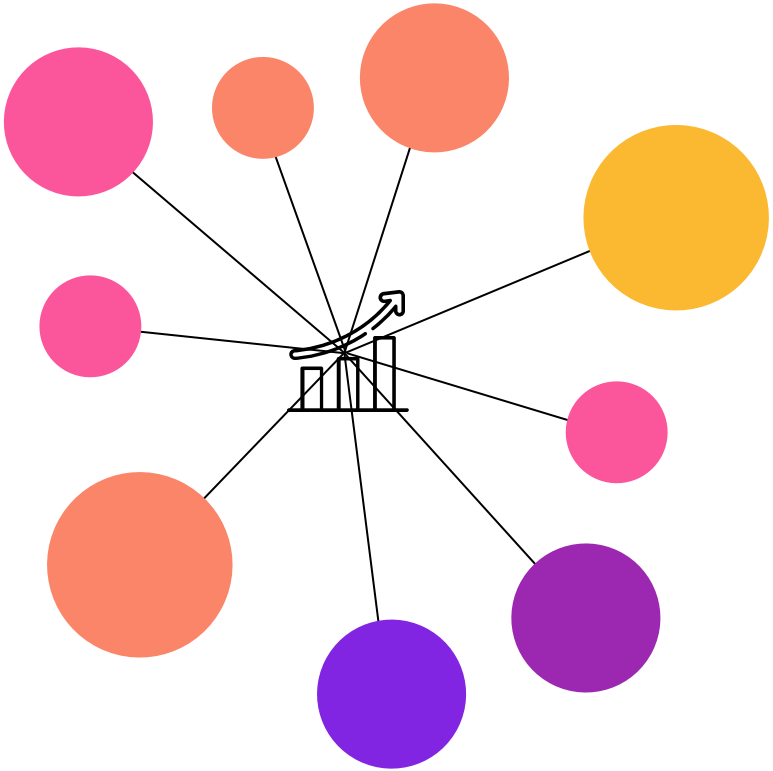
OFPPT



# Approche agile

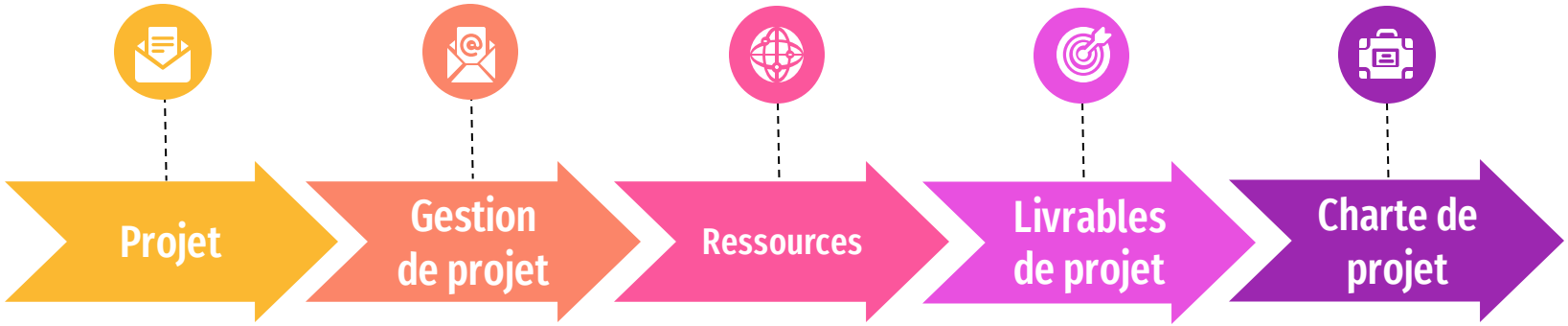
## Gestion de projet

Présenté par:
Berrouan Aya
FS201

# Chapitre 1

# Découvrir les Concepts de gestion de Projet

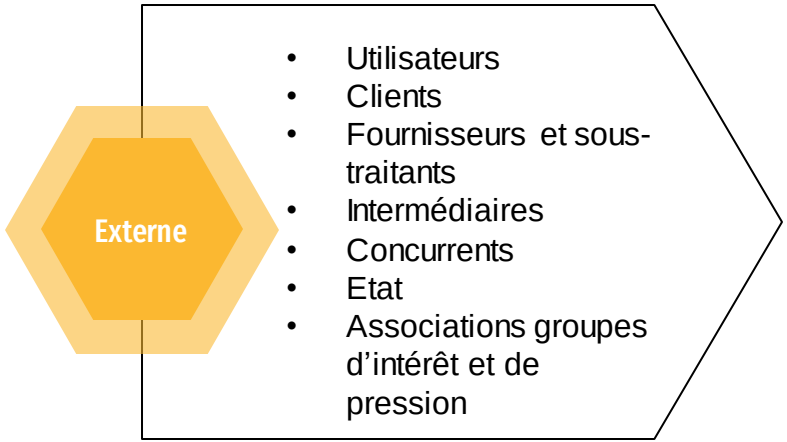
# Concepts de gestion de projet

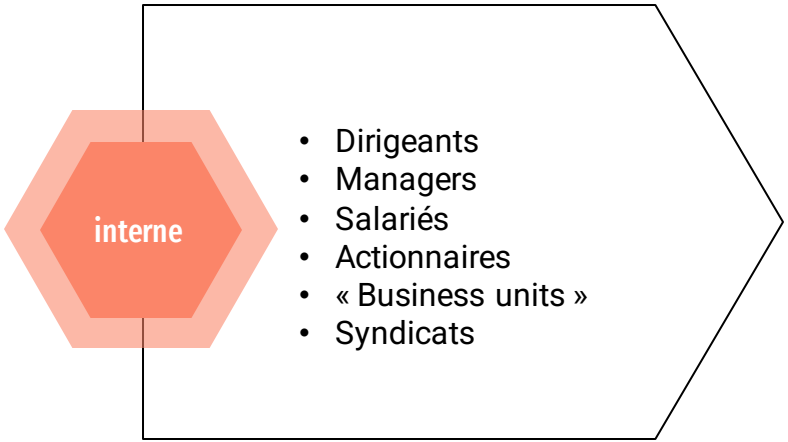



| Projet | Gestion de projet | Ressources | Livrables de projet | Charte de projet |
|---|---|---|---|---|
| Un projet est un ensemble des taches avec un début et une fin pour accomplir un objectif en utilisant des ressources. | La gestion de projet est l'ensemble des activités visant à organiser le bon déroulement d'un projet. | Les ressources sont les éléments essentiels, tels que les personnes, les outils, l'argent et le temps, nécessaires à la réalisation d'un projet. | Les livrables de projet sont les résultats concrets, produits ou documents obtenus à la fin d'un projet, répondant aux objectifs définis. | La charte de projet est un document initial décrivant les objectifs, les responsabilités et le cadre général d'un projet. |

# Parties prenantes de projet

Les parties prenantes de projet sont des acteurs internes ou externes qui ont un intérêt direct ou indirect dans le projet, affectant ou étant affectés par ses résultats.




# Principaux rôles dans un projet informatique

- **Le Chef de projet**

Un chef de projet informatique est un professionnel chargé de planifier, coordonner et superviser les activités liées à un projet informatique, de la conception à la mise en œuvre, en garantissant l'atteinte des objectifs, le respect des délais et du budget, tout en assurant la communication avec les parties prenantes.

- **Matrice d'assignation des responsabilités**

La matrice d'assignation des responsabilités est un outil de gestion de projet qui attribue des rôles et des responsabilités spécifiques à chaque membre de l'équipe, clarifiant ainsi les attentes.

# Caractéristiques de base d’un projet

## Objectifs

Chaque projet a des objectifs spécifiques qui définissent ce qu'il vise à accomplir pour satisfaire un besoin particulier.

## Activités

Les activités sont les tâches à réaliser pour répondre aux objectifs définis.

## Ressources

Les ressources sont les éléments essentiels, tels que les personnes, les outils, l'argent et le temps, nécessaires à la réalisation d'un projet.

## Résultats

Les résultats sont les produits tangibles ou les services créés à partir des activités du projet. Ils contribuent à la réalisation de l'objectif spécifique du projet.

# Contraintes dans la gestion d’un projet

## Contraintes de projet

Ce sont des limites générales qui influent sur un projet, telles que les délais, les coûts et les risques, ayant un impact sur ses performances.

## Contraintes de coûts

Les contraintes de coûts sont les règles sur l'argent qu'un projet peut dépenser ou les coûts qu'il ne doit pas dépasser.

## Contraintes de délais

Les contraintes de délais sont comme des limites de temps pour finir un projet.

## Contraintes dues aux clients

Les contraintes dues aux clients sont des règles ou des délais imposés par les clients pour la réalisation d'un projet.

## Contraintes de qualité

Les contraintes de qualité sont des exigences ou normes assurant que le projet respecte les critères de performance souhaités.

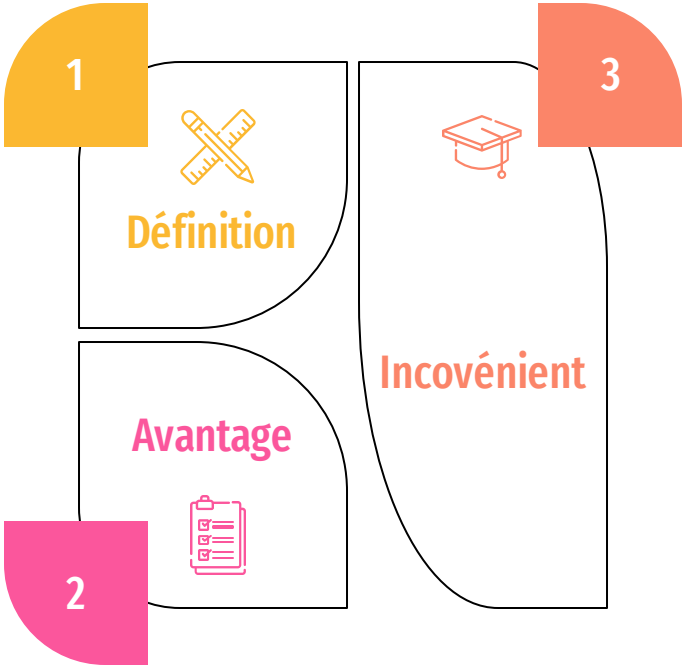
# Chapitre 2

# Découvrir les différentes méthodes de gestion de projet

# Le modèle en cascade

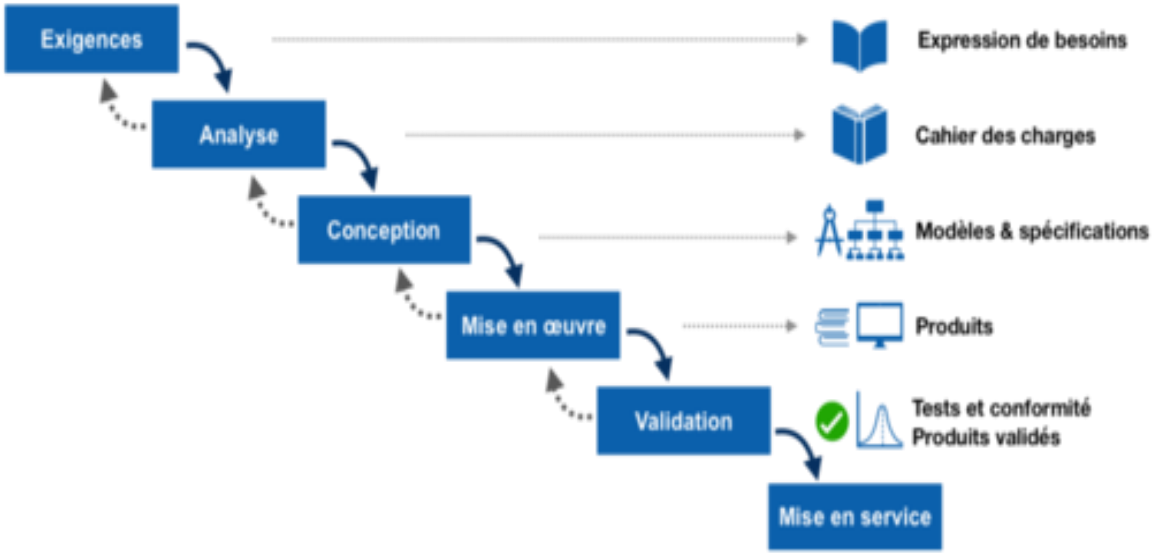
Un modèle en cascade est une approche de traitement séquentiel où chaque étape améliore la sortie de la précédente.

- Clarté du plan initial.
- Suivent une séquence d'étapes fixe, étape par étape.
- Chaque partie du projet est bien expliquée, ce qui aide tout le monde à comprendre ce qui se passe.




- Difficulté à diviser des projets complexes en phases claires.
- Peu de flexibilité pour s'adapter aux changements des exigences.
- Implication tardive de l'utilisateur final.
- Détection tardive des erreurs dans le processus de développement.

# Exemple de modèle en cascade



# Cycle en V

## Definition

Le cycle en V est un modèle de développement de projet qui associe les phases de conception et de test en parallèle pour assurer la qualité du produit.

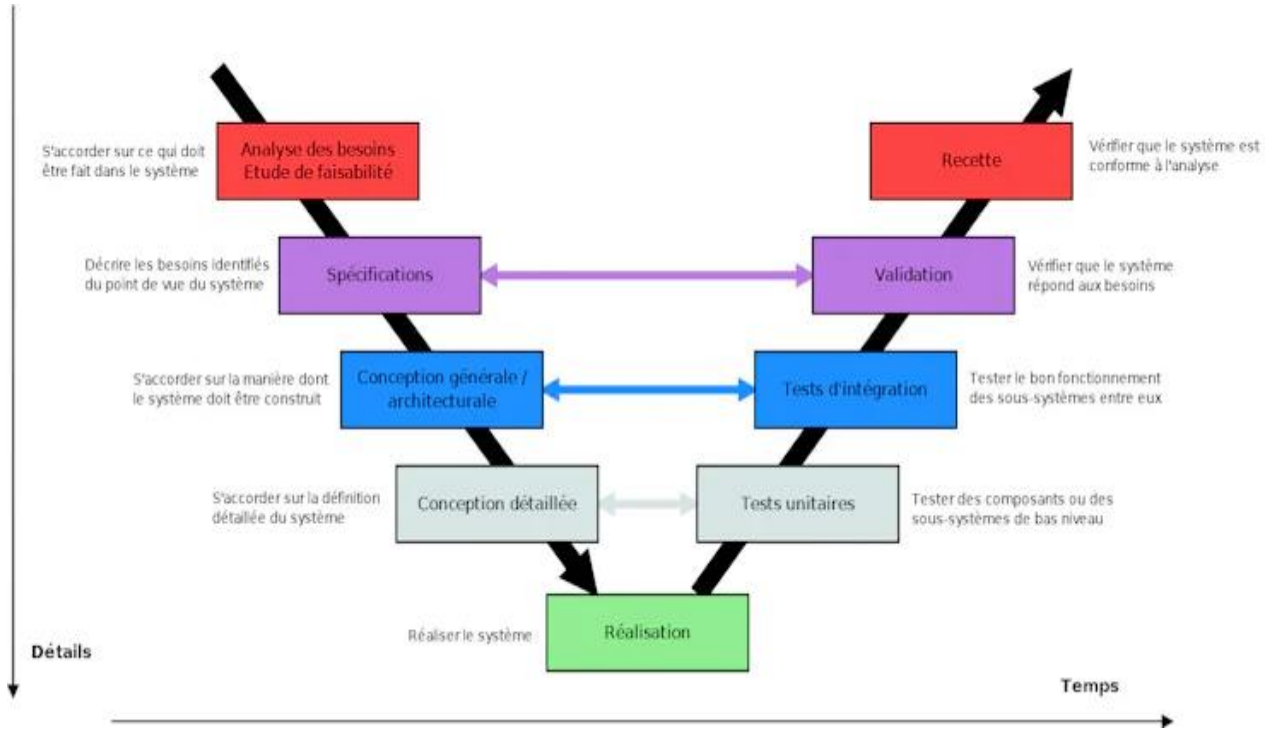
## Avantages

- Évite les retours en arrière fréquents pour redéfinir les spécifications initiales.
- Documentation détaillée et validation solide à chaque phase.
- Processus rigoureux et intuitif, facile à mettre en œuvre.
- Structure de documentation réutilisable d'un projet à l'autre.
- Adapté aux structures multi-sites grâce à des réunions de pilotage ponctuelles.

## Inconvénients

- Peu d'adaptabilité aux changements dans les spécifications.
- Moins réactif aux nouvelles technologies, aux demandes clients et aux imprévus.
- Faible flexibilité du processus, réduisant la prise de risque.
- Documentation précoce non ajustable.
- Délais longs entre les besoins et la livraison du produit final.

# Exemple de cycle en V



# Cycle en Y

## Definition

Le cycle en Y est une approche de gestion de projet qui combine la phase de conception et de test de manière itérative et interactive pour améliorer la qualité.

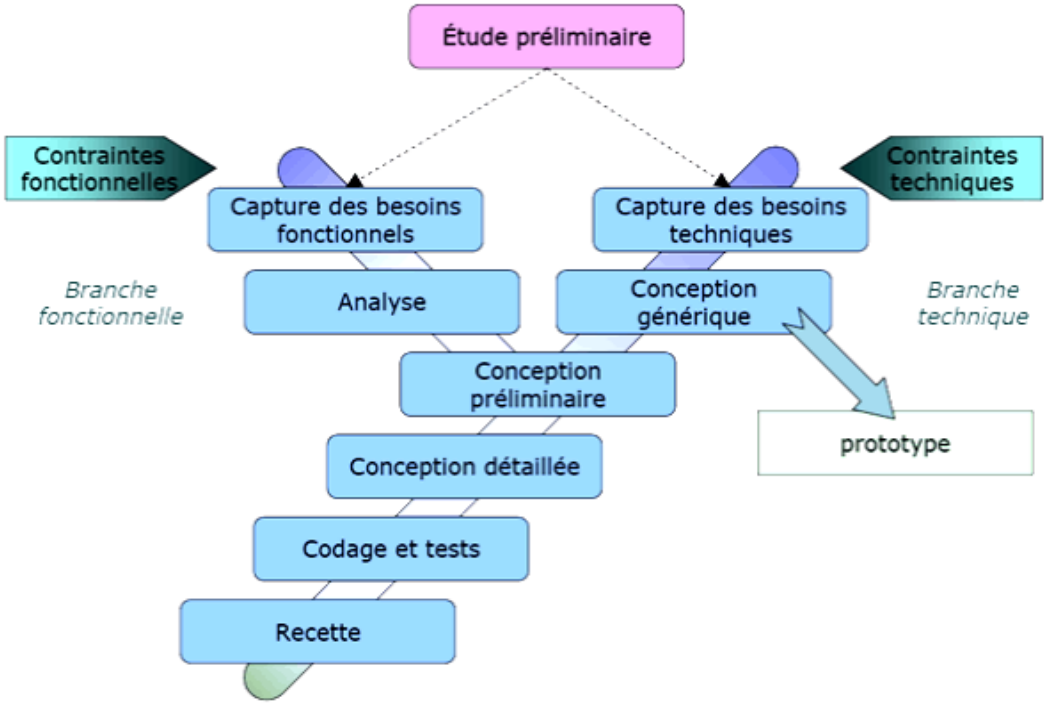
## Avantages

- Intègre les meilleures pratiques de développement.
- Approche itérative, centrée sur l'architecture et les besoins des utilisateurs.
- Gestion des risques et orientation composants.
- Combinaison d'une démarche structurée en phases avec une flexibilité itérative.

## Inconvenients

- Complexité et lourdeur dus aux détails des activités et des artefacts.
- Nécessite une qualification élevée de l'équipe projet, notamment en approches itératives et UML.
- Grande latitude d'adaptation peut conduire à des implémentations rigides malgré l'agilité théorique.

# Exemple de cycle en Y

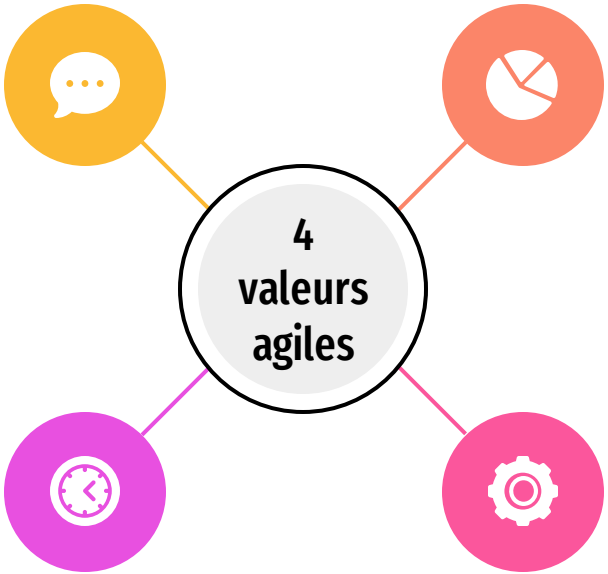


# Méthodes imprévisibles (Agile)

La méthodologie Agile est un processus qui permet à l'équipe de gérer un projet en le décomposant en plusieurs étapes. Elle implique une collaboration constante entre les parties prenantes, une amélioration et une itération continues à chaque étape.

## Aux individus et leurs interactions plutôt qu'aux processus et aux outils;

Les équipes engagées créent de la valeur. Les processus sont utiles, mais les personnes font la différence.




## À la collaboration avec les clients plutôt qu'a la négociation contractuelle;

Agile facilite les changements et la communication avec le client.

## À un logiciel fonctionnel plutôt qu'à une documentation exhaustive;

Les équipes doivent se concentrer sur la création de fonctionnalités, pas sur la paperasse.

## À l'adaptation au changement plutôt qu'au suivi d'un plan.

Agile permet de réagir rapidement aux évolutions, ce qui augmente la stabilité du projet

# Avantages des méthodes agiles

01 Avantage 1

02 Avantage 2

03 Avantage 3

L'Agile rejette la planification détaillée initiale, la considérant contre-productive.

Il favorise des objectifs à court terme et divise le projet en sous-projets, permettant l'adaptabilité aux imprévus.

Accorde une place aux changements et à l'ajustement constant.

# Méthodes traditionnels vs. Méthodes agiles

| Méthode traditionnelles | | Méthodes agiles |
| --- | --- | --- |
| **Processus séquentiel** : Développement étape par étape. | ⟷ | **Approche itérative et incrémentielle :** Développement par petites étapes itératives. |
| **Exigences définies au début :** Exigences initiales strictes. | ⟷ | **Flexibilité :** Capacité à s'adapter aux changements. |
| **Documentation approfondie :** Accent sur la documentation détaillée. | ⟷ | **Collaboration client continue:** Implication constante du client. |
| **Tests à la fin :** Les tests sont effectués après le développement. | ⟷ | **Tests continus** : Les tests sont effectués tout au long du processus. |
| **Rigidité face au changement** : Moins d'adaptabilité aux modifications. | ⟷ | **Livraison fréquente** : Les fonctionnalités sont livrées régulièrement. |
| **Contrôle centralisé :** Décisions centralisées et hiérarchiques. | ⟷ | **Auto-organisation des équipes** : Les équipes prennent des décisions collectivement. |

# Chapitre 3

## Analyser le cahier des charges

# Compréhension des besoins d’un client

## Besoins explicites

Les besoins explicites du client sont clairement exprimés, sans confusion, et servent de base aux objectifs du projet.

## Besoins implicites

Besoins implicites sont des attentes non exprimées, devinées pour répondre aux désirs sous-entendus des clients.

## Livrables potentiels

Livrables potentiels sont les résultats du projet à vérifier pour s'assurer qu'ils satisfont les attentes du client.

# Contexte du projet et ses avantages

Le contexte d'un projet englobe toutes les informations qui caractérisent son environnement, y compris l'histoire, la réglementation, la culture, l'économie, la concurrence, et l'environnement de travail.

1

Identifier les facteurs de réussite.

2

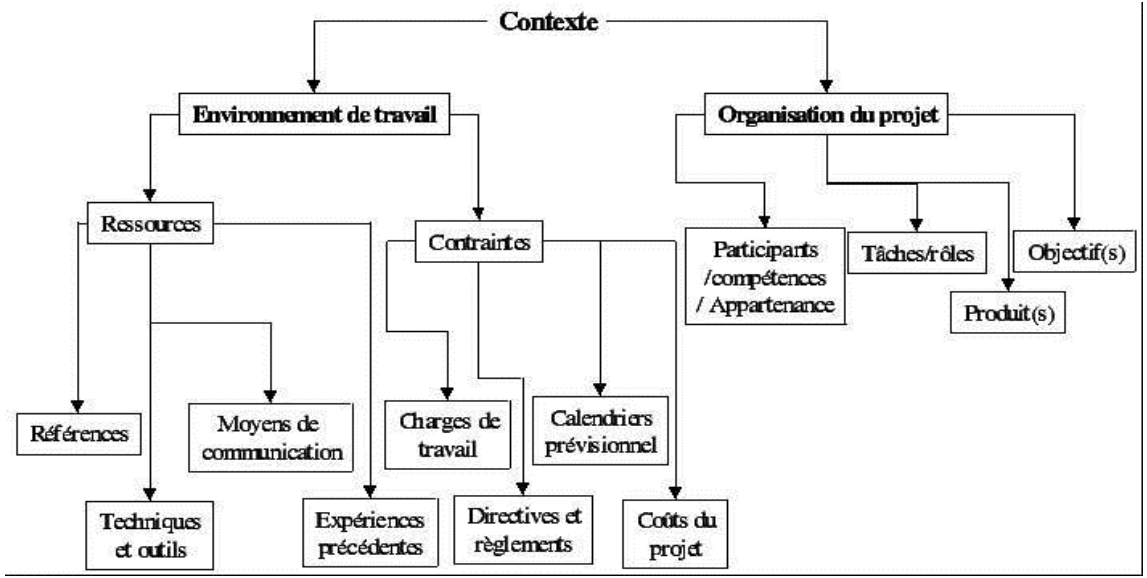
Gérer les obstacles et les motivations, y compris la résistance au changement.

3

Gérer les risques liés à l'environnement du projet.

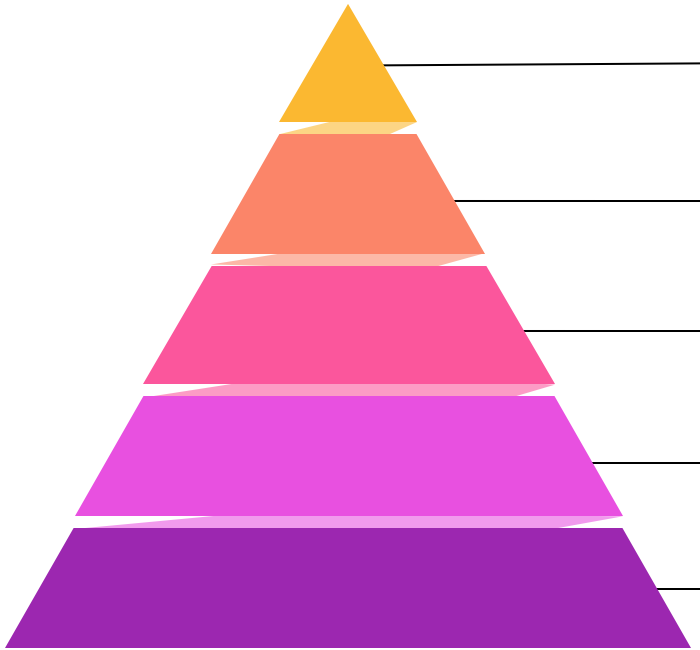
4

Apprendre des expériences d'autres acteurs du marché.



# Périmètre du projet

Le périmètre du projet signifie ce qui est inclus et ce qui ne l'est pas dans le projet, déterminant ainsi ses limites, objectifs, et ressources nécessaires.



**Objectif 1**

Le périmètre inclut tout ce qui doit être fait et assure la qualité du travail.

**Objectif 2**

Il est essentiel pour cadrer le projet, sécuriser son exécution et éviter les dérives.

**Objectif 3**

Il délimite précisément ce qui est inclus et exclu du projet.

**Objectif 4**

Il aide à estimer le temps, le planning et le coût.

**Objectif 5**

Le périmètre est essentiel en gestion de projet, avec les coûts et les délais

# Détection des risques liés à la nature du projet

La détection des risques liés à la nature du projet implique d'identifier les dangers spécifiques au projet afin de les anticiper, prévenir, et gérer efficacement.

**Financiers** — coût supérieur à l' estimation, manque de budget, etc.

**Humains** — manque de compétences, absentéisme, démission au cours du projet, conflits au sein de l'équipe, etc.

**Temporels** — retards des sous-traitants ou des fournisseurs, mauvaise estimation des délais, etc.

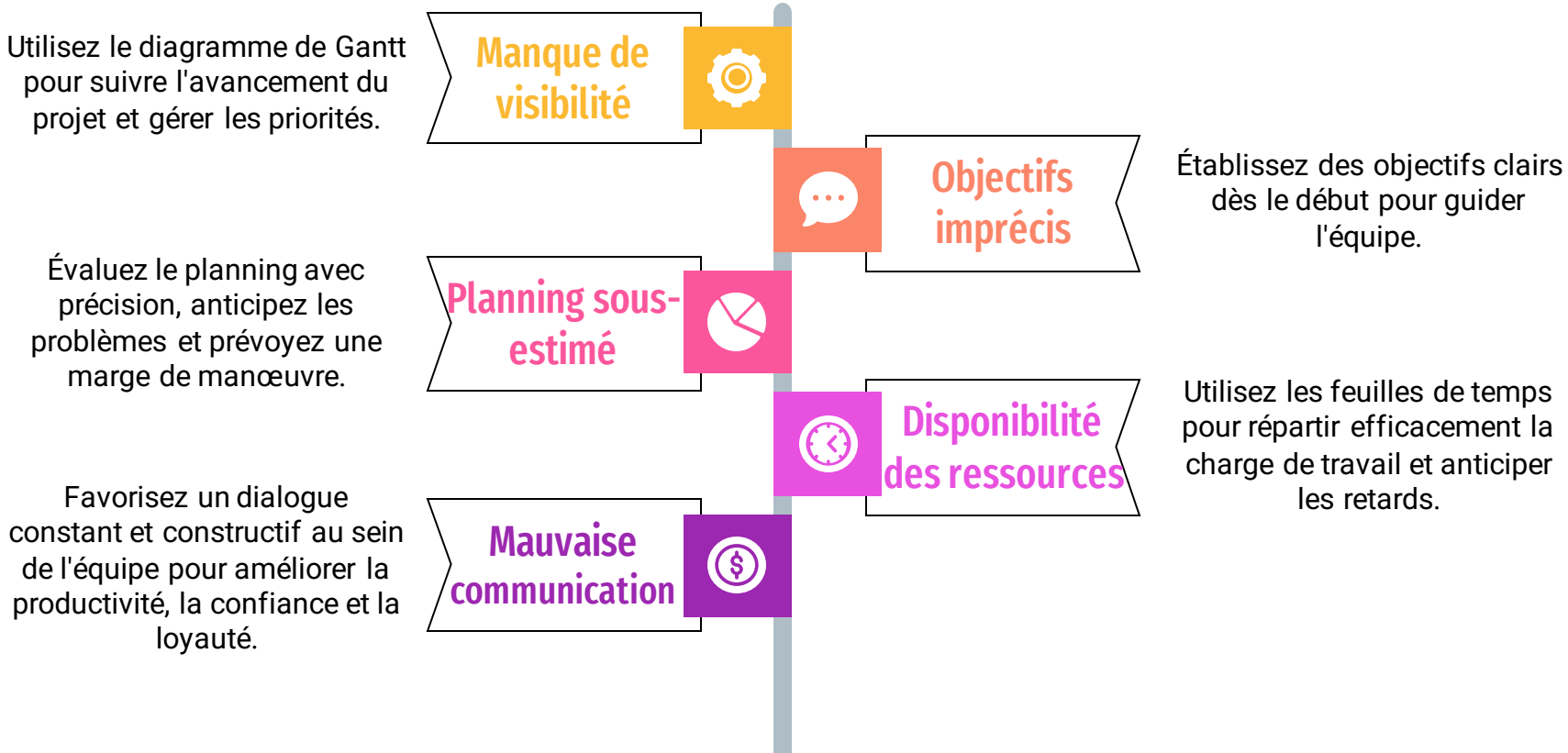
**Techniques** — logiciel inadapté, pannes, matériel obsolète, etc.

**Juridiques** — réglementations et lois à respecter, faillite d'un fournisseur, etc.

**Environnementaux** — impacts négatifs du projet sur l'environnement, ou environnement ayant un impact sur le projet (inondation, sécheresse, tempête...).

**Organisationnels** — changement dans la politique de l'entreprise, changements économiques, etc.

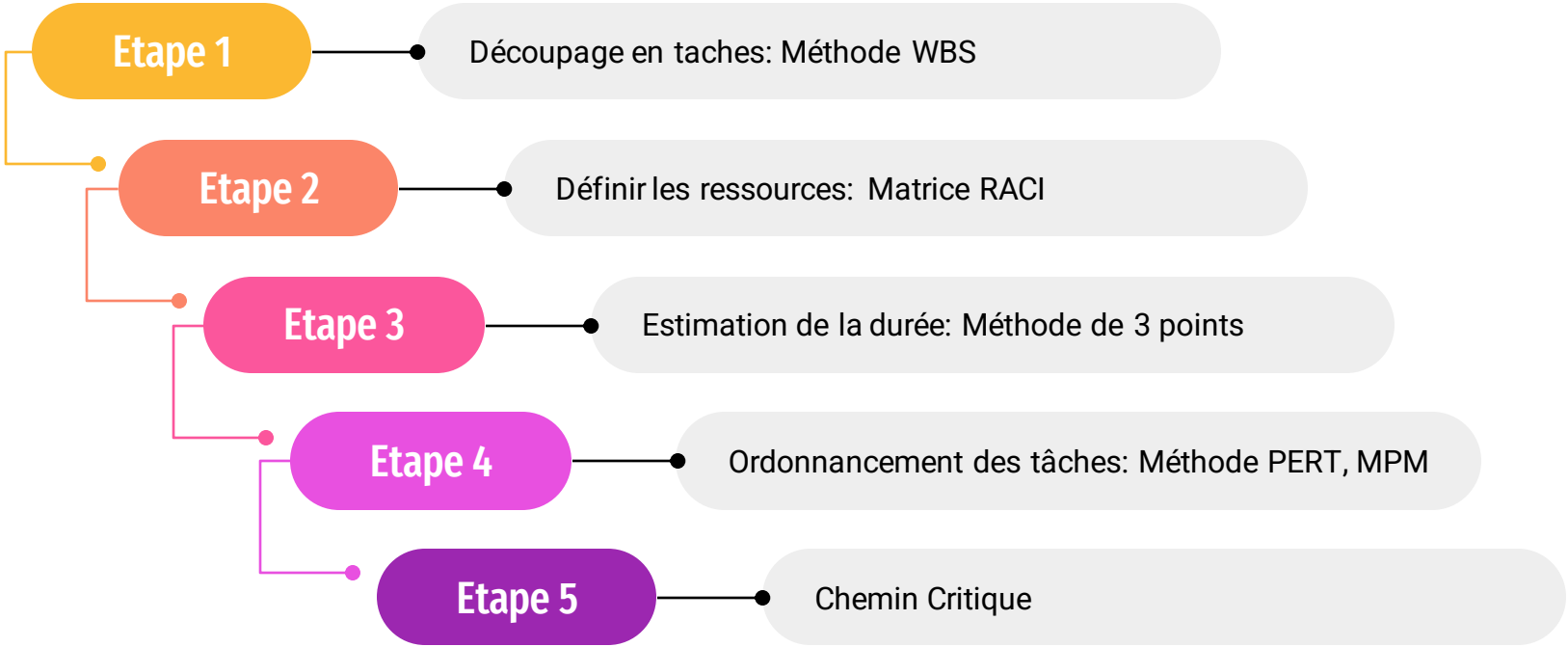
# Proposition des solutions possibles



# Chapitre 4

## Préparer le projet

# Ordonnancement des tâches



# Définition d'organigramme et ses avantages

Un organigramme est une représentation visuelle de la structure hiérarchique ou fonctionnelle d'une organisation, indiquant les relations et responsabilités des individus ou des unités.

Divise le projet en étapes claires pour une vision d'ensemble.

Favorise la communication transparente et compréhensible.

Réduit les risques de dépassement de coûts et d'imprévus en planifiant mieux.

# Répartition de l'ensemble des fonctionnalités en tâches

Décomposer le projet en sous-projets et tâches à partir du résultat global, adaptée aux projets bien connus.

**Approche descendante**

Commencer par les tâches de base, regrouper en sous-projets, utile pour des projets exploratoires.

**Approche ascendante**

Intègre éléments d'approche ascendante et descendante pour gérer projets complexes de manière itérative.

**Approche Combinée**

# La description des 6 phases de gestion d'un projet

| | |
|---|---|
| Avant projet | Évaluation initiale de la pertinence et de la faisabilité du projet, examinant les besoins, opportunités, contraintes et risques potentiels. |
| Initialisation | Définition claire des objectifs du projet, identification des membres de l'équipe, et décision sur la faisabilité du projet. |
| Planification | Élaboration détaillée du plan du projet, incluant la définition des tâches, l'allocation des ressources, l'estimation des coûts, la planification des échéanciers et l'identification des risques. |
| Execution | Mise en œuvre concrète du plan, affectation des ressources et réalisation des tâches nécessaires pour atteindre les objectifs du projet. |
| Contrôle | Suivi continu du progrès du projet, comparaison des résultats réels par rapport au plan, identification des problèmes potentiels, et prise de mesures correctives si nécessaire. |
| Clotûre | Évaluation des résultats du projet, documentation, libération des ressources et conclusion ordonnée du projet. |

# Définition et Avantages de méthode WBS

La méthode WBS est une technique de gestion de projet qui consiste à décomposer un projet mère en tâches enfants, elles-mêmes divisées en sous-tâches.

1. Décomposition du projet en parties gérables.
2. Montre tout le travail, les priorités et les urgences, évitant les surprises.
3. Aide à la gestion du temps.
4. Attribue des responsabilités.
5. Favorise la communication avec les parties prenantes, instillant un esprit de confiance chez le client.
6. Aide à estimer les coûts.
7. Identifie et gère les risques.
8. Base pour une planification détaillée.

# Etapes de méthode WBS

## Créer le découpage WBS

Dans cette étape, vous divisez le projet en catégories, sous-catégories et tâches plus spécifiques.

## Affecter les ressources

Une fois que le WBS est établi, vous identifiez les ressources nécessaires pour chaque tâche ou élément du WBS. Cela peut inclure des membres de l'équipe, des matériaux, des équipements, etc. Vous attribuez ensuite ces ressources aux tâches appropriées.

## Estimer la durée

L'estimation doit être réaliste, tenir compte des contraintes

Méthode des 3 points

- Estimation la plus optimiste (O)
- Estimation la plus probable (M)
- Estimation la plus pessimiste (P)

Durée moyenne avec la formule suivante :

**Durée = (O + 4*M + P) / 6**

# Méthode PERT et ses avantages

PERT (Program Evaluation and Review Technique) est une méthode de gestion de projet utilisant des modèles pour planifier, estimer, contrôler et anticiper les risques.

## Gestion efficace

PERT permet une planification et une gestion plus efficaces des projets, en identifiant les tâches critiques et en optimisant les ressources.

## Prévision des délais

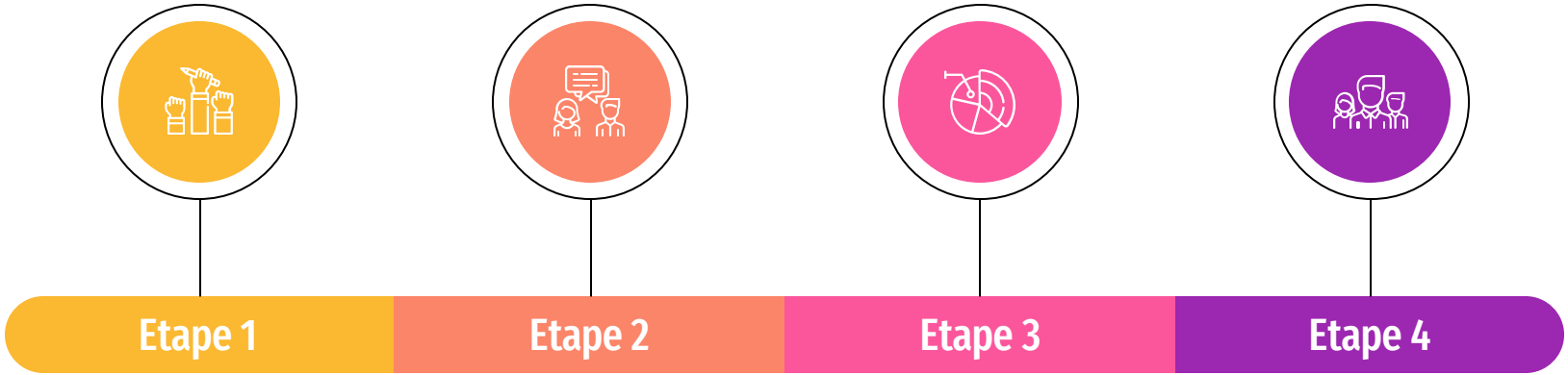
PERT permet d'estimer avec précision la durée totale du projet en prenant en compte les incertitudes et les variations possibles.

## Gestion des risques

PERT aide à anticiper et à atténuer les obstacles potentiels pour respecter les échéances.

# Méthode MPM

La MPM (Méthode des Potentiels et antécédents Métra) est une technique d'ordonnancement basée sur la théorie des graphes pour planifier un projet.



## Etape 1

Identifiez les tâches qui peuvent commencer sans aucune dépendance. Reliez-les au point de départ.

## Etape 2

Repérez les tâches qui dépendent uniquement de ces tâches de départ. Reliez-les à ces tâches.

## Etape 3

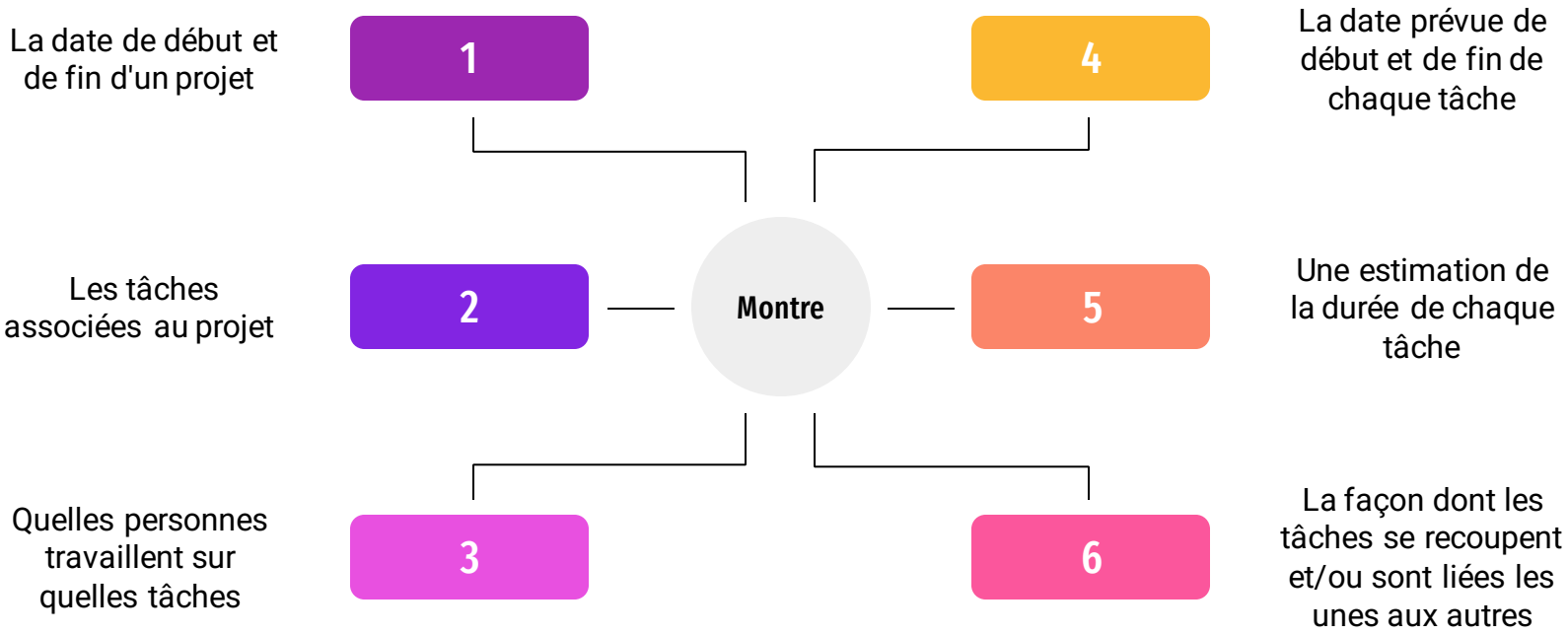
Continuez à identifier les tâches en fonction de leurs dépendances, en les reliant les unes aux autres.

## Etape 4

Enfin, reliez les tâches qui n'ont aucune dépendance au point de fin du projet.

# Diagramme de Gantt

Un diagramme de Gantt est un outil de planification qui représente visuellement les tâches d'un projet, indiquant leurs dates, durées, et relations, pour faciliter la gestion et la coordination.



# GanttProject et ses avantages

GanttProject est un logiciel de gestion de projet open-source qui permet de créer des diagrammes de Gantt, un type de diagramme de planning, pour organiser les tâches et les ressources d'un projet.

1. **Économie de Coûts:** Open-source gratuit.
2. **Création Gantt aisée:** Facilité d'utilisation pour la création de diagrammes de Gantt.
3. **Echanges de données:** Possibilité d'exporter et d'importer des données.

# TeamGantt et ses avantages

TeamGantt est une application en ligne conçue pour la gestion de projet collaborative, en mettant l'accent sur la simplicité et la visualisation de la planification.

1. **Convivialité :** Interface intuitive pour une adoption rapide.
2. **Collaboration en temps réel :** Les équipes peuvent travailler ensemble de manière transparente.
3. **Accessibilité** : Étant basé sur le cloud, TeamGantt est accessible de n'importe où.

# Chemin critique

Le chemin critique dans la gestion de projet est la séquence la plus longue d'activités qui doit être réalisée sans retard pour achever le projet selon le calendrier prévu.

- Découper le projet en tâches.
- Estimer la durée de chaque tâche.
- Identifier les dépendances et interdépendances.
- Calculer le chemin critique.
- Créer le diagramme de réseau des tâches (diagramme de PERT).
- Calculer la marge.

Marge libre est le temps que la tâche peut être retardée sans influencer le début de la tâche suivante sur le même chemin critique.

# Maîtrise des coûts

La maîtrise des coûts en gestion de projet consiste à superviser et gérer les dépenses du projet pour minimiser les risques financiers

# Détermination des points de validation

Le dossier de faisabilité est un document détaillé exposant les aspects techniques, qualitatifs, financiers, et temporels d'un projet pour évaluer son approbation.

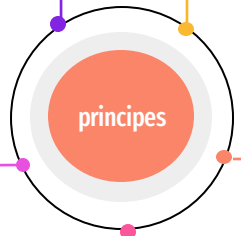
- Justifier le besoin du projet.
- Estimer les coûts du projet.
- Planifier les étapes de réalisation.
- Identifier les risques potentiels.
- Respecter les réglementations.
- Faciliter la compréhension du projet.
- Aider à prendre la décision d'approbation ou de rejet du projet

# Chapitre 5

# Appréhender la méthodologie Agile Scrum

# Définition de la méthode Agile Scrum

Scrum est une méthodologie Agile basée sur des sprints courts, une équipe auto-organisée et une communication quotidienne pour développer efficacement des produits.




## Livrer souvent

On partage des morceaux du projet régulièrement au lieu d'attendre tout à la fin.

## S'améliorer constamment

À la fin de chaque étape, on regarde ce qui a bien marché et ce qui peut être amélioré pour progresser en continu.

## Tout partager

On communique ouvertement sur les problèmes, les avancées et les besoins du projet.

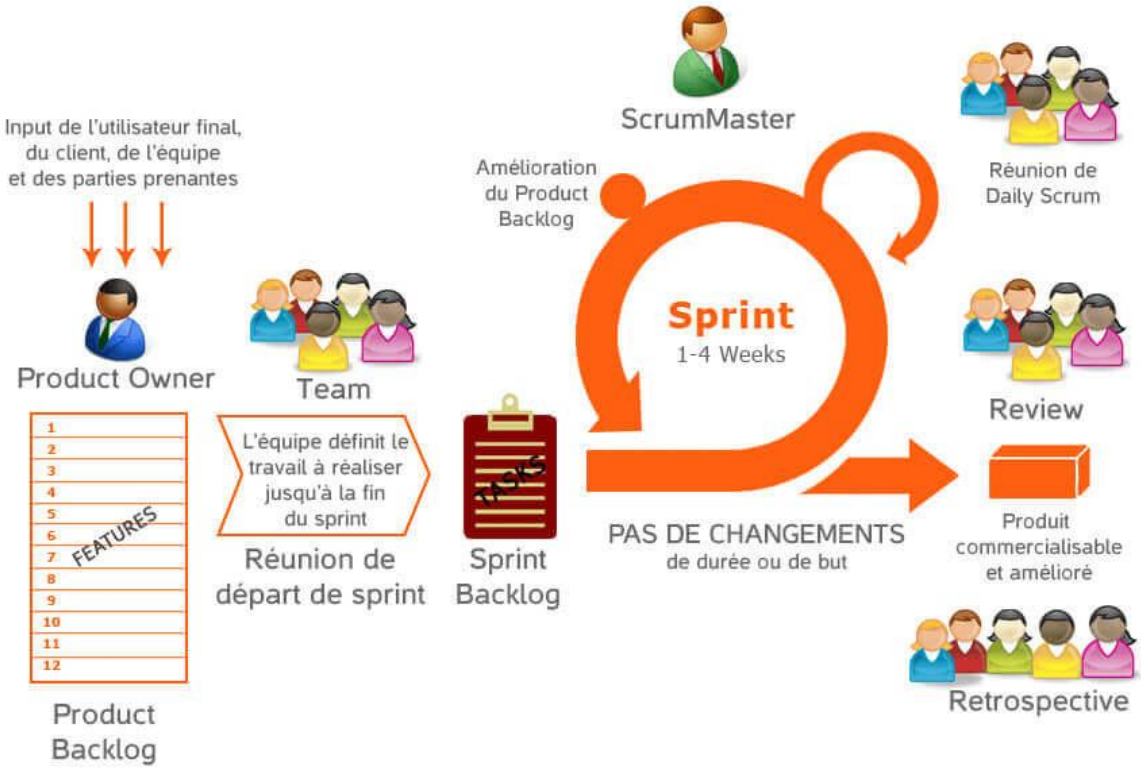
## Travailler ensemble et décider ensemble

L'équipe collabore étroitement et prend des décisions ensemble.

## S'adapter aux changements

On peut changer de direction si nécessaire pour mieux atteindre les objectifs.

# Méthode Scrum



# Processus de la méthode Scrum

## Préparation (Pre-Game)

- **Evénements:**

- Le client et le Product Owner se rencontrent pour discuter des besoins et des exigences du projet.

- Le Scrum Master transforme les userstories en tâches plus petites et gérables.

- Réunion sprint planning meeting pour planifier le sprint en affectant les tâches aux membres de l'équipe, fait par le Scrum Master.

- **Artefacts:** Product Backlog.

- Epics: Grandes fonctionnalités
- User stories: fonctionnalités détaillées décomposées en tâches.

## Réalisation (Game)

- **Evénements:**

- Démarrage de sprint: L'équipe de 6 à 10 personnes commence le travail sur les tâches planifiées.

- Réunion Daily Scurm de 5 à 10 minutes pour discuter des progrès et des problèmes, Les membres de l'équipe répondent à trois questions :
  * Qu'est-ce que j'ai fait hier ?
  * Qu'est-ce que je vais faire aujourd'hui ?
  * Quels problèmes ai-je rencontrés ?

- Fin de sprint

- **Artefacts:** Sprint Backlog.

## Post-traitement (Post-Game)

- **Evénements:**

- Réunion Sprint Review entre le client, Product Owner et l'équipe Scrum pour présenter le travail accompli au client pour valider le sprint et obtenir sa satisfaction.

- Réunion Sprint Retrospective entre l'équipe et Scrum Master pour discuter des points forts du sprint, des défis rencontrés et des aspects à améliorer pour les prochains sprints.

- Clôture(Doc, Demo).

- **Artefacts:** Increment de produit.

# Définitions des éléments de méthode Scrum

## Product Owner

chef de projet en mode agile responsable de définir et prioriser les besoins du produit

## Product Backlog

Une liste de toutes les fonctionnalités, tâches ou idées à réaliser pour le produit, triées par ordre de priorité.

## Sprint Backlog

Une liste de tâches à accomplir pendant la période de travail.

## Sprint

Une période de temps où l'équipe travaille sur certaines tâches, généralement 1 à 4 semaines.

## Scrum Master

Facilitateur de l'équipe Scrum, garantissant le respect des principes et processus Scrum.

## Review

Une réunion où l'équipe montre son travail aux autres pour avoir des avis.

## Rétrospective

Une réunion où l'équipe regarde comment elle a travaillé et comment elle peut s'améliorer.

## Réunion de Départ de Sprint

Une réunion au début d'une période de travail où l'équipe choisit ce qu'elle va faire.

## Réunion de Daily Scrum

Une courte réunion quotidienne où l'équipe parle de ce qu'elle a fait et de ce qui bloque son travail.

## Stakeholders

Ce sont les parties prenantes extérieures à l'équipe Scrum, ayant un intérêt ou une influence sur le produit, et contribuant au développement, à la promotion, et au soutien du produit.

## Team

Composée de développeurs, elle travaille de manière collaborative, sans hiérarchie interne, pour réaliser les tâches. Idéalement, elle compte de 6 à 10 membres.

# Le manifeste définit 12 principes

- Satisfaire le client rapidement.
- Accepter les changements pour plus de valeur.
- Livraisons fréquentes de fonctionnalités.
- Collaboration constante entre utilisateurs et développeurs.
- Autonomie et confiance aux personnes impliquées.
- Privilégier les interactions en face à face.
- Priorité à une application opérationnelle.
- Rythme de travail constant.
- Qualité technique et conception solides.
- Simplicité dans les méthodes de travail.
- Auto-organisation de l'équipe pour de meilleurs résultats.
- Adaptation continue des pratiques pour plus d'efficacité.

# Définition des artefacts

Les artefacts en SCRUM sont des éléments clés de documentation et de suivi, comme des listes de tâches et des versions de produit, utilisés pour organiser et gérer le travail.

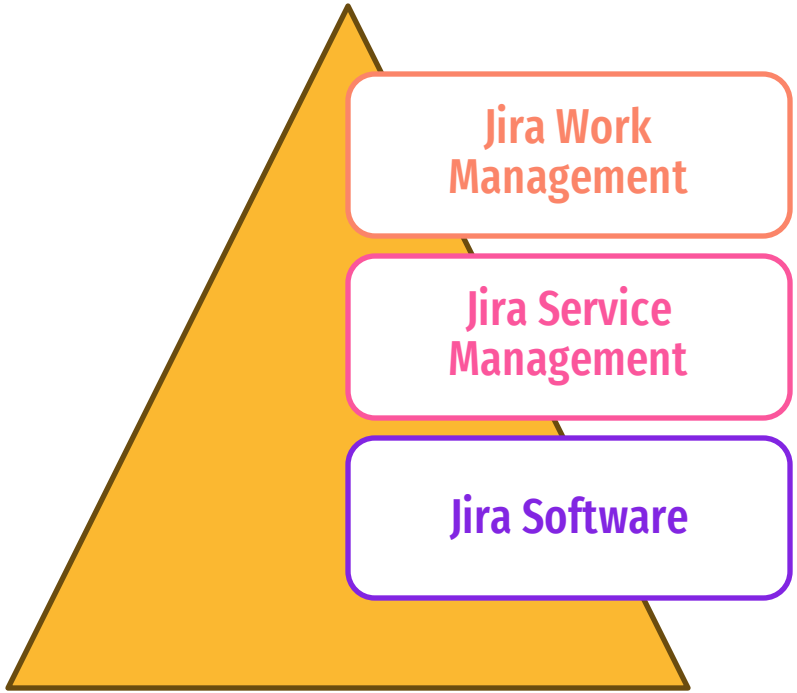
# Chapitre 6

# Manipuler l’outil de gestion de projet agile (Scrum/Jira)

# Définition de Jira et ses objectifs

Jira est un logiciel de gestion de projet d'Atlassian, favorisant l'organisation, la communication et la visualisation des tâches, principalement utilisé par les équipes de développement.

# Les équipes qui utilisent Jira Software




**pour les équipes métier (Marketing/RH/Finance…)**

**pour les équipes de support (DevOps)**

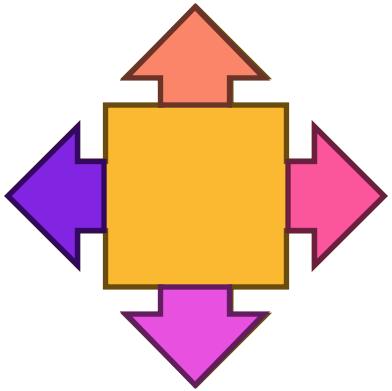
**pour les équipes de développement de logiciel (Équipe Agile) ou pour de la gestion de projet complexe**

# Qu'est-ce qu'un ticket dans Jira ?

Un ticket est une tâche à effectuer, c'est un élément de travail.
Un ticket suit plusieurs étapes : à faire, puis en cours, puis terminé

**Une epic (épopée) :**
Objectif majeur divisé en tâches pour organiser le travail des équipes agiles.

**Une story:**
représente une fonctionnalité à réaliser.



**Un bug:**
désigne un problème à corriger.

**Une tâche :**
est généralement une tâche technique à effectue.

# Chapitre 7

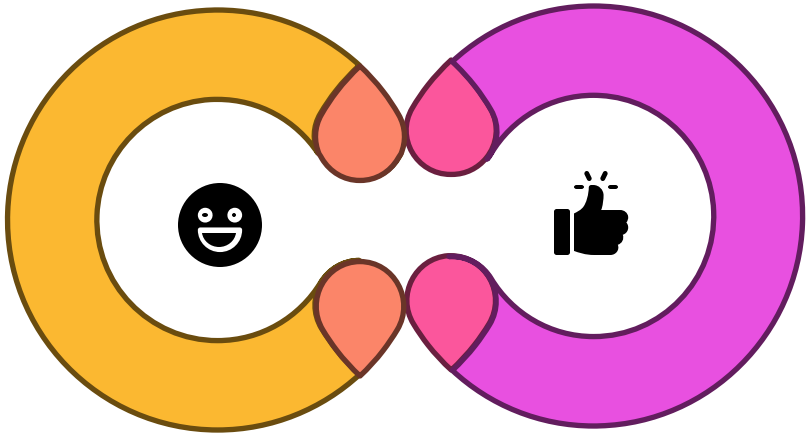
# Manipuler les outils de gestion de versions (Git/Gitlab)

# Définition de Git et ses objectifs

Git est un outil de gestion de versions qui trace les modifications dans le code source, facilitant la collaboration entre les développeurs et assurant un historique fiable des modifications apportées au projet.

## Suivi des modifications

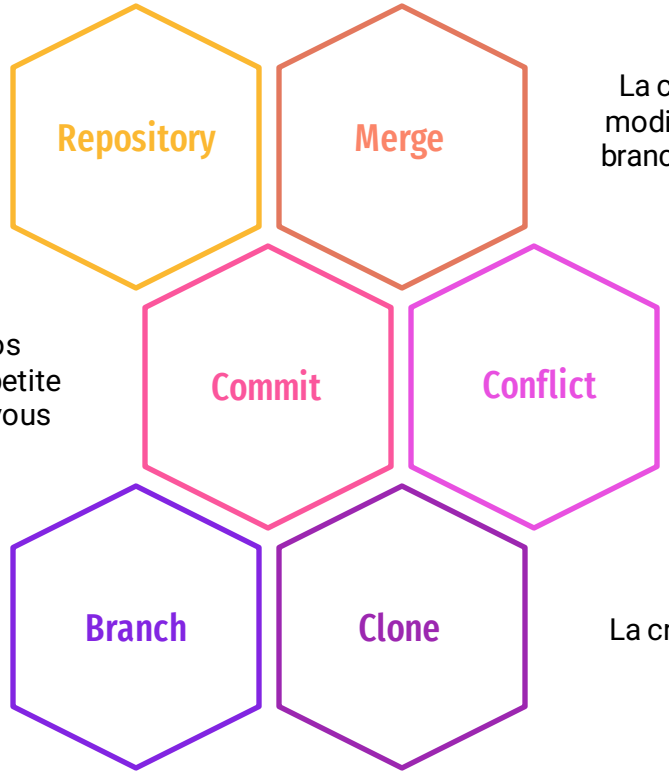
Git permet de suivre l'évolution des fichiers, en enregistrant chaque modification avec un message descriptif.



## Branching et merging

Git simplifie la création de branches pour développer des fonctionnalités isolées, puis les fusionner efficacement sans conflits.

# Les concepts de base de Git




Un espace de stockage contenant l'historique des fichiers et dossiers.

La combinaison des modifications de deux branches en une seule.

Une sauvegarde de vos changements avec une petite note expliquant ce que vous avez fait.

Une situation où Git ne peut pas fusionner automatiquement les modifications et nécessite une intervention manuelle.

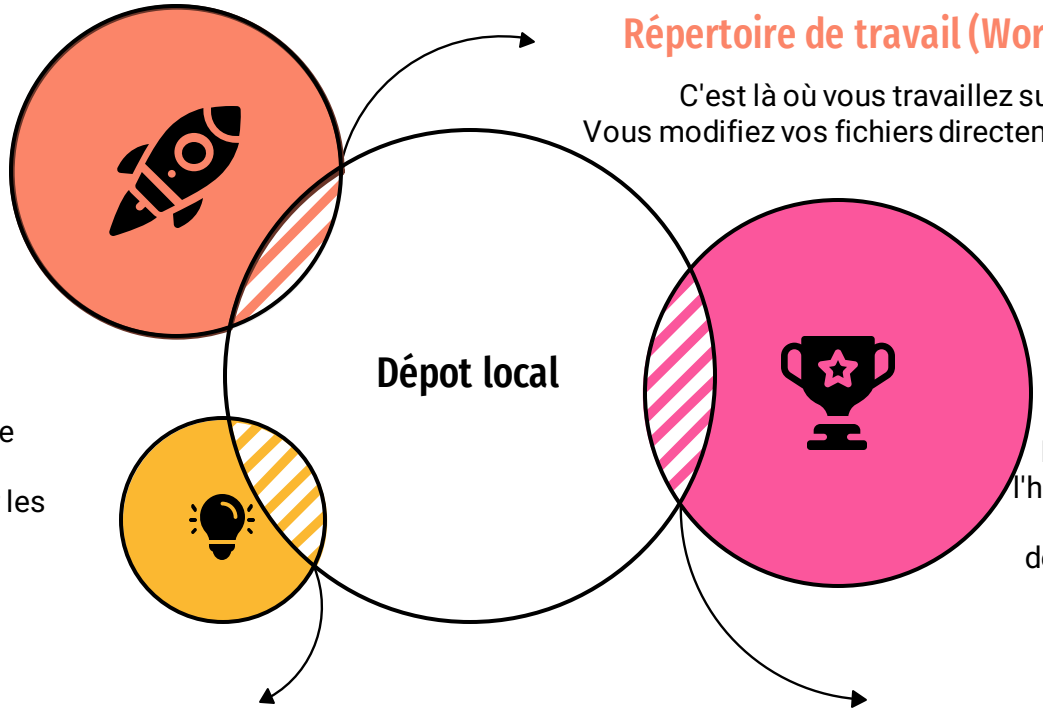
Une copie de votre projet pour développer et tester de nouvelles fonctionnalités sans impacter le projet de base.

La création d'une copie locale d'un dépôt distant.

# Zone de travail local

## Répertoire de travail (Working Directory)

C'est là où vous travaillez sur vos fichiers.
Vous modifiez vos fichiers directement dans cette zone.




## Index (Staging Area)

Zone intermédiaire entre le répertoire de travail et le référentiel pour sélectionner les modifications avant validation(commit)

## Référentiel local (Local Repository)

Espace central où Git stocke l'historique complet des versions d'un projet avec un message descriptif, facilitant le suivi des modifications.

# Définition de GitLab et sa fonctionnalités

GitLab est une plateforme de gestion de projets basée sur Git qui offre des fonctionnalités supplémentaires pour la collaboration et la gestion de projets.

Gestion des dépôts

Suivi des problèmes (Issues)

Intégration continue (CI/CD)

Wikis et documentation

Collaboration en équipe

# Différence entre Github et Gitlab

| Github | | Gitlab |
|---|---|---|
| Repositories privés payants | ⟷ | Repositories privés gratuits |
| Pas d'hébergement gratuit sur un serveur privé | ⟷ | Hébergement gratuit possible sur un serveur privé |
| Aucune plateforme de déploiement intégrée | ⟷ | Déploiement logiciel avec Kubernetes |
| Interface graphique un peu compliqué | ⟷ | Interface graphique clair et facile |

## Commandes de base pour le dépôt :

Configurer les paramètres spécifiques à Git sur votre système:

**Git config --global user. ...**

Créer un nouveau dépôt Git:

**git init**

Ajouter un ou tous fichier à l'index:

**git add <nom-fichier>** **ou** **.**

Affiche la liste des fichiers modifiés:

**git status**

Enregistrer les modifications dans le référentiel local avec un message explicatif:

**git commit -m "message"**

Supprimer des fichiers de l'index et du répertoire de travail

**Git rm**

## Commandes pour l'historique des commits :

Affiche l'historique des commits du dépôt:

**git log**

Affiche les différences entre les fichiers ou les commits:

**git diff**

Réinitialiser l'index et le répertoire de travail à l'état de dernier commit(annuler commit)

**git reset**

Marquer des commits spécifiques avec des poingnées simples:

**git tag**

Enregistrer les changements qui ne doivent pas être commit immédiatement:

**git stash**

## Commandes pour la gestion des dépôts distants:

Affiche la liste des dépôts distants:

**git remote**

Envoyer les modifications locales apportées à la branche principale associé:

**git push origin main**

Récupère et fusionne les modifications depuis un dépôt distant :

**git pull origin main**

Récupère les informations des dépôts distants:

**git fetch**

Affiche l'auteur de chaque ligne d'un fichier:

**git blame <fichier>**

## Commandes pour la gestion des branches:

Affiche la liste des branches:
**git branch**

Créer une branche:
**git branch <nom-branche>**

Supprimer une branche:
**git branch –d <nom-branche>**

Crée une branche et déplacer à cette branche:
**git checkout -b <nom-branche>**

Déplace à une autre branche:
**git checkout <nom-branche>**

Fusionne une branche dans la branche courante:
**git merge <nom-branche>**

## Commandes Avancées

Afficher des informations sur tout fichier git:
**git show**

Créer un fichier zip ou tar contenant les composants d'un arbre du dépôt:
**git archive**

Effectuer une vérification d'intégrité du système de fichiers git:
**git fsck**

Utiliser pour la réapplication des commits sur une autre branche
**git rebase**

Supprime les objets Git non référencés, optimise l'espace de stockage.
**git prune**

Afficher un fichier arborescent avec le nom et le mode de chaque élément et la valeur SHA-1 du blob:
**Git ls-tree**

Affiche le type de l'objet Git correspondant à la valeur SHA-1:
**git cat-file**

Afficher l'interface graphique du depôt local:
**Gitk**

Lancé un serveur web local pour interagir avec un dépôt.
**git instaweb**

Optimiser le dépôt en supprimant les fichiers inutiles et les optimiser:
**Git gc**

# Autres Commandes

Affiche l'historique des modifications des références (branches, HEAD)

**Git reflog**

Réinitialise l'index et le répertoire de travail pour qu'ils correspondent à un commit spécifique, supprimant toutes les modifications non commises. Utile pour revenir à un état propre du dépôt:

**git reset --hard <commit>**

Revenir au commit précédent et supprimer toutes les modifications non commises. Utile pour annuler complètement les changements récents:

**git reset --hard HEAD~1**

Ajouter et committer toutes les modifications des fichiers suivis avec un message de commit:

**git commit -am "message"**

Lister tous les tags du dépôt.

**git tag**

Créer un nouveau tag pour marquer un point important dans l'historique du projet:

**git tag <nom-tag>**

Envoie tous les tags locaux vers le dépôt distant. Utile pour partager les versions taguées avec les autres contributeurs:

**git push --tags**

Basculer sur le commit spécifié par un tag pour examiner l'état du projet à ce moment-là:

**git checkout <nom-tag>**

Redéfinir le message de commit précédent:

**git commit –amend -m "nouveau message"**

# Chapitre 8

# Manipuler les outils de mesure de la qualité du code (SonarQube)

# Définition de SonarQube et ses objectifs

C'est un logiciel open source pour mesurer la qualité du code source dans les projets de développement.

| | |
|---|---|
| **Serveur SonarQube** | une interface web pour visualiser les résultats et gérer les configurations de projet. Vous pouvez voir facilement les problèmes et décider comment les résoudre. |
| **Sonar Scanner** | est un outil utilisé par SonarQube pour scanner votre projet vers SonarQube pour que ce dernier puisse lire le langage de programmation utilisé. |
| **Base de donnée** | stockent données liées projets, analyses. Compatibles avec divers systèmes tels que PostgreSQL, MySQL, Oracle dans SonarQube. |


Sonar couvre les 7 axes de la qualité du code : **architecture & design , documentation ,respect des standards de codage ,non duplication du code , tests unitaires ,complexité ,bogues potentiels**

# Tableau de bord avec SonarQube

Affiche: les bugs détectés, vulnérabilités de sécurité, indicateurs de mauvaises pratiques, dette technique en jours, pourcentage de lignes de code couvertes par des tests, pourcentage de code dupliqué, complexité cyclomatique moyenne

Les catégories des défauts logiciels:

1

## Bugs

Une erreur de programmation entraînant un comportement indésirable dans le logiciel, nécessitant correction pour fonctionner correctement.

2

## Vulnérabilités

Faiblesses du code pouvant nuire au système, impactant la sécurité de l'application.

3

## Code Smells

se réfèrent à des structures de code source qui indiquent des problèmes potentiels ou des améliorations nécessaires.

# Scores de maintenabilité, fiabilité et sécurité (MFS)

| Score | Fiabilité | Sécurité | Maintenabilité |
|---|---|---|---|
| A | 0 bug | 0 vulnérabilité | 0,00<=ratio dette technique<=0,05 |
| B | Au moins 1 bug mineur | Au moins 1 vulnérabilité mineure | 0,06<=ratio dette technique<=0,1 |
| C | Au moins 1 bug majeur | Au moins 1 vulnérabilité majeure | 0,11<=ratio dette technique<=0,20 |
| D | Au moins 1 bug critique | Au moins 1 vulnérabilité minecritique | 0,21<=ratio dette technique<=0,50 |
| E | Au moins 1 bug bloquant | Au moins 1 vulnérabilité bloquante | 0,51<=ratio dette technique<=1 |

**Le ratio dette technique:** est le ratio entre le coût pour remédier aux problèmes de type code smell et le coût de développement de l'application. La formule de calcul est la suivante : **coût total de remédiation des issues / (coût pour développer une ligne de code x nombre de lignes de code)**

# Chapitre 9

# Introduire la chaîne DevOps

# Définition de DevOps et ses avantages

DevOps est une approche favorisant la collaboration et la communication entre développeurs et professionnels des opérations, automatisant la livraison logicielle et les changements d'infrastructure.

### Collaboration

Encourage le travail d'équipe pour améliorer le flux de travail.

### Vitesse

Permet d'introduire rapidement de nouveaux produits sur le marché.

### Agilité

S'adapte facilement à la demande et améliore la gestion des changements

### Sécurité

Simplifie la gestion de la sécurité et accélère la récupération en cas d'incidents (DevSecOps)

# Outils de DevOps

- **Gestion de Code Source:** Git, GitHub, GitLab, Bitbucket
- **Tests CI/CD :** Jenkins, GitlabCI
- **Conteneurs :** Docker, Kubernetes
- **Cloud Providers :** Google Cloud Platform, Azure, AWS
- **Automatisation et Configuration :**Terraform, Ansible, Puppet, Salt
- **Monitoring et Alerting :**Prometheus, Grafana, ELK
- **Gestion de Projet:** Jira, Trello
- **Gestion des Secrets:** Vault

# Difference entre Agilité et DevOps

| Comparaison | Agilité | DevOps |
|---|---|---|
| **Portée** | L'Agile se concentre principalement sur le processus de développement | DevOps englobe l'ensemble du cycle de vie du logiciel, y compris le déploiement, l'exploitation et la maintenance. |
| **Objectifs** | Agile vise à répondre rapidement aux changements dans les besoins du client | DevOps vise à améliorer la collaboration entre les équipes de développement et d'exploitation pour accélérer la livraison et améliorer la qualité. |
| **Automatisation** | Peu ou pas d'automatisation | Automatisation au cœur de la pratique |

# Chapitre 10

# Mettre en place la CI/CD avec Gitlab

# Définition des notions CI/CD

## Intégration continue (CI)

Pratique automatisée du développement où les modifications de code sont régulièrement intégrées, facilitant une détection précoce des erreurs.

Etape 1

## Livraison continue (CD)

Processus automatisé préparant chaque version pour déploiement, nécessitant validation manuelle avant la mise en production.

Etape 2

## Déploiement continue (CD)

Processus automatisé préparant chaque version pour déploiement, sans nécessite une intervention manuelle après chaque intégration réussie.

Etape 3

# Définition de Gitlab CI/CD

GitLab CI est un puissant système d'intégration continue qui automatise les étapes de build, test et deploy.

### Un pipeline

est le composant principal orchestrant les étapes de l'intégration, de la livraison et du déploiement continu dans GitLab.

01

### Un runner

est une application collaborant avec GitLab CI/CD pour exécuter les tâches définies dans un pipeline.

02

### Une tâche (Job)

est un ensemble d'instructions exécutées par un runner, potentiellement produisant des artefacts utiles.

03

### Les étapes (Stages)

sont des phases déterminant le moment d'exécution des tâches dans un pipeline, avec des conditions de passage à l'étape suivante.

04

### Les artefacts de pipeline

sont des fichiers générés après l'exécution d'un pipeline, utilisés pour collecter des informations telles que la couverture de test.

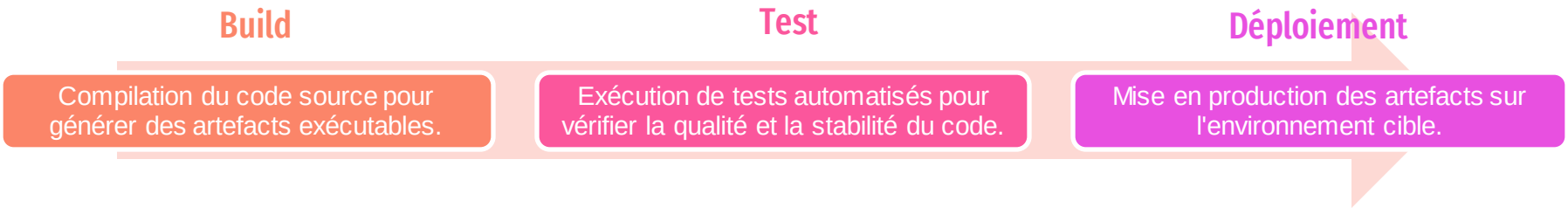
05

### Les tags

sont utilisés pour sélectionner un runner spécifique parmi la liste.

06

# Les stages de pipeline CI/CD




# Avantages pipeline CI/CD

- Automatisation du Processus
- Livraisons Fréquentes et Rapides
- Détection Précoce des Erreurs
- Amélioration de la Qualité du Code

# .gitlab-ci-yml

```
19 + stages:          # List of stages for jobs, and their order of execution
20 +   - build
21 +   - test
22 +   - deploy
23 +
24 + build-job:       # This job runs in the build stage, which runs first.
25 +   stage: build
26 +   script:
27 +     - echo "Compiling the code..."
28 +     - echo "Compile complete."
29 +
30 + unit-test-job:   # This job runs in the test stage.
31 +   stage: test    # It only starts when the job in the build stage completes successfully.
32 +   script:
33 +     - echo "Running unit tests... This will take about 60 seconds."
34 +     - sleep 60
35 +     - echo "Code coverage is 90%"
36 +
37 + lint-test-job:   # This job also runs in the test stage.
38 +   stage: test    # It can run at the same time as unit-test-job (in parallel).
39 +   script:
40 +     - echo "Linting code... This will take about 10 seconds."
41 +     - sleep 10
42 +     - echo "No lint issues found."
43 +
44 + deploy-job:      # This job runs in the deploy stage.
```

```yaml
stages :
  - build
  - test
  - deploy

build_job :
  stage : build
  script :
    - echo "Building the application"
    - npm install  # Install the dependencies
    - npm run build  # Build the application
  artifacts :  # Define the files that should be passed between jobs
    paths :
      - build/  # Pass the build directory to the next job
  only :  # Define the conditions under which the job should run
    - master  # Only run this job on the master branch

test_job :
  stage : test
  script :
    - echo "Testing the application"
    - npm install  # Install the dependencies
    - npm test  # Run the tests
  except :
    - master  # Do not run this job on the master branch

deploy_job :
  stage : deploy
  script :
    - echo "Deploying the application"
    - npm install  # Install the dependencies
    - npm run deploy  # Deploy the application
  rules :
    - if : $CI_COMMIT_BRANCH == "master" && $CI_COMMIT_TAG == "v*"
    # Only run this job on the master branch and for tags that start with 'v'
```

# Marge libre et Marge total

- Marge libre: date plutôt de tache suivante - date plutôt de tache précédente - durée tache
- Marge total: date plus tard de tache suivante - date plutôt de tache précédente - durée tache

# Exercices: PERT et Gantt

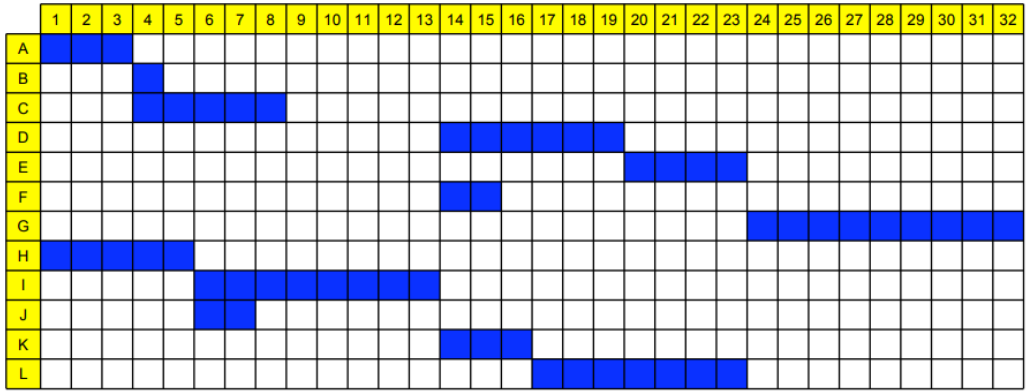
## Exercice 1

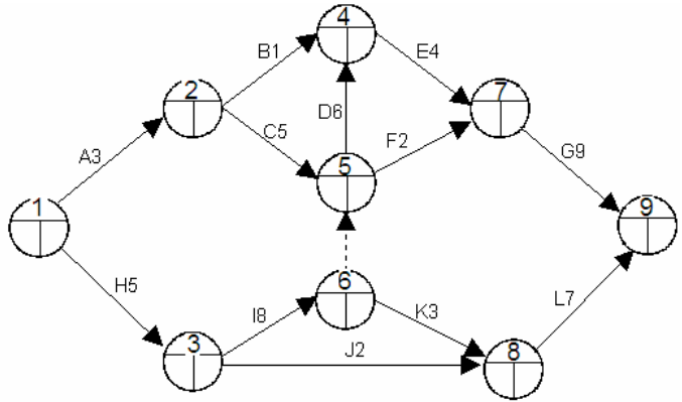
| tâches | antécédents | durée |
|---|---|---|
| A | / | 3 |
| B | A | 1 |
| C | A | 5 |
| D | C,I | 6 |
| E | B,D | 4 |
| F | C,I | 2 |
| G | E,F | 9 |
| H | / | 5 |
| I | H | 8 |
| J | H | 2 |
| K | I | 3 |
| L | K,J | 7 |

## Exercice 2

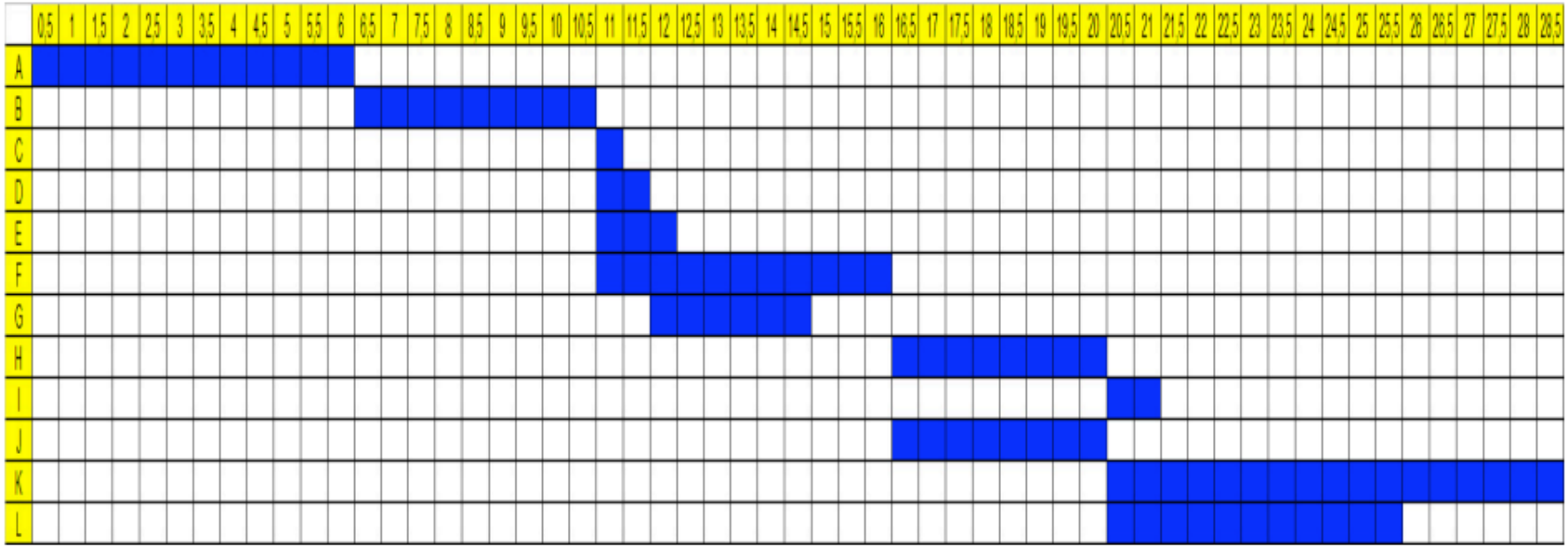
| Tâches élémentaires | Durée (en h) | Contraintes d'antériorité |
|---|---|---|
| A | 6 | - |
| B | 4,5 | A |
| C | 0,5 | B |
| D | 1 | B |
| E | 1,5 | B |
| F | 5, 5 | B |
| G | 3 | C, D |
| H | 4 | E, F, G |
| I | 1 | H, J |
| J | 4 | E, F, G |
| K | 8,5 | H, J |
| L | 5,5 | H, J |

# Réponse Exercice 1



# Réponse Exercice 2

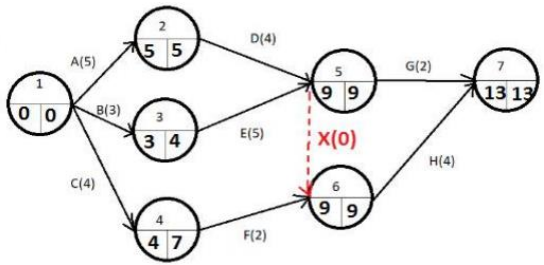


# Exercice 3

| Tâche | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| Antériorité | ------- | ------- | ------- | A | B | C | E , D | E , F |
| Durée | 5 | 3 | 4 | 4 | 5 | 2 | 2 | 4 |

1. Déterminer les niveaux des tâches.
2. Tracer le graphe PERT et calculer les dates au plus tôt et les dates au plus tard pour chaque sommet.
3. Calculer les marges libres et les marges totales.
4. Déterminer le chemin critique.

# Exercice 3: Réponses

1. Le Niveau 0 : ( A-B-C) ; Le Niveau 1 : (D–E–F) ; Le Niveau 2 : (G–H)
2. 

3.

| Tâches | Marge libre ( ML ) | | Tâche | Marge totale ( MT ) |
|---|---|---|---|---|
| A | 5-0-5 = 0 | | A | 5-0-5= 0 |
| B | 3-0-3= 0 | | B | 4-0-3= 1 |
| C | 4-0-4= 0 | | C | 7-0-4= 3 |
| D | 9-5-4= 0 | | D | 9-5-4= 0 |
| E | 9-3-5= 1 | | E | 9-3-5= 1 |
| F | 9-4-2= 3 | | F | 9-4-2= 3 |
| G | 13-9-2= 2 | | G | 13-9-2= 2 |
| H | 13-9-4= 0 | | H | 13-9-4= 0 |

4. Les tâches critiques : A - D - H
Le chemin critique ( ADH)

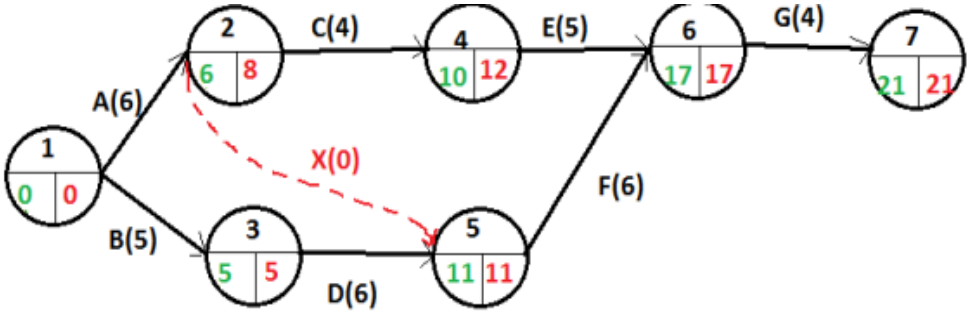
# Exercice 4

| Tâche | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| Tâche antérieur | ------ | -------- | A | B | C | A , D | E , F |
| Durée | 6 | 5 | 4 | 6 | 5 | 6 | 4 |

1. Tracer le graphe PERT et calculer les dates au plus tôt et les dates au plus tard pour chaque sommet..
2. Calculer les marges libres et les marges totales.
3. Déterminer le chemin critique.

# Exercice 3: Réponses

1.




2.

| tâche | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| ML | 0 | 0 | 0 | 0 | 2 | 0 | 0 |
| MT | 2 | 0 | 2 | 0 | 2 | 0 | 0 |

3. Les tâches critiques : B-D-F-G
Le chemin critique (BDFG)

# Exercice 5

| Tâche | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| Tâche antérieur | --- | --- | A | A | A,B | C | C,D | E | F,G | H,I |
| Durée | 6 | 2 | 3 | 7 | 2 | 4 | 2 | 3 | 4 | 1 |

1. Tracer le graphe PERT et calculer les dates au plus tôt et les dates au plus tard pour chaque sommet..

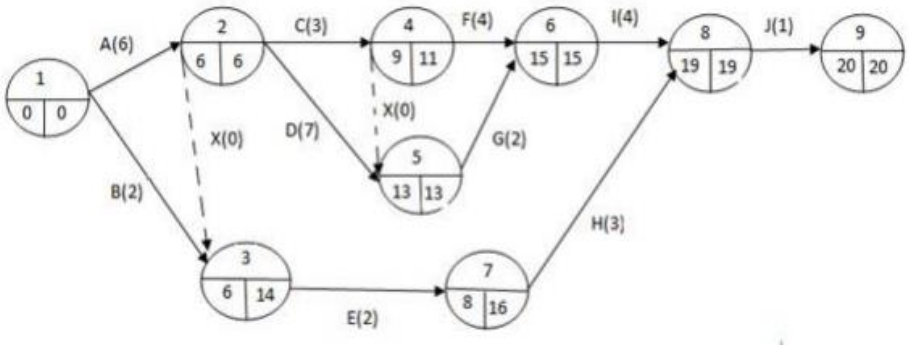
2. Calculer les marges libres et les marges totales.

3. Déterminer le chemin critique.

4. Si la durée de la tâche E devient 12 au lieu de 2
- quelle est la date au plus tôt pour commencer H ?
- quelle la durée du projet ?

# Exercice 5: Réponses

1.




2.

| tâche | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| ML | 0 | 4 | 0 | 0 | 0 | 2 | 0 | 8 | 0 | 0 |
| MT | 0 | 12 | 2 | 0 | 8 | 2 | 0 | 8 | 0 | 0 |

3. les tâches critiques (A –D-G-I-J) donc Le chemin critique : ( ADGIJ )

4-a- Normalement la date au plus tôt pour commencer H c'est 8 , mais si la durée de E devient 12 ( 12 au lieu de 2 c – à d un retard de 10 ) , et comme la marge libre de E est 0 ( c à d le retard n'est pas acceptable ) , donc on va retarder H aussi de 10 , donc la date au plus tôt pour commencer H c'est 10 + 8 = 18.

4- b - Normalement la durée du projet est 20 , mais si on retarde E de10 , et comme la marge totale de E est 8 ( c à d il y a un retard acceptable de 8 ) , donc on va retarder le projet de (10-8 = 2 ) par conséquent la durée du projet est 20+2=22.

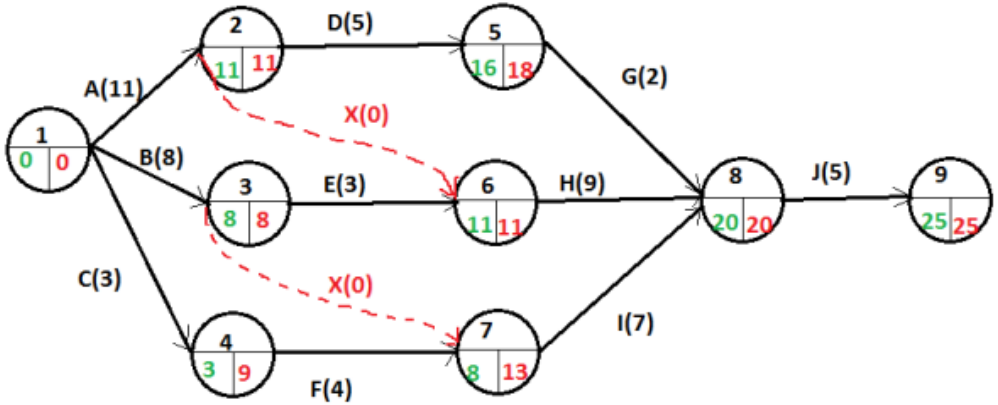
# Exercice 6

| Tâche | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| Tâche antérieur | --- | --- | --- | A | B | C | D | A, E | B, F | G, H,I |
| Durée | 11 | 8 | 3 | 5 | 3 | 4 | 2 | 9 | 7 | 5 |

1. Tracer le graphe PERT et calculer les dates au plus tôt et les dates au plus tard pour chaque sommet..
2. Calculer les marges libres et les marges totales.
3. Déterminer le chemin critique.

# Exercice 6: Réponses

1.




2.

| tâche | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| ML | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 5 | 0 |
| MT | 0 | 0 | 6 | 2 | 0 | 6 | 2 | 0 | 5 | 0 |

3. Les tâches critiques sont : A - B - E - H - J.

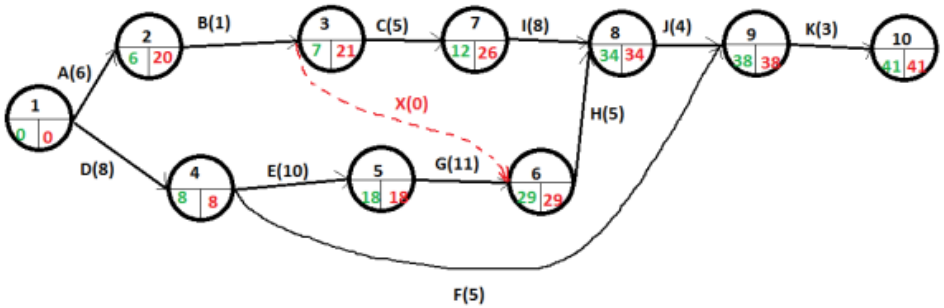
# Exercice 7

| Tâche | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Tâches précédentes | --- | A | B | --- | D | D | E | G,B | C | I,H | J,F |
| Durée | 6 | 1 | 5 | 8 | 10 | 5 | 11 | 5 | 8 | 4 | 3 |

1. Tracer le graphe PERT et calculer les dates au plus tôt et les dates au plus tard pour chaque sommet..
2. Calculer les marges libres et les marges totales.
3. Déterminer le chemin critique.
4. Supposons que la durée de E devient 20, quelle la date au plus tôt pour commencer G ?
5. - supposons que la durée de I devient 24, quelle la durée totale du projet ?

# Exercice 7: Réponses

1.




2.

| Tâches | Marge libre ( ML ) | Marge totale ( MT ) |
|---|---|---|
| A | 6 – 0 – 6 = 0 | 20 – 0 – 6 = 14 |
| B | 7 – 6 – 1 = 0 | 21 – 6 – 1 = 14 |
| C | 12 – 7 – 5 = 0 | 26 – 7 – 5 = 14 |
| D | 8 – 0 – 8 = 0 | 8 – 0 – 8 = 0 |
| E | 18 – 8 – 10 = 0 | 18 – 8 – 10 = 0 |
| F | 38 – 8 – 5 = 25 | 38 – 8 – 5 = 25 |
| G | 29 – 18 – 11 = 0 | 29 – 18 – 11 = 0 |
| H | 34 – 29 – 5 = 0 | 34 – 29 – 5 = 0 |
| I | 34 – 12 – 8 = 14 | 34 – 12 – 8 = 14 |
| J | 38 – 34 – 4 = 0 | 38 – 34 – 4 = 0 |
| K | 41 – 38 – 3 = 0 | 41 – 38 – 3 = 0 |

3. le chemin critique est composé des tâches critiques ( les tâches dont la marge totale est nulle ) : le chemin = ( D E G H J K )

4. la durée de E devient 20 au lieu de 10 , donc un retard de 10 dans la réalisation de E , et comme la marge libre de E est nulle , ce retard va impacter la ou les tâche (s) suivante (s) , donc G au lieu de commencer à 18 , elle va commencer à 10+18 = 28.

5. la durée de I devient 24 au lieu de 8 , donc un retard de 16 , et comme la marge totale de I est 14 ( c à d on a le droit de retarder I de 14 sans influencer la fin du projet ) , donc I va retarder le projet de 16-14=2 par conséquent la durée totale du projet devient : 41 + 2 = 43.

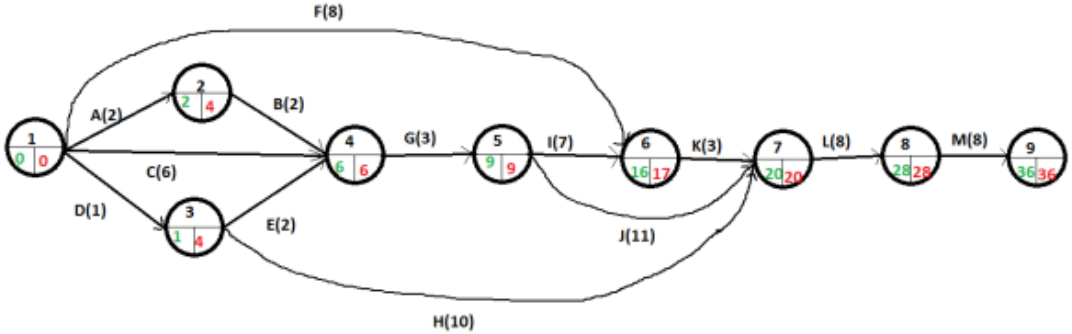
# Exercice 8

| tâche | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Antériorité | --- | A | --- | --- | D | --- | BCE | D | G | G | IF | KJH | L |
| Durée | 2 | 2 | 6 | 1 | 2 | 8 | 3 | 10 | 7 | 11 | 3 | 8 | 8 |

1. Tracer le graphe PERT et calculer les dates au plus tôt et les dates au plus tard pour chaque sommet..
2. Calculer les marges libres et les marges totales.
3. Déterminer le chemin critique.

# Exercice 8: Réponses

1.




2.

| Tâche | Marge libre (ML) | Marge totale ( MT) |
|---|---|---|
| A | 2-0-2 = 0 | 4-0-2 = 2 |
| B | 6-2-2 = 2 | 6-2-2 = 2 |
| C | 6-0-6 = 0 | 6-0-6 = 0 |
| D | 1-0-1 = 0 | 4-0-1 = 3 |
| E | 6-1-2 = 3 | 6-1-2 = 3 |
| F | 16 -0 -8 =8 | 17-0-8 = 9 |
| G | 9-6-3 = 0 | 9-6-3 = 0 |
| H | 20-1-10 = 9 | 20-1-10 = 9 |
| I | 16-9-7 = 0 | 17-9-7 = 1 |
| J | 20-9-11 = 0 | 20-9-11 = 0 |
| K | 20-16-3 = 1 | 20-16-3 = 1 |
| L | 28-20-8 = 0 | 28-20-8 = 0 |
| M | 36-28-8 = 0 | 36-28-8 = 0 |

3. Les tâches critiques sont les tâches qui ont une marge totale nulle.
donc c'est : C - G - J - L - M
le chemin critique ( CGJLM)