Analyse et conception de Système

- 026047 - Rôle de l'analyste et cycles chronologiques de développement de systèmes
- 026049 - Langage de modélisation unifié
- 026051 - Techniques utilisant les méthodologies Agiles

PLAN DU COURS

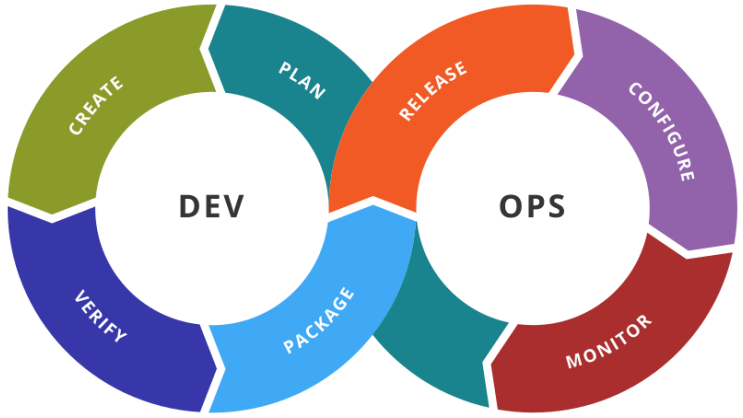
Partie 1 : Cycle de développement d’un logiciel

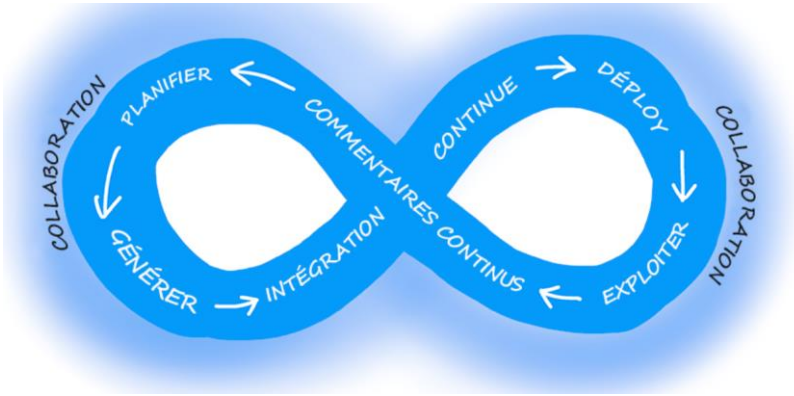
Partie 2 : Langage de modélisation unifié

Partie 3 : Approches Agiles

Partie 4 : Étude de cas – Évaluation sommative

Méthodologie de gestion de projet
DevOps





Méthodologie de gestion de projet
DevOps

Micro-services

Diviser une application monolithique en composants autonomes. Chaque microservice est une unité indépendante, gérant une fonctionnalité spécifique de l'application.

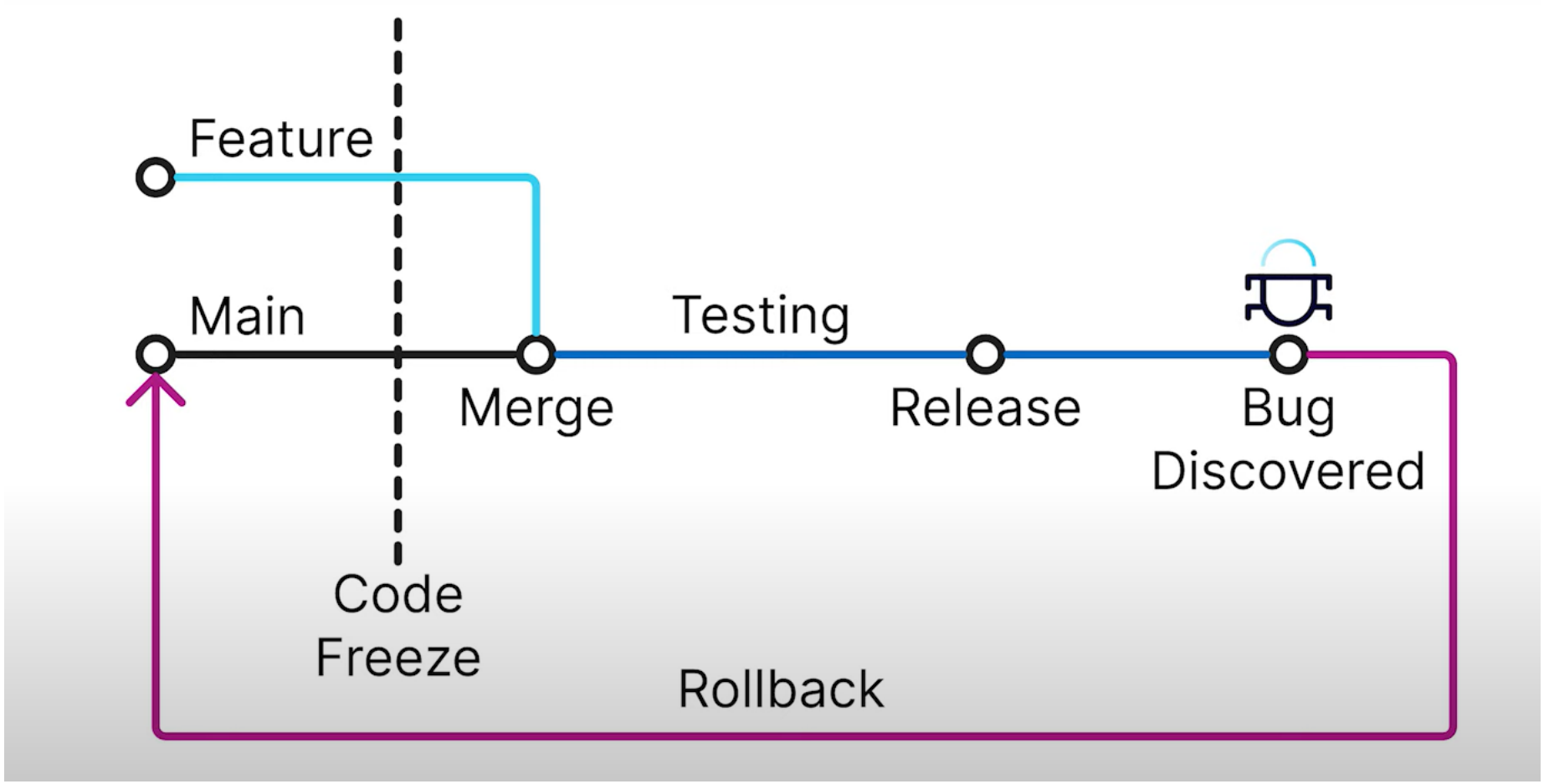
Intégration continue (CI)

Les développeurs fusionnent régulièrement leur code dans un référentiel partagé. À chaque fusion, des tests automatisés sont déclenchés pour identifier rapidement les erreurs. L'objectif principal de CI est d'assurer une détection précoce des problèmes d'intégration, favorisant ainsi un développement plus fluide, fiable et rapide.

Déploiement continue (CD)

Automatiser le processus de mise en production des applications. Après chaque modification de code réussie et intégrée, le déploiement continu permet de libérer automatiquement les nouvelles fonctionnalités ou corrections, assurant une livraison rapide et fiable des mises à jour logicielles.

Méthodologie de gestion de projet
Long cycle de déploiement



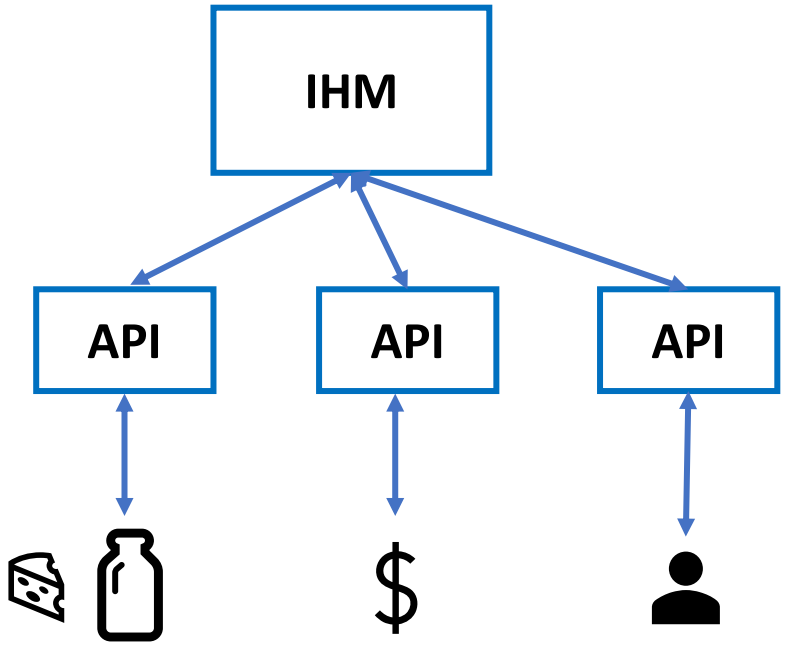
Méthodologie de gestion de projet
Microservices

Les microservices sont une approche de conception logicielle qui divise une application en petits services autonomes et indépendants.

Chaque microservice représente une fonctionnalité spécifique de l'application et communique avec les autres microservices via des API bien définies.

Exemples

Robot

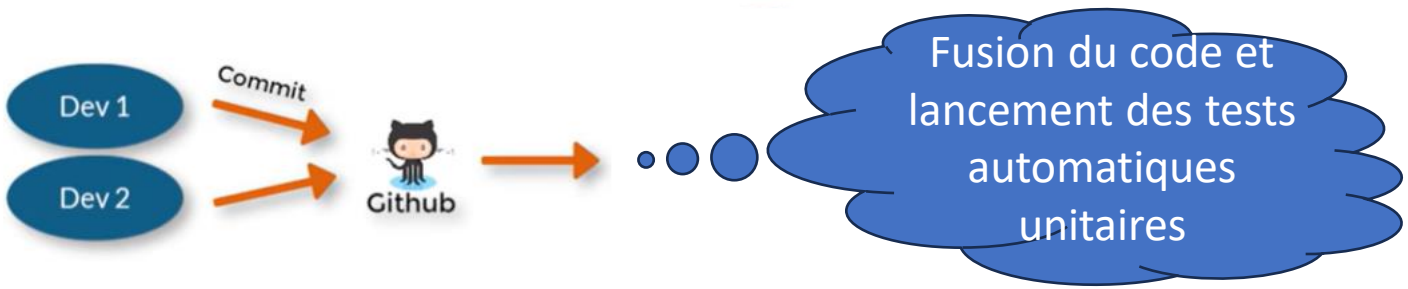


eCommerce

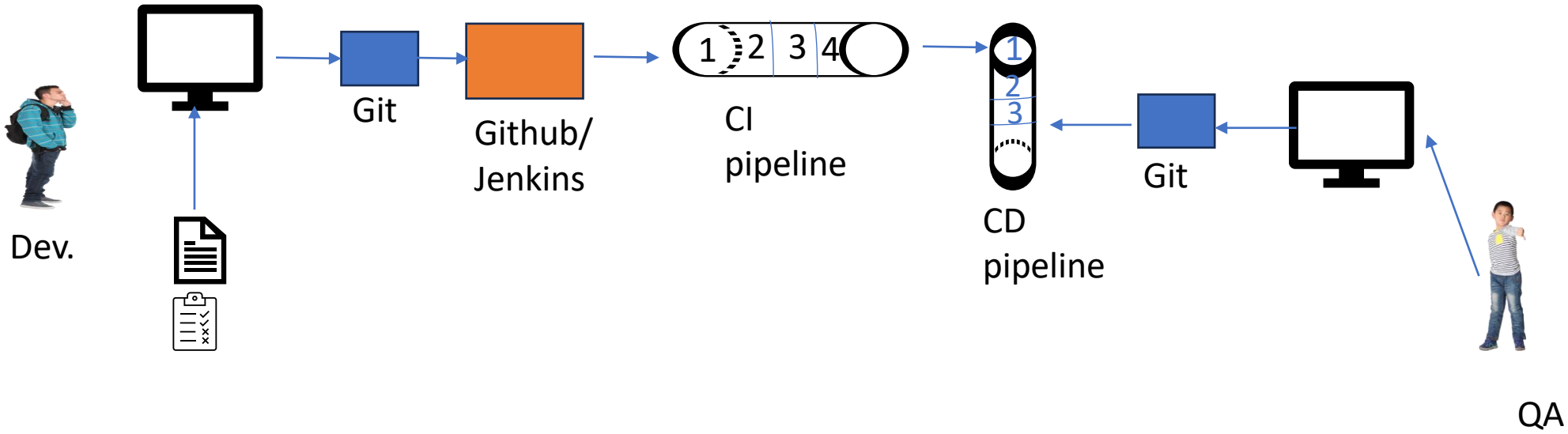
Méthodologie de gestion de projet
CI / CD

Intégration continue (CI)

Les développeurs fusionnent régulièrement leur code dans un référentiel/répertoire partagé. À chaque fusion, des tests automatisés sont déclenchés pour identifier rapidement les erreurs. L'objectif principal de CI est d'assurer une détection précoce des problèmes d'intégration, favorisant ainsi un développement plus fluide, fiable et rapide.



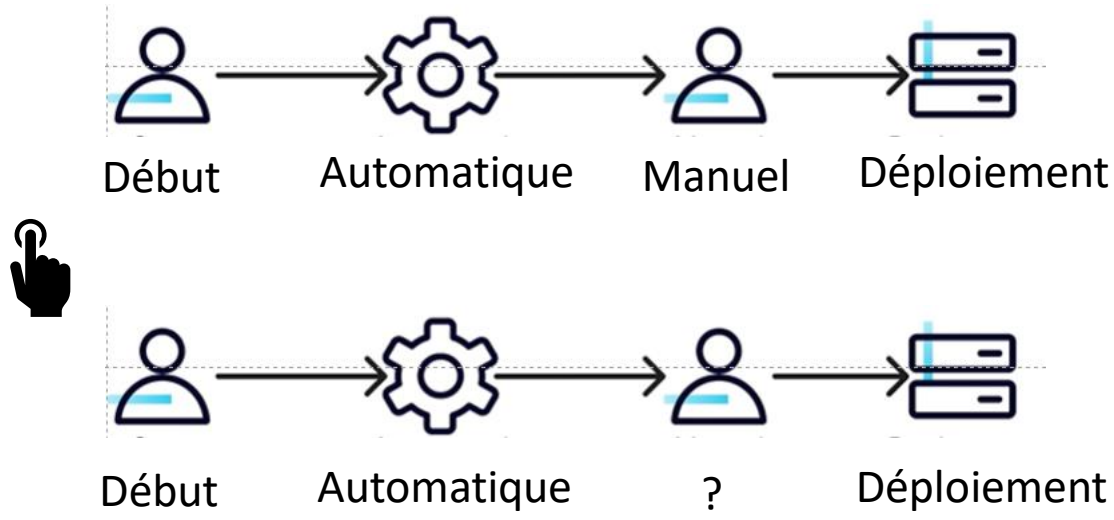
Méthodologie de gestion de projet
CI / CD

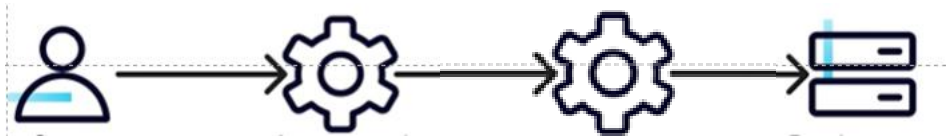


Méthodologie de gestion de projet
CI / CD

Livraison /ou Déploiement continu (CD)

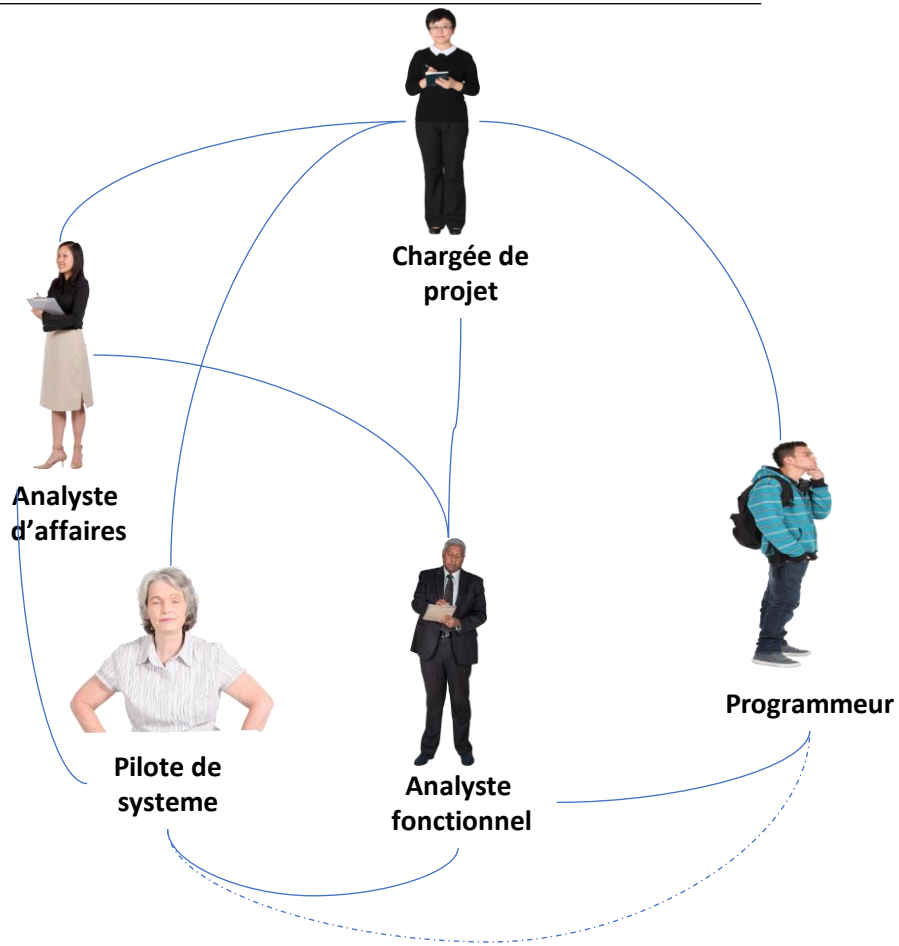
Automatiser le processus de mise en production des applications. Après chaque modification de code réussie et intégrée, le déploiement continu permet de libérer automatiquement les nouvelles fonctionnalités ou corrections, assurant une livraison rapide et fiable des mises à jour logicielles.





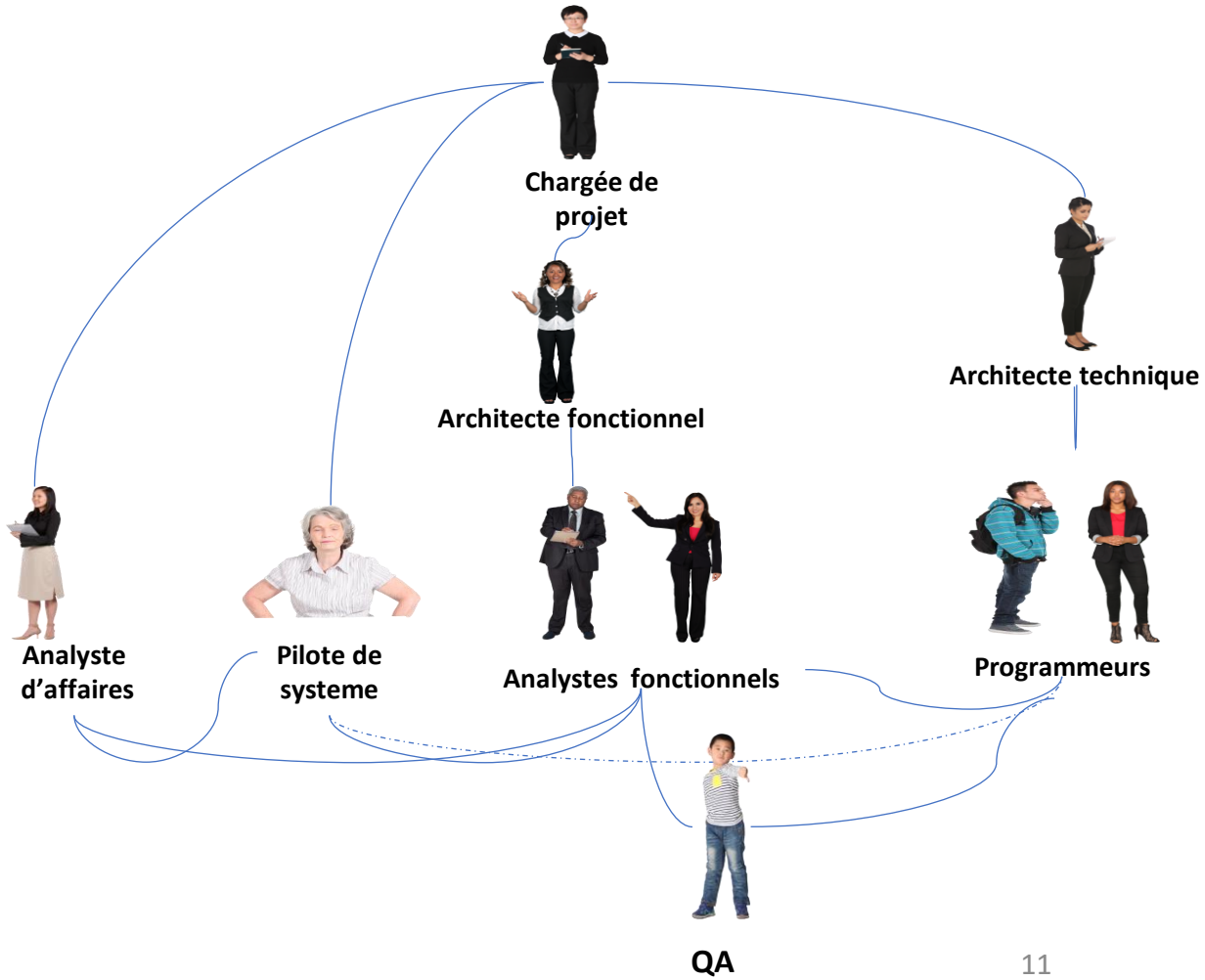
Cycle de vie d'un projet
Phase d'initiation – Acteurs du projet

- **Analyste fonctionnel:** Recueille les besoins d'affaires, puis analyse et conçoit le système. Il fournit les dossiers fonctionnels qui contiennent toutes les spécifications sur le système attendu.
- **Programmeur**. Traduit les spécifications en langage machine, écrit le code du système, fait les compilations et installations.
- **Analyste d'affaires** est spécialement dédié pour consigner le besoin des utilisateurs, et la vision de la direction de l'entreprise. Il priorise les besoins et évalue les impacts sur l'entreprise.
- **Pilote de système,** appelé aussi super utilisateur ou utilisateur clef. C'est lui qui fait les configurations avancées dans le système, c'est aussi lui la personne de référence pour les utilisateurs lorsqu'ils ont des problèmes ou questions avec le système. Cette personne est aussi responsable de faire des tests utilisateurs et d'acceptation avant les mises en production du système.
- **Chargé de projet** a le rôle d'encadrer l'équipe de développement afin de s'assurer de respecter les échéanciers, la portée et les budgets. Il facilite la collaboration entre les membres de l' équipe et contribue à trouver des solutions lorsqu'il y a des valeurs qui entrent en conflit. Il s'assure constamment que le développement avance bien.

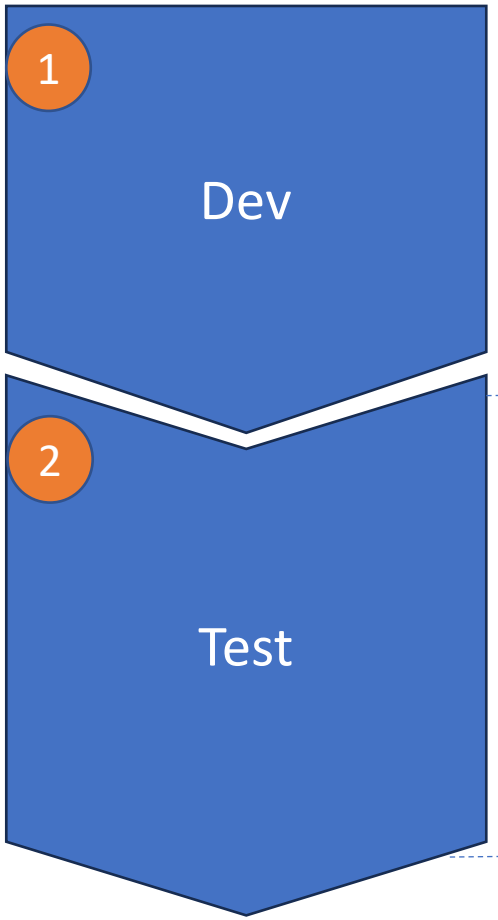


Cycle de vie d'un projet
Phase d'initiation – Acteurs du projet

- **Architectes** : ils travaillent sur les besoins et les solutions à haut niveau. Ils s'assurent de la cohérence de tout ce qui est fait. Ils aident pour l'évaluation des temps. Ils recensent et pèsent les impacts avant qu'un ajout ou une modification d'une fonctionnalité soit fait au système selon leur secteur (affaires, fonctionnel ou technique).
  Ils prennent ensemble les grandes décisions quant aux orientations à appliquer. Ils répondent aux questions des membres de leur équipe, encadrent les nouveaux venus, diffusent les informations des décisions prises dans les comités et sollicitent de l'aide aux gens de leur équipe pour faire des prototypes et des simulations.



Cycle de vie d'un projet
Phases de développement d'un système – environnements utilisés



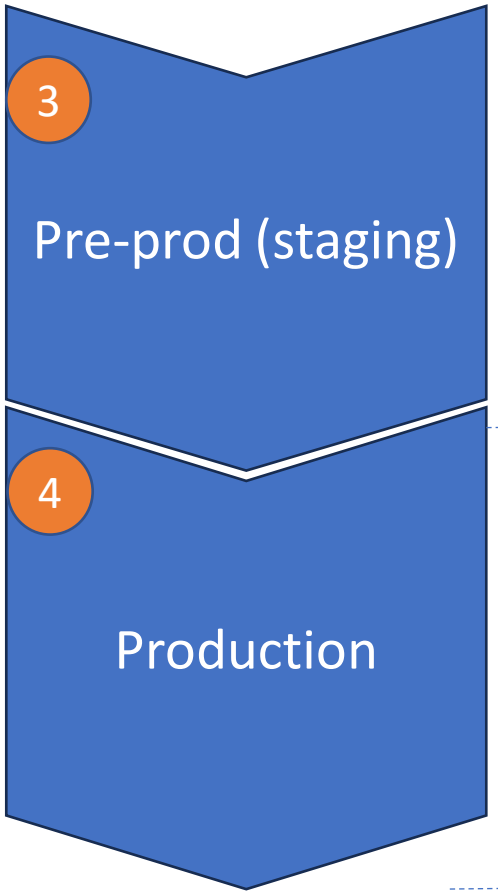
C'est l'environnement où les développeurs écrivent, modifient et testent le code source.

- Environnement isolé de la production pour éviter d'impact sur les utilisateurs finaux.
- Généralement, une copie locale de la base de données est utilisée pour faciliter le développement et le débogage

Tester le logiciel dans un environnement similaire à la production avant le déploiement.

- Utilisation de jeux de données réalistes.
- Les tests peuvent inclure des tests unitaires, des tests d'intégration, des tests de système, et des tests de performance.

Cycle de vie d'un projet
Phases de développement d'un système – environnements utilisés



Simuler ou « répliquer » l'environnement de production pour valider le déploiement et détecter d'éventuels problèmes avant la mise en production.

- Environnement très proche de la production en termes de configuration matérielle et logicielle.
- Tests de charge et de performance intensifs sont souvent réalisés.

Environnement de déploiement final pour les utilisateurs finaux.

- Configuration optimisée pour les performances et la disponibilité.
- Surveillances en temps réel pour détecter et résoudre rapidement les problèmes.

Outils de développement d'un système
SCM : Git & GitHub

Git

- Outil de gestion de versions décentralisé.
- Permet de suivre les modifications du code source au fil du temps.
- Utilisé pour coordonner le travail d'équipes sur des projets.

- **Dépôt** (Repository) : Dossier qui contient tous les fichiers nécessaires à votre projet. Un dépôt peut être local (sur votre machine) ou distant (sur un serveur).
- **Commit** : Image du projet à un moment donné. Enregistre les modifications avec un message descriptif.
- **Branches :** Chemins de développement isolés. Elles permettent de travailler sur des fonctionnalités distinctes sans affecter la branche principale (généralement appelée "master" ou "main").
- **Clone** : Copie d'un dépôt distant sur votre machine locale.

Github

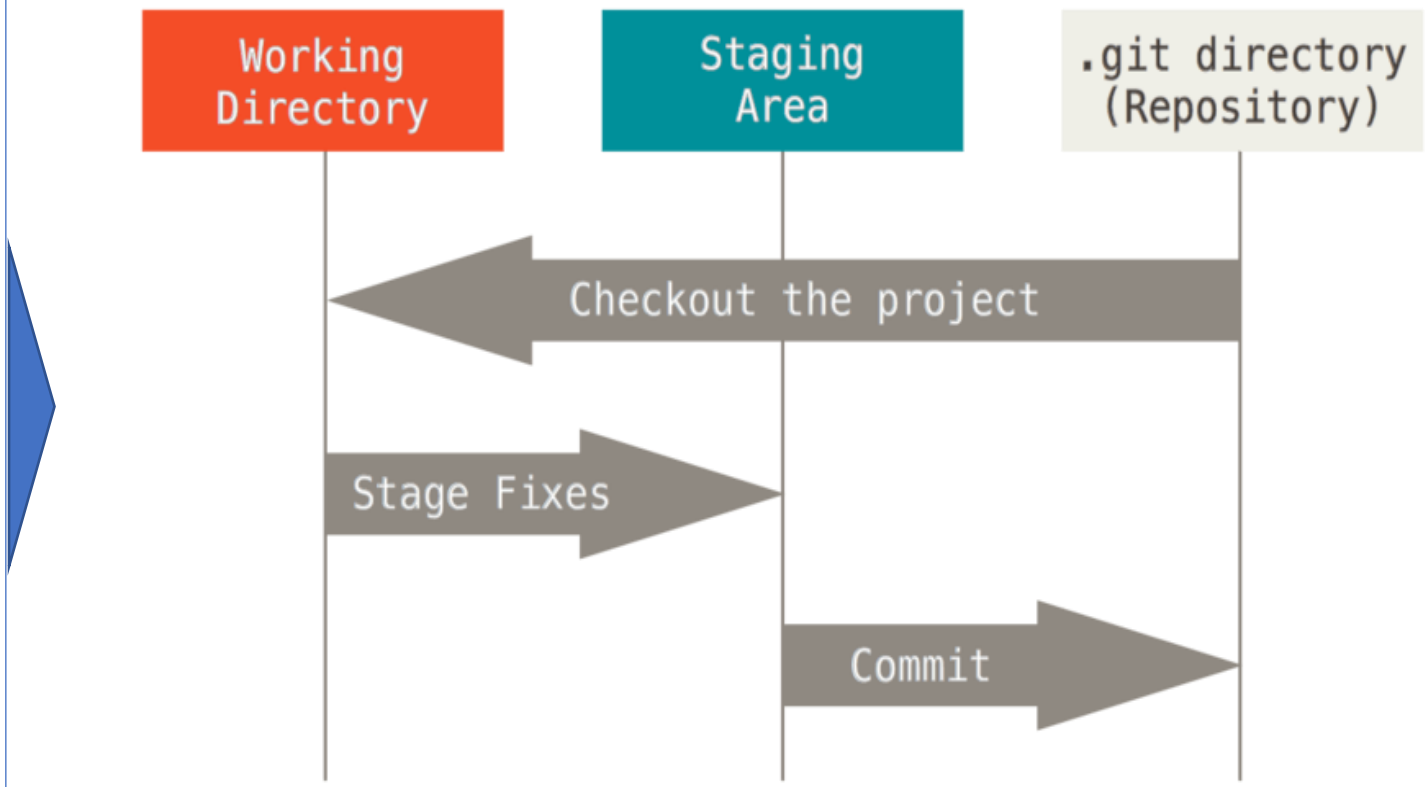
- Plateforme web basée sur Git.
- Facilite la collaboration et le partage de code entre développeurs.
- Héberge des dépôts Git distants

- **Pull (Demande de tirage) :** Mécanisme pour proposer des changements à un dépôt. Elle permet la révision et la discussion avant de fusionner les modifications.
- **Forks :** Création d'une copie personnelle d'un dépôt. Elle Utile pour contribuer à des projets open source.
- **Issues :** Suivi des problèmes, idées, et améliorations. Elle facilite la communication au sein de l'équipe.
- **Actions :** Automatisation des flux de travail (tests, déploiement, etc.).

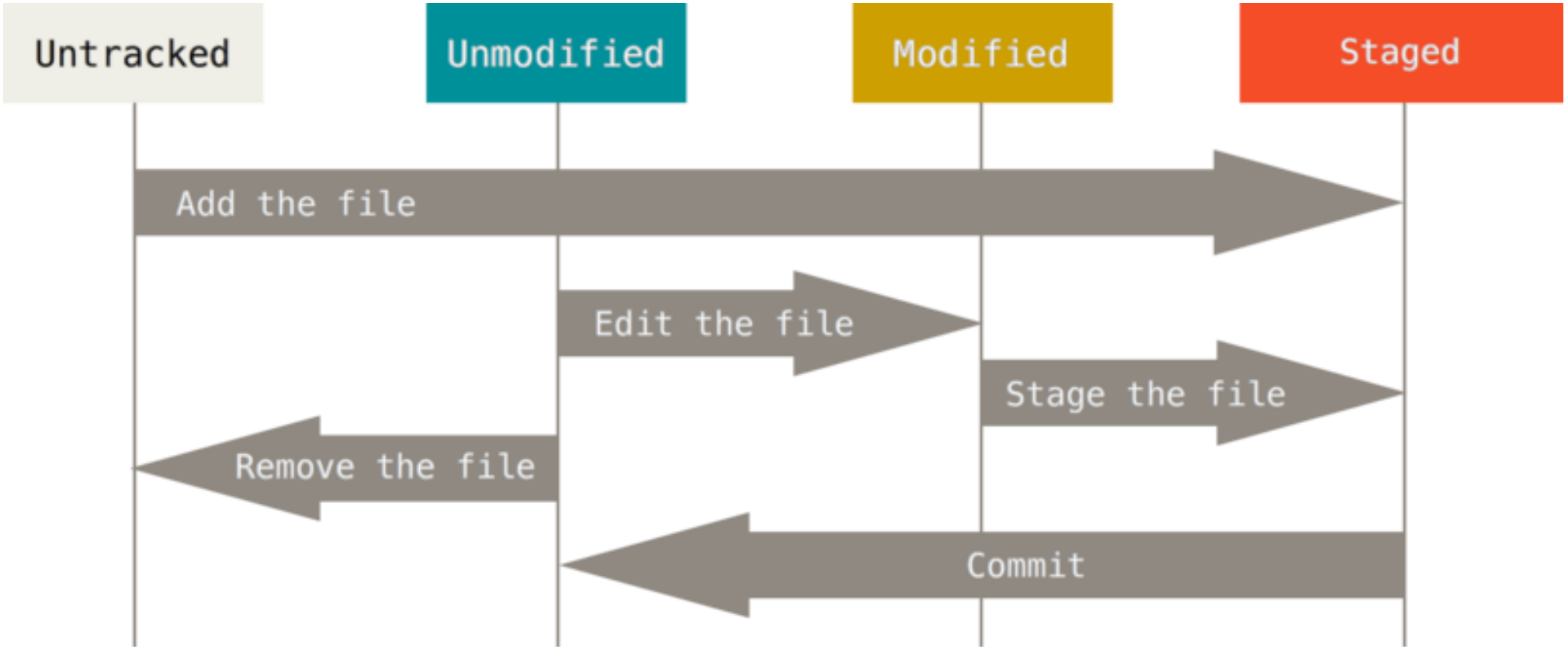
Outils de développement d'un système
SCM : Git

Les trois états
Git gère trois états dans lesquels les fichiers peuvent résider : **modifié**, **indexé** et **validé**.

- **Modifié** signifie que vous avez modifié le fichier mais qu'il n'a pas encore été validé en base.
- **Indexé** signifie que vous avez marqué un fichier modifié dans sa version actuelle pour qu'il fasse partie du prochain instantané du projet.
- **Validé** signifie que les données sont stockées en sécurité dans votre base de données locale.



Outils de développement d'un système
SCM : Git



Outils de développement d'un système
Git : Premiers pas

1. Vérification de la version ou installation
2. Configuration (git config global user.name / user.email
    1. verification de la config : `git config --list`
3. creation d'un repertoire local
    1. git init ⇒ initialiser un repertoire existant comme repertoire Git
        1. se positionner sur le repertoire local
        2. initialisation (git init)
    2. git clone [url] ⇒ recuperer un repertoire existant via un URL (ex. a partir de Github)
4. Création du fichier (.gitignore)
5. Ajout des fichier au staging (git add –all)
6. Verifier le statut
7. Faire le premier commit

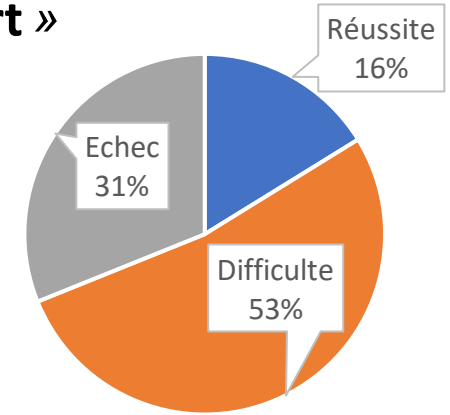
Outils de développement d'un système
Git : Premiers pas – Exercice d'application

Gestion de projet et versions

1. **Création du Projet :** Créez un nouveau répertoire pour votre projet.
2. **Initialisation de Git :** Initialisez un nouveau dépôt Git dans le répertoire.
3. **Ignorer des Fichiers :** Créez un fichier .gitignore et ajoutez des règles pour ignorer certains fichiers.
4. **Modification et Commit :**
   - Ajoutez un fichier au répertoire.
   - Modifiez ce fichier et effectuez un commit.
5. **Création de Branche :** Créez une nouvelle branche appelée "NouvelleBranche".
6. **Fusion des Branches :**
   - Effectuez des modifications dans la nouvelle branche.
   - Fusionnez la nouvelle branche dans la branche principale.
7. **Gestion des Conflits :**
   - Créez un conflit en modifiant le même fichier dans deux branches.
   - Résolvez le conflit.
8. **Historique des Commits :** Utilisez la commande **git log** pour afficher l'historique des commits.
9. **Suppression de Branche :** Supprimez la branche que vous avez créée.
10. **Revenir à une Version Antérieure :** Revenez à une version antérieure en utilisant la commande **git reset**.
11. **Se connecter à votre compte Github** et synchronier votre repertoire avec Github

Cycle de vie d'un projet
Facteurs clés de succès d'un projet informatique

Résultat d' étude de Standish Group « Standish Group Chaos Report »



- Projet réussi : le projet a été achevé dans les temps et dans l'enveloppe budgétaire initialement prévue, et le logiciel comporte les fonctionnalités demandées.
- Projet en difficulté : le projet a été achevé et est opérationnel. Néanmoins, il a dépassé les coûts et les délais initialement prévus, et propose moins de fonctionnalités que ce qui avait été demandé initialement.
- Echec du projet : le projet a été arrêté pendant son cycle de développement.

Principaux facteurs d' échec

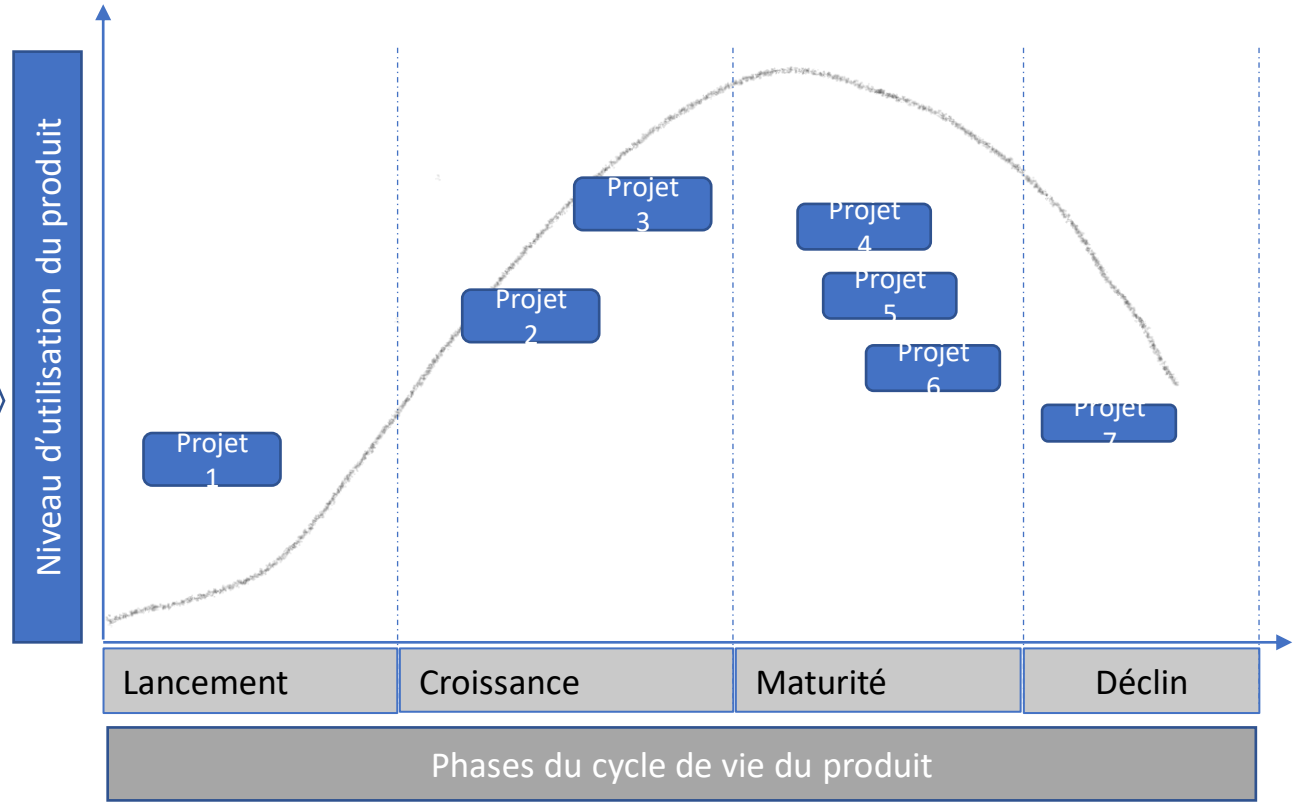
- Spécifications du système incomplètes où changeantes
- objectifs vagues
- faible implication des utilisateurs
- manque de soutien des dirigeants de l'entreprise
- absence de soutien technique
- mauvaise planification du projet
- manque de ressources essentielles

Cycle de vie d'un produit

Un produit est un objet quantifiable, pouvant aussi bien être un produit final qu'un composant.
Cycle de vie du produit est une série de phases qui représentent l'évolution d'un produit, du lancement jusqu'à son retrait du marché.

Il y a la possibilité de lancer des projets à tout moment du cycle de vie du produit dans le but de créer de la valeur et d'améliorer les composants, des fonctions et/ou des capacités données.

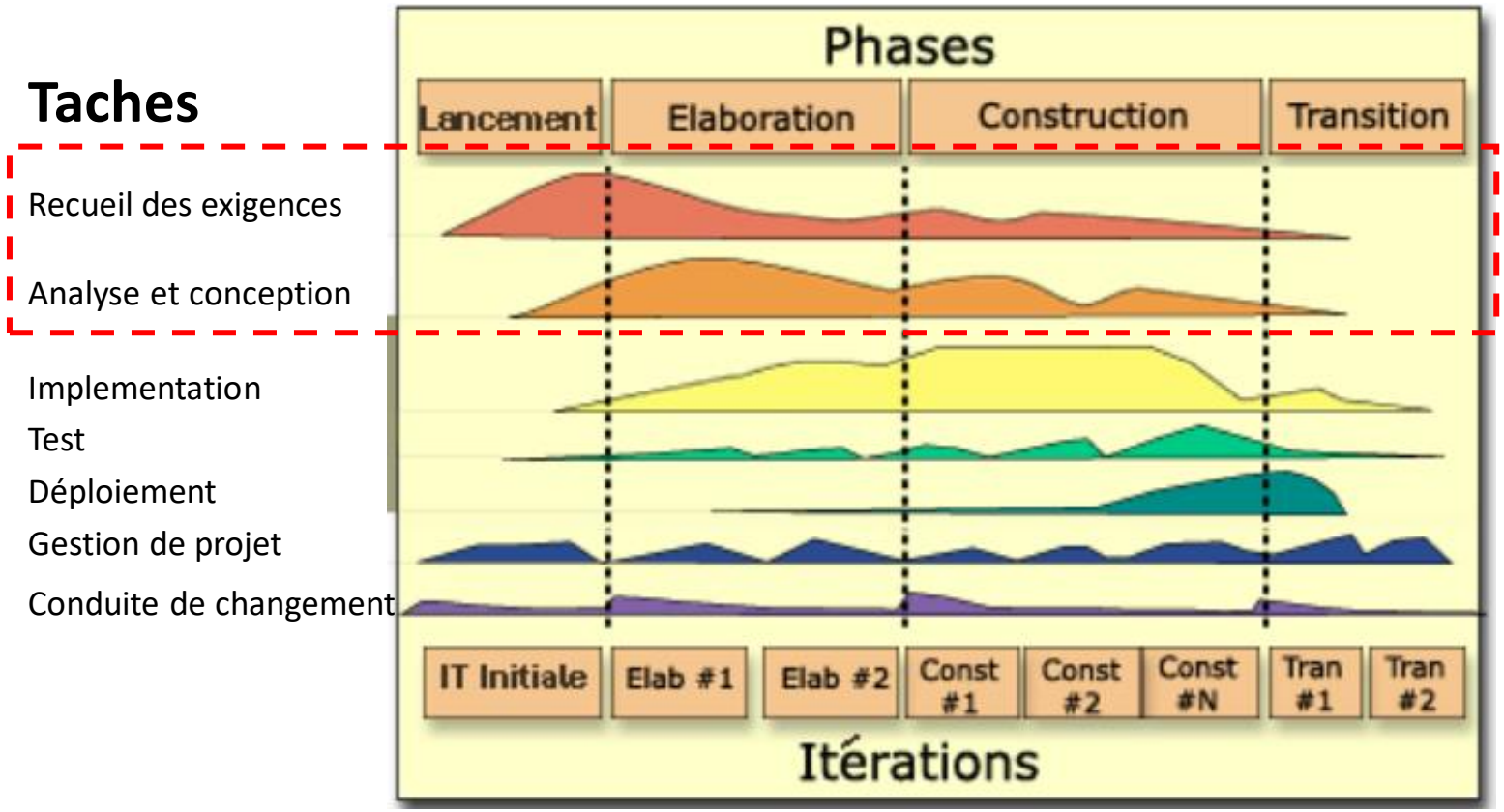
Le produit initial peut démarrer sous la forme d'un livrable d'un projet. Au cours de son cycle de vie, un nouveau projet peux ajouter aux améliorer les composants, des attributs ou des capacités qui crée une valeur supplémentaire pour les clients et l'organisation.



Cycle de développement logiciel
Modèle Rational Unified Process(RUP) (1/3)

Un logiciel est un produit (manufacturé)

- Il est comme une voiture ou une machine à outils
- Il requiert un processus de développement pour :
  - Déterminer les besoins
  - Définir le plan ou/et l'architecture du système
  - Concevoir le système
  - Réaliser le système
  - Etc.



Cycle de développement logiciel
Modèle RUP (Rational Unified Process) (2/3)

Principes du Processus Unifié (UP)

- **Orientation cas d'utilisation :** le développement est organisé autour d'une liste des cas d'utilisation
- **Centré autour de l'architecture**: le processus de développement mène à la construction d'une architecture du système qui permettra l'implémentation des exigences. Cette architecture est basée sur l'identification d'une structure qui sera construite en plusieurs itérations
- **Itératif et incrémental** : le développement est subdivisé en itération ou cycles de développement. Chaque itération ou nouvelle fonction est ajoutée à l'architecture du système, ou corrige et affine le système, permet de s'approcher de la version finale désirée.
- **Orientation risque** : les éléments qui représentent un risque élevé sont traitées très tôt dans le projet. Par exemple, un cas d'utilisation critique est identifié, détaillé est implémenté avant les autres.

Cycle de développement logiciel
Modèle RUP – Recueil des besoins (1/5)

Activités

- Abstraction et séparation des problèmes
- Modularisation : séparation des besoins fonctionnels
- Ateliers de travail collaboratif
- Modélisation

Objectifs

Découverte et compréhension de la situation actuelle.
À partir du cahier des charges, description du problème à traiter ou de l'opportunité à saisir

- identification des besoins de l'utilisateur
- spécification du **« pourquoi"** développer ce logiciel.

Sorties

- Déclaration de l'envirgure et de la portée
- Cas d'utilisation

Cycle de développement logiciel
Modèle RUP – Recueil des besoins (2/5)

L'**analyste** devrait agir plus en tant que **psychanalyste** ou **coach** que consultant ou conseiller

- Éviter de se prononcer à la place du client/utilisateur
- Aider le client à s'exprimer et parler de son problème et de ses besoins
- Pratiquer l'écoute active et poser des questions
- Se focaliser plus sur les problèmes des affaires
- **Éviter de se lancer dans des solutions**

Questions posées par l'analyste à ce stade

- C'est quoi le problème que l'organisation cherche à résoudre?
- Pourquoi l'organisation souhaite investir dans ce système? Quels sont la vision et les objectifs de l'organisation?
- Acheter ou developer?

Cycle de développement logiciel
Modèle RUP – Recueil des besoins (3/5)

L'observation des utilisateurs en action ou en situation Cette technique est souvent utilisée par les ergonomes pour comprendre le comportement des utilisateurs sur leur poste de travail.
Exemple, nbre de cliques faits pour atteindre un menu, les astuces que note l'utilisateur pour palier au manquement de son application (post-it, prise de notes, etc….)

L'analyse de l'existant : il s'agit d'examiner le système en place et de voir ses forces et ses faiblesses

Le questionnaire : On utilise un questionnaire lorsque nous voulons avoir l'avis d'un grand nombre d'utilisateurs sur un certains nombres de besoins particuliers. Combinant des questions fermées, ciblées et questions ouvertes le questionnaire ne doit pas induire (guider) les réponses.
Les questions du questionnaire doivent être claires et précises. Le questionnaire peut être anonyme ou non.
Exemple pour améliorer l'utilisation de eCite par les étudiants, on pourrait envoyer un questionnaire à tous les étudiant en ciblant des problèmes bien précis. Il serait inutile de rencontrer les étudiants un à la fois.

Cycle de développement logiciel
Modèle RUP – Recueil des besoins (4/5)

Le Brainstorming : Technique idéale pour « défricher » les besoins encore flous ou mal organisés par les utilisateurs lors du démarrage du projet. Le brainstorming peut se faire sous forme de petites rencontres (1 à 2) durant lesquelles les utilisateurs peuvent exprimer ce qui leur parait important dans le projet. Personne ne se censure. Un facilitateur pourra guider le groupe à hiérarchiser les besoins.

L'interview : C'est la technique la plus directe pour approfondir un besoin. Un utilisateur à la fois.
C'est une technique simple mais qui doit être préparée d'avance. L'interview doit se planifier. Fixer une date, une heure début et une durée.
Les questions de l'interview doivent être minutieusement préparées. C'est à vous de guider l'interview. Lors de l'interview, un utilisateur qui s'ennuie peut vous mener en dehors de votre objectif (il va vous raconter sa vie). C'est à vous de veiller à ce que l'objectif de l'interview soit atteint. À la fin de l'interview, vous devez peut-être valider avec le supérieur hiérarchique de l'utilisateur.

Cycle de développement logiciel
Modèle RUP – Recueil des besoins (5/5)

Besoin Fonctionnel (BF)

Décrire ce qu'un système doit faire et le comportement du système en ce qui concerne les fonctionnalités du système.

Besoin non-Fonctionnel (BNF)

Elabore les caractéristique de performance du système, les BNF sont généralement des contraintes sur la manière dont le système fonctionnera.
Ils représentent des indicateurs de qualité de l'exécution des besoins fonctionnels.

Cas utilisation – use case
Histoire d'utisation – Story

Liste ou grille des BNF

Cycle de développement logiciel
Modèle RUP – Recueil des besoins

EXERCICE

- Prendre un projet fictif (développement d'un site web, conception et développement d'une application mobile iOS/Android, construction d'une maison, ou autres)

1. Identifier 2-3 spécifications fonctionnelles
2. Identifier 4-5 spécifications non-fonctionnelles

Cycle de développement logiciel
Recueil des besoins – Cas d'utilisation

Un cas d'utilisation permet de décrire une séquence d'événements qui, pris tous ensemble, définissent ce qui est attendu d'un système. Chaque cas d'utilisation contient un ou plusieurs scénarios qui définissent comment le système devrait interagir avec les utilisateurs pour atteindre un but ou une fonction spécifique.

- Ils centrent l'expression des exigences du système sur ses utilisateurs
- Ils se limitent aux préoccupations "réelles" des utilisateurs ;
- ils ne présentent pas de solutions d'implémentation et ne forment pas un inventaire fonctionnel du système.
- Ils identifient les utilisateurs du système et leur interaction avec celui-ci

Diagrammes de cas d'utilisation est une représentation graphique:

- hiérarchique et structuré.
- exprime les diverses relations entre les cas d'utilisation et les acteurs

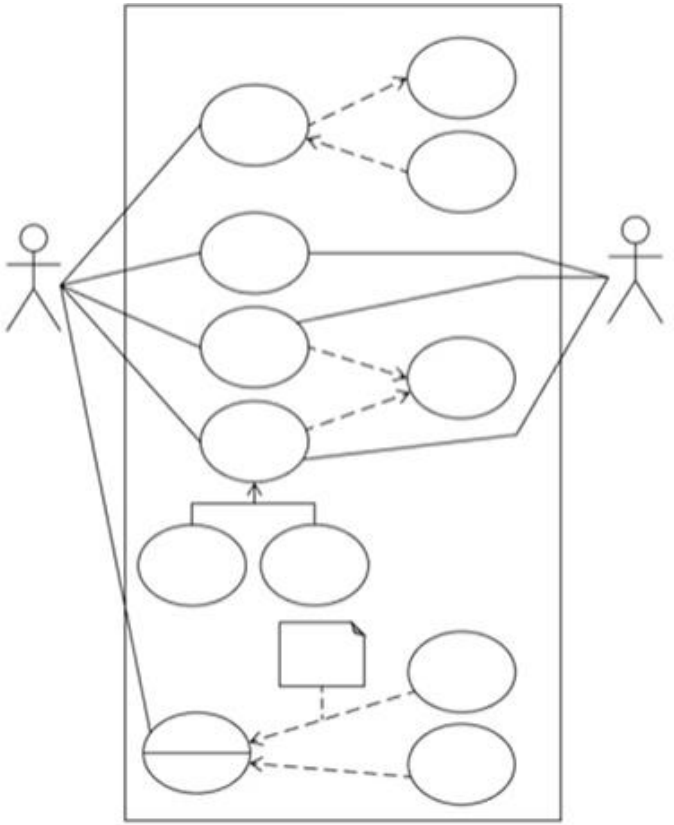
Histoires d'utilisateurs est une représentation littéraire.

- une description du besoin/ ou de la fonctionnalité décrite en mots compréhensibles.

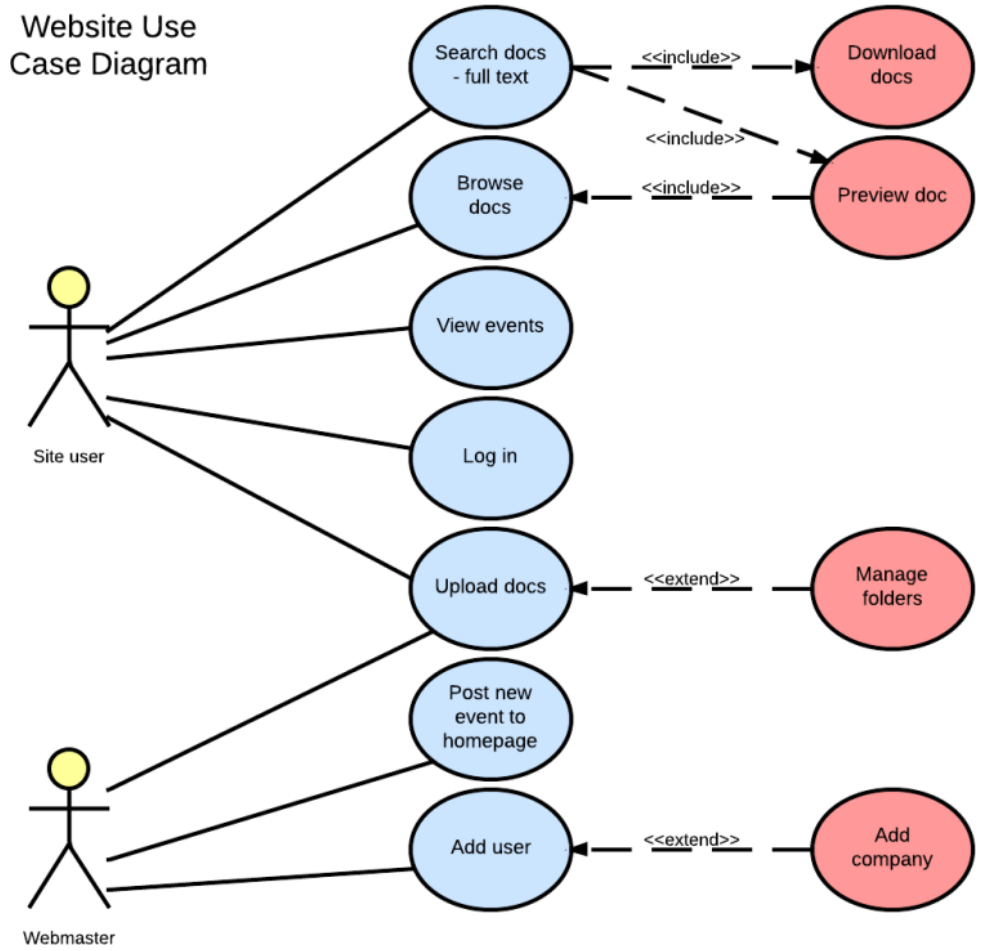
Cycle de développement logiciel
Diagramme des cas d'utilisation- Généralités

Un diagramme de cas d'utilisation est constitué de quatre différents composants:

1. **Systèmes:** est-ce que vous développez (un site internet, module d'un logiciel, un processus d'affaires, une application, ou autre).
2. **Acteurs:** est quelqu'un ou quelque chose qui utilise le système afin d'accomplir un objectif (une personne, une organisation, un autre système, ou un dispositif externe)
3. **Cas d'utilisation:** c'est l'action qui accomplit une tâche dans notre système, Ils seront placés dans notre rectangle parce qu'ils sont des actions qui se passent dans le système
4. **Relation**s: le lien/dépendance entre les acteurs et les cas d'utilisation (association, inclusion, extension et generalisation)



Cycle de développement logiciel
Diagramme des cas d'utilisation- Exemple



Cycle de développement logiciel
Diagramme des cas d'utilisation- Relations

On distingue quatre types de relation entre deux cas d'utilisation.

1. Association Indiquent que des instances d'un élément de modèle sont connectées aux instances d'un autre élément de modèle

<<include>>

2. Une inclusion <<include>> montre les dépendances entre un cas d'utilisation de base et un cas d'utilisation inclus. A chaque fois qu'un cas d'utilisation est exécuté, le cas d'utilisation inclus est exécuté en même temps.

<<extend>>

3. Une extension <<extend>> montre une dépendance potentiel (pas obligatoire) entre deux cas d'utilisation indépendants. Ex. un cas d'utilisation A (consulter la liste des abonnés) peut étendre son action sur un autre cas d'utilisation B (imprimer la liste des abonnés). Les deux cas peuvent s'exécuter de manière indépendante.
4. Une généralisation : les cas d'utilisation descendants héritent des propriétés de leur parent. Un cas A est une généralisation d'un cas B si B est un cas particulier de A.

Cycle de développement logiciel
Diagramme des cas d'utilisation- Relations

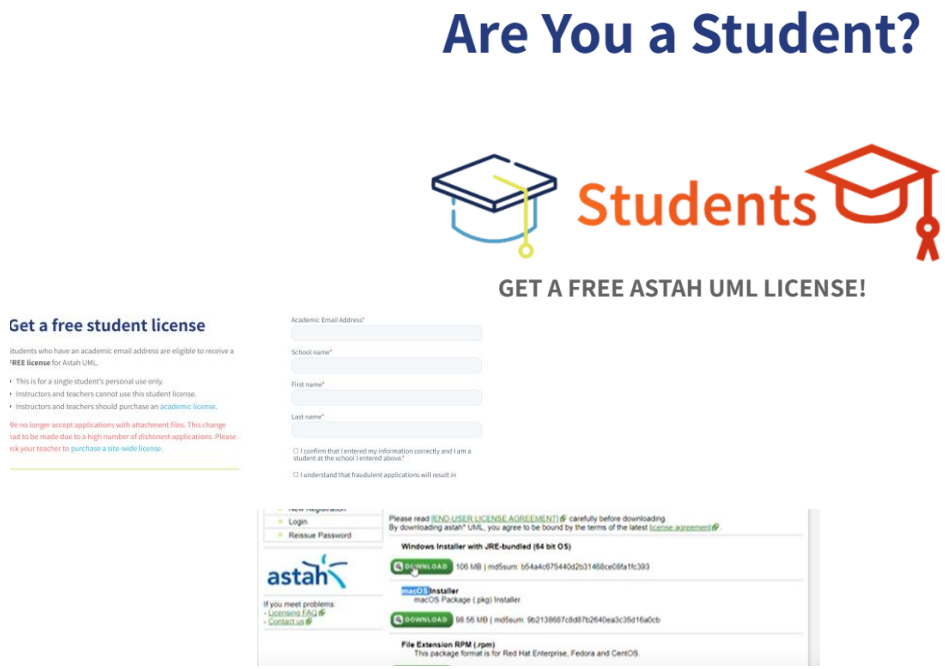
Intérêt des cas d'utilisation dans le cycle de développement d'un logiciel.

1. Modéliser les exigences d'affaires
2. Définir l'architecture du système
3. Extraire les cas de test
4. Élaborer le planning des itérations
5. Servir comme base de documentation du projet TI

Cycle de développement logiciel
Diagramme des cas d'utilisation- Modélisation

Comment télécharger et installer Astah UML 2021 gratuitement

1. Se connecter sur le site pour télécharger l'application Download Astah Software – Astah
2. Appliquer pour une licence étudiant
3. Télécharger la version correspondante (MS, Mac)
4. Installer le fichier executable
5. Ajouter la licence
    1. Télécharger la licence suivant le lien reçu par courriel
    2. Décompresser le fichier



Cycle de développement logiciel
Diagramme des cas d'utilisation- Modélisation

Thank you for your free student license request!

Here's your license file. Please download from the link here and unzip it.

- Download your student license file

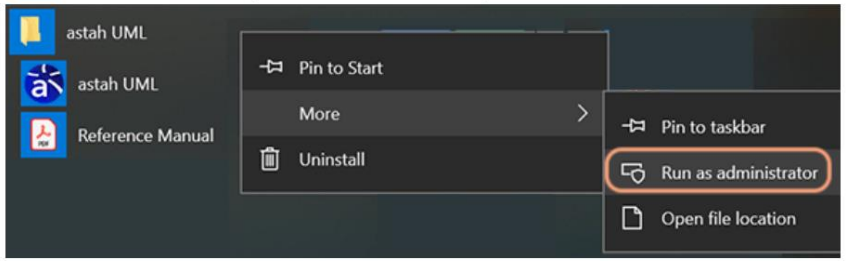
Please refer to the installation guide to get you started!

- How to set your license file

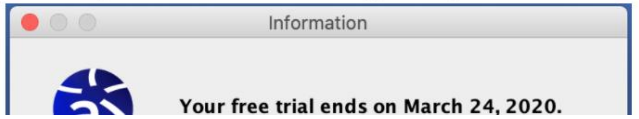
Happy studies! :)

How to set license

1. Download Astah UML.
2. Launch Astah UML. If you are on Windows, run it as Administrator.



3. If this is your first time launching Astah, it'll run as a trial and you will get a pop-up like this. Click [OK].



Cycle de développement logiciel
Diagramme des cas d'utilisation- Modélisation

Développer une application bancaire (app bancaire) qui va permettre à un client de se connecter, de vérifier son solde bancaire, transférer des fonds entre comptes, et payer des factures.

Questions:

1. Quels sont les acteurs et les cas d'utilisation de cette initiative?
2. Modéliser ce diagramme d'utilisation
3. Compléter le diagramme avec les cas d'utilisation additionnels ci-après
    - Lorsqu'un client entre ses informations de connexion, l'app bancaire vérifie le mot de passe avant de finir le processus de connexion.
    - Si le mot de passe est incorrect, l'app bancaire affichera un message d'erreur.
    - Quand un client transfert des fonds ou fait un paiement, l'app bancaire va s'assurer que le client a suffisamment d'argent sur son compte pour effectuer ces transactions.
    - Lorsque qu'un client veut faire un paiement, l'app bancaire leur donnera l'option de payer avec leur compte courant ou leur compte d'épargne.

Cycle de développement logiciel
Histoire utilisateur

Une histoire des utilisateurs est une exigence du système à développer, formulée en une ou 2 phrases dans un langage utilisateur normal.
Le format utilisé permet d'identifier le persona/acteur, la fonction et la valeur ou l'objectif a atteindre.

En tant que <persona>, je veux <fonctionnalité> pour que <satisfaire un besoin >.

Quelques exemples :

- En tant qu'utilisateur je peux réserver des chambres d'hôtel pour passer mon séjour
- En tant que en tant que recruteur je peux déposer les offres d'emploi pour diffuser les offres des postes à pourvoir disponibles
- En tant que membre régulier, je veux m'inscrire à des cours en ligne afin de pouvoir me qualifier pour un poste avancé.
- En tant que membre du personnel, je veux envoyer des courriels aux membres des comités afin de leur communiquer les horaires, les procès-verbaux et les notes.

Cycle de développement logiciel
Histoire utilisateur

Ron Jeffries recommande d'utiliser les 3C pour la décrire :

1. **Carte** : l'histoire est courte et écrite sur une carte
2. **Conversation:** les détails de l'histoire sont discutés
3. **Confirmation** : l'histoire est confirmée par des tests d'acceptation rédigés au même moment que celle-ci, au dos de la carte.

Histoires utilisateurs formalisent le besoin de l'utilisateur et elles sont orientées but.

- Elles font l'objet d'ateliers de travail avec les utilisateurs pour les découvrir et les expliciter.
- Elles sont rédigées par les analystes et parfois directement par le client
- Elles sont textuelles et obéissent à des règles de construction très précises.
- Après vont être priorisées et elles vont être ainsi guider les développements
- Elles mettent en avant les rôles, des différents profils d'utilisateurs.
- Affecte le choix des contenus des itérations
- Elles ne traitent que des exigences fonctionnelles, notamment les aspects interface et ergonomie.

Cycle de développement logiciel
Histoire utilisateur – critères d'accéptation

Donne la définition de "Terminé" pour une histoire..

Qu'est-ce qui est "assez suffisant" ? Concentrez-vous sur les critères d'acceptation qui vous donnent l'ensemble minimal de fonctions à mettre en service.

Formaliser les critères obligatoire d'acceptation d'une histoire (Must have)

Fournit des informations suffisamment spécifiques pour être testables.

Essuie-glaces

Vous achetez une nouvelle voiture Haut de Gamme et votre histoire d'utilisation indique que vous voulez des essuie-glaces à vitesse variable afin de pouvoir voir clairement la route.

Les critères d'acceptation doivent définir le nombre et les niveaux de vitesse des essuie-glaces. Imaginez la différence entre un véhicule d'entrée de gamme doté uniquement d'un réglage haut/bas et un véhicule haut de gamme doté de la technologie de détection de pluie.

Imaginez que vous avez supposé que la technologie de détection de pluie était incluse mais que vous n'avez obtenu que le réglage haut/bas ? Avez-vous pensé à mentionner un essuie-glace arrière ? Quel niveau est acceptable pour vous, en tant que consommateur ?

Cycle de développement logiciel
Diagramme des cas d'utilisation vs Histoire utilisateur

User Story	Use case
C'est une brève description d'une fonctionnalité telle que vue par l'utilisateur.	Représente une séquence d'action qu'un système peut accomplir en interagissant avec les acteurs du système
Mode orale et collaboratif	Mode écrit et souvent distant
Grande lisibilité vu sa simplicité	Pourrait souvent manquer de lisibilité (volume)
Format écrit court laissant part à de belles discussions orales	Format écrit très riche en information (postcondition, scénario …) peu de place à l'oral
Utilisée pour spécification des exigences mais surtout pour estimation de la planification	Utilisé seulement en tant que spécification
émergence rapide au travers des ateliers de collaboration	Long travail d'analyse et de formalisation
Implémentée et testé en une itération seulement	Implémenté et testé en plusieurs opérations
Contient des tests d'acceptation	Ne contient pas les cas de test qui en découle.
Difficile à lier les unes aux autres: Absence de vue globale	Liaison et vue globale faciles au travers du diagramme des cas d'utilisation qui lient les use case
Associée au méthodes agiles comme SCRUM, XP	Associée généralement au Processus Unifié
Très facile à maintenir	Plus difficile à maintenir

Fin

Merci pour votre attention