CYCLE DE DÉVELOPPEMENT D'UN LOGICIEL ET L'ASSURANCE QUALITÉ LOGICIELLE

27 avril 2024

S. Jule César Francky B. ADEOSSI
Ing. Audit informatique et Génie logiciel

INF1425

OBJECTIFS DU COURS

Au terme de ce cours, les apprenants seront capables :

- d'identifier les différentes activités comprises dans le cycle de développement de logiciel
- d'assurer leur enchaînement conformément aux bonnes pratiques.
- d'expliquer les différentes mesures pour évaluer un logiciel
- d'apprécier la qualité logicielle
- d'apporter des solutions pour la qualité d'un logiciel
- de comprendre les normes pour la qualité d'un logiciel notamment la norme ISO 9126

PLAN DU COURS

Partie I : Cycle de vie de développement d'un logiciel

- Généralités
- Notion de logiciel et ses caractéristiques
- Notion de Génie logiciel et de cycle de développement logiciel
- Les différentes catégories de logiciel
- Les différents types de logiciels
- Pourquoi se préoccuper d'un « cycle de vie » ?
- Les différentes phases du cycle de développement de logiciel
- Étude des modèles de cycle de vie
- Étude de quelques méthodologies
- Comparaison des approches Cascade, V et Itératif

PLAN DU COURS

Partie II : Assurance qualité d'un logiciel

- La qualité d'un logiciel
- Les critères de la qualité d'un logiciel
- La norme ISO 9126

Quand commence la réalisation d'un logiciel ?

- quand on écrit la première ligne de code ?
- quand on a planifié son développement ?
- quand on a écrit la spécification ?
- quand on a écrit le cahier des charges ?
- quand on a terminé l'étude de marché
- ...

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
GÉNÉRALITÉS

Quand est ce qu'on peut dire qu'un logiciel est terminé ?

- quand on a fini de le programmer ?
- quand on l'a compilé ?
- quand il s'exécute sans se planter ?
- quand on l'a testé ?
- quand on l'a documenté ?
- quand il est livré au premier client ?
- quand il n'évolue plus ?
- quand il n'est plus maintenu ?

Qu'est ce qu'un logiciel ?

- Le logiciel est un ensemble de programmes, de procédures, de données et de documentation associée, qui permet d'exécuter des tâches spécifiques ou de fournir des fonctionnalités particulières.
- Il est la composante immatérielle d'un système informatique, par opposition au matériel (hardware).

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
NOTION DE LOGICIEL ET SES CARACTÉRISTIQUES

Les catégories de logiciel

Il existe plusieurs catégories de logiciels, voici les principales :

- Logiciels système
- Logiciels applicatifs
- Logiciels de développement
- Logiciels embarqués
- Logiciels de gestion
- Logiciels en ligne (web)
- Logiciels mobiles

Les types de logiciels

- **Les logiciels payants**

Les logiciels payants appartiennent à leurs concepteurs, en l'occurrence à des entreprises qui créent des logiciels. Les logiciels payants ne peuvent pas être modifiés ni revendus par ceux et celles qui les achètent.
Ex: Microsoft office 365, Suite Adobe…

- **Les logiciels gratuits**

Les logiciels gratuits, en plus de pouvoir être obtenus sans frais, peuvent aussi être copiés par ceux et celles qui se les procurent. La plupart du temps, un logiciel gratuit est utilisé comme outil promotionnel. Il s'agit donc d'un produit d'appel qui sera gratuit pour une période donnée.
Ex : Avast, VLC…

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
NOTION DE LOGICIEL ET SES CARACTÉRISTIQUES

- **Les logiciels libres**

Les logiciels libres sont des logiciels libres de droit, c'est-à-dire des logiciels dont le code source est ouvert. Les personnes qui utilisent des logiciels libres sont donc en mesure de faire des copies et d'y apporter les modifications qu'elles désirent.
Ex: Linux, Notepad++…

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
NOTION DE LOGICIEL ET SES CARACTÉRISTIQUES

Caractéristiques d'un logiciel

- Intangibilité
- Malléabilité
- Complexité
- Reproductibilité
- Portabilité
- Évolutivité

Ces caractéristiques uniques du logiciel influencent grandement son développement, sa gestion et son utilisation dans les systèmes informatiques.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
NOTION DE GÉNIE LOGICIEL ET DE CYCLE DE DÉVELOPPEMENT LOGICIEL

Génie logiciel

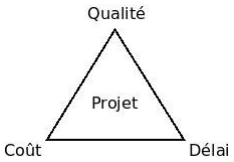
- est un domaine des sciences de l'ingénieur dont l'objet d'étude est la conception, la fabrication, et la maintenance des systèmes informatiques complexes.
- Il vise à appliquer une approche méthodique, rigoureuse et quantifiable au développement de logiciels.
- L'objectif du génie logiciel est de produire des logiciels de qualité, fiables, maintenables et évolutifs.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
NOTION DE GÉNIE LOGICIEL ET DE CYCLE DE DÉVELOPPEMENT LOGICIEL

Génie logiciel (software engineering)

- est un domaine des sciences de l'ingénieur dont l'objet d'étude est la conception, la fabrication, et la maintenance des systèmes informatiques complexes.
- Il vise à appliquer une approche méthodique, rigoureuse et quantifiable au développement de logiciels.
- L'objectif du génie logiciel est de produire des logiciels de qualité, fiables, maintenables et évolutifs.

Comme tout projet, la réalisation d'un logiciel est soumise à des exigences contradictoires et difficilement conciliables (triangle coût-délai-qualité).



PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
NOTION DE GÉNIE LOGICIEL ET DE CYCLE DE DÉVELOPPEMENT LOGICIEL

Cycle de développement logiciel

Le cycle de développement logiciel décrit les différentes phases par lesquelles passe un projet de développement de logiciel, dans le but d'obtenir des produits de qualité, fiables et évolutifs.

Le développement comprend un ensemble d'activités :

- Spécifications des besoins,
- Conception générale,
- Conception détaillée,
- Codage et tests unitaires,
- Intégration des modules,
- Intégration du logiciel,
- Recette.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
NOTION DE GÉNIE LOGICIEL ET DE CYCLE DE DÉVELOPPEMENT LOGICIEL

Pourquoi un cycle de vie de logiciel

- **Assurer la qualité du logiciel**

Le cycle de vie permet de définir un processus structuré pour concevoir, développer, tester et déployer le logiciel de manière rigoureuse.
Cela contribue à réduire les erreurs et à améliorer la qualité du produit final.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
NOTION DE GÉNIE LOGICIEL ET DE CYCLE DE DÉVELOPPEMENT LOGICIEL

- **Maîtriser les coûts et les délais**

Le cycle de vie aide à planifier et à organiser le projet de développement de manière à respecter les budgets et les échéances prévus.
Il permet d'anticiper et de gérer les différentes phases du projet de manière efficace.

- **Faciliter la maintenance et l'évolution**

Le cycle de vie définit des pratiques pour documenter le logiciel et le rendre plus maintenable à long terme.
Cela permet d'adapter et de faire évoluer le logiciel pour répondre aux nouveaux besoins.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
NOTION DE GÉNIE LOGICIEL ET DE CYCLE DE DÉVELOPPEMENT LOGICIEL

- **Améliorer la collaboration**

Le cycle de vie établit une méthodologie commune pour impliquer les différentes parties prenantes (développeurs, utilisateurs, gestionnaires, etc.) tout au long du projet.
Cela favorise une meilleure coordination et une communication plus efficace.

- **Répondre aux exigences réglementaires**

Certains secteurs, comme l'aéronautique ou le médical, imposent des normes et des processus de développement spécifiques.
Le cycle de vie permet de se conformer à ces exigences réglementaires.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
NOTION DE GÉNIE LOGICIEL ET DE CYCLE DE DÉVELOPPEMENT LOGICIEL

- **Capitaliser sur l'expérience :**

Le cycle de vie favorise la capitalisation des bonnes pratiques et des leçons apprises d'un projet à l'autre.
Cela permet d'améliorer continuellement la qualité et l'efficacité du processus de développement.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
NOTION DE GÉNIE LOGICIEL ET DE CYCLE DE DÉVELOPPEMENT LOGICIEL

Les différentes phases du cycle de développement de logiciel

Le développement comprend un ensemble d'activités :

- Spécifications des besoins,
- Conception générale,
- Conception détaillée,
- Codage et tests unitaires,
- Intégration des modules,
- Intégration du logiciel,
- Recette.

Ce cycle peut suivre différentes méthodologies de développement, comme le modèle en cascade, le modèle itératif (Agile) ou le modèle en spirale…

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

La plupart des modèles des processus reprennent les activités fondamentales mais les organisent différemment

- De nombreux modèles ont été définis
- Un modèle peut être spécifique à une organisation et à un type de logiciels (ex: embarqué)
- Il existe malheureusement peu d'outils supportant les processus

Il existe 3 principales familles :

- Modèle en cascade
- Modèle en V
- Modèle itératif
- Modèle en spirale

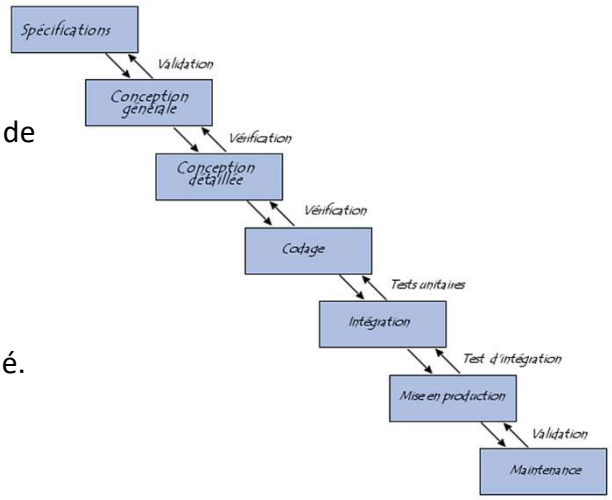
PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Modèles en cascade

Considérer le développement logiciel comme une succession d'étapes réalisées de façon strictement séquentielle.

Chaque étape correspond à une activité de base

- Chaque étape est validée
- Il n'y a pas (ou peu) de retours en arrière
- Les tests commencent une fois le développement du logiciel terminé.



PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Avantage du modèle en cascade

- ✓ Simple & convivial
- ✓ Étapes clairement définies
- ✓ Détection précoce des erreurs
- ✓ Livrable spécifique à chaque phase

Inconvénients du modèle en cascade

- ✓ Robuste aux exigences dynamiques
- ✓ Ne convient pas aux projets complexes
- ✓ Cycle de vie coûteux et plus long

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Exemple : Projet de réalisation du site web de l'IFRI

1- Interviewer les professeurs et les étudiants

2. Rédiger des spécifications du site

3. Analyser le besoin

4. Identifier et formaliser une architecture

5. Coder

6. Intégrer, tester

7. Vérifier la qualité du produit

8. Livrer

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Modèles en V

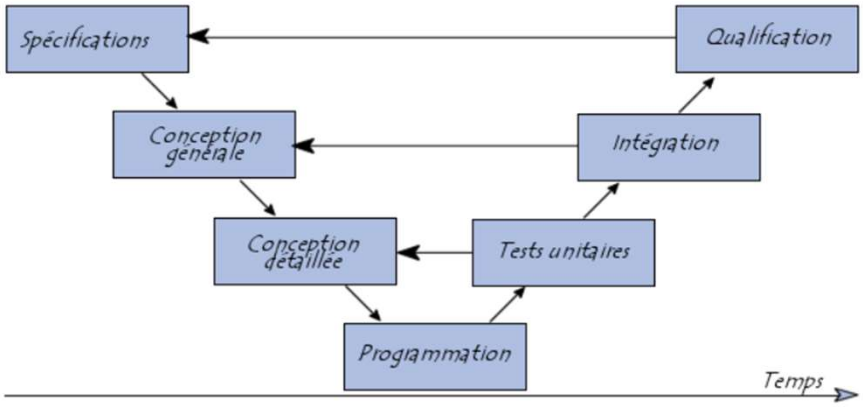
Le modèle de cycle de vie en V part du principe que les procédures de vérification de la conformité du logiciel aux spécifications doivent être élaborées dès les phases de conception.

Il est également connu sous le nom de modèles de vérification et de validation.

Chaque phase du modèle en V sera achevée avant le début de la phase suivante.

- Structuration de la phase de validation
- Le test du produit se fait en parallèle avec les phases de développement correspondantes.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE



PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Avantage du modèle en V

- ✓ Modèle hautement discipliné
- ✓ Simple et convivial
- ✓ Vérification et validation à chaque phase de développement

Inconvénients du modèle en V

- ✓ Rigide et inflexible
- ✓ Ne convient pas aux projets complexes
- ✓ Cycle de vie coûteux et plus long

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Exemple : Cycle de développement d'une voiture

- Durée : 3 ans

- Découpage en 4 niveaux différents :
  - Prestation
  - Système
  - Composant
  - Code

- Chaque niveau :
- Découpe son problème en sous éléments
- Rédige des spécifications (par aller retour avec l'étage inférieur)
- Établit un plan de validation
- Effectue la validation une fois le travail de la partie inférieur terminé

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Cycle de développement à Renault

- **Prestation**
  - Correspond aux besoins clients
  - Gérée par l'équipe prestation
  - Fourni une documentation contenant les contraintes à respecter
  - Fourni une documentation contenant la manière de les valider
  - ex: climatisation: pour une commande donnée, on veut tel débit d'air et tel température
  - la validation ne peut se faire qu'une fois la voiture terminée

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Cycle de développement à Renault

- **Système**
  - Les prestations sont décomposées en fonctions système
  - Chaque fonction système est associé à un composant
  - Géré par l'équipe système
  - Fourni une documentation contenant les exigences
  - Fourni une documentation contenant un plan de validation
  - ex: climatisation: produire du chaud / produire du froid / mixer / souffler / calculer
  - Validation à partir de tous les composants

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Cycle de développement à Renault

- **Composant**
  - Chaque composant reçoit les différentes fonctions systèmes qui lui sont associées
  - Découpage en fonctions logicielles
  - Géré par l'équipe composant
  - Crée un modèle de la fonction à réaliser (Matlab)
  - Fourni une documentation contenant les spécifications
  - Fourni une documentation contenant les tests
  - ex : calculateur de climatisation : fonction qui retourne une certaine valeur pour une certaine entrée
  - Valide l'assemblage de toutes les fonctions logicielles

Cycle de développement à Renault

- **Code**
  - Production du code correspondant à la fonction logicielle
  - Géré par un fournisseur
  - Fourni du code sur un composant
  - Valide les tests
  - En général, 5 aller retour avec l'équipe composant pour une fonction

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

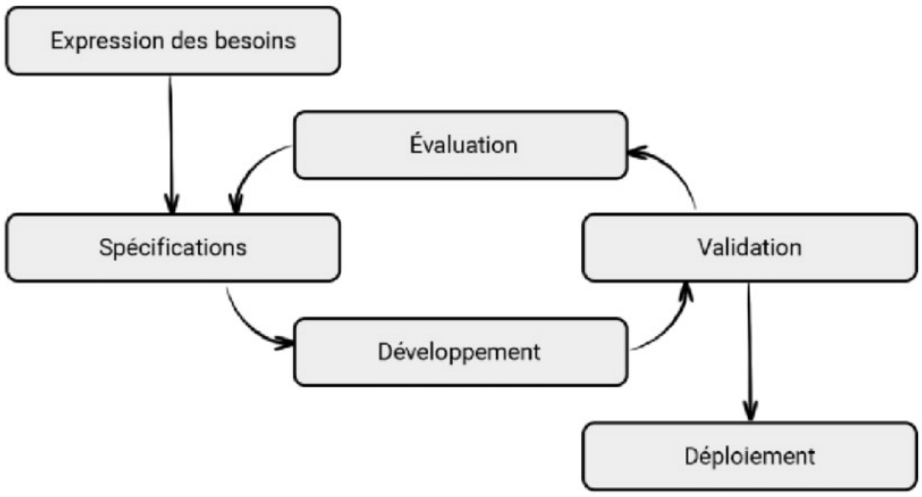
Modèle itératif

Le modèle de cycle de vie itératif ou incrémentiel commence par la simple implémentation d'un petit ensemble d'exigences logicielles et améliore progressivement le logiciel jusqu'à ce que le système complet soit implémenté et prêt à être déployé.

Le modèle itératif ne commence pas avec l'ensemble des spécifications du client. Au lieu de cela, le développement logiciel démarre avec un sous-ensemble de l'exigence et s'améliore progressivement avec plusieurs itérations avant de développer le produit final.

L'idée fondamentale derrière cette méthode est de développer un produit logiciel par cycles répétés (Itératif) et par portions (Incrémental).

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE



PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Avantage du modèle itératif

- ✓ Les résultats sont obtenus tôt et périodiquement
- ✓ Commentaires réguliers après chaque itération à l'équipe de développement
- ✓ Chaque itération rend les tests et débogage facile
- ✓ Meilleure analyse des risques et gestion

Inconvénients du modèle en itératif

- ✓ Plus de ressources & l'attention de la direction est requise
- ✓ Ne convient pas aux petits projets
- ✓ Risque de changement d'avis fréquent du client

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Exemple : Projet de réalisation du site web de l'IFRI

- Les différents cycles (itérations)
  - Gestion des données liées à l'IFRI
  - Définition de l'architecture du site
  - Définition de la charte graphique du site
  - Intégration des données au site

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Modèle en spirale

Il part du principe que le développement d'applications représente un cycle itératif, qui doit être répété jusqu'à ce que le but fixé soit atteint. Par une analyse régulière des risques et des contrôles réguliers du produit intermédiaire, le modèle en spirale diminue considérablement le risque d'échec lors des projets logiciels de grande taille.

Chaque cycle de la spirale se déroule en quatre phases :

- **Phase 1 : définition des objectifs et d'alternatives**

Un cycle-type dans le modèle en spirale commence par la **détermination des objectifs** à associer aux différentes étapes individuelles du processus de développement. Il peut ici s'agir par exemple **d'améliorer des performances** ou **d'étendre des fonctionnalités**. Dans le même temps, il convient de définir des alternatives pour la mise en œuvre (conception A vs. conception B par exemple) et de déterminer le cadre général ainsi que les **coûts** ou le **temps de travail** nécessaire.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

- **Phase 2 : examen des alternatives**

L'étape suivante correspond à **l'évaluation des alternatives**, dans laquelle les objectifs et le cadre général servent de valeurs de référence. Dans cette phase du cycle du modèle en spirale, le but est d'identifier les zones d'incertitude, c'est-à-dire les zones du projet qui **comportent un risque non négligeable pour l'avancement du projet de développement**. Ensuite a lieu **l'élaboration de la stratégie la moins risquée et la plus économique,** où des méthodes telles que le prototypage, les simulations, les tests d'étalonnage, les modèles d'analyse et les sondages d'usagers peuvent être employées.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

- **Phase 2 : examen des alternatives**

L'étape suivante correspond à **l'évaluation des alternatives**, dans laquelle les objectifs et le cadre général servent de valeurs de référence. Dans cette phase du cycle du modèle en spirale, le but est d'identifier les zones d'incertitude, c'est-à-dire les zones du projet qui **comportent un risque non négligeable pour l'avancement du projet de développement**. Ensuite a lieu **l'élaboration de la stratégie la moins risquée et la plus économique,** où des méthodes telles que le prototypage, les simulations, les tests d'étalonnage, les modèles d'analyse et les sondages d'usagers peuvent être employées.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

- **Phase 3 : développement et contrôle de l'état intermédiaire**

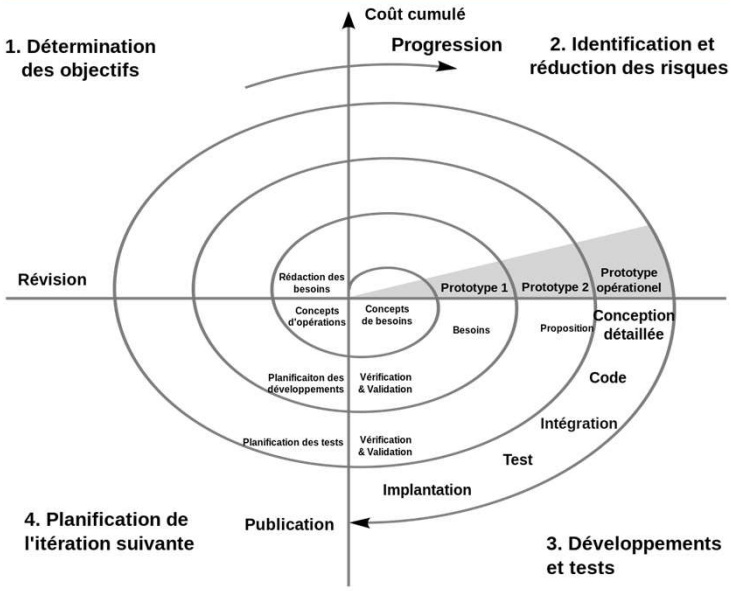
Après l'analyse des risques, place au **développement logiciel** à proprement parler, une phase toujours caractérisée par des risques résiduels relatifs. Si des risques liés aux performances ou aux interfaces utilisateur ou encore des risques concernant le contrôle des interfaces internes pèsent sur le processus de développement, une **stratégie de développement évolutive** est d'abord possible, dans laquelle le projet est spécifié plus précisément et les prototypes optimisés. Le code est écrit et testé plusieurs fois jusqu'à obtenir le résultat voulu, lequel servira ensuite de base à faible risque pour les étapes de développement ultérieures.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

- **Phase 4 : planification du cycle suivant**

Avec la fin d'un cycle commence déjà la planification du cycle suivant. Il peut s'agir de l'avancement régulier du projet, si **l'objectif du cycle a été atteint** et l'objectif suivant doit être défini. Mais il peut s'agir également de trouver des solutions, **si l'étape de développement précédente ne s'est pas déroulée comme prévu**. Ainsi, par exemple, la stratégie suivie jusqu'alors peut être remplacée par l'une des alternatives déjà définies au préalable ou bien par une nouvelle alternative. Avec celle-ci, il est ensuite possible de démarrer une nouvelle tentative pour atteindre l'objectif fixé.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE



PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Avantage du modèle en spirale

- ✓ Modèle générique flexible
- ✓ Implication précoce du client et des utilisateurs possible
- ✓ Contrôle périodique dû aux risques
- ✓ Coordination parfaite entre exigences techniques et conception
- ✓ Maîtrise maximale des coûts, ressources et qualité du projet logiciel
- ✓ Adapté aux environnements techniques novateurs
- ✓ Convient aux grands projets

Inconvénients du modèle en spirale

- ✓ Effort de gestion important
- ✓ Les décisions régulières peuvent retarder le processus de développement
- ✓ À cause de la subdivision du processus de développement, des erreurs et incohérences de conception peuvent facilement se retrouver dans le produit final
- ✓ Connaissance en analyse et gestion des risques indispensable, mais souvent manquante
- ✓ Inadapté aux petits projets aux risques raisonnables

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Modèle agile

En termes simples, agile n'est rien d'autre qu'une chaîne de développement et de déploiement rapides.

Le modèle agile aide à créer des logiciels de manière incrémentielle à l'aide de courtes itérations en divisant l'application en petits morceaux de code en suivant le modèle de microservices pour s'aligner sur l'évolution des besoins de l'entreprise.

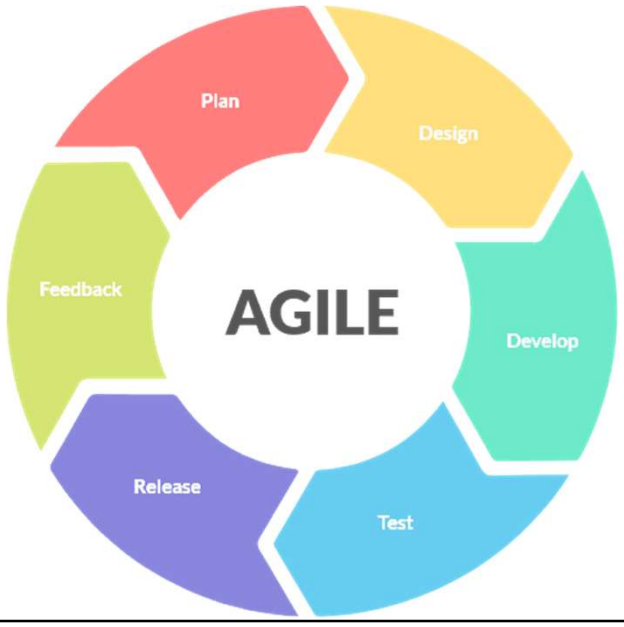
Au lieu d'un développement en une seule passe de 6 à 18 mois où toutes les exigences et tous les risques sont prédits à l'avance, Agile s'adapte aux retours d'information fréquents en livrant un produit exploitable après chaque itération.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Dans un modèle agile, les itérations sont appelées sprints. Et, chaque sprint dure de 1 à 4 semaines.

Une fois chaque sprint terminé, le chef de produit vérifie le produit mis à jour et le déploie dans l'environnement client. Après le déploiement, les commentaires des clients sont rassemblés et le produit est amélioré dans le sprint de suivi.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE



PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Principes des méthodes agiles	
Utilisateur	implication dans le développement fourniture des exigences et évaluation des itérations
Incréments	fourniture incrémentale du logiciel
People	reconnaissance du talent des développeurs pas de processus imposé
Changements	conception orientée évolution
Simplicité	Chasser toute forme de complexité
Tests	jouent le rôle de spécification
Binômes	les développeurs travaillent par binômes

Quelques méthodologies Agiles

Une fois qu'une organisation décide d'adopter une gestion de développement Agile, il reste encore à choisir la méthodologie la plus adaptée à son projet.
Les méthodologies Agiles les plus populaires en usage sont:

- L'eXtrême Programming (XP),
- Scrum,
- Feature Driven Development (FDD),
- Lean Software Development,
- Agile Unified Process (Agile UP ou AUP),
- Crystal
- et Dynamic Systems Development Method (DSDM) .

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Extreme Programming (XP)

Conçu par Kent Beck, Extreme Programming, ou XP, est une méthode agile de gestion de projet particulièrement bien adaptée aux projets de développement informatique.
Le principe fondamental de la méthode XP est de faire collaborer étroitement tous les acteurs du projet et d'opter pour des itérations de développement très courtes.
L'Extreme Programming préconise également le travail en binôme des développeurs, facilitant ainsi la production d'un code simple, facilement lisible et maintenable.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Scrum

Destinée à la gestion de projets informatiques, Le principe de Scrum est de pouvoir modifier la direction prise par le projet au fur et à mesure de son avancement.
Si les conditions de réussite ne sont pas remplies, alors il faut réorienter le projet pour repartir sur de meilleures bases. Le client est étroitement impliqué grâce à la livraison régulière de prototypes opérationnels permettant de valider les développements. Cette gestion dynamique permet de s'assurer de la correspondance entre le besoin exprimé et le produit livré, et de réorienter au besoin les futurs développements.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Feature Driven Development (FDD)

Le Développement Dirigé par les Fonctionnalités, est une méthode de gestion de projet basée sur la gestion des risques. Les développements sont organisés en itérations courtes autour de fonctionnalités testables par l'utilisateur. L'utilisateur est ainsi impliqué dans les développements, peut suivre l'avancement du projet et la validation des fonctionnalités. Aucune méthode de programmation n'est priorisée, c'est réellement la fonctionnalité qui est mise en avant.
Un projet géré par FDD est découpé en plusieurs grandes étapes :

- la constitution d'un modèle général du produit
- la construction de la liste complète des fonctionnalités à réaliser
- la conception technique des fonctionnalités,
- la réalisation.

Lean Software Development

Le Lean Software Development est basé sur sept grands principes.

- Éliminer les gaspillages (finition partielle, processus inutiles, fonctionnalités non nécessaires, modification de l'équipe, retards…),
- Favoriser l'apprentissage (multiplication des sources d'apprentissage, synchronisation des équipes…),
- Reporter les décisions (jusqu'au dernier moment raisonnable, pour éviter de longues discussions sources de pertes de temps et les décisions irrévocables),
- Livrer vite (livraisons rapides et régulières de façon à avoir un retour client rapide également),
- Responsabiliser l'équipe (favoriser l'autonomie et le leadership des équipes, partir du principe que les intervenants connaissent leur travail, faciliter le développement de l'expertise),

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

- Construire la qualité (elle doit être placée au cœur du projet, de la conception à la réalisation),
- Optimiser le système dans son ensemble (mise en place de mesures de performances complètes, pour avoir en permanence une vision globale du produit, et gérer les différentes interactions et dépendances).

Avec la méthode Lean, la qualité est réellement placée au cœur de la gestion du projet, en optimisant notamment l'ensemble des processus d'apprentissage, de décision, de livraison et de mesure de performances.

Agile Unified Process (Agile UP or AUP)

Agile Unified Process (ou Processus Unifié Agile) est une version simplifiée du Rational Unified Process, ou RUP. Il s'agit d'une méthode de développement d'applications métier utilisant les techniques agiles du TDD (Test Driven Development ou développement piloté par les tests), du MDD (Model Driven Development ou développement piloté par le modèle) et de la gestion du changement.

La méthode est divisée en quatre phases :

- Lancement : identification du périmètre du projet, définition de la ou des architectures potentielles pour le système, implication des intervenants et obtention du budget.
- Conception : définir l'architecture du système et démonstration de sa pertinence.
- Réalisation : développement du logiciel lors d'un processus incrémental dans l'ordre de priorité des fonctionnalités.
- Livraison : validation et déploiement du système en production.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Crystal (Clear/Orange)

La méthode Crystal Clear est particulièrement adaptée aux petites équipes de développement. Idéalement, l'équipe est composée d'un architecte et de deux à six ou sept développeurs, situés à proximité les uns des autres, de façon à faciliter la communication, dans un local calme.

Dynamic Systems Development Method (DSDM)

Cette méthode s'articule autour de neufs grands principes qui sont la participation des utilisateurs, l'autonomie de l'équipe projet, la transparence des développements, l'adéquation avec le besoin, le développement itératif et incrémental, la réversibilité permanente, la synthèse du projet, les tests automatisés et continus et enfin la coopération entre tous les intervenants.

PARTIE I : CYCLE DE VIE DE DÉVELOPPEMENT D'UN LOGICIEL
ÉTUDE DES MODÈLES DE CYCLE DE VIE

Avantage du modèle agile

- ✓ Diffusion de logiciels persistants
- ✓ Satisfaction accrue des parties prenantes
- ✓ Flexibilité aux changements dynamiques à tout moment
- ✓ L'interaction quotidienne avec le client élimine les conjectures
- ✓ Un produit de haute qualité en un minimum de temps

Inconvénients du modèle agile

- ✓ Interactions fréquentes avec les clients
- ✓ Des professionnels hautement qualifiés sont requis
- ✓ Des professionnels hautement Relativement cher pour les petits projets

PARTIE II : ASSURANCE QUALITÉ D'UN LOGICIEL
LA QUALITÉ D'UN LOGICIEL

Définition

La qualité est la conformité avec les besoins, l'adéquation avec l'usage attendu, le degré d'excellence ou, tout simplement, la valeur de quelque chose pour quelqu'un.

Dans le domaine du logiciel, satisfaire les besoins de l'utilisateur suppose une démarche qualité qui prenne en compte :

- la qualité de son processus de développement (coûts, délais, méthodes, organisation, personnel, techniques, outils),
- la qualité intrinsèque du produit (modularité, simplicité, ...),
- la qualité du service fourni par le logiciel en exploitation.

PARTIE II : ASSURANCE QUALITÉ D'UN LOGICIEL
LA QUALITÉ D'UN LOGICIEL

L'assurance qualité

C'est l'ensemble des mesures, procédures, méthodes utilisées dans le cadre du processus de développement du logiciel afin d'obtenir le niveau de qualité souhaitée.
La mise en œuvre d'une politique d'assurance qualité passe par la rédaction d'un Manuel Qualité présentant toutes les procédures qui pourront (devront) être utilisées dans le cadre de l'activité informatique de l'entreprise.

L'évaluation de la qualité du logiciel

➢ **La métrologie du logiciel**

La métrologie du logiciel est un ensemble de méthodes qui permettent d'évaluer la qualité du logiciel.

PARTIE II : ASSURANCE QUALITÉ D'UN LOGICIEL
LA QUALITÉ D'UN LOGICIEL

Elle a pour objet :

- de définir un ensemble de caractéristiques mesurables du logiciel,
- de définir des méthodes d'évaluation,
- de définir des outils d'évaluation (analyseurs, jeux de tests),
- enfin, d'évaluer les logiciels par la mise en œuvre de ces méthodes et l'utilisation de ces outils.

➢ **Le modèle d'évaluation**

Ce modèle définit la qualité du logiciel à travers la qualité du produit, du processus et du service rendu. On peut représenter ce modèle sous la forme d'une arborescence.

PARTIE II : ASSURANCE QUALITÉ D'UN LOGICIEL
LA QUALITÉ D'UN LOGICIEL

Un **facteur** est une caractéristique du logiciel, du processus ou du service contribuant à sa qualité telle qu'elle est ressentie par l'utilisateur.

Un **critère** est un attribut du logiciel par l'intermédiaire duquel un facteur peut être obtenu. C'est également une caractéristique du logiciel sur laquelle le développeur peut agir. (par exemple, sa simplicité)

Une **métrique** est la mesure d'une propriété d'un critère. (par exemple, la taille d'un module pour le critère "Simplicité").

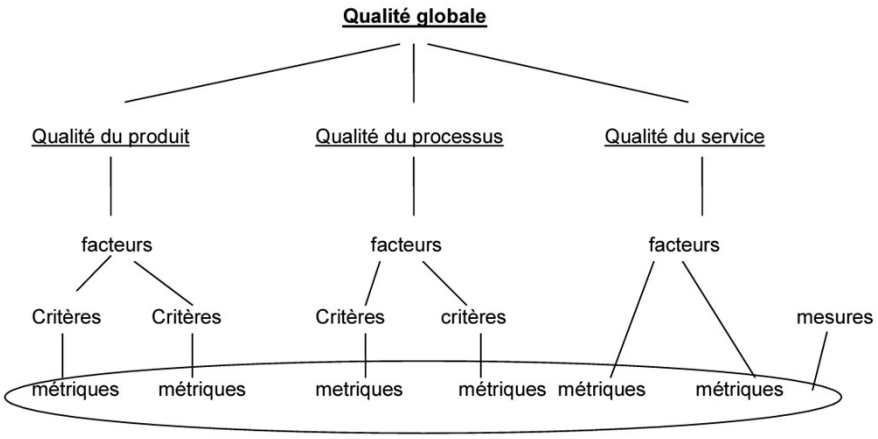
PARTIE II : ASSURANCE QUALITÉ D'UN LOGICIEL
LA QUALITÉ D'UN LOGICIEL

Les principaux facteurs de qualité d'un logiciel

Les principaux facteurs de qualité d'un logiciel sont:
- la conformité aux besoins,
- la fiabilité,
- l'ergonomie (dont la facilité d'emploi),
- la flexibilité,
- la maintenabilité,
- l'intégrité
- et la disponibilité.

Au vu de son utilisateur, un logiciel de qualité doit donc présenter ces caractéristiques sans que son efficacité, ses performances (temps de réponse, place mémoire minimum, ...) en pâtissent.

PARTIE II : ASSURANCE QUALITÉ D'UN LOGICIEL
LA QUALITÉ D'UN LOGICIEL



PARTIE II : ASSURANCE QUALITÉ D'UN LOGICIEL
LA QUALITÉ D'UN LOGICIEL

Les principaux critères de qualité d'un logiciel

Les principaux critères de qualité d'un logiciel sont:
- la cohérence,
- la complétude,
- la compréhensibilité,
- le contrôle des accès,
- la modularité,
- la protection des accès,
- et la simplicité.

Les indicateurs pour la qualité logicielle sont régis par la norme ISO 9126,

PARTIE II : ASSURANCE QUALITÉ D'UN LOGICIEL
LA QUALITÉ D'UN LOGICIEL

➢ **Les méthodes et les outils de mesure**

Il existe plusieurs méthodes et outils permettant d'effectuer des mesures sur la qualité d'un logiciel , de son processus de développement ou du service rendu.
Les principales méthodes sont les suivantes : **les audits ; les essais.**
Les principaux outils sont des outils de scrutation de programmes ou de réseaux.

PARTIE II : ASSURANCE QUALITÉ D'UN LOGICIEL
LA NORME ISO/CEI 9126

La norme ISO 9126 (« Technologies de l'Information : Qualités des produits logiciels ») définit un ensemble d'indicateurs pour la qualité logicielle, et « facilite » ainsi le processus d'évaluation logiciel et la spécification d'exigences fonctionnelles ou non-fonctionnelles.

Cette norme définit 6 caractéristiques permettant de décrire la qualité du logiciel, elles-mêmes détaillées en sous-caractéristiques.

- **Capacité fonctionnelle**

Il s'agit d'un ensemble d'attributs permettant de vérifier si le logiciel répond aux besoins fonctionnels exprimés. On y retrouve notamment les notions de pertinence, d'exactitude, d'interopérabilité, de sécurité et de conformité.

PARTIE II : ASSURANCE QUALITÉ D'UN LOGICIEL
LA NORME ISO/CEI 9126

- **Fiabilité**

Il s'agit ici de décrire les attributs permettant d'évaluer l'aptitude d'un système à maintenir son niveau de service : tolérance aux pannes, conditions de remise en service, maturité…

- **Facilité d'utilisation**

On retrouve dans cette caractéristique un ensemble d'attributs permettant de caractériser l'effort nécessaire à un utilisateur potentiel pour utiliser le système : facilité de compréhension, d'apprentissage, d'exploitation ou encore pouvoir d'attraction.

- **Rendement ou efficacité**

Cette caractéristique permet de qualifier le rapport entre le service rendu par le logiciel et les efforts qu'il faut entreprendre pour le faire fonctionner (quantité de ressources utilisées notamment).

PARTIE II : ASSURANCE QUALITÉ D'UN LOGICIEL
LA NORME ISO/CEI 9126

- **Maintenabilité**

Il s'agit ici d'évaluer les possibilités de faire évoluer le système dans le cas de nouveaux besoins : facilité d'analyse, de modification, stabilité et testabilité de la solution.

- **Portabilité**

Cette caractéristique décrit la possibilité de transférer le logiciel d'une plateforme à une autre, et les efforts nécessaires pour le faire : facilité d'adaptation et d'installation, coexistance, interchangeabilité.

MERCI DE VOTRE ATTENTION

S. Jule César Francky B. ADEOSSI
Ing. Audit informatique et Génie logiciel