Centre universitaire Abdelhafid Boussouf Mila

Faculté des sciences et de la technologie

Département math et informatique

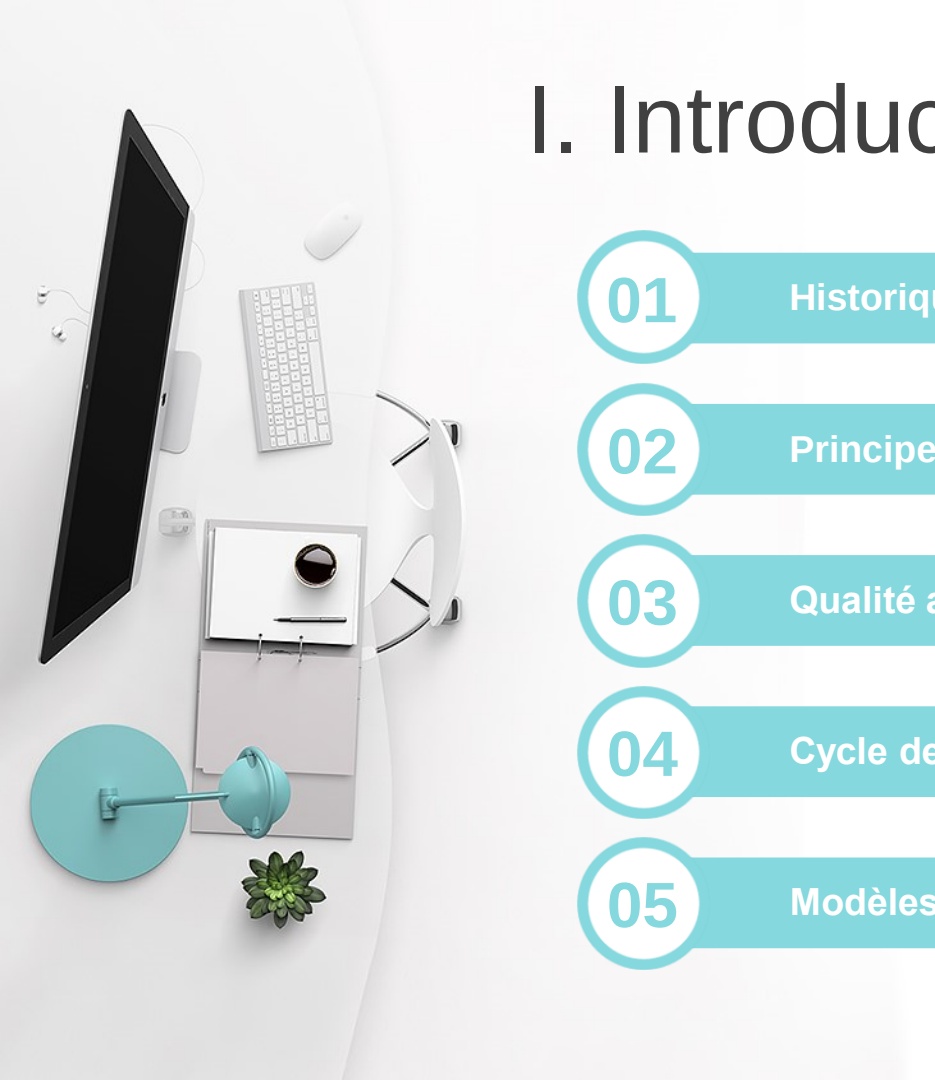
Génie logiciel

Chapitre 1

Introduction au génie logiciel

Mme. S.HEDJAZ

2022/2023



I. Introduction au GL

01 Historique, définitions et objectifs

02 Principes de génie logiciel

03 Qualité attendues d’un logiciel

04 Cycle de vie d’un logiciel

05 Modèles de cycle de vie d’un logiciel

01 Historique, définitions et objectifs

Le génie logiciel « **GL** » est apparut à la fin des années 60 pour répondre à **la crise logiciel**

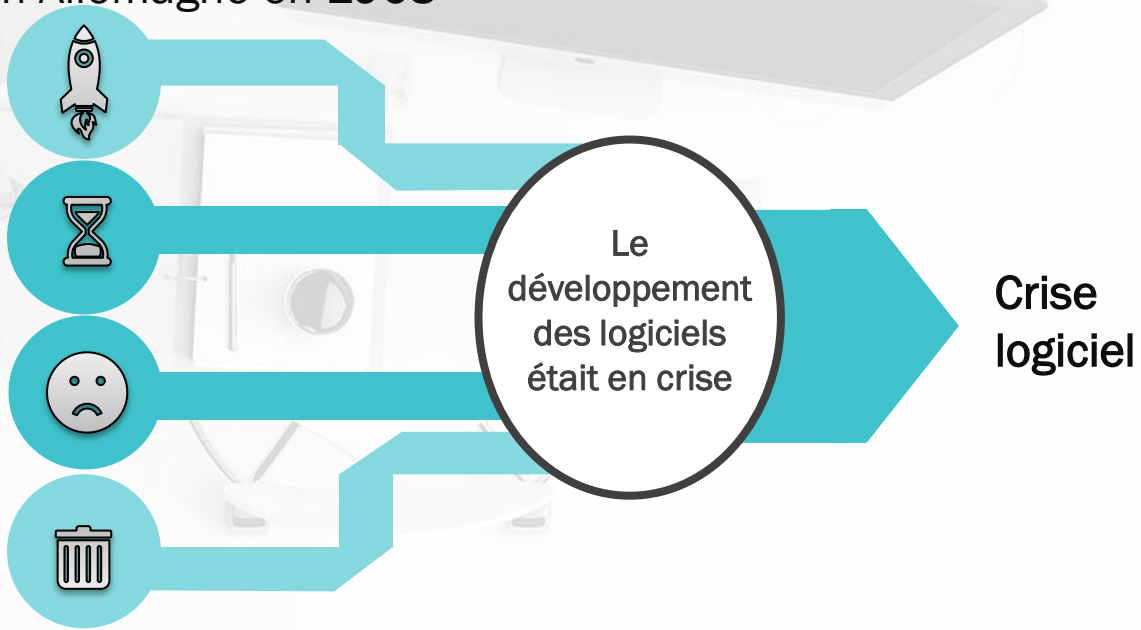
L'expression « **génie logiciel** » ou « **software engineering** » est introduit la première fois à une conférence en Allemagne en 1968

La construction des logiciels coutait très cher (Dépassement moyen 70%)

Les délais de livraison n'était pas respectés (Dépassement moyen 50%)

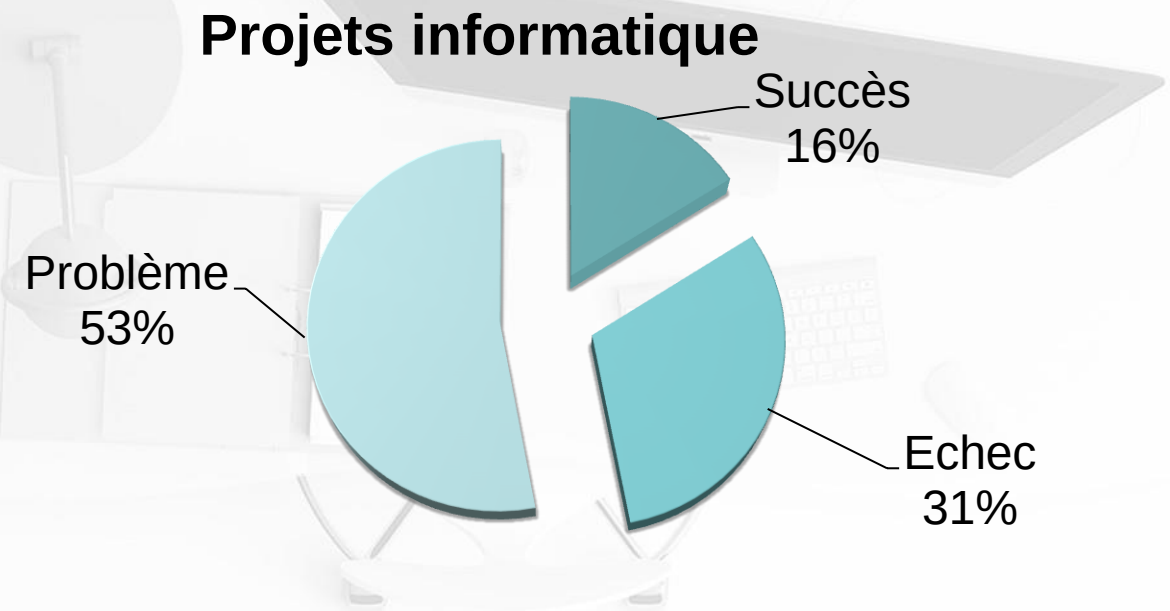
Les logiciels ne répond pas aux besoins des utilisateurs

Difficile à utiliser, maintenir ou faire évoluer (40% - 70% du coût logiciel)



01 Historique, définitions et objectifs

Selon une étude sur 8380 projets « Standish Group, 1995 » :



01 Historique, définitions et objectifs

Bugs célèbres:

1962: Destruction en vol de la sonde spatiale **Mariner 1** peu de temps après son envol

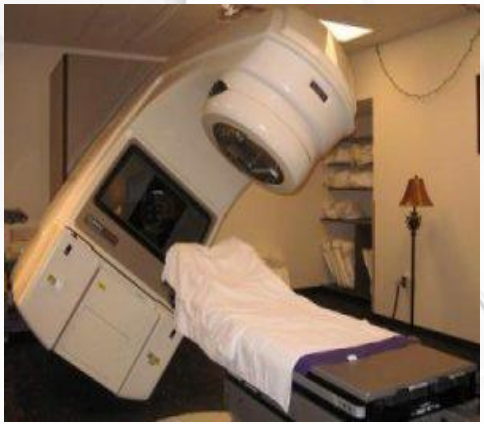
- Coût : 18,5 millions de dollars
- Mission: survol de la planète Vénus
- Cause: Erreur de transcription manuelle d'un symbole mathématique dans la spécification

01 Historique, définitions et objectifs

Bugs célèbres:

1985/1987: Problème du **Therac-25**, surdosage en radiothérapie « appareil d'irradiation thérapeutique »

- Au moins 5 morts par dose massive de radiations
- Cause: des fautes de conception dans le matériel et le logiciel



01 Historique, définitions et objectifs

Bugs célèbres:

1994: le bug de la division du **Pentium**, une erreur était introduite en raison d'une faute de conception de l'algorithme de division flottante

1996: **Ariane 5**, explosion du premier vol d'Arian 5 due à une erreur de débordement lors de la conversion d'un nombre flottant 64 bits vers un entier 16 bits.
Coût 500 M $.(Le bug informatique le plus couteux de l'histoire)

2000 : Le bug de, dysfonctionnements lorsque les dates sont postérieures au 31 décembre 1999

2011: **PlayStation Network**, des millions de données personnelles et bancaires piratées, pertes financières de plusieurs milliards de dollars, a cause des vulnérabilité du réseau

2014: **Outil de chiffrement, Open SSL,** 500000 serveurs web concernés par la faille. vulnérabilité permettant de lire une portion de la mémoire d'un serveur distant

01 Historique, définitions et objectifs

Idée:

L'idée de **génie logiciel** est d'appliquer les méthodes classique d'ingénierie au domaine du logiciel.

Ingénierie (ou génie): Ensemble de fonction allant de **la conception** et des études à la responsabilité de la **construction** et au **contrôle des équipement** d'une **installation techniques ou industrielle.**

Définitions:

Tel que défini par Pollice, 2005 et IEEE 1993, le génie logiciel est l'application d'une approche disciplinée quantifiable systématique, au développement, à l'exploitation et à la maintenance des logiciels, ainsi que l'étude de ces approches

Le génie logiciel est un ensemble **des méthodes**, **des techniques**, et des **outils dédiés** à **la conception**, au **développement** et à **la maintenance** des systèmes informatiques.

01 Historique, définitions et objectifs

Objectif GL (Critère CQFD):

Le GL se préoccupe des procédés de fabrication de logiciel de façon à s'assuré que les quatre critères suivant soit satisfait:

- Coût (C): les coûts reste dans les limites prévu au départ.
- Qualité (Q): La qualité correspond au contrat du service initial.
- Fonctionnalité (F): Le système qu'est fabriqué répond aux besoins des utilisateurs.
- Délai (D): Les délais reste dans les limites prévu au départ.

02 Principes de génie logiciel

Rigueur:

Les principales sources de défaillances d'un logiciel sont d'origine humaine. À tout moment, il faut se questionner sur la validité de son action.
Des outils de vérification accompagnant le développement peuvent aider à réduire les erreurs. Cette famille d'outils s'appelle CASE (Computer Aided Software Engineering).

Abstraction:

Extraire des concepts généraux sur lesquels raisonner, puis instancier les solutions sur les cas particuliers.

Décomposition en sous problèmes:

Traiter chaque aspect séparément, chaque sous-problème plus simple que problème global.

02 Principes de génie logiciel

Modularité:

Partition du logiciel en modules interagissant, remplissant une fonction et ayant une interface cachant l'implantation aux autres modules.

Construction incrémentale:

Construction pas à pas, intégration progressive.

Généricité:

Proposer des solutions plus générales que le problème pour pouvoir les réutiliser et les adapter à d'autres cas. Un logiciel réutilisable a beaucoup plus de valeur qu'un composant dédié.

Anticipation des évolutions:

Liée à la généricité et à la modularité, prévoir les ajouts/modifications possibles de fonctionnalités.

Documentations:

Essentielle pour le suivi de projet et la communication au sein de l'équipe de projet.

03 Qualité attendues d'un logiciel

Définition 1:

Ensemble d'entité nécessaire au fonctionnement d'un processus de traitement automatique de l'information.
Parmi ces entités, on trouve: **Programmes, Documentations d'utilisation, informations de configuration, … etc.**

Définition 2 :

Ensemble de **programmes** qui permet à un **système informatique d'assurer une tâche** ou une **fonction en particulier**.

Exemple: un jeux, site web, application mobile, …etc.

03 Qualité attendues d'un logiciel

Des facteurs de qualité du logiciel ont été donnée par B.MEYER dans (conception et programmation objet)

Externe

La validité: aptitude d'un produit logiciel à réaliser exactement les tâches définies par sa spécification.

La robustesse: aptitude d'un logiciel à fonctionner même dans des conditions anormales.

L'extensibilité: Facilité d'adaptation d'un logiciel aux changement de spécification.

Réutilisabilité: aptitude d'un logiciel à être réutiliser en tout ou en partie pour des nouvelles applications.

La compatibilité: aptitude d'un logiciel à pouvoir être combiner les un avec les autres.

03 Qualité attendues d'un logiciel

D'autres facteurs de qualité du logiciel sont moins cruciales:

L'efficacité: bonne utilisation des ressources du matériel.
La portabilité: facilité aux laquelle le produit être adapté à des différents environnements matériel ou logiciel.
Vérifiabilité: facilité de préparation des procédures de recette et de certification (test, ...).
L'intégrité: aptitude des logiciels à protéger leurs différents composants contre accès et des modifications non autorisés.
Facilité d'utilisation: facilité avec lesquelles les utilisateurs d'un logiciel peuvent apprendre comment l'utiliser, comment le faire fonctionner, comment préparer les données, mais aussi comment interpréter les résultats et les effets en cas d'erreur.

03 Qualité attendues d'un logiciel

Des critères interne permettent d'atteindre ces facteurs externe de qualité

Interne

La modularité: c'est la décomposition du logiciel en composant facilement appréhendable et relativement indépendants.

La complétude: c'est le degré d'implantation des spécification. Un logiciel est complet si toutes ses spécifications externes sont opérationnelles.

La cohérence: c'est la possibilité de faire des retours en arrières dans le cycle de développement. En particulier de faire remonter une erreur détectée en maintenance au niveau de l'implantation, de la conception, ou de l'analyse.

La généralité: plage d'application potentielle des composants logiciel

L'auto documentation et lisibilité: possibilité d'extraction de la documentation depuis les composants logiciel.

04 Cycle de vie d’un logiciel

Le cycle de vie du logiciel modélise l'enchaînement des différentes activités du processus technique de développement du logiciel.

Une activité comprend: des tâches, des contraintes, des ressources, une façon d’être réalisée.

Les grandes activités sont:

- Analyse des besoins
- spécification globale
- Conception architecturel et détaillé
- Programmation
- Gestion de configuration et d’intégration
- Validation et vérification
- Livraison et maintenance

04 Cycle de vie d'un logiciel

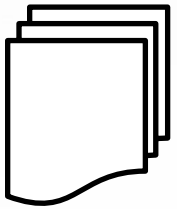
1.1. Analyse des besoins:

But: Eviter de développer un logiciel non adéquat. On va étudier le domaine d'application ainsi que l'état actuel et futur de l'environnement du système afin d'en déterminer: Les frontières, Le Rôle, Les ressources disponibles et requises, Les contraintes d'utilisation et performance … etc.

- Experts du domaines d'application
- Futur utilisateurs du système

Analyse des besoins

- Entretien
- Questionnaire
- Observation de l'existence
- Etude de situation similaire



- Cahier des charges (CC)
- Manuel d'utilisation préliminaire

04 Cycle de vie d'un logiciel

1.2. Spécification des besoins:

But: Etablir une première description du futur système. pour produire une description de ce que doit faire le système mais sans préciser comment il le fait

- Cahier des charges (CC)
- Manuel d'utilisation préliminaire

- Modèle E/A
- Diagramme de flot de données
- Diagramme d 'état transition
- Réseaux de pétri et les grafcet
- Diagramme des méthodes O.O ...

- Cahier des charges fonctionnel
- Technique et faisabilité informatique

1.3. Conception:

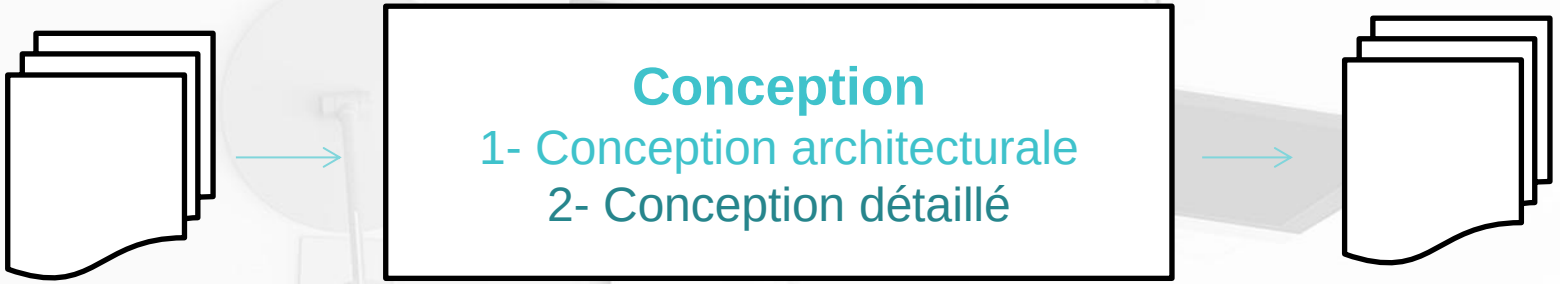
But: Enrichir la description du logiciel de détails d'implémentation afin d'aboutir à une description très proche d'un programme (décrire le comment).

✓ **La conception architecturale** (ou conception globale) a pour but de décomposer le logiciel en composants plus simples, définis par leurs interfaces et leurs fonctions (les services qu'ils rendent).

✓ **La conception détaillée** fournit pour chaque composant une description de la manière dont les fonctions ou les services sont réalisés : algorithmes, représentation des données.

04 Cycle de vie d'un logiciel

1.3. Conception:



- Cahier des charges fonctionnel
- Technique et faisabilité informatique

- Conception fonctionnelle et conception orienté objet
- Pour chaque composant, écrire les algorithmes

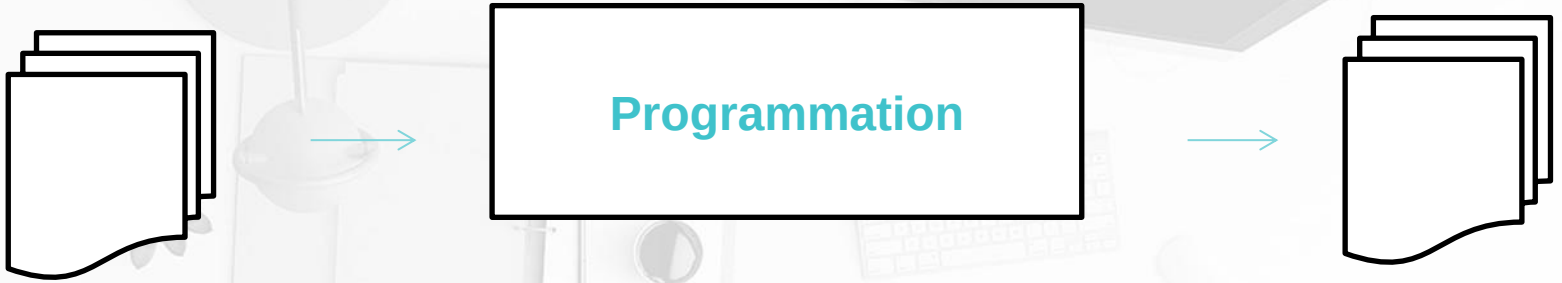
- Dossier de conception: (Architecture et Algorithme)

04 Cycle de vie d'un logiciel

1.4. Programmation:

Implantation de la solution conçue

✓ Choix de l'environnement de développement, du/des langage(s) de programmation, de normes de développement…



- Dossier de conception:

(Architecture et Algorithme)

- Code source
- Manuel final d'utilis

ateur
- Documentations

1.5. Gestion de configuration et intégration:

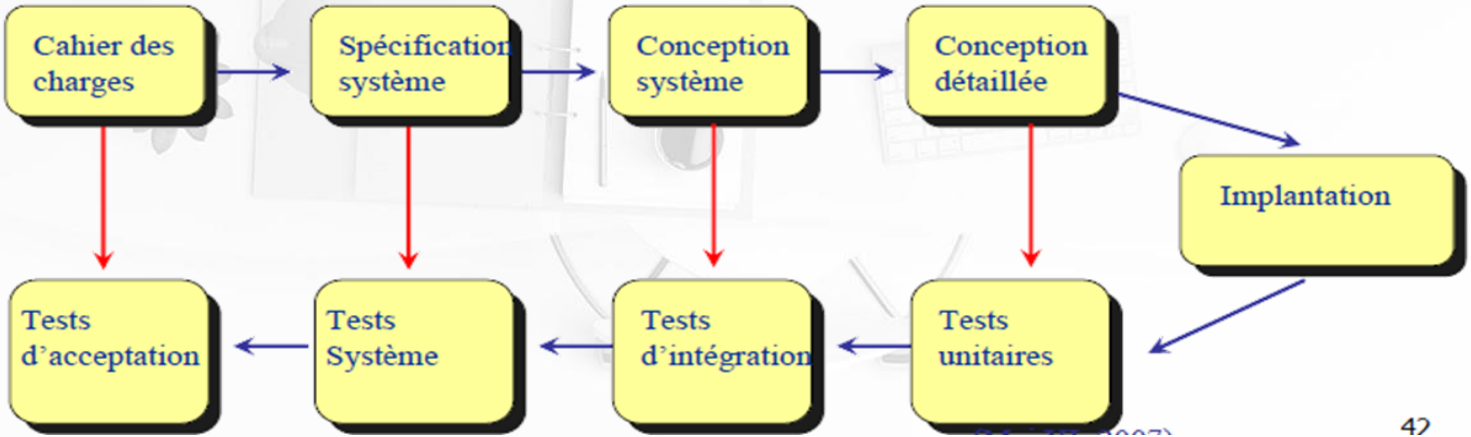
La gestion de configurations a pour but de maîtriser l'évolution et la mise à jour des composants tout au long du processus de développement.

L'intégration a pour but de réaliser un ou plusieurs systèmes exécutables à partir des composants (Combiner les composants).

1.6. Validation et vérification:

La validation a pour but de répondre à la délicate question : a-t-on décrit le bon système , celui qui répond à l'attente des utilisateurs.

La vérification répond à la question : le développement est-il correct par rapport à la spécification globale ? Ce qui consiste à s'assurer que les description successives et le logiciel lui-même satisfont la spécification.



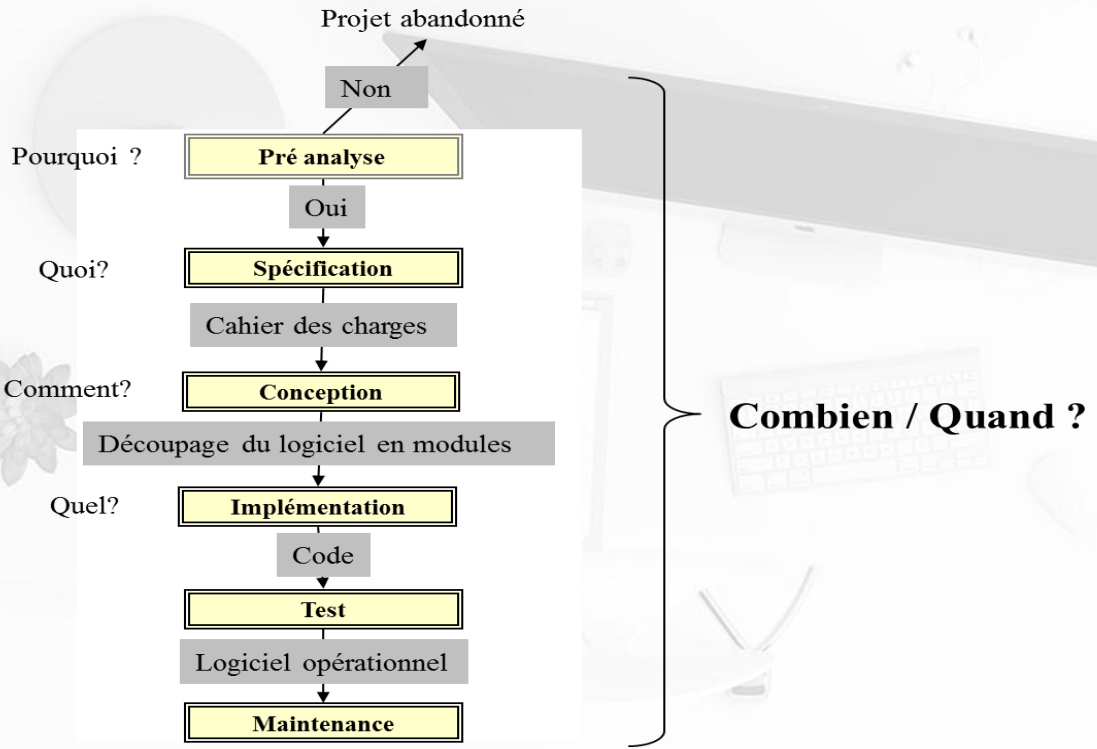
1.7. Maintenance:

Il s'agit d'apporter des modifications à un logiciel existant. C'est la phase la plus coûteuse (70% du coût total)

Types:

Maintenance corrective, Maintenance perfective, Maintenance adaptative.

04 Cycle de vie d'un logiciel



04 Cycle de vie d’un logiciel

Documents:

Manuel utilisateur final	Implémentation
Dossier de conception architecturale	Conception
Code source	Implémentation
Cahier des charges	Analyse des besoins
Manuel utilisateur préliminaire	Analyse des besoins
Dossier de conception détaillée	Conception
Rapport des tests	Tests
Documentation	Implémentation
Cahier des charges fonctionnel	Spécification

05 Les modèles de cycle de vie d'un logiciel

Modèles linéaires

- modèle en cascade
- modèle en V

...

Modèles non linéaires

- Prototypage
- modèle en spirale
- modelés incrémentaux
- Modèles unifiés
- ...

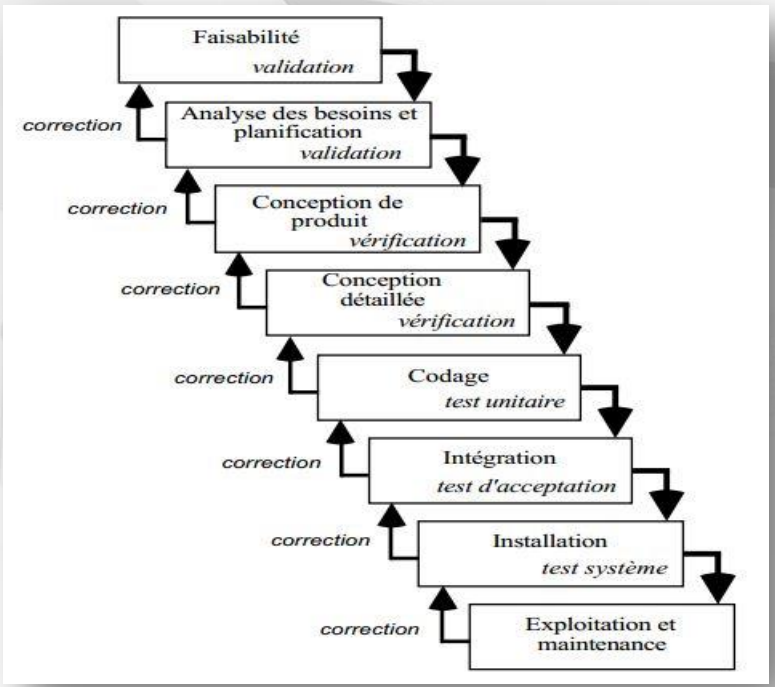
05 Les modèles de cycle de vie d’un logiciel

1. Modèle en cascade

- Date des années 70 mais reste pertinent
- Modèle linéaire avec phases séquentielles
- Vérification de chaque phase avant de passer à la suivante
- On ne peut passer à l’étape suivante tant la précédente n’est pas terminée
- Production de documents à l'issue de chaque phase
- la modification d’une étape a un impact important sur les étapes à venir
- …

05 Les modèles de cycle de vie d'un logiciel

1. Modèle en cascade



05 Les modèles de cycle de vie d'un logiciel

Avantages

- Le planning est établi à l'avance et le chef du projet sait précisément ce qui va lui être livré et quand il pourra en prendre livraison

Inconvénients

- Modèle trop séquentiel c.à.d. dure trop longtemps
- Validation trop tardive (remise en question coûteuse des phases précédentes)
- Sensibilité à l'arrivée de nouvelles exigences (refaire toutes les étapes)
- **Bien adapté lorsque les besoins sont clairement identifiés et stables**

05 Les modèles de cycle de vie d'un logiciel

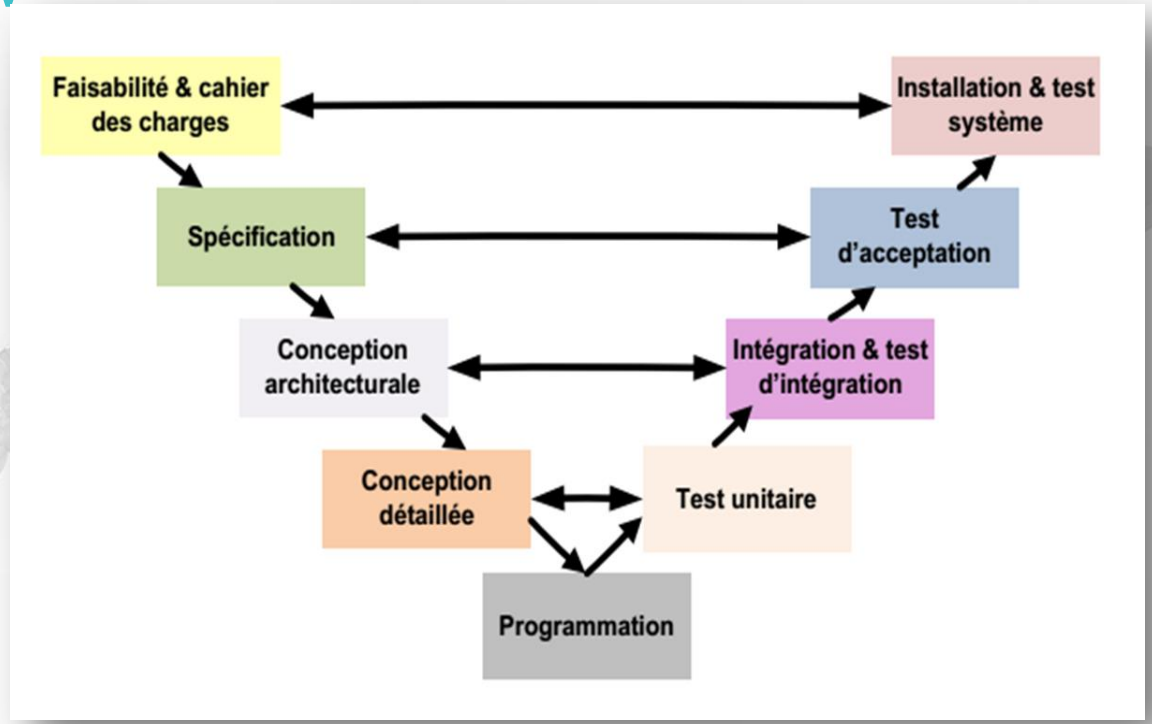
2. Modèle en V

A ce jour, le cycle de V reste le cycle de vie le plus utilisé. C'est un cycle de vie Orienté test:

- A chaque activité créative « spécification, conception, et codage » correspond une activité de vérification « validation, intégration, tests unitaires »
- Chaque phase en amant prépare la phase correspondante de vérification « la vérification est prise en compte au moment même de la création »

05 Les modèles de cycle de vie d'un logiciel

2. Modèle en V



05 Les modèles de cycle de vie d'un logiciel

Avantages

- La préparation des **dernières phases** (validation et vérification) par **les premières** (construction logiciel), permet d'éviter d'énoncer une propriété qu'il est impossible de vérifier après la réalisation
- Chaque livrable doit être testable

Inconvénients

- Le logiciel est utilisé très tard, il faut attendre longtemps pour savoir si on a construit le bon logiciel
- Ne contient pas d'activités d'analyse de risque
- **Idéal quand les besoins sont bien connus, « Quand l'analyse et la conception sont claires »**

05 Les modèles de cycle de vie d'un logiciel

3.a. Prototypage jetable

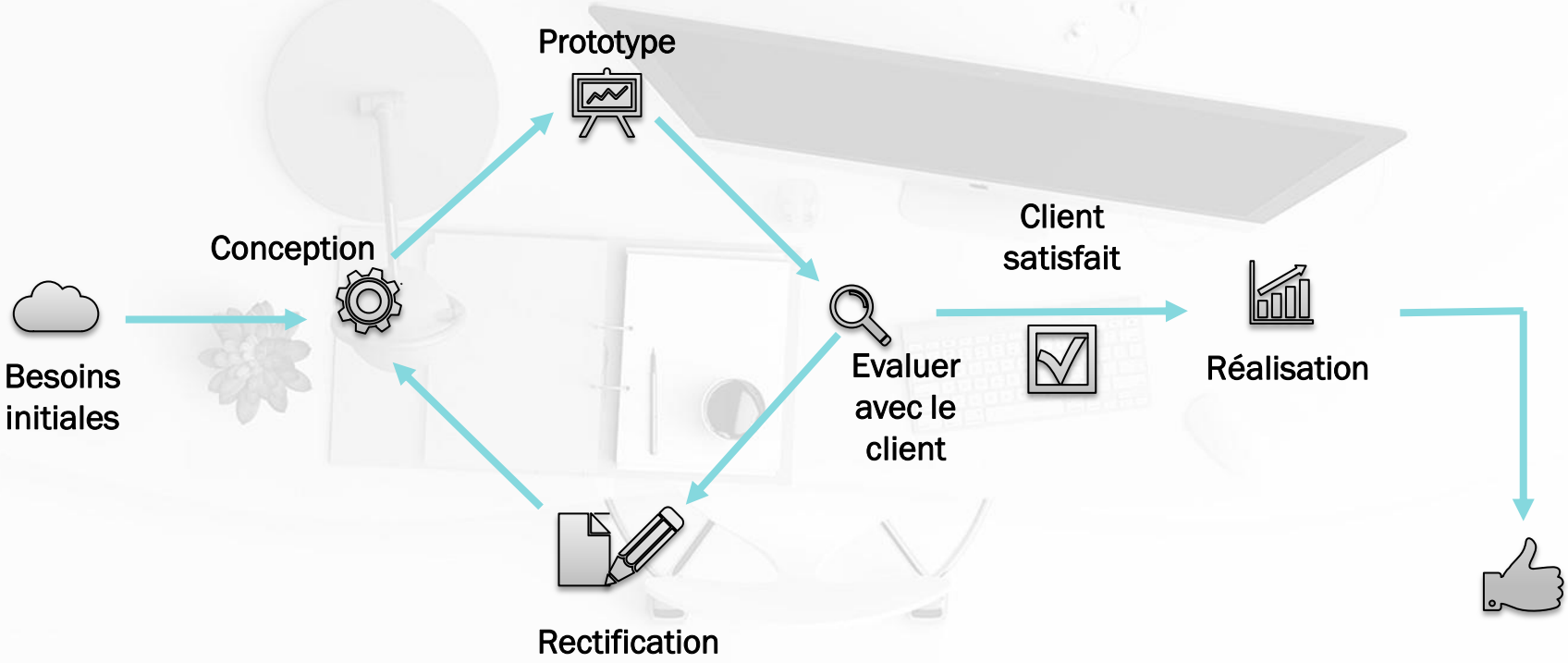
- Squelette du logiciel qui n'est crée que dans un but et dans une phase particulière de développement, si ce prototype est gardé; alors il s'appelle prototypage évolutif

3.b. Prototypage évolutif

- Le projet se fait sur plusieurs itérations
- Les développeurs construisent un prototype selon les attentes du client
- Le prototype est évalué par le client
- Le client donne son feedback
- Les développeurs adaptent le prototype selon les feedbacks et les nouvelles exigences client
- Quand le prototype satisfait le client, le code est normalisé selon les standards et les bonnes pratiques

05 Les modèles de cycle de vie d'un logiciel

3.b. Prototypage évolutif



05 Les modèles de cycle de vie d'un logiciel

Avantages

- Implication active du client, le développeur apprend directement du client
- S'adapter rapidement aux changements des besoins

Inconvénients

- Difficile d'établir un planning
- Le processus peut ne jamais s'arrêter
- **Idéal quand les besoins sont instables et/ou nécessitent des clarifications, conseillé pour de très petits projets impliquant très peu de personnes**

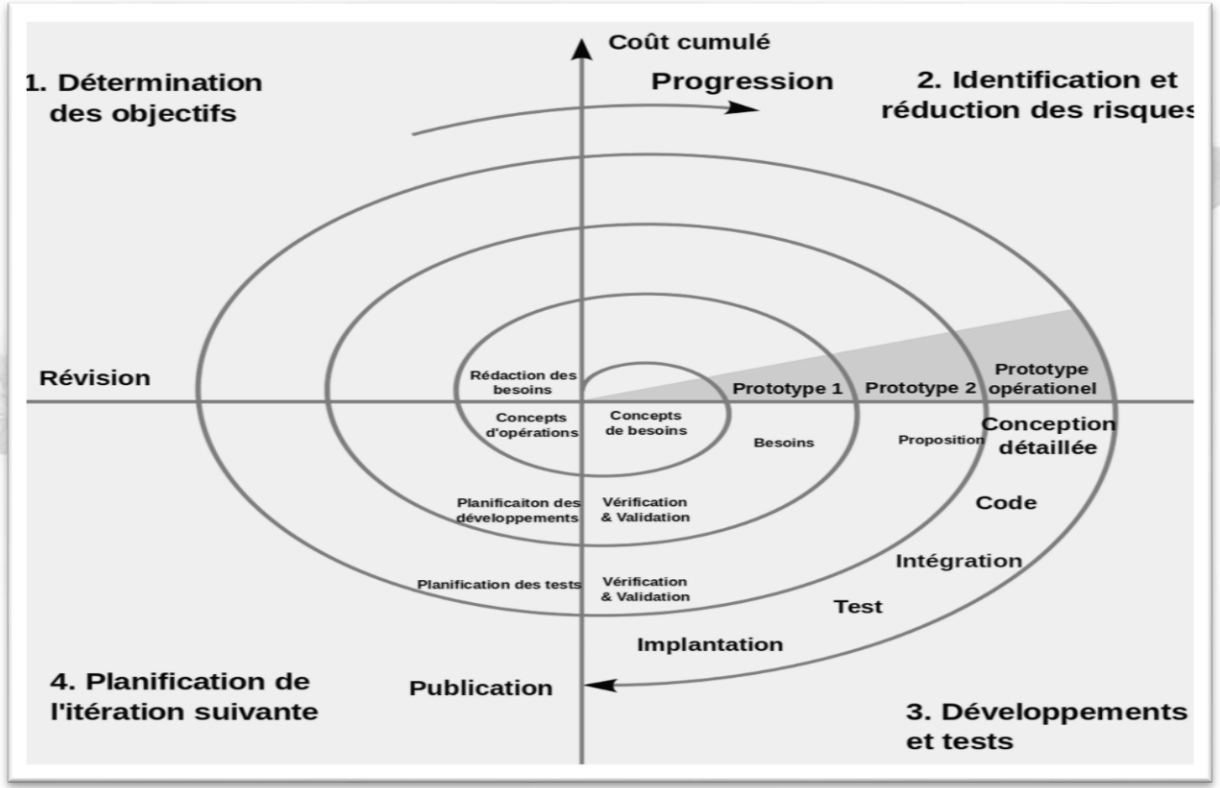
4. Modèle en spirale

Chaque cycle de la spirale se déroule en quatre phases:

- Détermination des objectifs du cycle, des alternatives pour atteindre et des contraintes à partir des résultats des cycles précédents, ou de l'analyse préliminaire des besoins
- Analyse des risque, évaluation des alternatives et éventuellement maquettage
- Développement et vérification de la solution retenue, un modèle classique peut être utilisé ici « Cascade, V, … »
- Revue des résultats et vérification du cycle suivant

05 Les modèles de cycle de vie d'un logiciel

4. Modèle en spirale



05 Les modèles de cycle de vie d’un logiciel

Avantages

- Permet d’établir le modèle de développement le plus adéquat en regard du risque
- Tous les autres modèles de développement constituent une variante de la spirale

Inconvénients

- A besoin de compétences dans l'évaluation approfondie des incertitudes et des risques associés au projet et leur réduction
- Évaluer les risques impliqués dans le projet peut prendre jusqu'à le coût et il peut être plus élevé que le coût de la construction du système.

Bibliographies

- **Introduction to Software Engineering**, Ronald J. Leach, CRC Press, Taylor & Francis Group, 2016
  https://1library.net/document/rz3r50mz-introduction-to-software-engineering-pdf.html

- **Génie logiciel, principes, méthodes et techniques,** Presses Polytechniques et Universitaires Romande, 1996,
  https://www.leslibraires.fr/livre/596107-genie-logiciel-principes-methodes-et-techniques-alfred-strohmeier-didier-buchs--presses-polytechniques-et-universitaires-romandes