# Cours de Génie Logiciel

**Introduction au Génie logiciel et à la modélisation**

**(Chapitre 1 & Chapitre 2)**

**2022-2023**

**Dr. Kouah S.**

# Chapitre 1: Introduction

1. Définitions et objectifs
2. Principes du Génie Logiciel
3. Qualités attendues d’un logiciel
4. Cycle de vie d’un logiciel
5. Modèles de cycle de vie d’un logiciel

# 1. Définitions et objectifs

- Un logiciel (Software) est un **ensemble de programmes**, de **procédés**, de **règles**, de **documents** relatifs au fonctionnement d'un ensemble de traitement de l'information.
- Logiciel = ensemble de programmes coopérant ensemble via des interfaces et de procédures dédiés à la réalisation d'un ensemble de besoins de ses utilisateurs.
- **Logiciel : Pas uniquement du code informatique (code+ structures de données + documentation + jeux de test + informations de configuration…etc.)**

# 1. Définitions et objectifs

- **Constat du développement de logiciels en fin des années 60 :**

Les **techniques hardware** ont atteint un degré de maturité relatif:

- Le matériel est relativement fiable
- Le marché est standardisé

La place des logiciels dans les systèmes complexes augmente régulièrement, or, les producteurs de logiciel ne respectent que rarement les coûts ou les délais de développement, quand ce ne sont pas les besoins eux même de leurs clients!

- Les problèmes liés à l'informatique sont essentiellement des problèmes de Logiciel.

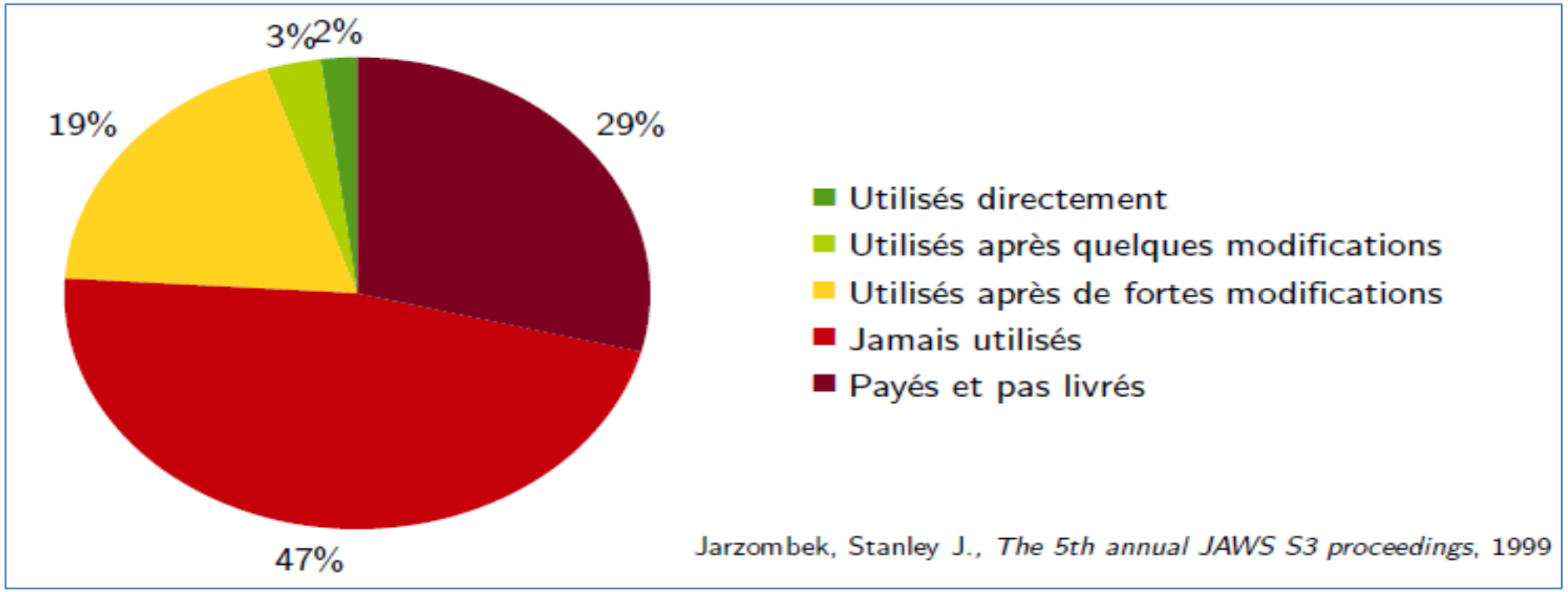
**Crise de Logiciel !!!**

## Crise de Logiciel

- ✓ Délais de livraison non respectés
- ✓ Cout prévu non respectés
- ✓ Ne répond pas aux besoins de l'utilisateur ou du client
- ✓ Difficile à utiliser, à maintenir, et à faire évoluer

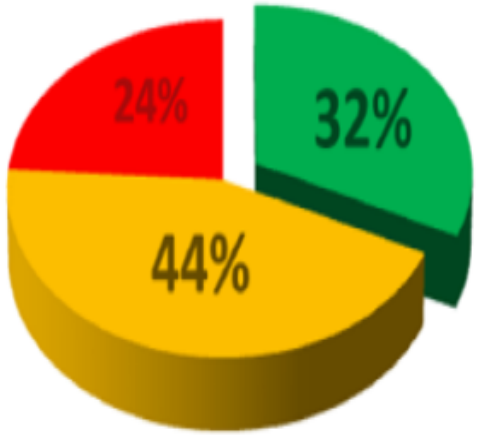
**Selon une Étude du DoD (Department of Defense des États-Unis), 70 % des projets initiés sont abandonnés avant leur achèvement.**

**Etude DoD en 1995, sur** les logiciels produits dans le cadre de **9 gros projets militaires**.



Source: The Standish Group

The Chaos Report 2009




**Successful**: on-time, on-budget, and with all features and functions as defined in the initial scope;

**Challenged**: late, over budget, and/or with less and functions than defined in the initial scope;

**Failed**: cancelled prior to completion, or delivered but never used.

Plus la complexité et la taille sont importantes, plus le risque d'echec est fort…

| | | COMPLEXITY | | | | |
|---|---|---|---|---|---|---|
| | | C1 | C2 | C3 | C4 | C5 |
| SIZE | S1 | 100 | 250 | 400 | 550 | 700 |
| | S2 | 175 | 325 | 475 | 625 | 775 |
| | S3 | 250 | 400 | 550 | 700 | 850 |
| | S4 | 325 | 475 | 625 | 775 | 625 |
| | S5 | 400 | 550 | 700 | 850 | 1000 |

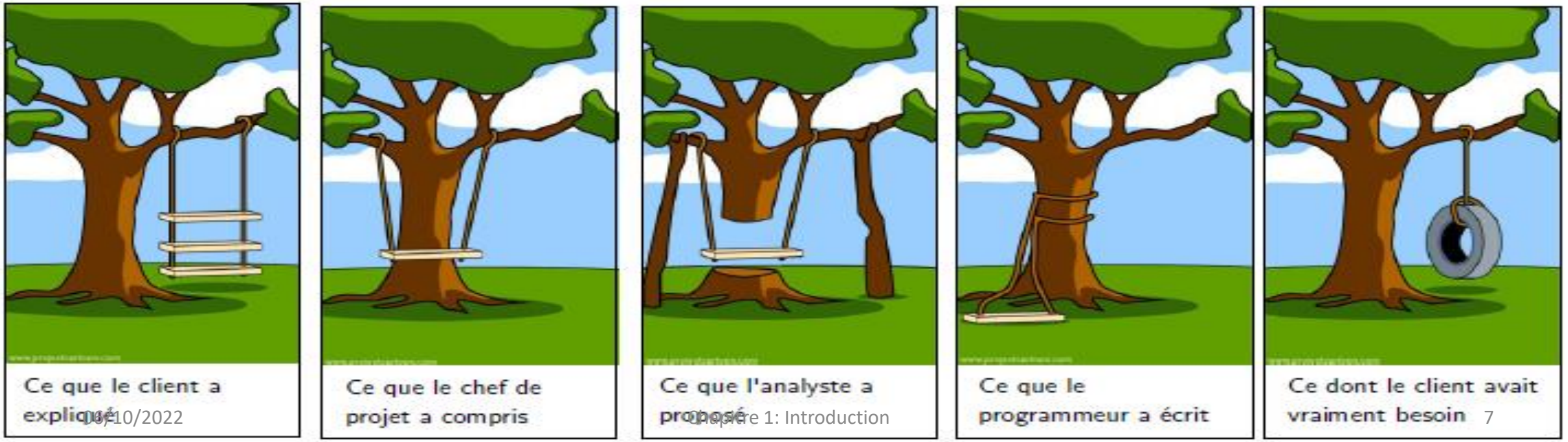
SUCCESSFUL
CHALLENGED
FAILED

Le **génie logiciel** est né de ce constat, en 1968 dans une conférence sur le thème « **La crise du Logiciel** »

# Quelques Causes de la crise de logiciel

- Taille et complexité des logiciels.
- Diversification des domaines d'application.
- Absence de cadre méthodologique, de méthodes et d'outils de développement.
- Mauvaise compréhension des besoins.
- Absence de métrologie de la qualité des processus et des produits logiciels.
- Etc

Il y avait donc une nécessité de passer d'une démarche artisanale à une discipline d'ingénieur.



# 1. Définitions et objectifs

## Génie logiciel (GL) =

- Génie =science de l'ingénieur à l'instar des génies aéronautiques, maritime, civil...
- Logiciel = objet manufacturé complexe avec ses propres techniques de fabrication.

# 1. Définitions et objectifs

## Génie logiciel (GL) = Génie (l'art de spécifier, réaliser, faire évoluer) + Logiciel

1. Ensemble des méthodes, des techniques et d'outils dédiés à la conception, au développement et à la maintenance des systèmes informatiques.

2. Le génie logiciel est un domaine des sciences de l'ingénieur dont la finalité est la conception, la fabrication, et la maintenance des *systèmes informatiques complexes* sûrs *et de* qualité.

# 1. Définitions et objectifs

## Génie logiciel (GL) = Génie (l'art de spécifier, réaliser, faire évoluer) + Logiciel

3. Le génie logiciel est donc l'art de spécifier, de concevoir, de réaliser, et de faire évoluer, avec des moyens et dans des délais raisonnables, des programmes, des documentations et des procédures de qualité en vue d'utiliser un ordinateur pour résoudre certains problèmes. (Gaudel, 1996)

# 1. Définitions et objectifs

**Objectif du Génie logiciel:** Le GL se préoccupe des procédés de fabrication des logiciels en s'assurant que les **4 critères** suivants soient satisfaits:

- ✓ **Coût** prévu initialement (**Cost**)
- ✓ **Qualité** attendue (**Quality**)
- ✓ Répond aux besoins des utilisateurs (correction **fonctionnelle**) (**Functionnality**)
- ✓ **Délais** prévus initialement (**Delay**).

**Règle de CQFD ( Cost + Quality + Functionnality + Delay)**

# 1. Définitions et objectifs

- Autrement dit:

✓ Avoir une **méthodologie** bien définie qui tient compte du **cycle de vie** du logiciel

✓ Avoir un **ensemble d'éléments** qui **documentent** chaque étapes du **cycle de vie** et montrent les traces d'une étapes à l'autre.

✓ Un ensemble de **jalons** (repères) qui peuvent être révisés à des intervalles réguliers du **cycle de vie** de logiciel.

# 2. Principes du Génie Logiciel

Le GL est basé sur plusieurs grands principes (de bon sens), nous citons à titre d'exemples les principes suivants:

- Rigueur
- Généralisation
- Structuration
- Abstraction
- Modularité
- Documentation
- Vérification
- Anticipation du changement
- Construction incrémentale (raffinement)

# 2. Principes du Génie Logiciel

- Rigueur: le  logiciel **doit offrir une solution précise** → Développement sérieux, soigné, non approximatif.
- Généralisation : regroupement d'un ensemble de fonctionnalités semblables en une fonctionnalité paramétrable (généricité, héritage). → Solution pourra être réutilisée.
- Structuration : façon de décomposer un logiciel (utilisation d'une méthode **bottom-up** ou **top-down**)
- Abstraction (séparation des aspects): mécanisme qui permet de présenter un contexte en exprimant les éléments pertinents et en omettant ceux qui ne le sont pas (considération des aspects jugés importants d'un système).
- Modularité (séparation des parties logiques): Principe qui consiste à décomposer un système en sous-systèmes plus simples  afin de maîtriser la complexité.

# 2. Principes du Génie Logiciel

- Documentation : gestion des documents incluant leur identification, acquisition, production, stockage et distribution
- Vérification : détermination du respect des spécifications établies sur la base des besoins identifiés dans la phase précédente de développement.
- Anticipation du changement: Concevoir un système suffisamment riche pour que l'on puisse le modifier incrémentalement.
- Construction incrémentale (raffinement) : Consiste à construire une solution étape par étape en s'approchant de plus en plus du but. (Ex.: réaliser les fonctions essentielles d'un système puis ajouter les aspects secondaires).
- …etc.

# Qualités attendues d'un logiciel

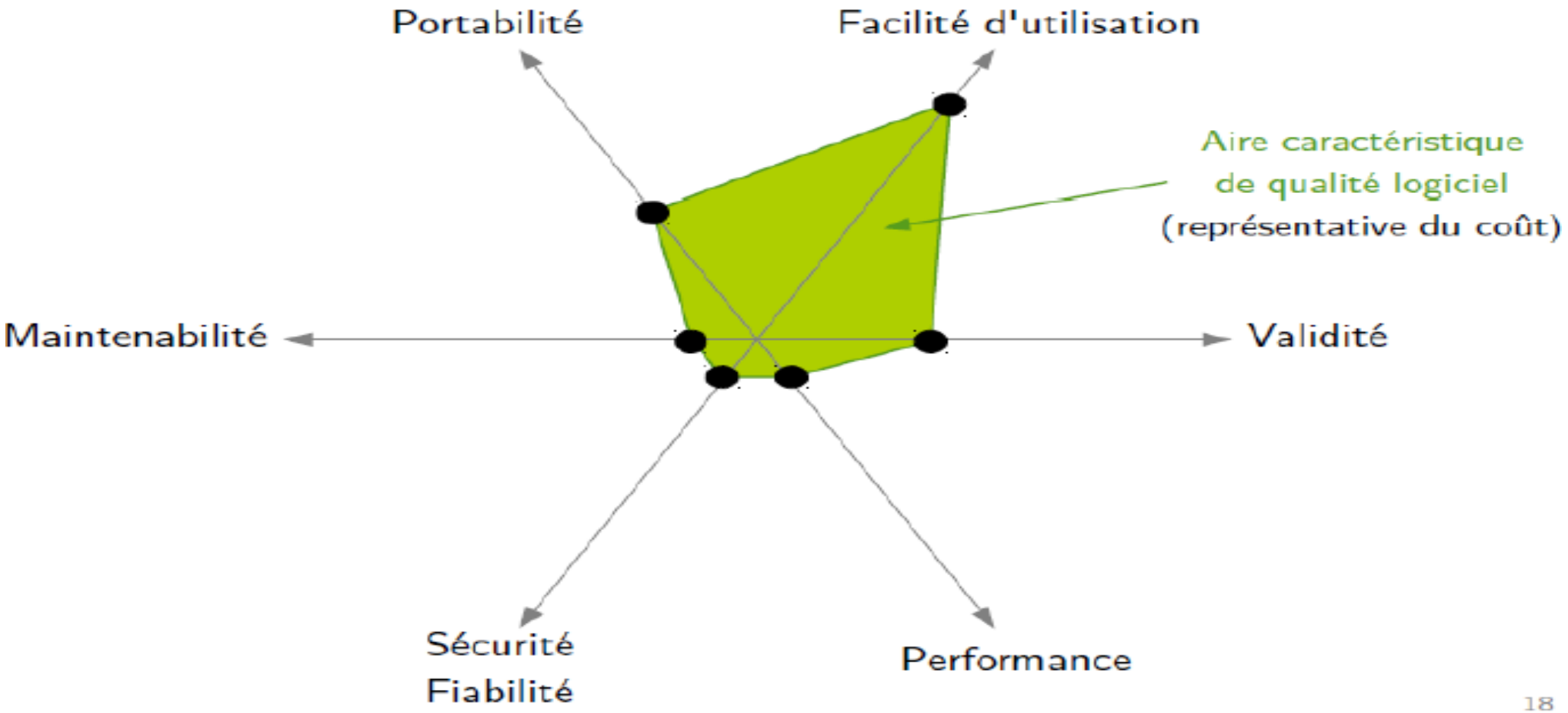
- En génie logiciel, divers travaux ont menés à la définition de la **qualité du logiciel** en termes de facteurs, qui dépendent, entre autres, du domaine d'application et des outils utilisés. Parmi ces derniers nous pouvons citer :
- **Validité :** remplir exactement ses fonctions, définies par le cahier des charges et les spécifications.
- **Fiabilité ou robustesse :** fonctionner dans des conditions anormales.
- **Facilité d'emploi :** facilité d'apprentissage, d'utilisation, de préparation des données, d'interprétation des erreurs et de rattrapage en cas d'erreur d'utilisation.

# Qualités attendues d'un logiciel

- **Compatibilité :** facilité avec laquelle un logiciel peut être combiné avec d'autres logiciels.
- **Efficacité :** Utilisation optimales des ressources matérielles.
- **Portabilité :** facilité avec laquelle un logiciel peut être transféré sous différents environnements matériels et logiciels.
- **Extensibilité (maintenance) :** facilité avec laquelle un logiciel se prête à sa maintenance, c'est-à-dire à une modification ou à une extension des fonctions qui lui sont demandées.
- **Réutilisabilité :** peut être réutilisé, en tout ou en partie, dans de nouvelles applications.
- **Vérifiabilité :** facilité de préparation des procédures de test.
- **Intégrité :** protéger son code et ses données contre des accès non autorisés.

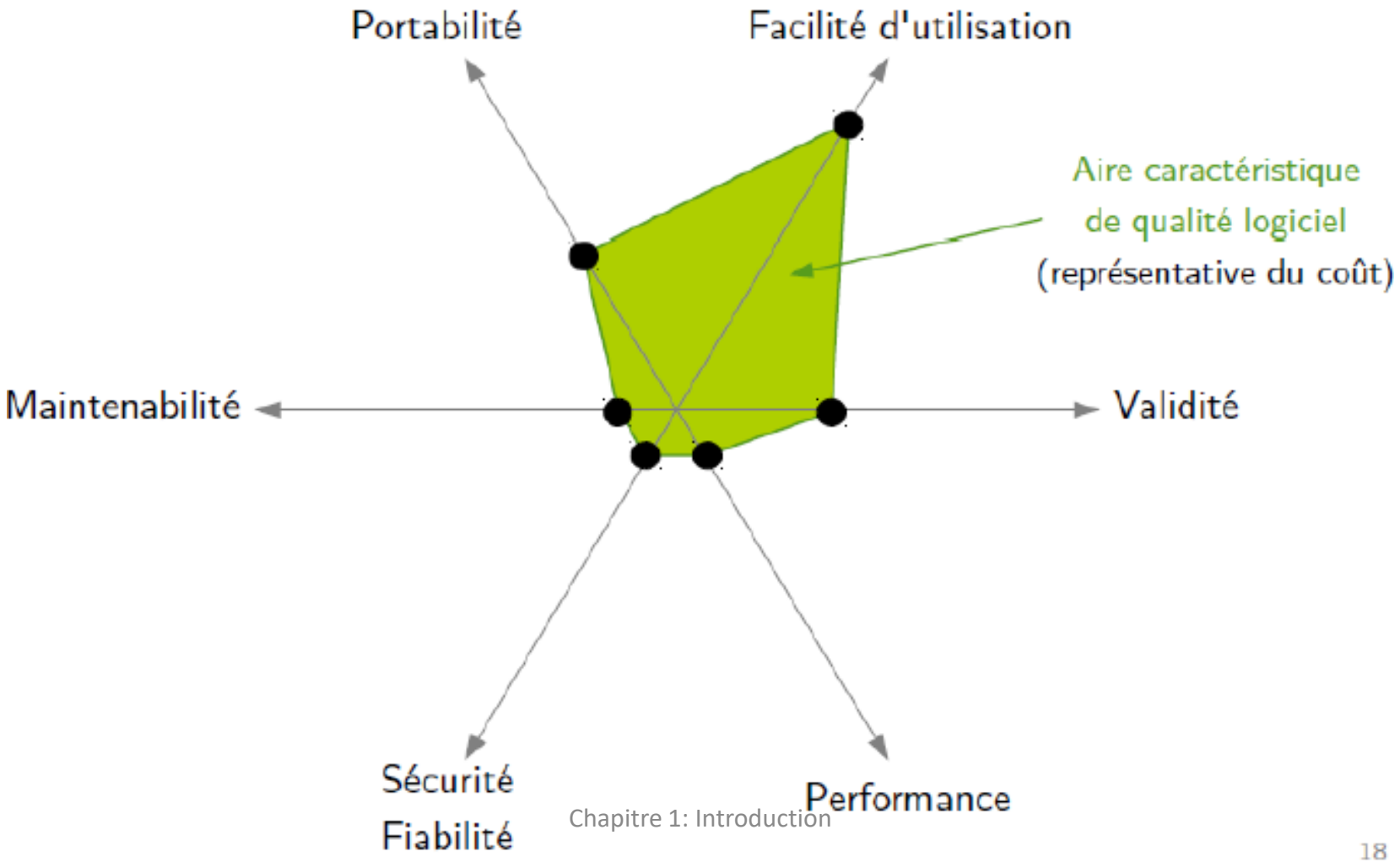
# Quelle (s) qualité (s)?

Contrôleur de télécommande



# Quelle (s) qualité (s)?

Jeu Vidéo



# Quelle (s) qualité (s)?

- Certaines qualités au détriment des autres.

- Choix dépend de plusieurs critères, tel que la nature de l'application et les exigences d'utilisateur.

# 4.cycle de vie d’un logiciel

- Le **processus de développement** influence la **qualité du logiciel**. Donc, pour obtenir un logiciel de qualité, il faut en maîtriser son processus de développement.
- Un **processus de développement logiciel** est un ensemble *structuré* d’activités qui conduisent à la production d’un logiciel. **La vie d'un logiciel** est composée de différentes étapes, depuis sa **naissance** jusqu’à sa **mort** ou **retrait du service**. Donc, un

**Cycle de vie du logiciel =**

- Naissance du système (l’arrive d’une demande d’un client).
- Développement (Cycle de développement)
- Utilisation et Maintenance.
- Mort (Démontage : logiciel n’est plus désiré ou ne répond plus aux besoins évolutifs)

# 4.cycle de vie d’un logiciel

- Le **cycle de vie du logiciel** est la période de temps s’étalant du début à la fin du processus du logiciel. Il **commence** donc avec la **proposition ou la décision de développer un logiciel** et **se termine par sa mise hors service**.

# 4.cycle de vie d'un logiciel

Il existe de nombreux modèles de cycle de vie, les plus courants comportent les activités suivantes :

Analyse des besoins
Spécification Globale
Conception
Programmation (Codage)
Intégration
Installation
Maintenance
Vérification et Validation
Test

Pour chaque activité on a **Utilisation et production de documents**

# **Analyse des besoins** :

**-** Comprendre les **besoins du client (Déterminer les <u>fonctionnalités</u> du système et ses <u>contraintes</u> d'utilisation)**

- **Données**: Fournies par les experts du domaine d'application et les futurs utilisateurs.

- **Résultat** = **<u>Cahier de charges</u>**

**<u>Cahier de charges :</u>** ensemble de documents décrivant les aspects pertinents de l'environnement du futur système, son rôle, sa future utilisation,  dans un langage compréhensible par les utilisateurs et les développeurs

# Spécification Globale

- Établir une description claire de **ce que doit faire** le système (**exigences fonctionnelles et non fonctionnelles, …**). **Décrire le QUOI.**
- **Données:** Cahier de charges + considérations techniques et de faisabilité informatique.
- **Résultat**: **Document de spécification technique des besoins**

**Document de spécification technique des besoins :** il couvre toutes les situations, plus compréhensible que le cahier de charges.

**Remarque:** Généralement, il est parfois difficile, voire impossible de séparer les activités d'analyse des besoins et de spécification. Elles apparaissent regroupées.

**Conception** : **Élaborer une solution concrète** réalisant la spécification (Description très proche d'un programme), généralement décomposée en deux phases successives :

1. **conception générale**, **conception globale**, **conception préliminaire** ou **conception architecturale:**
2. **Conception détaillée**

- **Résultat**: **document de conception détaillée**.

1. **conception générale**, **conception globale**, **conception préliminaire** ou **conception architecturale:**

Description architecturale en terme de composants (les composants, leurs fonctionnalités les, interfaces, les interactions entre composants, le choix d'une architecture logicielle). Le résultat de cette démarche est un **document de conception générale**.

- **Données**: Document de spécification technique
- **Résultat**: Document de conception générale

**2. Conception détaillée** : Raffine la conception générale. Elle commence par décomposer les composants découvertes lors de la conception générale en composants plus élémentaires. Cette décomposition doit être poursuivie jusqu'au niveau où les composants sont faciles à implémenter et à tester (des composants logiciels élémentaires).

- dépend fortement du langage de programmation. Il faut ensuite décrire chaque composant logiciel en détail : son interface, les algorithmes utilisés, le traitement des erreurs, ses performances, etc.

**Programmation** : Lors de cette phase, **la conception détaillée** est **traduite dans un langage de programmation**.

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

**<u>Intégration</u>: Assembler tout ou parties des composants** pour créer un **système exécutable**. Cette activité utilise la **gestion de configurations** pour assembler des versions cohérentes de chaque composant.

**Installation:** Après avoir intégré le logiciel, on peut l'installer dans son environnement d'exploitation, ou dans un environnement qui simule cet environnement d'exploitation, et le tester pour s'assurer qu'il se comporte comme requis dans la spécification élaborée.

# 4.cycle de vie d'un logiciel

**Maintenance**

- Après l'installation suit la **phase d'exploitation et de maintenance**. Le logiciel est maintenant employé dans son environnement opérationnel, son comportement est surveillé et, si nécessaire, il est modifié. Cette dernière activité s'appelle la maintenance du logiciel (software maintenance).
- Il peut être nécessaire de modifier le logiciel pour corriger des défauts, pour améliorer ses performances ou autres caractéristiques, pour adapter le logiciel à un nouvel environnement ou pour répondre à des nouveaux besoins ou à des besoins modifiés. On peut donc distinguer entre la maintenance corrective, la maintenance perfective et la maintenance adaptative.

## Validation et Vérification:

**Validation**: A t on décrit le bon système ?(Building the right Product?)? Bon signifie ici qui répond aux attentes des utilisateurs et aux contraintes de l'environnement.

**Vérification**: Le développement est-t-il correct para port à sa spécification?( Building the product right?).

**Test: (plusieurs activités de test)**

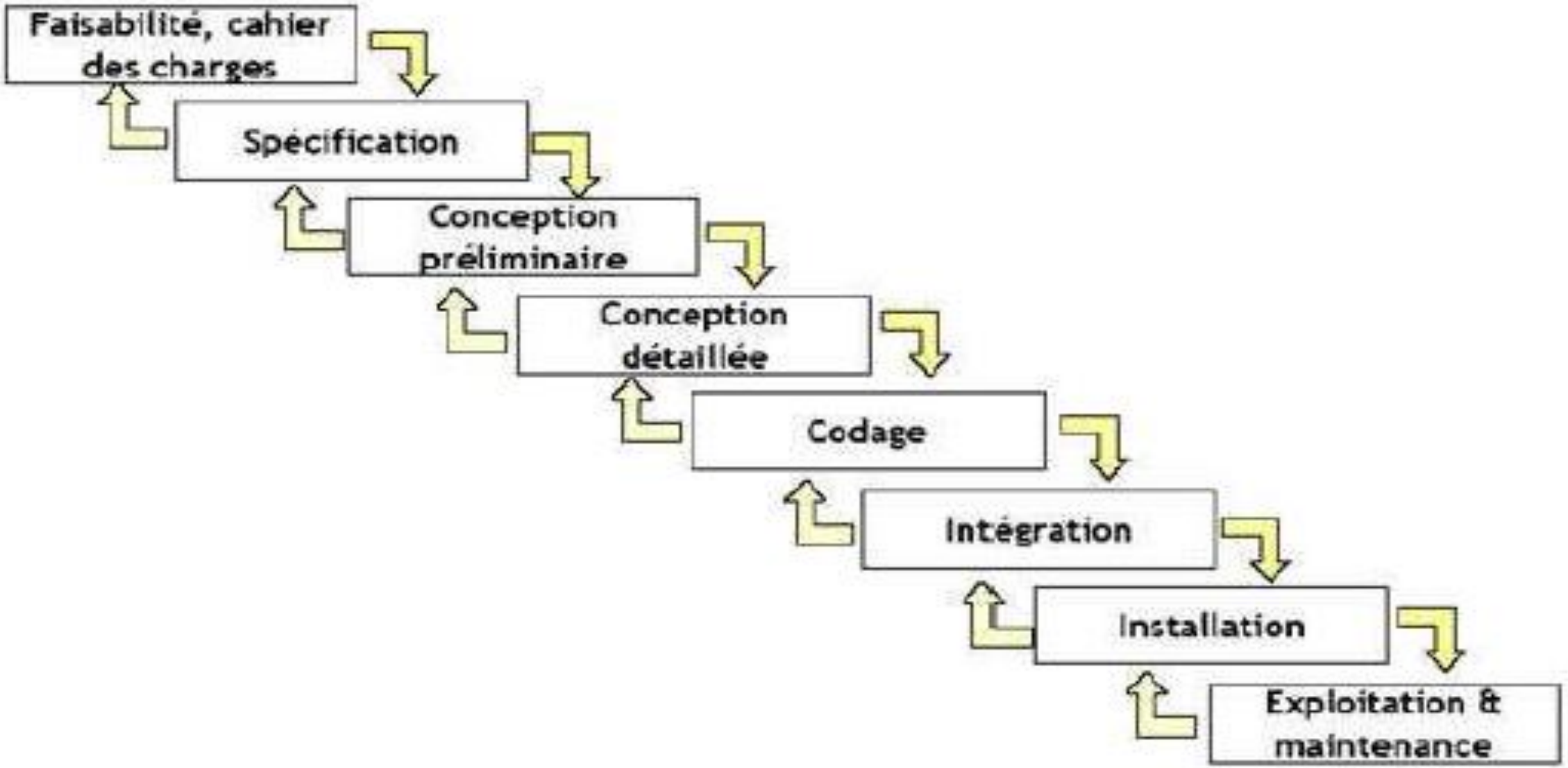
- **Test unitaires:** Tester les composants isolés.
- **Test d'intégration:** Tester un ensemble de composants assemblés.
- **Test Système:** tester le système sur son futur site d'exploitation.
- **Test de non régression:** Tester le système suite à une modification.

# 5. Modèles de cycle de vie d'un logiciel

Il existe différents modèles de processus logiciel qui organisent de façon différente ces activités, tels que :

- Modèle en cascade
- Modèle en V
- En Spirale
- Par prototypage
- Par incrément
- En Y
- En W
- …Etc

# Le cycle de vie en " Cascade " (waterfall model)



# Le cycle de vie en " Cascade " (Waterfall model)

Le cycle de vie en « cascade » date de 1970, il est l'œuvre de Royce.

- **Cycle de vie linéaire** sans aucune évaluation entre le début du projet et la validation.

- Le projet est découpé en phases successives dans le temps et à chaque phase correspond une activité principale bien précise produisant un certain nombre de livrables.

- On ne passe à l'étape suivante que si les résultats de l'étape précédente sont jugés satisfaisants.

- L'activité d'une étape se réalise avec les résultats fournis par l'étape précédente ; ainsi, chaque étape sert de contrôle du travail effectué lors de l'étape précédente.

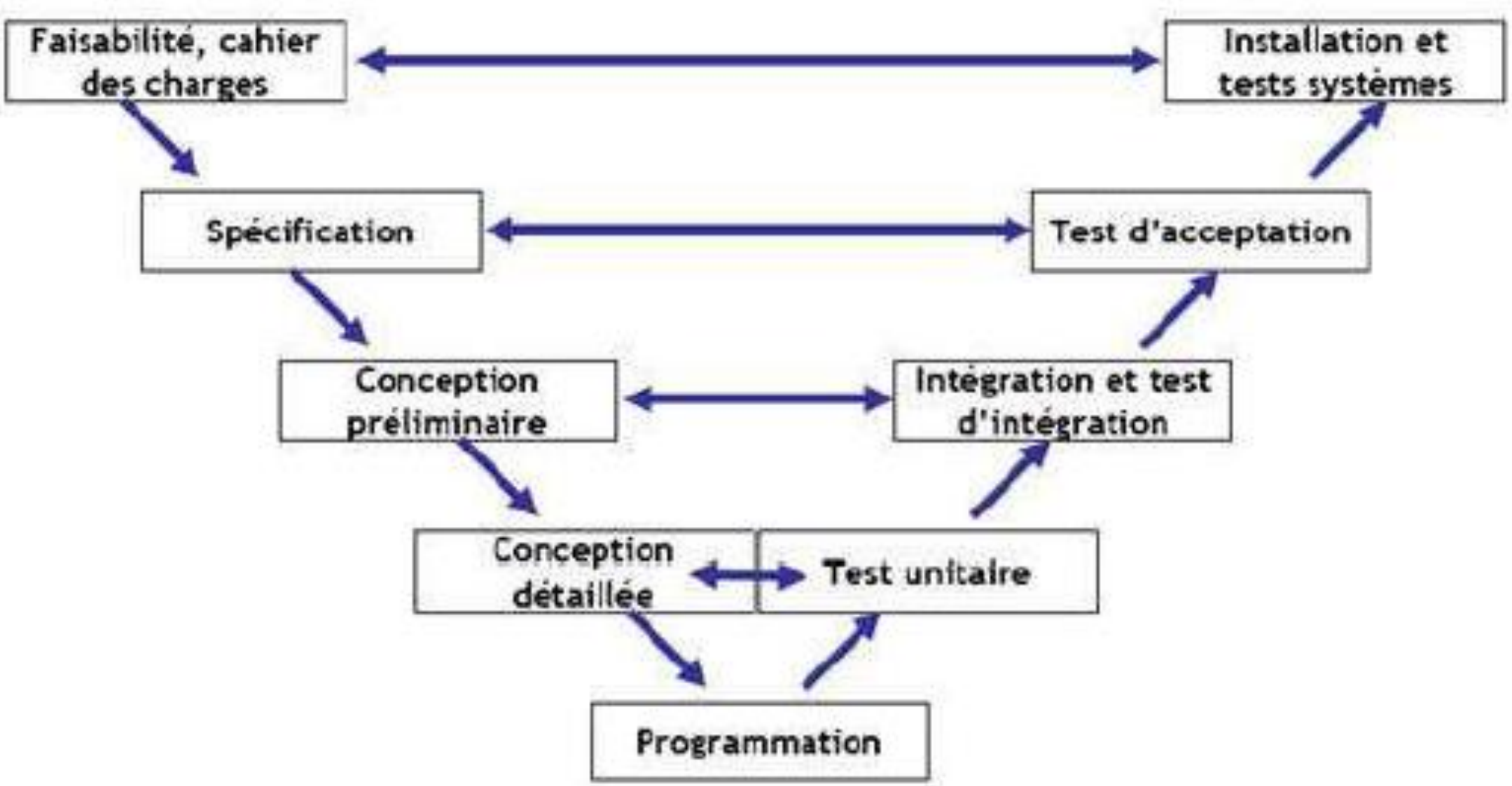
- Chaque phase ne peut remettre en cause que la phase précédente ce qui, dans la pratique, s'avère insuffisant.

# Le cycle de vie en " Cascade " (Waterfall model)

- les erreurs de spécifications sont généralement détectées au moment des tests, voire au moment de la livraison du logiciel à l'utilisateur. Leur correction nécessite alors de reprendre toutes les phases du processus.

- Ce modèle est **mieux adapté aux petits projets** ou à ceux dont **les spécifications sont bien connues et fixes.**

# Le cycle de vie en " V "



# Le cycle de vie en " V "

– Dérivé du modèle de la cascade.

– le début du processus de développement conditionne ses dernières étapes.

– Adapté aux **projets de taille et de complexité moyenne**.

– La première branche correspond à un modèle en cascade classique. Toute description d'un composant est accompagnée de définitions de tests.

Avec les **jeux de tests préparés dans la première branche**, les **étapes de la deuxième branche peuvent être mieux préparées et planifiées**.

# Le cycle de vie en " V "

– La **seconde branche correspond à des tests effectifs** effectués sur des composants réalisés.

– **L'intégration** est ensuite réalisée jusqu'à l'obtention du système logiciel final.

– L'avantage d'un tel modèle est d'éviter d'énoncer une propriété qu'il est impossible de vérifier objectivement une fois le logiciel réalisé.

– le cycle en V est normalisé, il est largement utilisé, notamment en informatique industrielle et télécoms

# Le cycle de vie en spirale