



UNIVERSITÉ  
DE MONTPELLIER



# Introduction

**Bases de la Conception Orientée Objet - AS**

**Nadjib Lazaar ([nadjib.lazaar@umontpellier.fr](mailto:nadjib.lazaar@umontpellier.fr))**

# Organisation

## Plan

- PARTIE 1 : COO
  - Introduction au Génie Logiciel
  - Modélisation UML
  - Les concepts de base de l'orienté objet
  - La modélisation des exigences
  - La modélisation de la structure
  - La modélisation de la dynamique
  - La modélisation du cycle de vie des objets
- PARTIE 2 : COO avancée
  - Principes SOLID
  - Patrons de conception de construction
  - Patrons de conception de structuration
  - Patrons de conception de comportement

# Organisation

## EDT et évaluation

- **Dis séances de cours/TD/TP (1h + 1h + 1h)**
  - Du 08/09/2021 au 17/12/2021
  - TP sur machine avec StartUML / Java
  - Dépôts Github : [github.com/IUTInfoMontpSete-AS-GL](https://github.com/IUTInfoMontpSete-AS-GL)
- **Note TD/TP** : 20% de la note finale
- **Projet à rendre** : 20% de la note finale
- **Contrôle final sur table** : 60% de la note finale
- **Cours et TD/TP disponibles sur MOODLE (AS/GL)**

# Logiciel - Software

## Définition

- Ensemble de **programmes**, des **moyens d'utilisation** et de la **documentation** nécessaires au fonctionnement d'un processus de traitement automatique de l'information.

# Logiciel - Software

## Définition

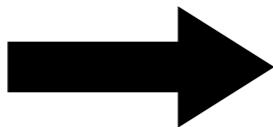
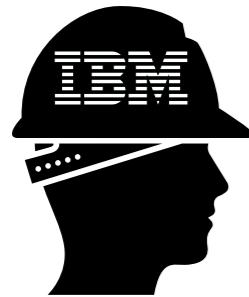
- Ensemble de **programmes**, des **moyens d'utilisation** et de la **documentation** nécessaires au fonctionnement d'un processus de traitement automatique de l'information.

**LOGICIEL = PROGRAMMES + UTILISATION + DOCUMENTATION**

# « Crise du logiciel »

Fin des années 1960

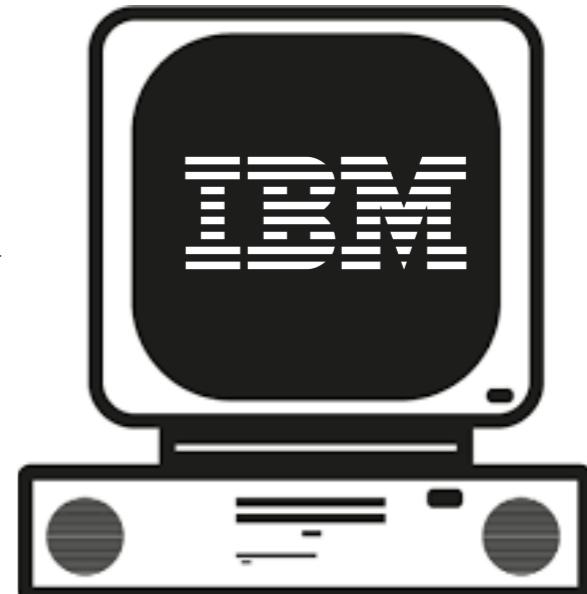
## INFORMATIQUE



```
3800000000400526 <main>:
400526: 55 push rbp
400527: 48 89 e5 mov rbp,rsp
40052a: 48 83 ec 20 sub rsp,0x20
40052e: 89 7d ec mov DWORD PTR [rbp-0x14],edi
400531: 48 89 75 e0 mov QWORD PTR [rbp-0x20],rsi
400535: c7 45 fc 00 00 00 00 mov DWORD PTR [rbp-0x4],edi
40053c: eb 0e jmp 40054c <main+0x26>
40053e: bf e4 05 40 00 mov edi,0x4005e4
400543: e8 b8 fe ff ff call 400400 <puts@plt>
400548: 83 45 fc 01 add DWORD PTR [rbp-0x4],0x1
40054c: 83 7d fc 09 cmp DWORD PTR [rbp-0x4],0x9
400550: 7e ec jle 40053a <main+0x18>
400552: b8 00 00 00 00 mov eax,0x0
400557: c9 leave
400558: c3 ret
400559: 0f 1f 80 00 00 00 00 nop DWORD PTR [rax+0x0]

3800000000400560 <_llbc_csu_init>:
400560: 41 57 push r15
400562: 41 56 push r14
```

..

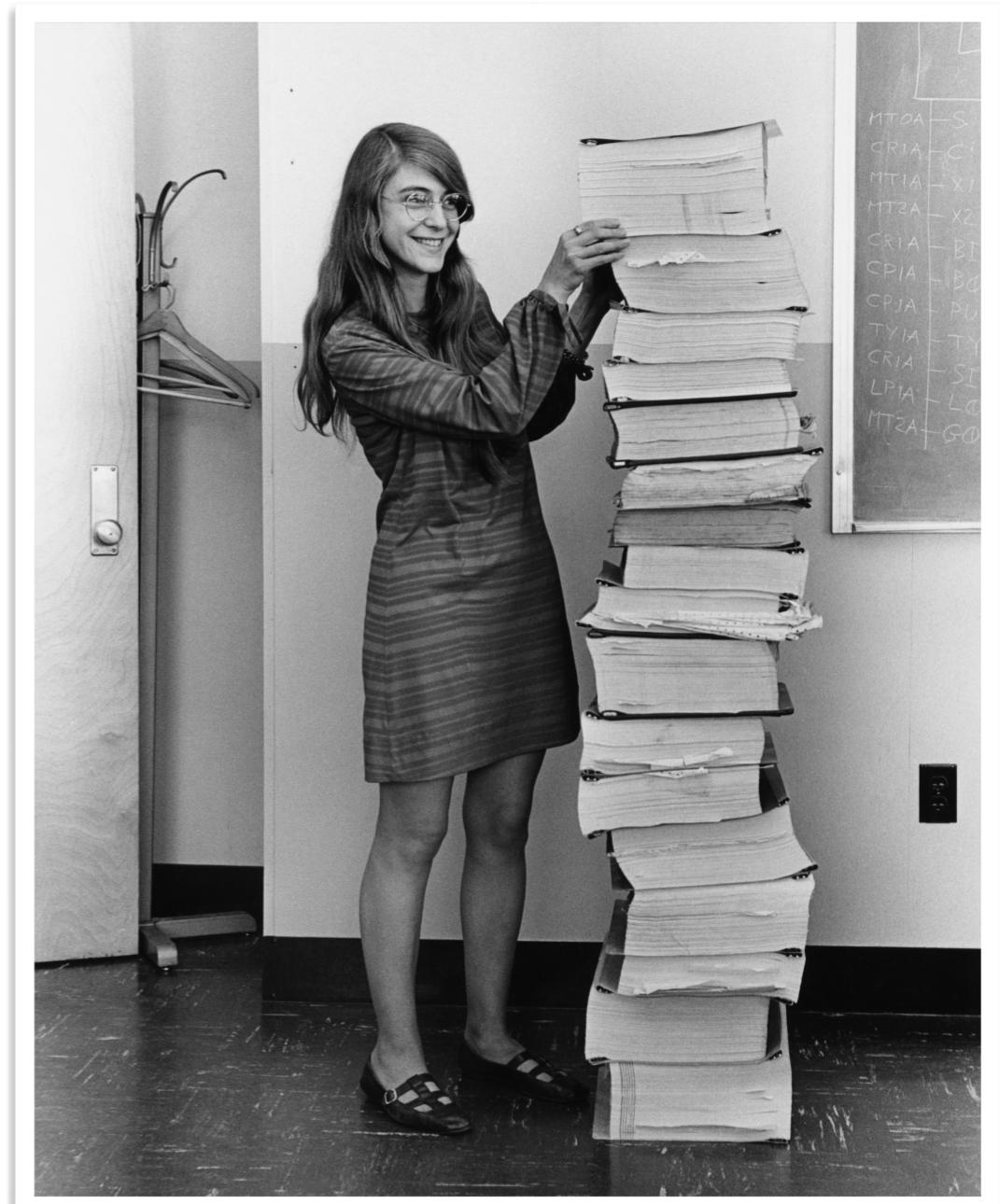


- L'informatique : du calcul scientifique aux champs d'application diverses
- Croissance exponentielle de la taille des logiciels

# « Crise du logiciel »

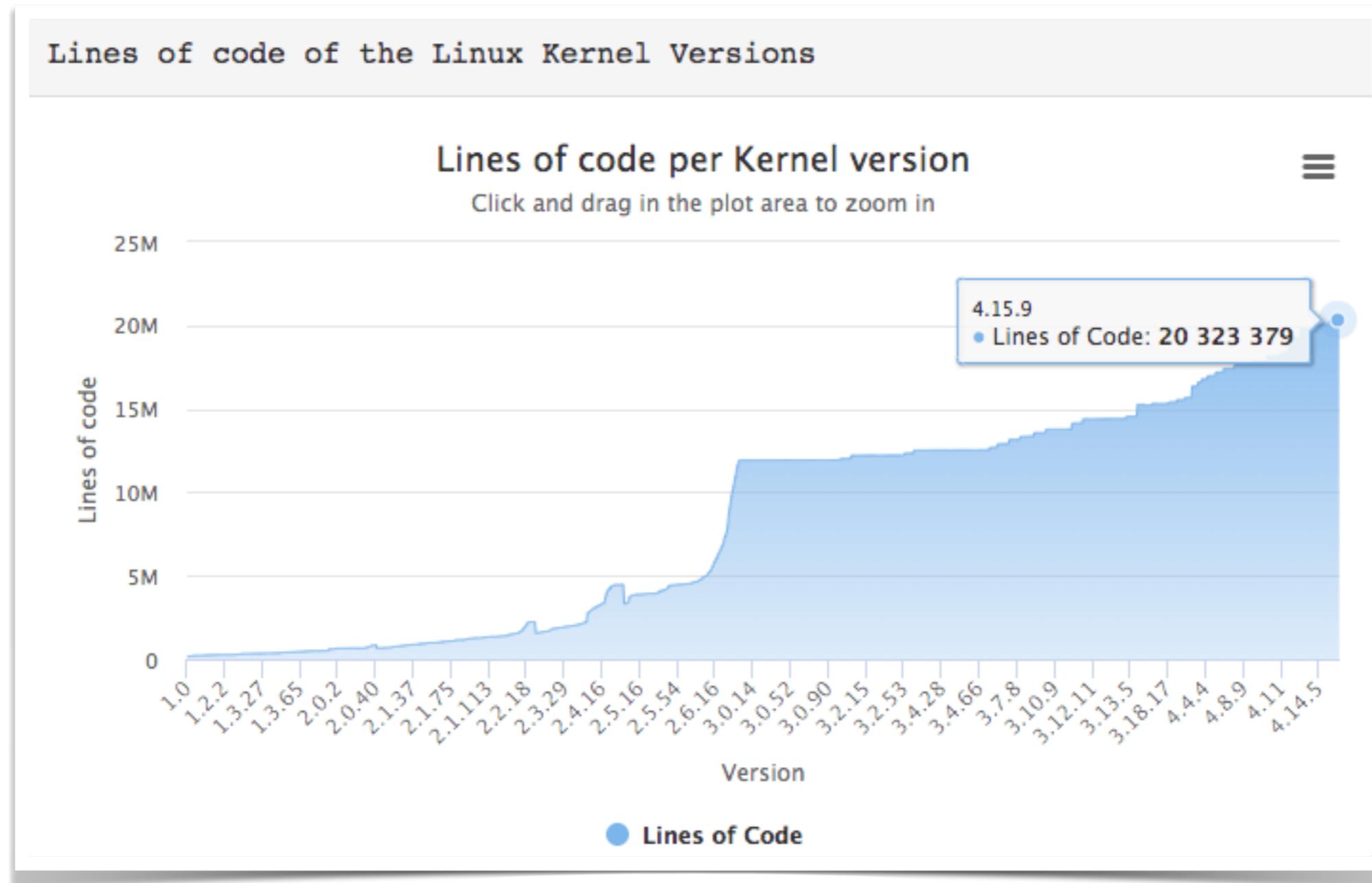
## Fin des années 1960

- Margaret Hamilton debout à côté du code source du logiciel qu'elle et son équipe du MIT ont produit pour le projet Apollo (Apollo 11 Guidance Computer (~60.000 lignes), 1969)
- « *When I first got into it, nobody knew what it was that we were doing. It was like the Wild West. There was no course in it. They didn't teach it* » **Margaret Hamilton**



# « Crise du logiciel »

Fin des années 1960



# Génie Logiciel - Software Engineering

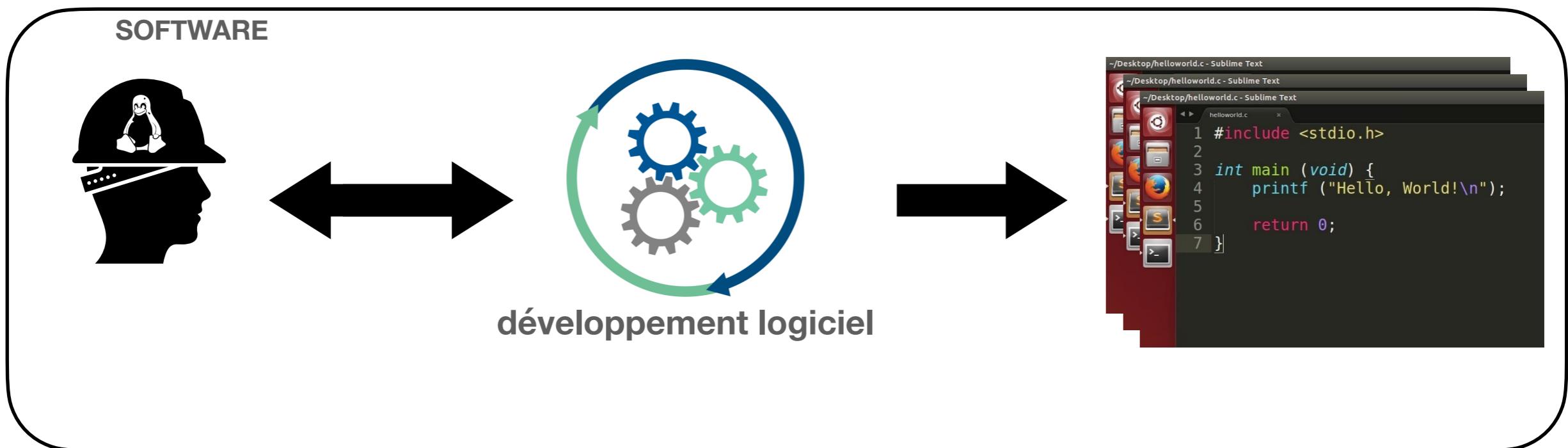
## Idée



- « Working conference on Software Engineering » OTAN, 1968  
(F. L Bauer Pr. Univ Munich)
- Appliquer les méthodes classiques d'ingénierie au domaine du logiciel
- L'**ingénierie (Génie)** est l'ensemble des tâches qui mènent de l'analyse et la conception, à la construction et à la mise en service d'une installation technique ou industrielle.
  - Ex : Génie civil, naval, génétique, mécanique, chimique,...
- Introduction de l'expression « **Génie Logiciel** »
  - Comment faire des logiciels de qualité ? Qu'attend-on d'un logiciel ?

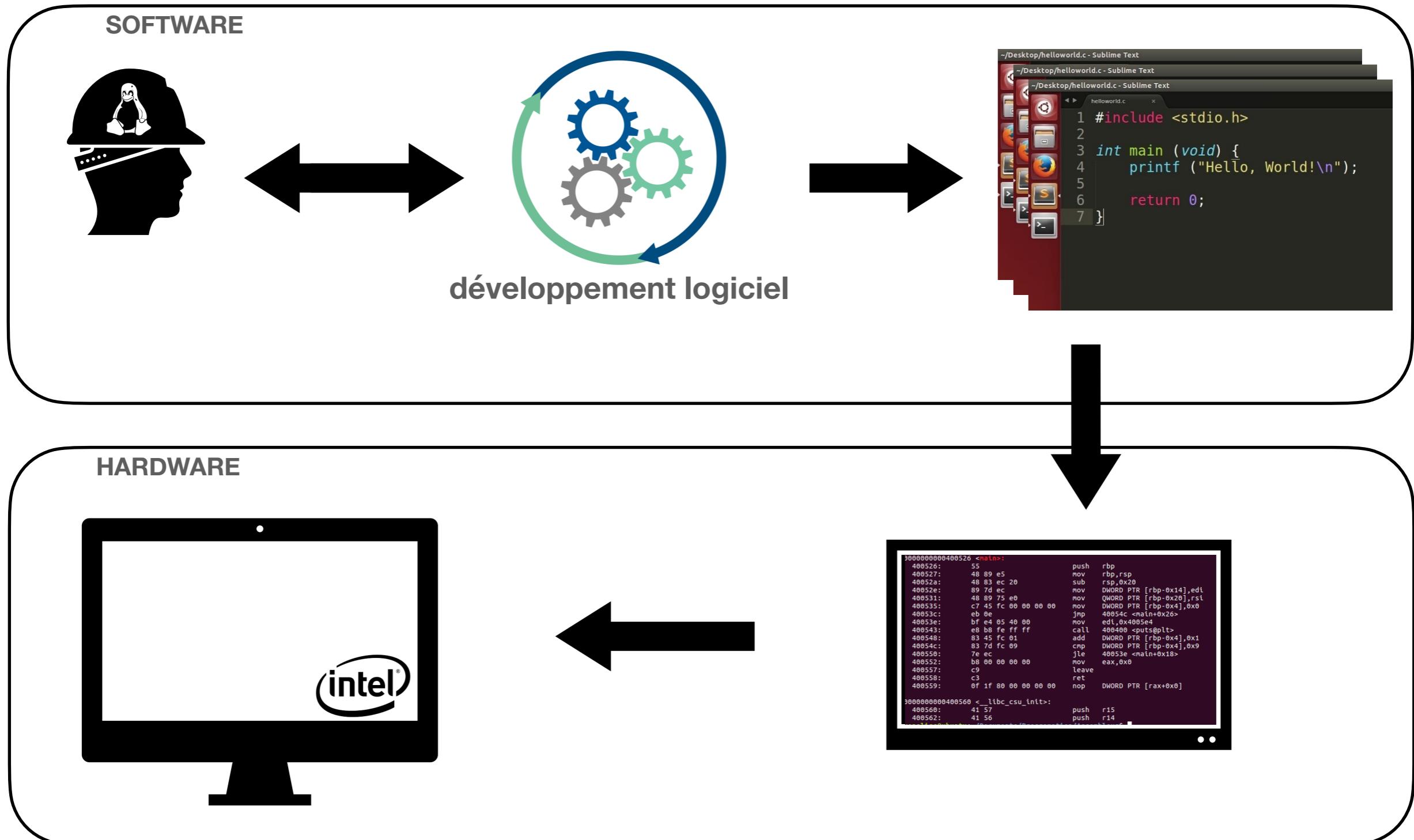
# Génie Logiciel - Software Engineering

## Idée



# Génie Logiciel - Software Engineering

## Idée



# Génie Logiciel - Software Engineering

Génie Logiciel, Ingénierie Logicielle ou Ingénierie du Logiciel

# Génie Logiciel - Software Engineering

**Génie Logiciel, Ingénierie Logicielle ou Ingénierie du Logiciel**

- Le **génie logiciel (GL)** est l'ensemble de :
  - méthodes de travail,
  - bonnes pratiques,
  - techniques,
  - outils

# Génie Logiciel - Software Engineering

**Génie Logiciel, Ingénierie Logicielle ou Ingénierie du Logiciel**

- Le **génie logiciel (GL)** est l'ensemble de :
  - méthodes de travail,
  - bonnes pratiques,
  - techniques,
  - outils

Dédiés à la **conception, le développement, le test et à la maintenance** des logiciels

# Génie Logiciel - Software Engineering

**Génie Logiciel, Ingénierie Logicielle ou Ingénierie du Logiciel**

- Le **génie logiciel (GL)** est l'ensemble de :
  - méthodes de travail,
  - bonnes pratiques,
  - techniques,
  - outils

Dédiés à la **conception, le développement, le test et à la maintenance** des logiciels

- Le GL s'intéresse particulièrement aux méthodes automatiques qui permettent d'arriver « *rapidement* » à des logiciels de « *qualité* ».

# Génie Logiciel - Software Engineering

Génie Logiciel, Ingénierie Logicielle ou Ingénierie du Logiciel

- Le **génie logiciel (GL)** est l'ensemble de :

- méthodes de travail,
- bonnes pratiques,
- techniques,
- outils



Ne pas confondre avec  
Génie Informatique  
(Computer Engineering)

Dédiés à la **conception**, le **développement**, le **test** et à la **maintenance** des logiciels

- Le GL s'intéresse particulièrement aux méthodes automatiques qui permettent d'arriver « *rapidement* » à des logiciels de « *qualité* ».

# Génie Logiciel - Software Engineering

## Principes

# Génie Logiciel - Software Engineering

## Principes

- **Généralisation** pour une réutilisation/adaptation des solutions

# Génie Logiciel - Software Engineering

## Principes

- **Généralisation** pour une réutilisation/adaptation des solutions
- **Structuration**, décomposition et traitement séparé des sous-problèmes

# Génie Logiciel - Software Engineering

## Principes

- **Généralisation** pour une réutilisation/adaptation des solutions
- **Structuration**, décomposition et traitement séparé des sous-problèmes
- **Abstraction** des concepts pertinents pour raisonner et instancier les cas particuliers

# Génie Logiciel - Software Engineering

## Principes

- **Généralisation** pour une réutilisation/adaptation des solutions
- **Structuration**, décomposition et traitement séparé des sous-problèmes
- **Abstraction** des concepts pertinents pour raisonner et instancier les cas particuliers
- **Modularité**, partition du logiciel en composants discrets

# Génie Logiciel - Software Engineering

## Principes

- **Généralisation** pour une réutilisation/adaptation des solutions
- **Structuration**, décomposition et traitement séparé des sous-problèmes
- **Abstraction** des concepts pertinents pour raisonner et instancier les cas particuliers
- **Modularité**, partition du logiciel en composants discrets
- **Construction incrémentale** et anticipation des évolutions

# Génie Logiciel - Software Engineering

## Principes

- **Généralisation** pour une réutilisation/adaptation des solutions
- **Structuration**, décomposition et traitement séparé des sous-problèmes
- **Abstraction** des concepts pertinents pour raisonner et instancier les cas particuliers
- **Modularité**, partition du logiciel en composants discrets
- **Construction incrémentale** et anticipation des évolutions
- **Documentation** du logiciel pour le suivi et la communication

# Génie Logiciel - Software Engineering

## Principes

- **Généralisation** pour une réutilisation/adaptation des solutions
- **Structuration**, décomposition et traitement séparé des sous-problèmes
- **Abstraction** des concepts pertinents pour raisonner et instancier les cas particuliers
- **Modularité**, partition du logiciel en composants discrets
- **Construction incrémentale** et anticipation des évolutions
- **Documentation** du logiciel pour le suivi et la communication
- **Vérification** pour assurer que le logiciel répond bien à la spécification de départ

# Qualité du Logiciel

## Critères de qualité

# Qualité du Logiciel

## Critères de qualité

- **Validité** : réponse aux attentes du client

# Qualité du Logiciel

## Critères de qualité

- **Validité** : réponse aux attentes du client
- **Utilisabilité** : prise en main, facilité d'utilisation et d'apprentissage

# Qualité du Logiciel

## Critères de qualité

- **Validité** : réponse aux attentes du client
- **Utilisabilité** : prise en main, facilité d'utilisation et d'apprentissage
- **Performance** : temps de réponse, complexité des algos, fluidité.

# Qualité du Logiciel

## Critères de qualité

- **Validité** : réponse aux attentes du client
- **Utilisabilité** : prise en main, facilité d'utilisation et d'apprentissage
- **Performance** : temps de réponse, complexité des algos, fluidité.
- **Fiabilité** : robustesse, sûreté et tolérance aux pannes

# Qualité du Logiciel

## Critères de qualité

- **Validité** : réponse aux attentes du client
- **Utilisabilité** : prise en main, facilité d'utilisation et d'apprentissage
- **Performance** : temps de réponse, complexité des algos, fluidité.
- **Fiabilité** : robustesse, sûreté et tolérance aux pannes
- **Sécurité** : intégrité des données et protection des accès

# Qualité du Logiciel

## Critères de qualité

- **Validité** : réponse aux attentes du client
- **Utilisabilité** : prise en main, facilité d'utilisation et d'apprentissage
- **Performance** : temps de réponse, complexité des algos, fluidité.
- **Fiabilité** : robustesse, sûreté et tolérance aux pannes
- **Sécurité** : intégrité des données et protection des accès
- **Maintenabilité** : facilité à corriger ou transformer le logiciel

# Qualité du Logiciel

## Critères de qualité

- **Validité** : réponse aux attentes du client
- **Utilisabilité** : prise en main, facilité d'utilisation et d'apprentissage
- **Performance** : temps de réponse, complexité des algos, fluidité.
- **Fiabilité** : robustesse, sûreté et tolérance aux pannes
- **Sécurité** : intégrité des données et protection des accès
- **Maintenabilité** : facilité à corriger ou transformer le logiciel
- **Portabilité** : changement d'environnement matériel ou logiciel

# Qualité du Logiciel

## Critères de qualité

- **Validité** : réponse aux attentes du client
- **Utilisabilité** : prise en main, facilité d'utilisation et d'apprentissage
- **Performance** : temps de réponse, complexité des algos, fluidité.
- **Fiabilité** : robustesse, sûreté et tolérance aux pannes
- **Sécurité** : intégrité des données et protection des accès
- **Maintenabilité** : facilité à corriger ou transformer le logiciel
- **Portabilité** : changement d'environnement matériel ou logiciel
- **Interopérabilité** : interagir en synergie avec d'autres logiciels

# Cycle de vie

## Processus de développement logiciel

# Cycle de vie

## Processus de développement logiciel

- **Spécification** : Définition des objectifs, étude de marché, comprendre les besoins du client, spécification des besoins, analyse du domaine, organisation du projet... [Analyste]
  - Document : cahier des charges, calendrier du projet, estimation des coûts, premier manuel d'utilisation

# Cycle de vie

## Processus de développement logiciel

- **Spécification** : Définition des objectifs, étude de marché, comprendre les besoins du client, spécification des besoins, analyse du domaine, organisation du projet... [Analyste]
  - Document : cahier des charges, calendrier du projet, estimation des coûts, premier manuel d'utilisation
- **Conception** : Comment le logiciel assurera les fonctionnalités recherchées, conception générale (structure et architecture du logiciel), conception détaillée (algorithmes, structures de données,...) [Concepteur]
  - Document : cahier de conception

# Cycle de vie

## Processus de développement logiciel

- **Spécification** : Définition des objectifs, étude de marché, comprendre les besoins du client, spécification des besoins, analyse du domaine, organisation du projet... [Analyste]
  - Document : cahier des charges, calendrier du projet, estimation des coûts, premier manuel d'utilisation
- **Conception** : Comment le logiciel assurera les fonctionnalités recherchées, conception générale (structure et architecture du logiciel), conception détaillée (algorithmes, structures de données,...) [Concepteur]
  - Document : cahier de conception
- **Programmation** : écriture du logiciel, choix des langages, ... [Programmeur]
  - Document : documentation du code + manuel d'utilisation

# Cycle de vie

## Processus de développement logiciel

# Cycle de vie

## Processus de développement logiciel

- **Verification et validation (V&V)** : mesurer la distance la conception et la programmation (vérification), distance entre l'analyse et la programmation (validation) [Testeur]
  - Document : rapport des tests, certificats, preuves...

# Cycle de vie

## Processus de développement logiciel

- **Verification et validation (V&V)** : mesurer la distance la conception et la programmation (vérification), distance entre l'analyse et la programmation (validation) [Testeur]
  - Document : rapport des tests, certificats, preuves...
- **Déploiement** : Installation, formation, assistance [Formateur]
  - Document : Manuel d'utilisation, support

# Cycle de vie

## Processus de développement logiciel

- **Verification et validation (V&V)** : mesurer la distance la conception et la programmation (vérification), distance entre l'analyse et la programmation (validation) [Testeur]
  - Document : rapport des tests, certificats, preuves...
- **Déploiement** : Installation, formation, assistance [Formateur]
  - Document : Manuel d'utilisation, support
- **Maintenance** : pérennité du logiciel avec les mises à jour. [All]
  - Document : Manuel d'utilisation à jour

# Cycle de vie

## Processus de développement logiciel

- **Verification et validation (V&V)** : mesurer la distance la conception et la programmation (vérification), distance entre l'analyse et la programmation (validation) [Testeur]
  - Document : rapport des tests, certificats, preuves...
- **Déploiement** : Installation, formation, assistance [Formateur]
  - Document : Manuel d'utilisation, support
- **Maintenance** : pérennité du logiciel avec les mises à jour. [All]
  - Document : Manuel d'utilisation à jour
- **Documentation** : Une activité qui accompagne chaque phase [All]

# Cycle de vie

## Processus de développement logiciel

# Cycle de vie

## Processus de développement logiciel

	Effort	Erreurs	Coût de la maintenance
Spécification	15 %	55 %	80 %
Conception	15 %	25 %	15 %
Programmation	20 %	10 %	2 %
V&V	50 %	10 %	3 %

# Cycle de vie

## Standish group, Chaos Report 2015

Softwares	Developers	Duration	Size (LOC)	Successful	Challenged	Failed
Grand	2K - 5K	5 - 10 yrs.	> 1M	6 %	51 %	43 %
Large	100 - 1K	4 - 5 yrs.	100K - 1M	11 %	59 %	30 %
Medium	5 - 20	2 - 3 yrs.	50K- 100K	12 %	62 %	26 %
Moderate	2 - 5	1 - 2 yrs.	3K - 20K	24 %	64 %	12 %
Small	1	1 - 6 mos.	1K - 2K	61 %	32 %	7 %

Successful: OnTime, OnBudget, OnTarget || 50,000 software projects

# Cycle de vie

## Standish group, Chaos Report 2015

Softwares	Developers	Duration	Size (LOC)	Successful	Challenged	Failed
Grand	2K - 5K	5 - 10 yrs.	> 1M	6 %	51 %	43 %
Large	100 - 1K	4 - 5 yrs.	100K - 1M	11 %	59 %	30 %
Medium	5 - 20	2 - 3 yrs.	50K- 100K	12 %	62 %	26 %
Moderate	2 - 5	1 - 2 yrs.	3K - 20K	24 %	64 %	12 %
Small	1	1 - 6 mos.	1K - 2K	61 %	32 %	7 %

Successful: OnTime, OnBudget, OnTarget || 50,000 software projects

Softwares	Successful	Challenged	Failed
Banking	30 %	55 %	15 %
Financial	29 %	56 %	15 %
Government	21 %	55 %	24 %
Healthcare	29 %	53 %	18 %
Manufacturing	28 %	53 %	19 %
Retail	35 %	49 %	16 %
Services	29 %	52 %	19 %
Telecom	24 %	53 %	23 %
Other	29 %	48 %	23 %

# Cycle de vie

Raisons d'une faible qualité logiciel

# Cycle de vie

## Raisons d'une faible qualité logiciel

- Taille des logiciels et des équipes de conception/développement

# Cycle de vie

## Raisons d'une faible qualité logiciel

- Taille des logiciels et des équipes de conception/développement

# Cycle de vie

## Raisons d'une faible qualité logiciel

- Taille des logiciels et des équipes de conception/développement
- Manque de rigueur dans la phase d'analyse et absence de du client dans le processus du développement.

# Cycle de vie

## Raisons d'une faible qualité logiciel

- Taille des logiciels et des équipes de conception/développement
- Manque de rigueur dans la phase d'analyse et absence de du client dans le processus du développement.

# Cycle de vie

## Raisons d'une faible qualité logiciel

- Taille des logiciels et des équipes de conception/développement
- Manque de rigueur dans la phase d'analyse et absence de du client dans le processus du développement.
- Manquement dans les activités du processus de développement logiciel

# Cycle de vie

Raisons d'une faible qualité logiciel

# Cycle de vie

## Raisons d'une faible qualité logiciel



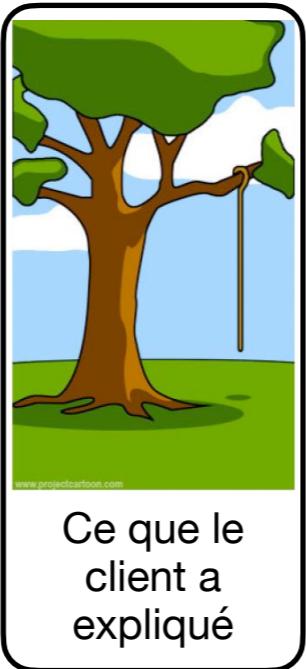
Ce que le client attend

# Cycle de vie

## Raisons d'une faible qualité logiciel



Ce que le client attend



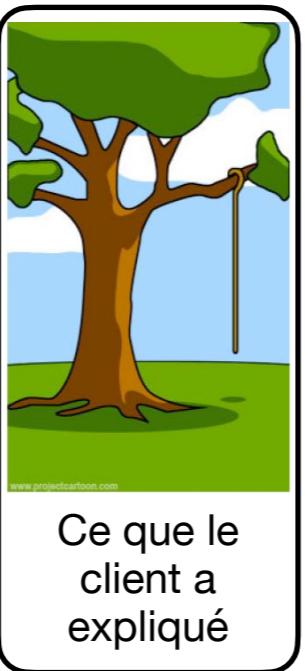
Ce que le client a expliqué

# Cycle de vie

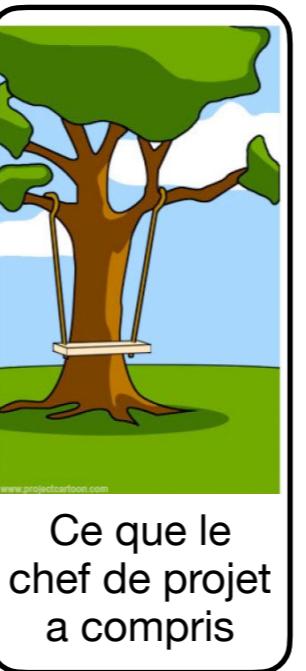
## Raisons d'une faible qualité logiciel



Ce que le client attend



Ce que le client a expliqué



Ce que le chef de projet a compris

# Cycle de vie

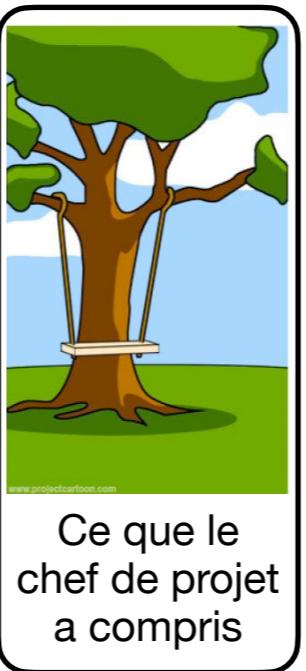
## Raisons d'une faible qualité logiciel



Ce que le client attend



Ce que le client a expliqué



Ce que le chef de projet a compris



Ce que l'analyste a proposé

# Cycle de vie

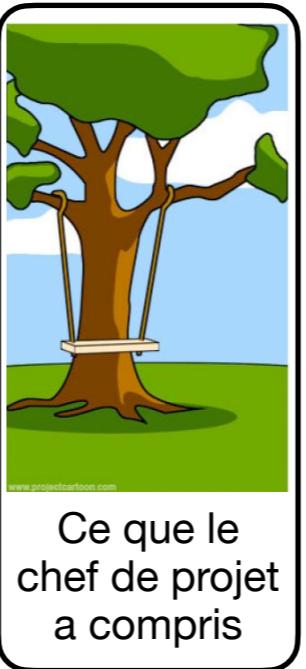
## Raisons d'une faible qualité logiciel



Ce que le client attend



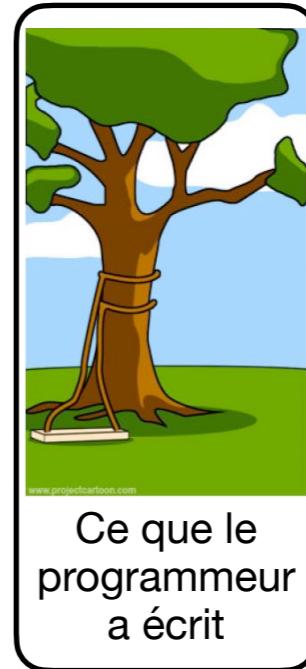
Ce que le client a expliqué



Ce que le chef de projet a compris



Ce que l'analyste a proposé



Ce que le programmeur a écrit

# Cycle de vie

## Raisons d'une faible qualité logiciel



Ce que le client attend



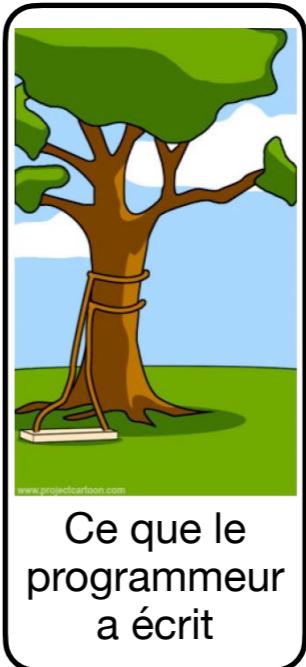
Ce que le client a expliqué



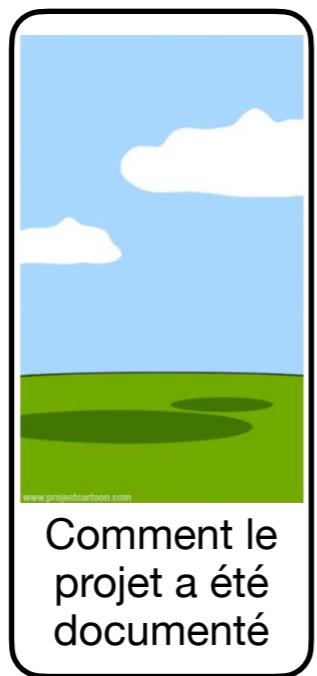
Ce que le chef de projet a compris



Ce que l'analyste a proposé



Ce que le programmeur a écrit



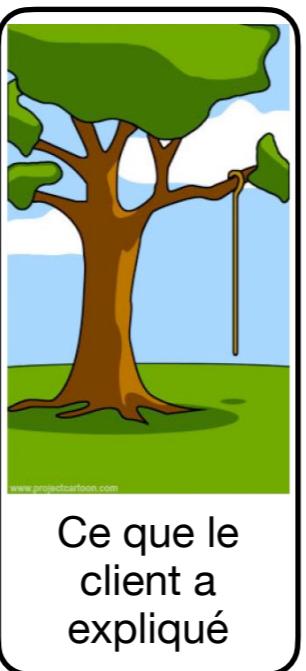
Comment le projet a été documenté

# Cycle de vie

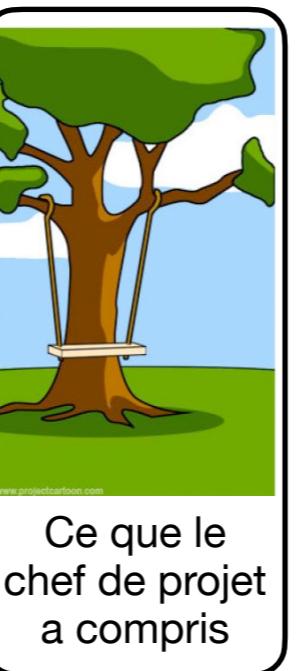
## Raisons d'une faible qualité logiciel



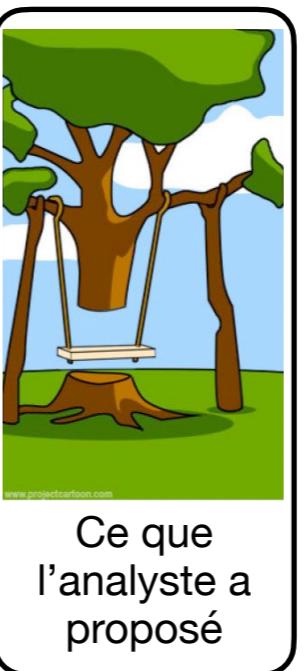
Ce que le client attend



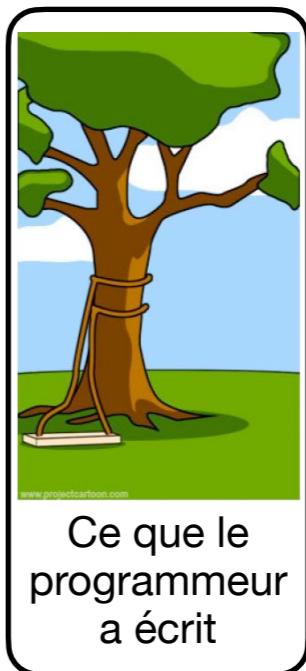
Ce que le client a expliqué



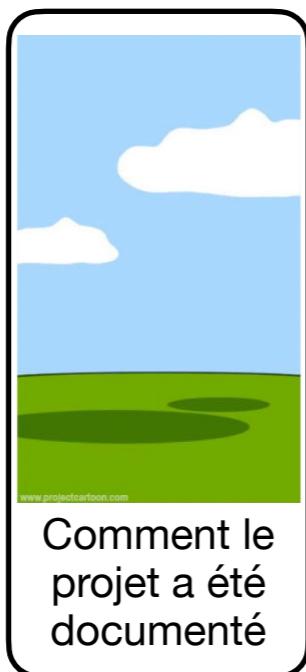
Ce que le chef de projet a compris



Ce que l'analyste a proposé



Ce que le programmeur a écrit



Comment le projet a été documenté



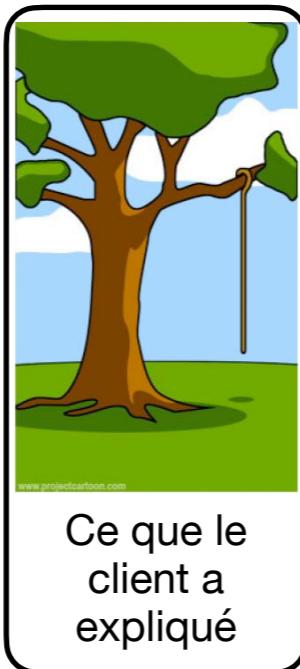
Quand le projet a été livré

# Cycle de vie

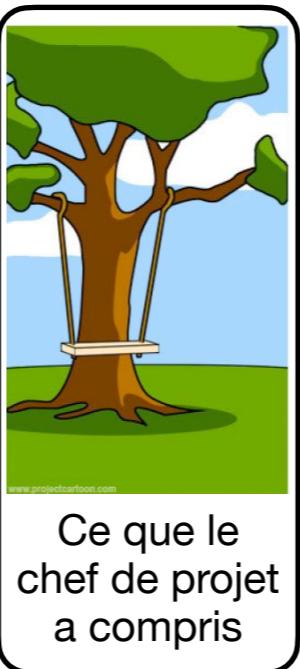
## Raisons d'une faible qualité logiciel



Ce que le client attend



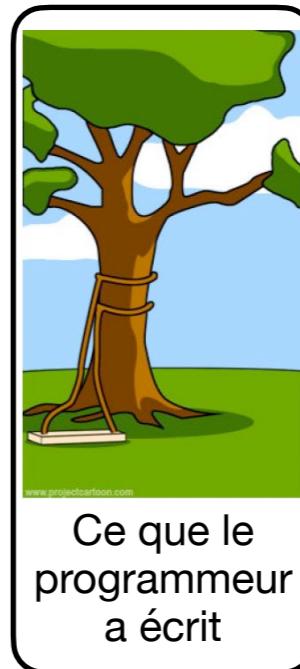
Ce que le client a expliqué



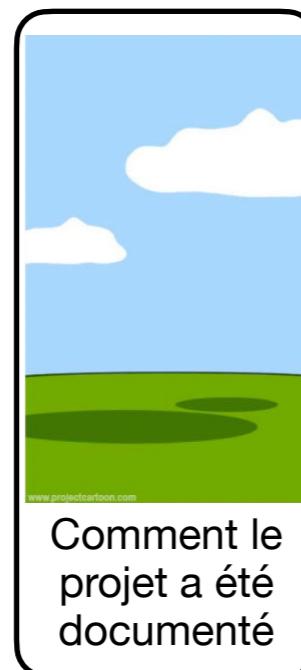
Ce que le chef de projet a compris



Ce que l'analyste a proposé



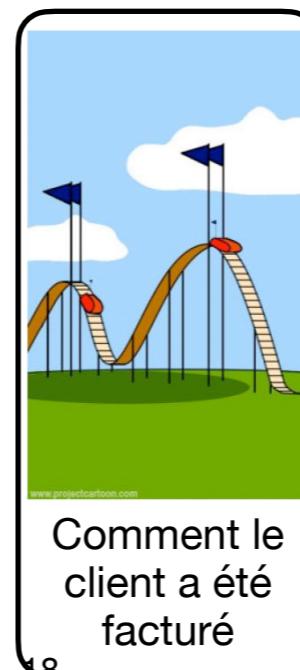
Ce que le programmeur a écrit



Comment le projet a été documenté



Quand le projet a été livré



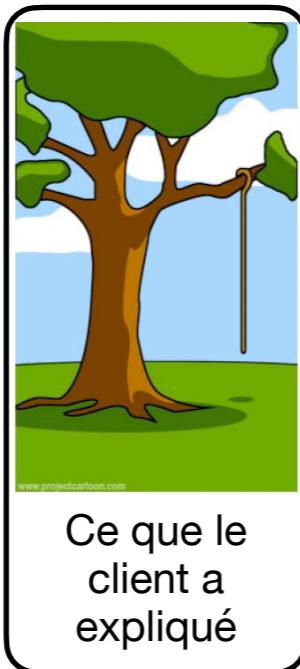
Comment le client a été facturé

# Cycle de vie

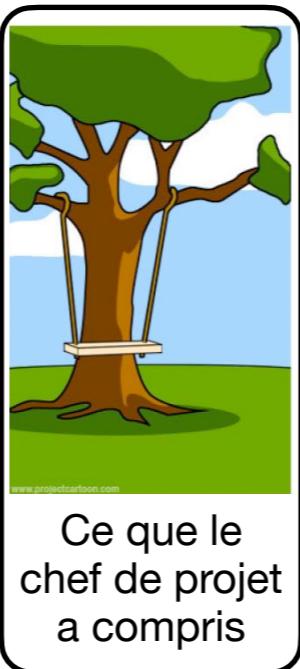
## Raisons d'une faible qualité logiciel



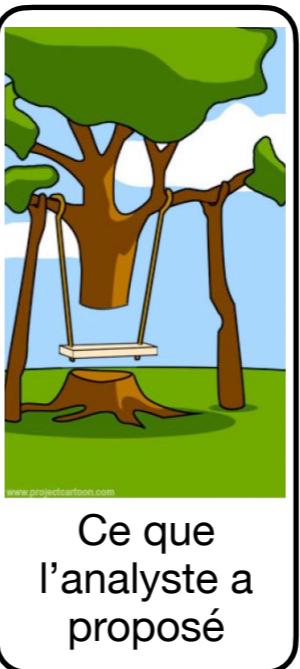
Ce que le client attend



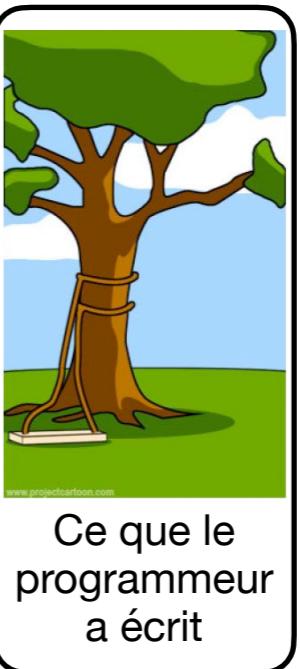
Ce que le client a expliqué



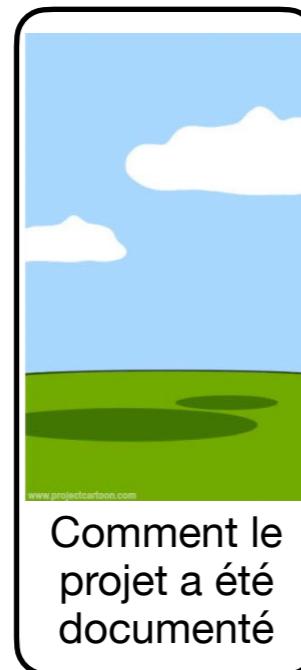
Ce que le chef de projet a compris



Ce que l'analyste a proposé



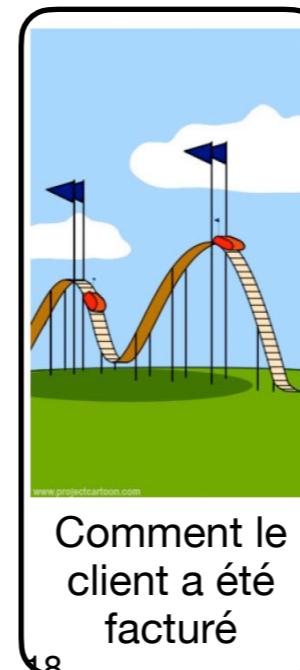
Ce que le programmeur a écrit



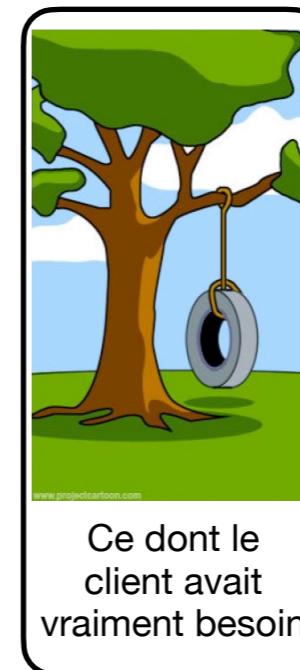
Comment le projet a été documenté



Quand le projet a été livré



Comment le client a été facturé



Ce dont le client avait vraiment besoin

# Cycle de vie

## Exercice 1

# Cycle de vie

## Exercice 1

- Quel est le seuil minimal (entre min, moyen et max) à avoir pour chaque facteur des systèmes suivants :

# Cycle de vie

## Exercice 1

- Quel est le seuil minimal (entre min, moyen et max) à avoir pour chaque facteur des systèmes suivants :

# Cycle de vie

## Exercice 1

- Quel est le seuil minimal (entre min, moyen et max) à avoir pour chaque facteur des systèmes suivants :
  - Contrôleur de machine à laver

# Cycle de vie

## Exercice 1

- Quel est le seuil minimal (entre min, moyen et max) à avoir pour chaque facteur des systèmes suivants :
  - Contrôleur de machine à laver
  - Jeu vidéo

# Cycle de vie

## Exercice 1

- Quel est le seuil minimal (entre min, moyen et max) à avoir pour chaque facteur des systèmes suivants :
  - Contrôleur de machine à laver
  - Jeu vidéo
  - Client de messagerie

# Cycle de vie

## Exercice 1

- Quel est le seuil minimal (entre min, moyen et max) à avoir pour chaque facteur des systèmes suivants :
  - Contrôleur de machine à laver
  - Jeu vidéo
  - Client de messagerie
  - Application mobile d'une banque en ligne

# Cycle de vie

## Exercice 1

- Quel est le seuil minimal (entre min, moyen et max) à avoir pour chaque facteur des systèmes suivants :
  - Contrôleur de machine à laver
  - Jeu vidéo
  - Client de messagerie
  - Application mobile d'une banque en ligne
  - Simulateur en génétique

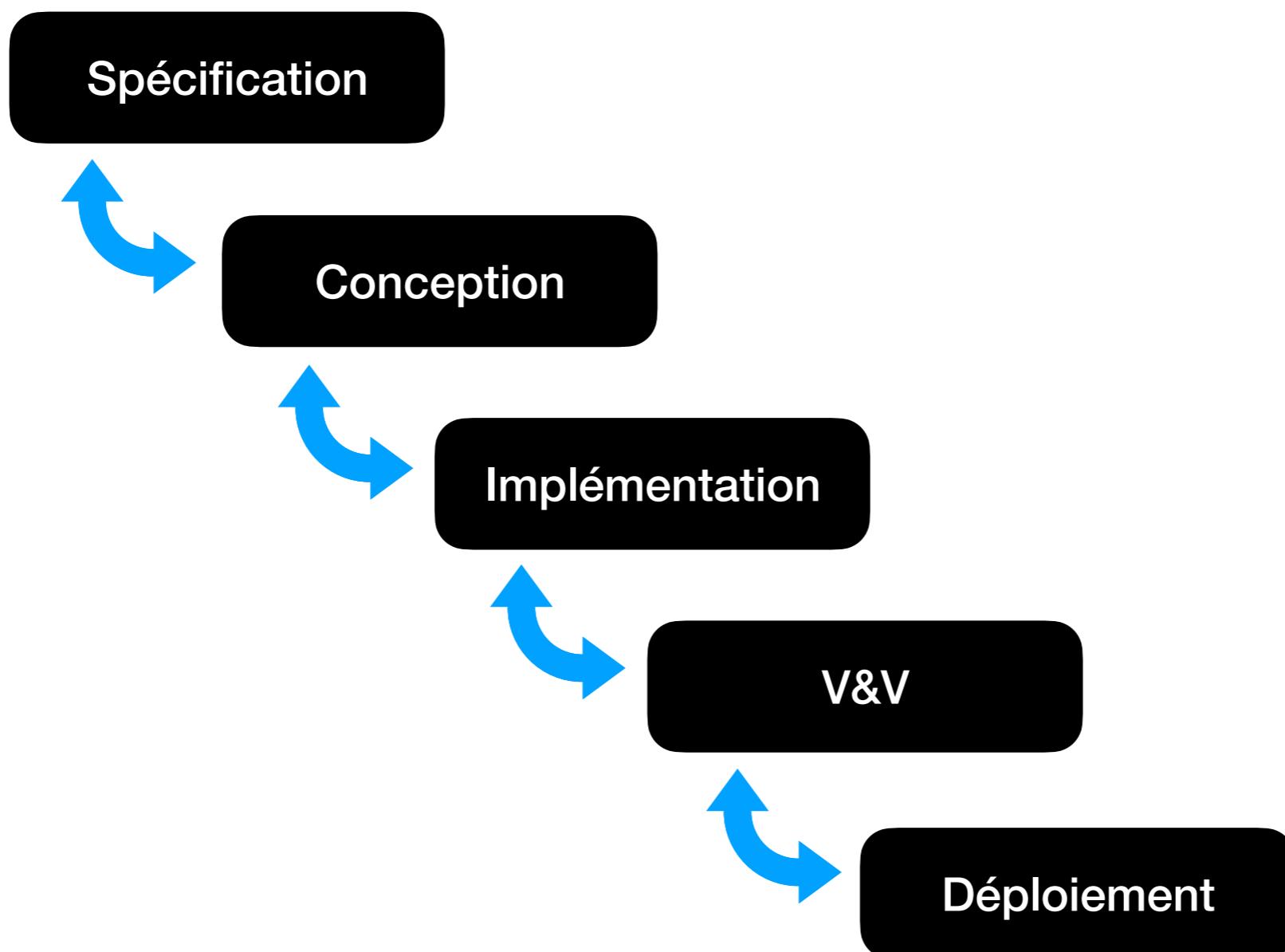
# Cycle de vie

## Exercice 1

	port.	util.	val.	perf.	séc.	maint.
Machine à laver	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max
Jeu vidéo	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max
Mail	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max
BankApp	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max
SimGen	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max	min/moy/ max

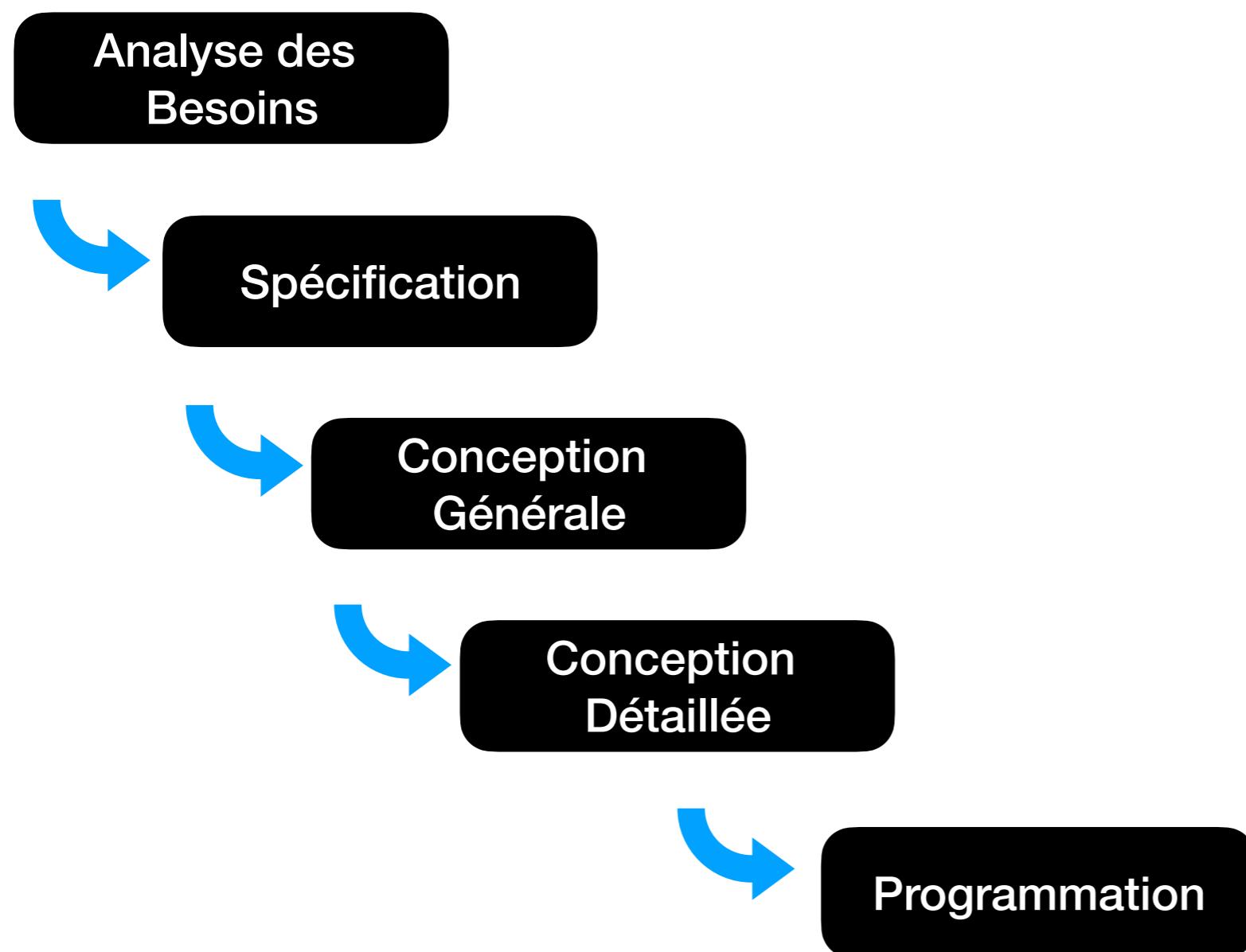
# Cycle de vie

## Modèle en Cascade (Waterfall Model)



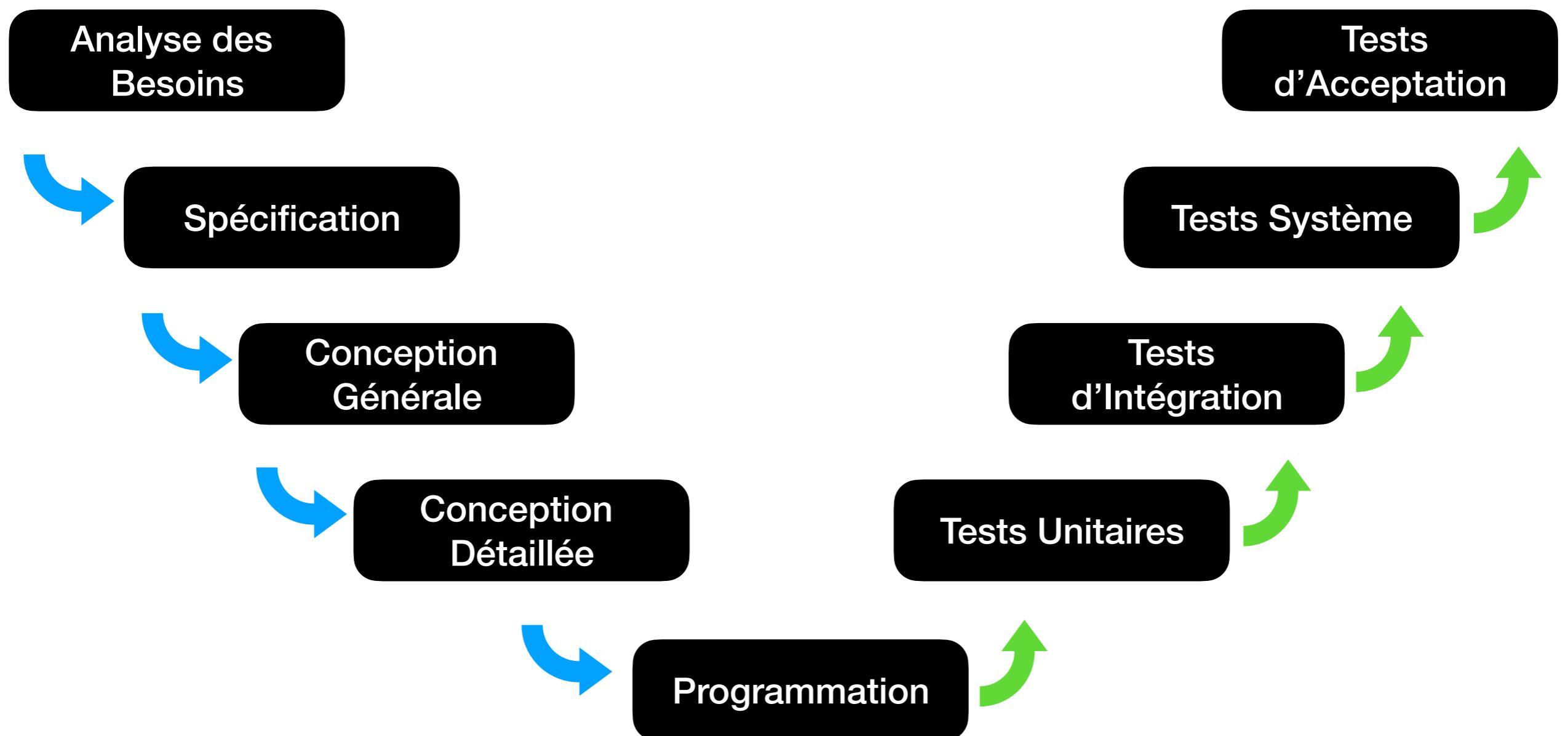
# Cycle de vie

## Modèle en V (V-Shaped Model)



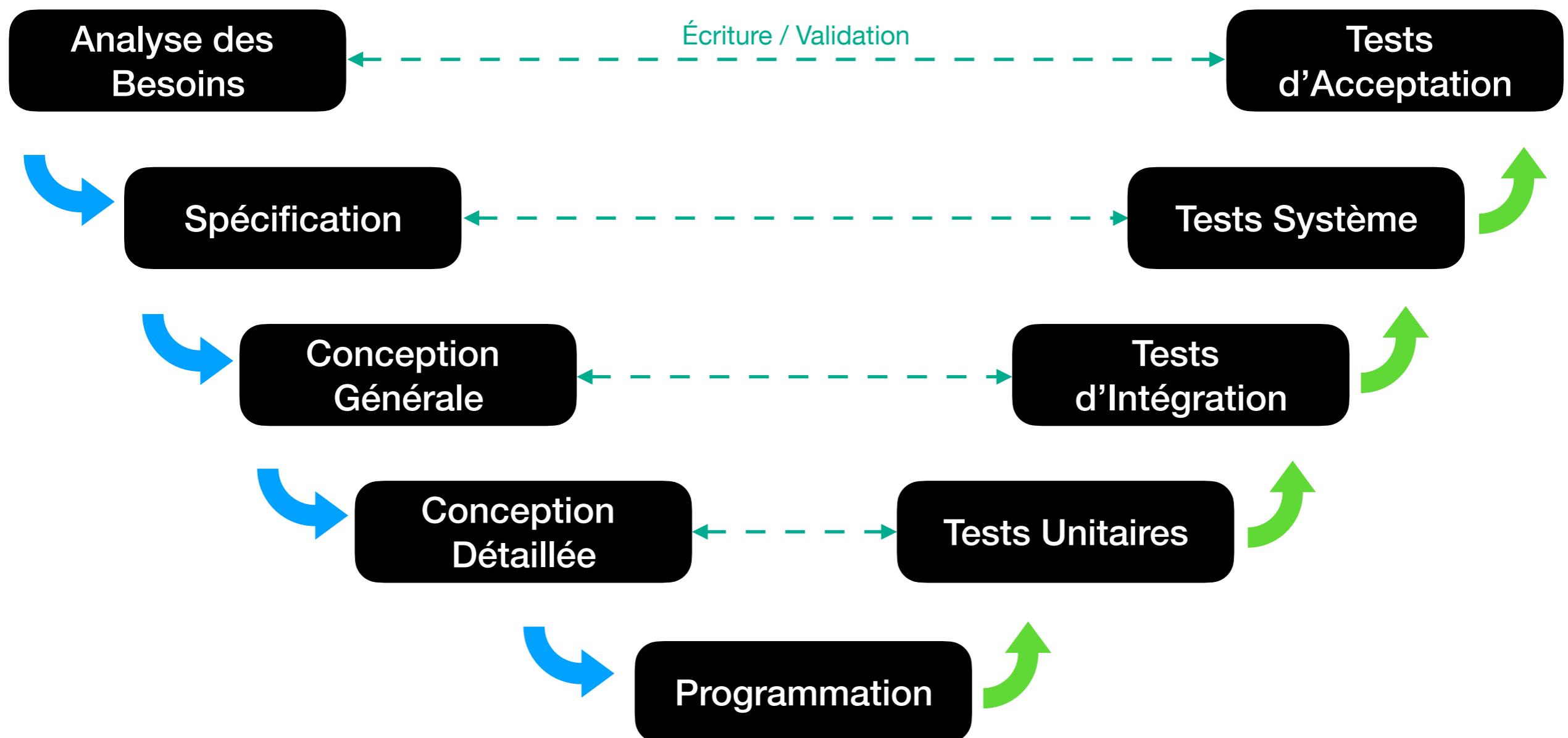
# Cycle de vie

## Modèle en V (V-Shaped Model)



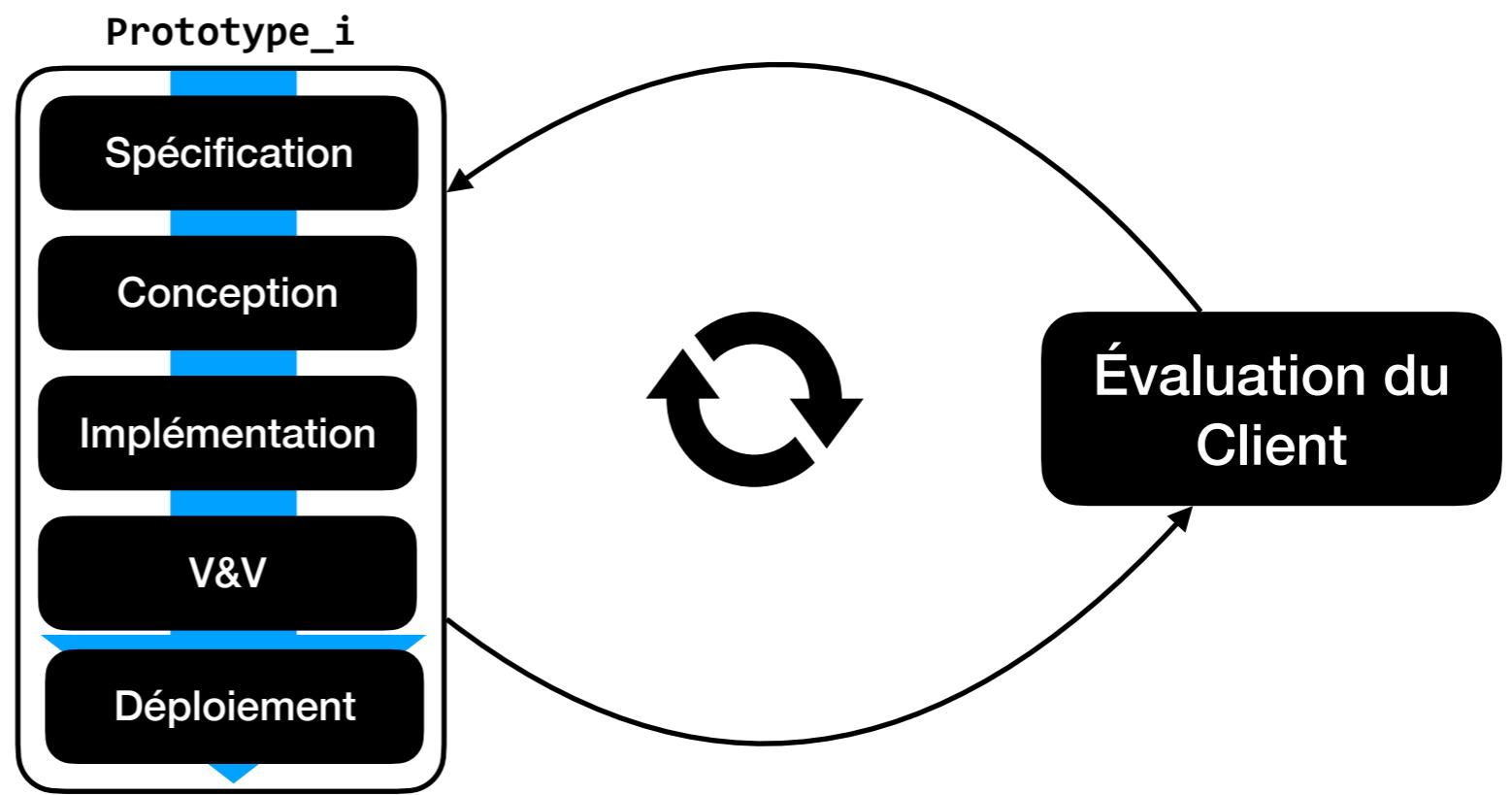
# Cycle de vie

## Modèle en V (V-Shaped Model)



# Cycle de vie

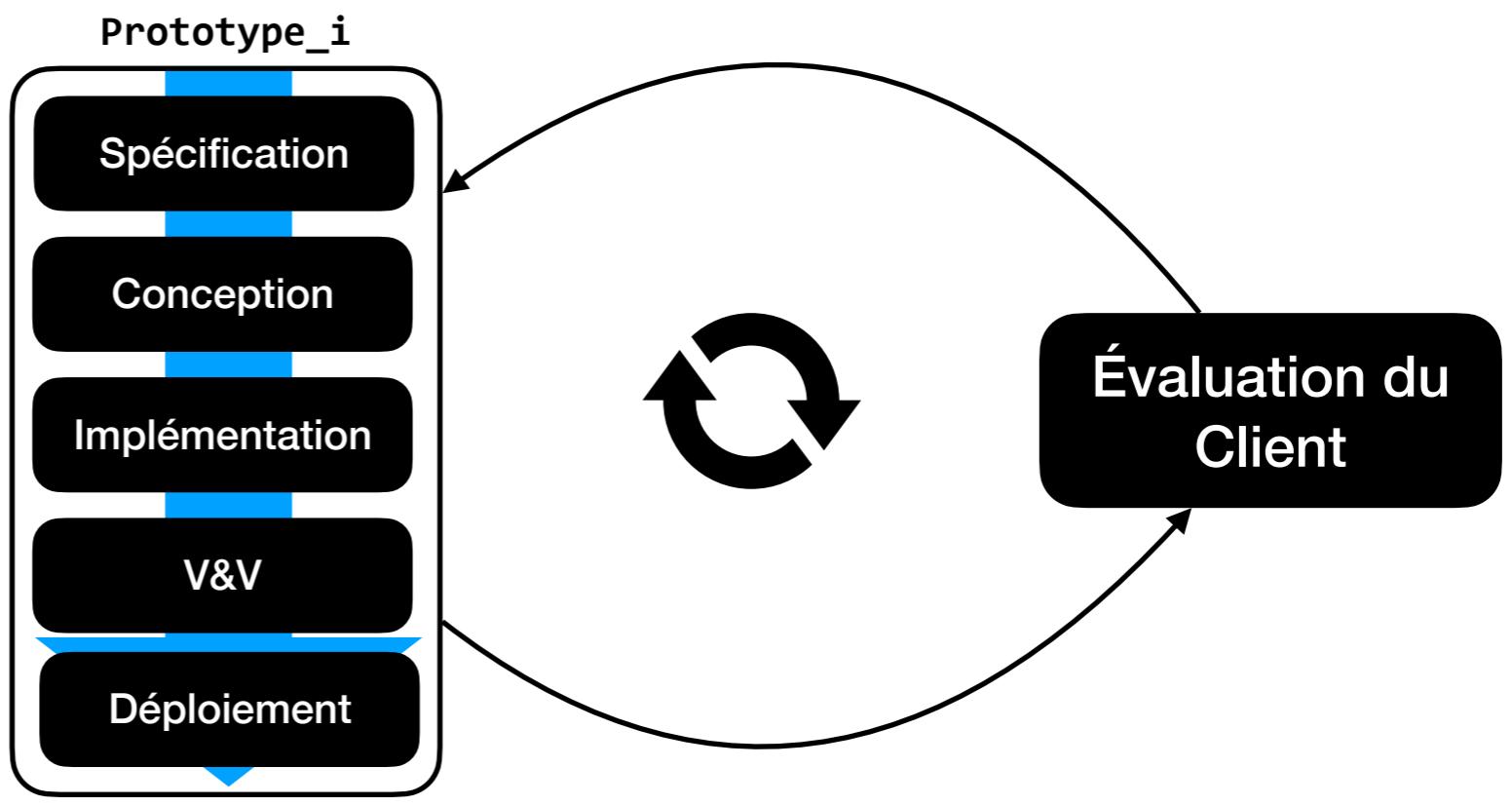
## Modèle par prototypage (Prototyping Model)



# Cycle de vie

## Modèle par prototypage (Prototyping Model)

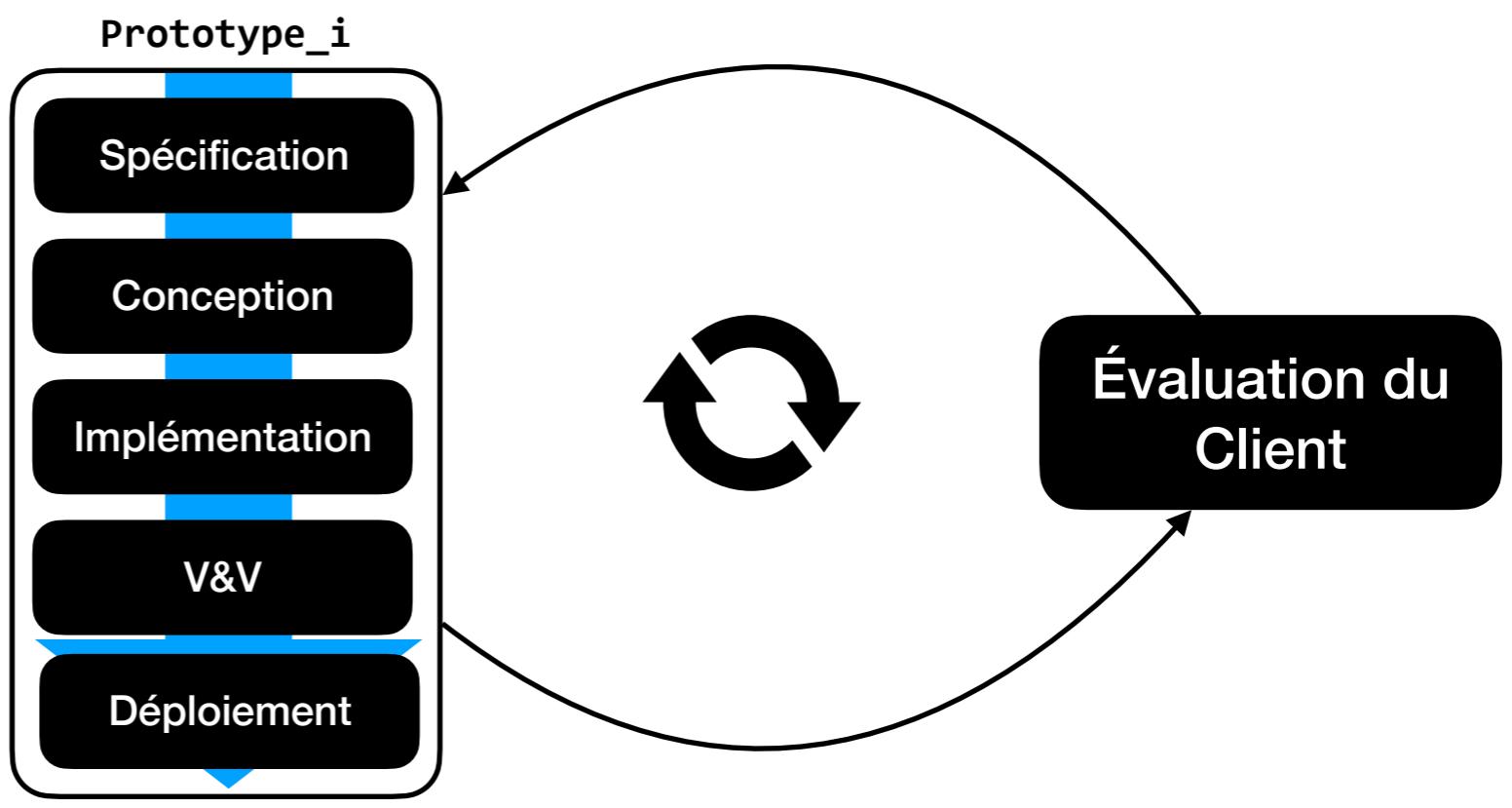
- Implication du client dans le développement



# Cycle de vie

## Modèle par prototypage (Prototyping Model)

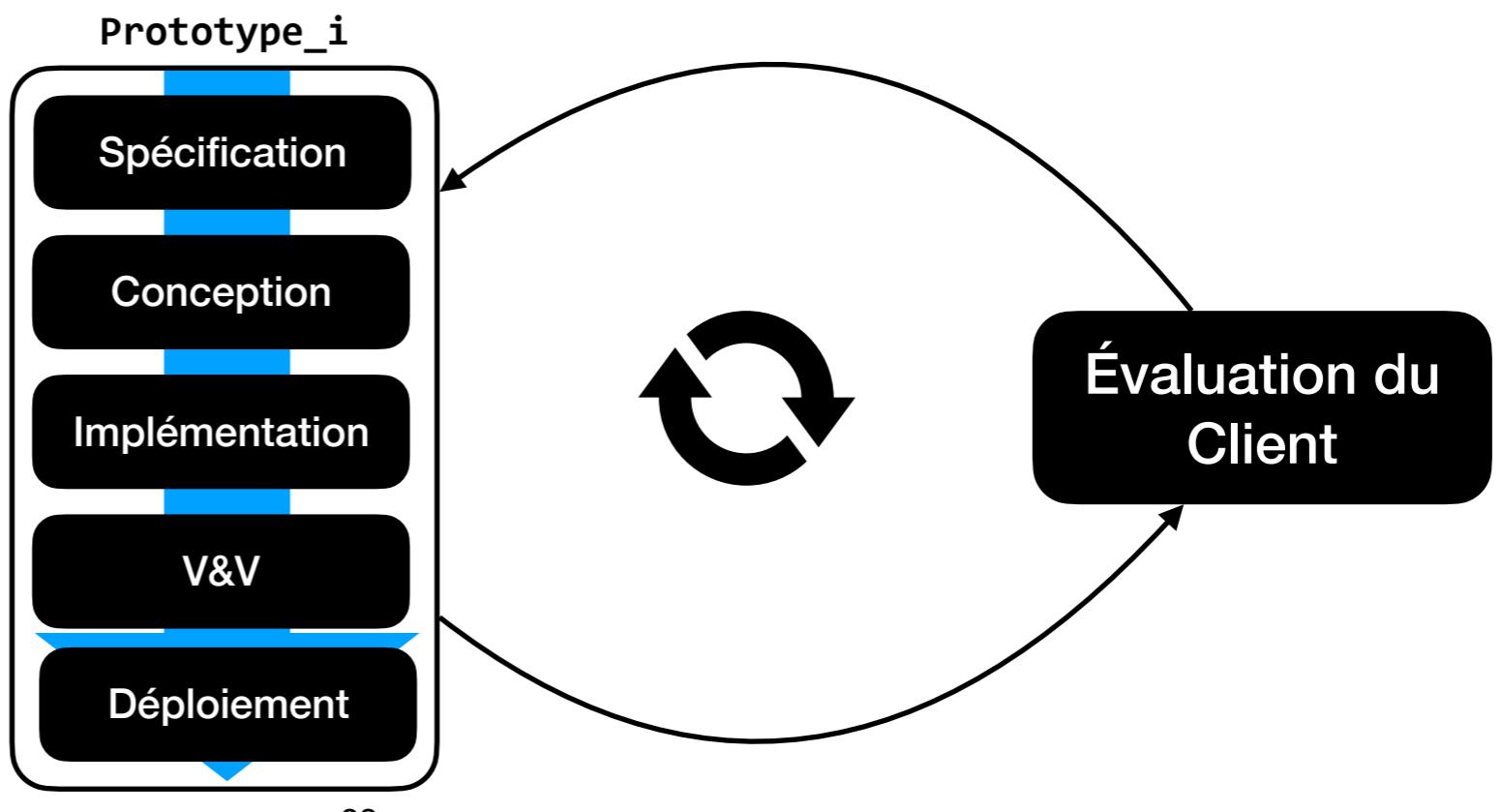
- Implication du client dans le développement
- À chaque étape, un prototype est présenté au client pour évaluation



# Cycle de vie

## Modèle par prototypage (Prototyping Model)

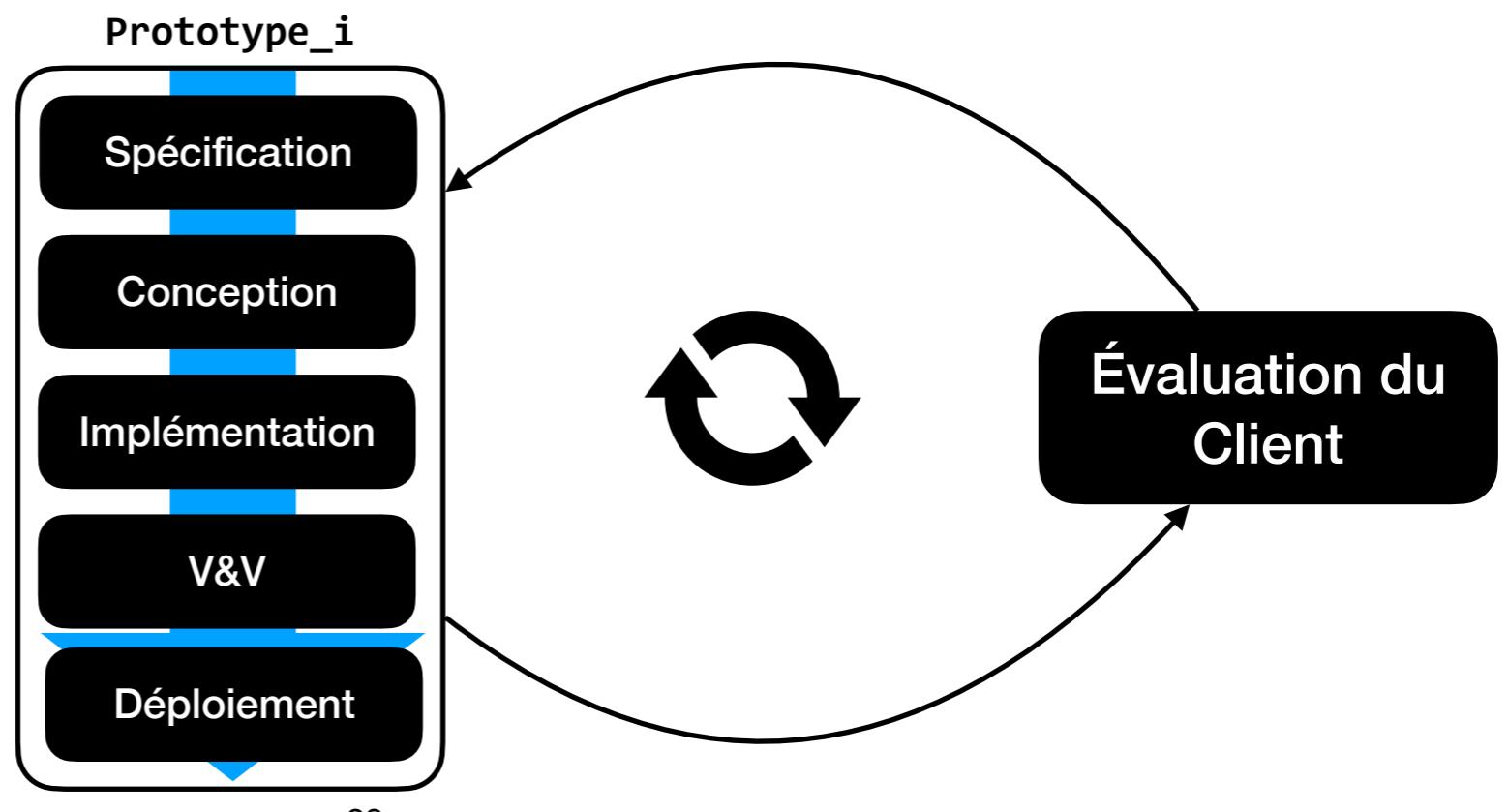
- Implication du client dans le développement
- À chaque étape, un prototype est présenté au client pour évaluation
- Permet de présenter au client des fonctionnalités par ordre de priorité



# Cycle de vie

## Modèle par prototypage (Prototyping Model)

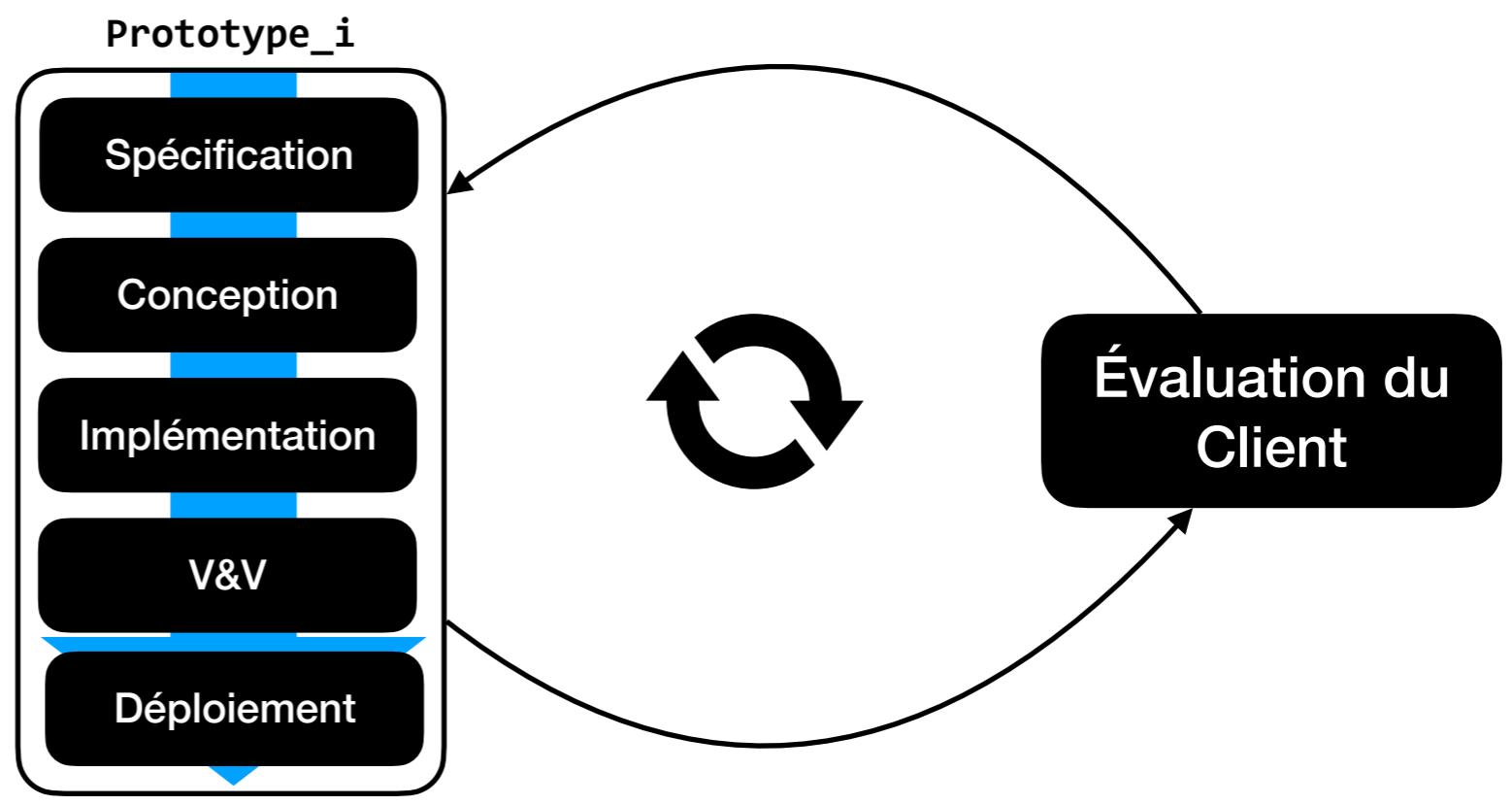
- Implication du client dans le développement
- À chaque étape, un prototype est présenté au client pour évaluation
- Permet de présenter au client des fonctionnalités par ordre de priorité
- Prototype jetable, maquette exploratoire développée rapidement



# Cycle de vie

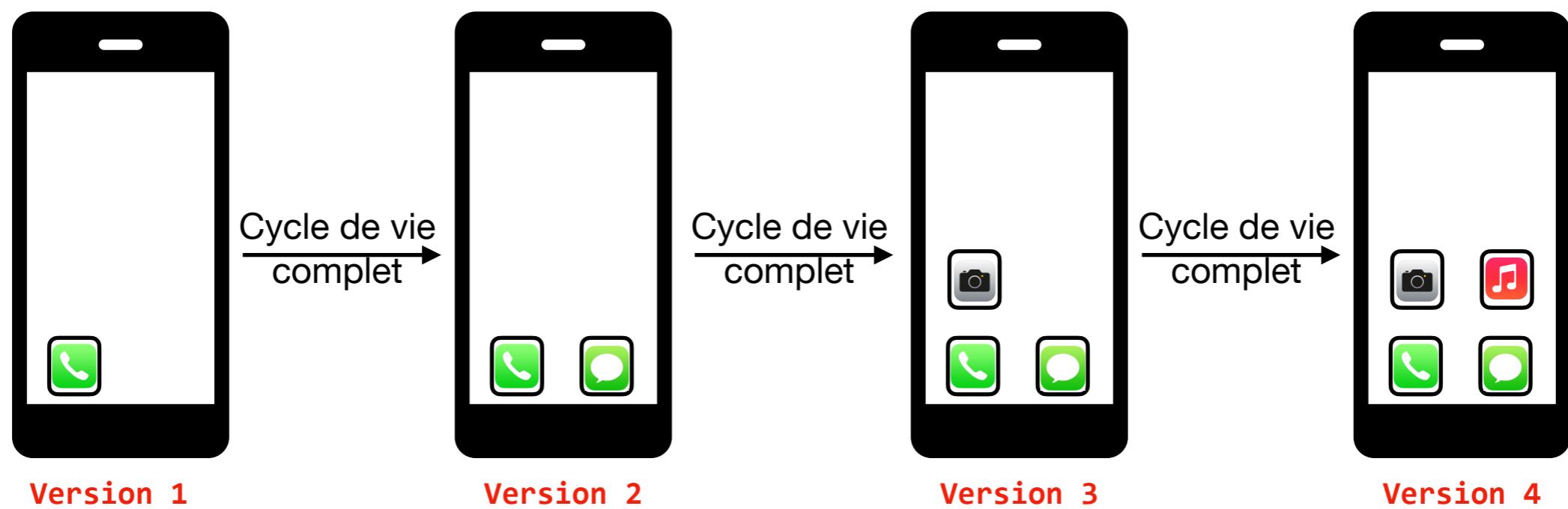
## Modèle par prototypage (Prototyping Model)

- Implication du client dans le développement
- À chaque étape, un prototype est présenté au client pour évaluation
- Permet de présenter au client des fonctionnalités par ordre de priorité
- Prototype jetable, maquette exploratoire développée rapidement
- Prototype évolutif, réutilisable, maquette à compléter / améliorer pour atteindre le produit final



# Cycle de vie

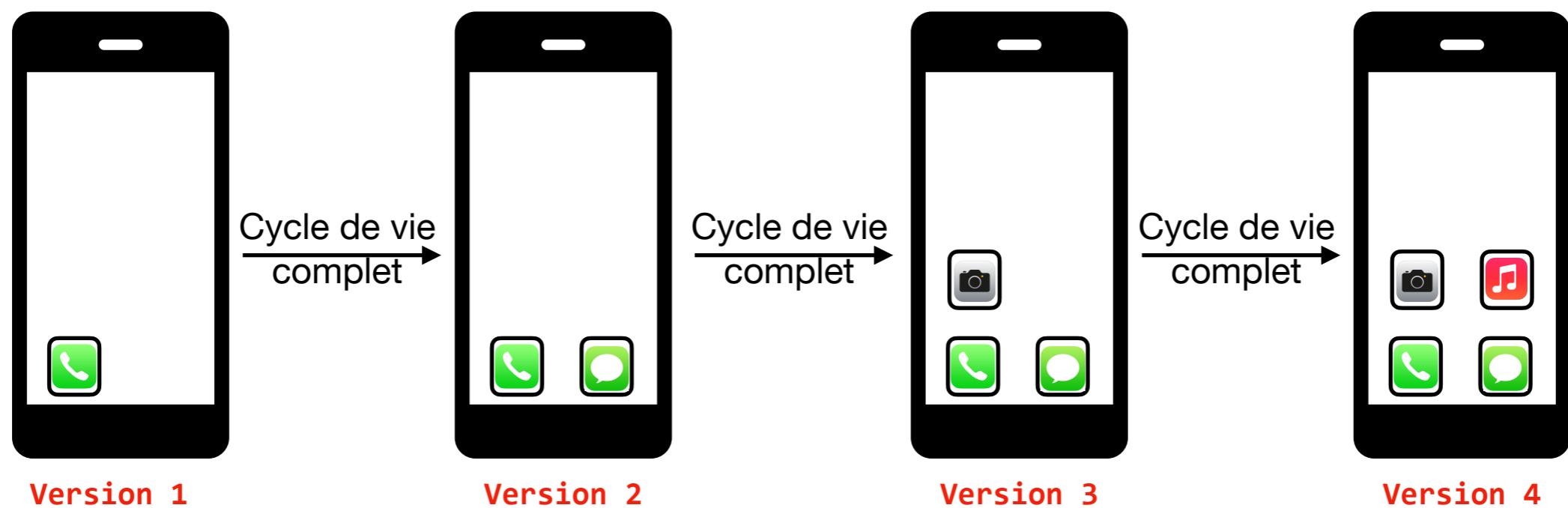
## Modèle évolutif - Incrémental



# Cycle de vie

## Modèle évolutif - Incrémental

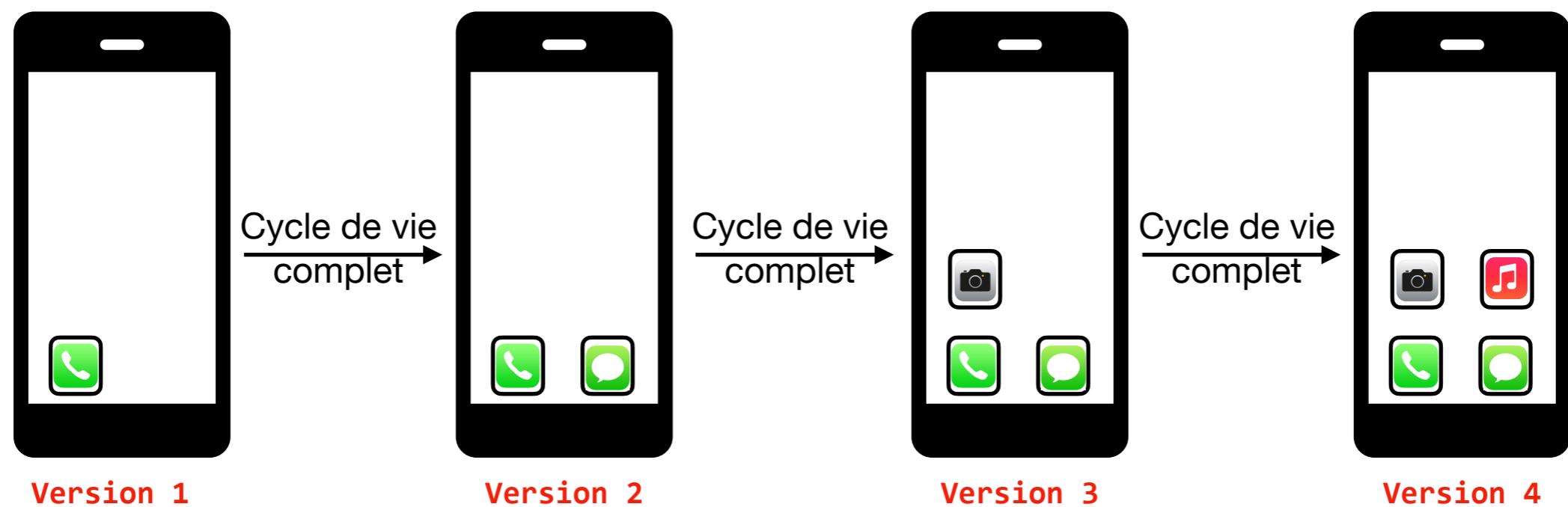
- Intégration du client dans le développement



# Cycle de vie

## Modèle évolutif - Incrémental

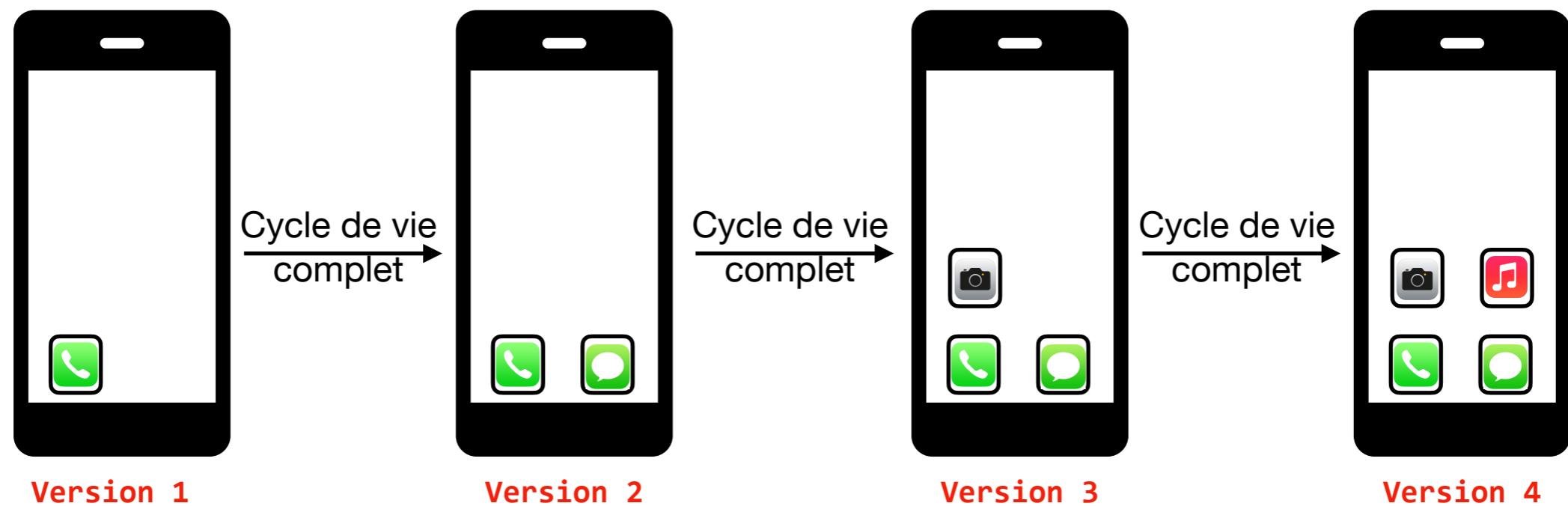
- Intégration du client dans le développement
- Concevoir et livrer une version minimale du système (sous-système)



# Cycle de vie

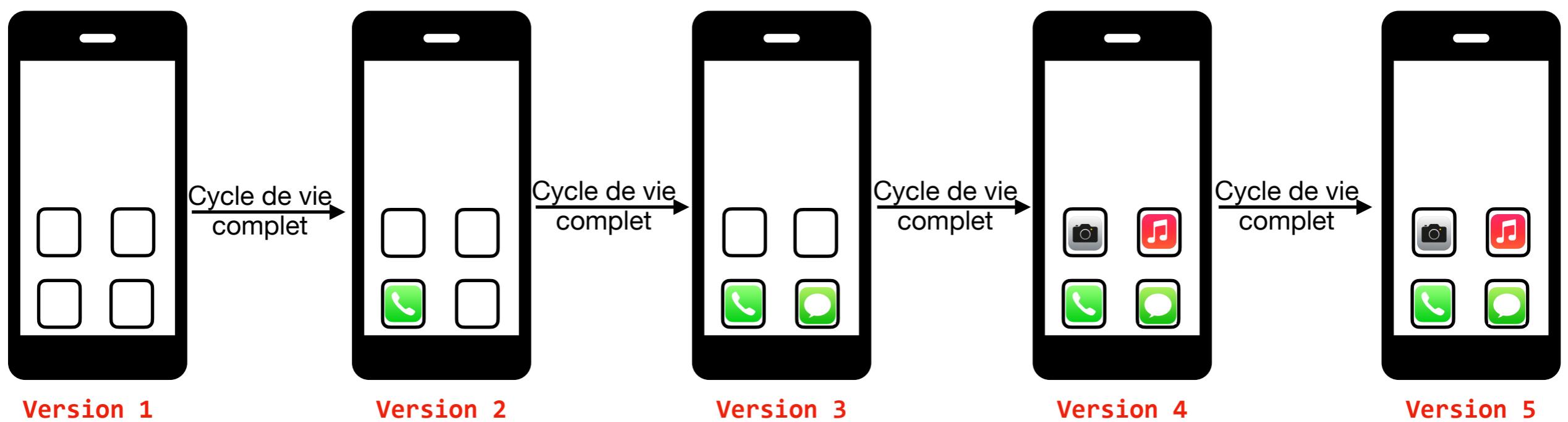
## Modèle évolutif - Incrémental

- Intégration du client dans le développement
- Concevoir et livrer une version minimale du système (sous-système)
- Ajout incrémental des fonctionnalités



# Cycle de vie

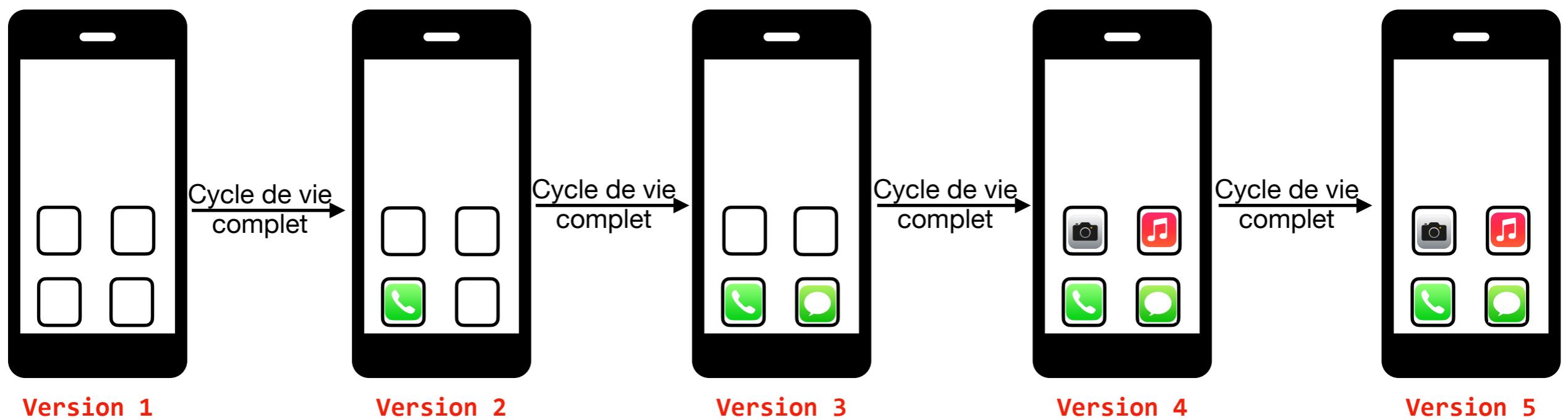
## Modèle évolutif - Itératif



# Cycle de vie

## Modèle évolutif - Itératif

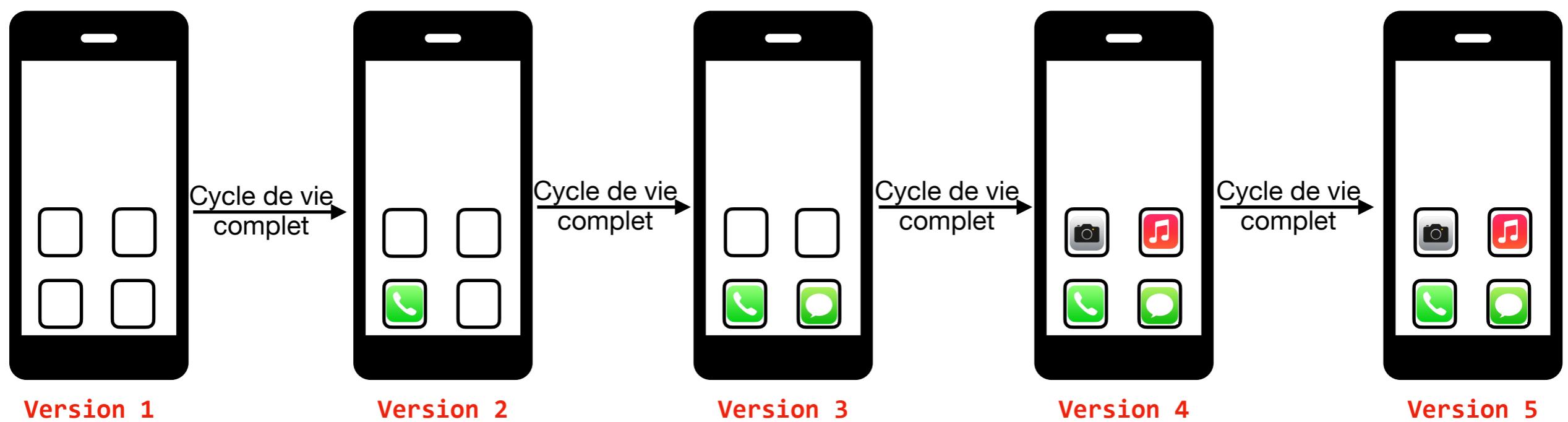
- Intégration du client dans le développement



# Cycle de vie

## Modèle évolutif - Itératif

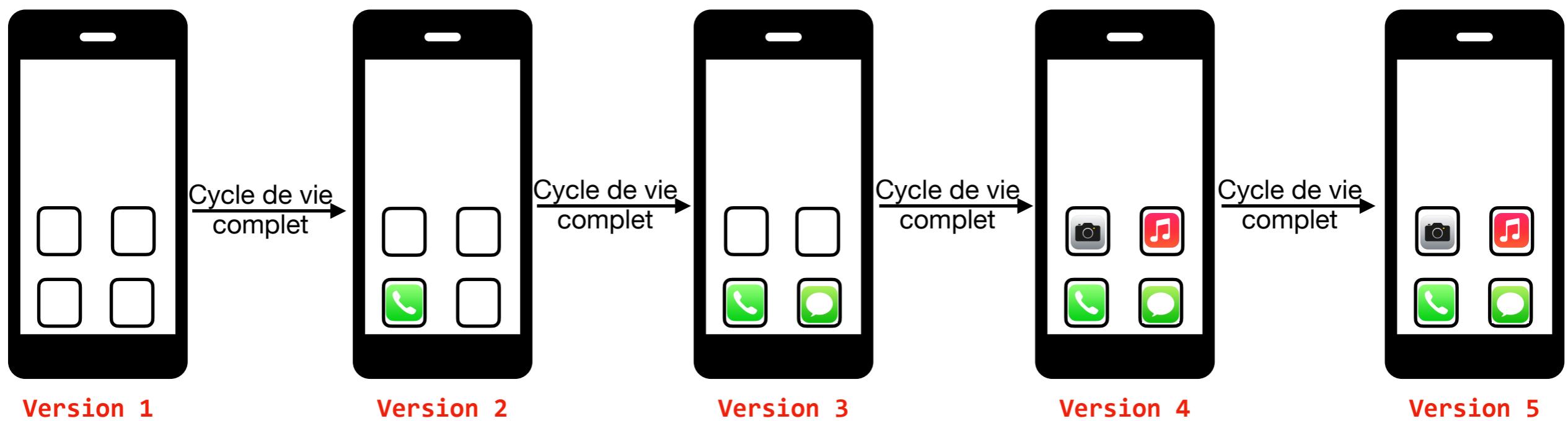
- Intégration du client dans le développement
- Concevoir et livrer une coquille complète du système (sous-système)



# Cycle de vie

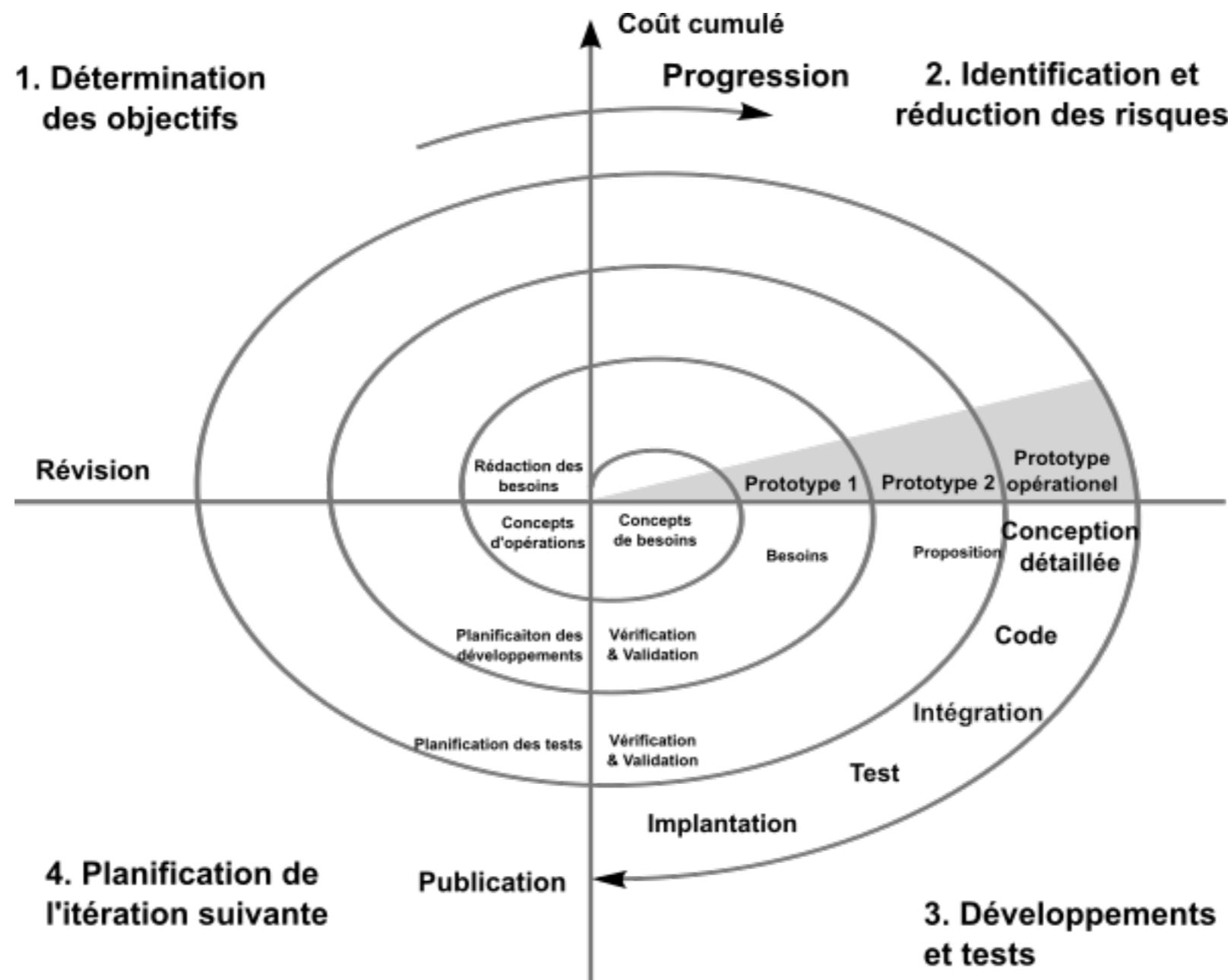
## Modèle évolutif - Itératif

- Intégration du client dans le développement
- Concevoir et livrer une coquille complète du système (sous-système)
- Chaque nouvelle version ajoute/améliore une fonctionnalité



# Cycle de vie

## Modèle en Spirale(Boehm, 1988)



# Méthodes Agiles

## Approche Agile

# Méthodes Agiles

## Approche Agile

- Une forte implication du client,

# Méthodes Agiles

## Approche Agile

- Une forte implication du client,
- Des livraisons fréquentes des versions

# Méthodes Agiles

## Approche Agile

- Une forte implication du client,
- Des livraisons fréquentes des versions
- Prise en compte des changements de spécifications en cours de projet

# Méthodes Agiles

## Approche Agile

- Une forte implication du client,
- Des livraisons fréquentes des versions
- Prise en compte des changements de spécifications en cours de projet
- Basées sur des modèles évolutifs (itératif / incrémental)

# Méthodes Agiles

## Approche Agile

- Une forte implication du client,
- Des livraisons fréquentes des versions
- Prise en compte des changements de spécifications en cours de projet
- Basées sur des modèles évolutifs (itératif / incrémental)
- Des cycles de vie relativement courts

# Méthodes Agiles

## Approche Agile

- Une forte implication du client,
- Des livraisons fréquentes des versions
- Prise en compte des changements de spécifications en cours de projet
- Basées sur des modèles évolutifs (itératif / incrémental)
- Des cycles de vie relativement courts

# Méthodes Agiles

## Approche Agile

- Une forte implication du client,
- Des livraisons fréquentes des versions
- Prise en compte des changements de spécifications en cours de projet
- Basées sur des modèles évolutifs (itératif / incrémental)
- Des cycles de vie relativement courts
- Méthodes les plus connues :

# Méthodes Agiles

## Approche Agile

- Une forte implication du client,
- Des livraisons fréquentes des versions
- Prise en compte des changements de spécifications en cours de projet
- Basées sur des modèles évolutifs (itératif / incrémental)
- Des cycles de vie relativement courts
- Méthodes les plus connues :
  - La méthode XP (eXtreme Programming)

# Méthodes Agiles

## Approche Agile

- Une forte implication du client,
- Des livraisons fréquentes des versions
- Prise en compte des changements de spécifications en cours de projet
- Basées sur des modèles évolutifs (itératif / incrémental)
- Des cycles de vie relativement courts
- Méthodes les plus connues :
  - La méthode XP (eXtreme Programming)
  - La méthode SCRUM

# Méthodes Agiles

## Approche Agile

- Une forte implication du client,
- Des livraisons fréquentes des versions
- Prise en compte des changements de spécifications en cours de projet
- Basées sur des modèles évolutifs (itératif / incrémental)
- Des cycles de vie relativement courts
- Méthodes les plus connues :
  - La méthode XP (eXtreme Programming)
  - La méthode SCRUM
  - La méthode RAD (Rapid Application Development)

# Méthodes Agiles

## Approche Agile

- Une forte implication du client,
- Des livraisons fréquentes des versions
- Prise en compte des changements de spécifications en cours de projet
- Basées sur des modèles évolutifs (itératif / incrémental)
- Des cycles de vie relativement courts
- Méthodes les plus connues :
  - La méthode XP (eXtreme Programming)
  - La méthode SCRUM
  - La méthode RAD (Rapid Application Development)
  - La méthode DSDM (Dynamic Systems Development Method)

# Modèles et méthodes

## Standish group, Chaos Report 2016

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
ALL SIZE PROJECTS	AGILE	40%	51%	9%
	WATERFALL	13%	60%	27%
	OTHER (AINO)	32%	52%	16%
LARGE SIZE PROJECTS	AGILE	21%	60%	19%
	WATERFALL	4%	56%	40%
MEDIUM SIZE PROJECTS	AGILE	30%	60%	10%
	WATERFALL	9%	69%	22%
SMALL SIZE PROJECTS	AGILE	55%	40%	5%
	WATERFALL	47%	40%	13%

# Références

## Books

- **UML Distilled (Third Edition): A Brief Guide to the Standard Object Modeling Language.** M Fowler 2004.
- **Object-Oriented Software Engineering (Second Edition): Practical Software Development Using UML and Java.** T. Lethbridge and R. Laganière 2005.
- **UML in Practice: The Art of Modeling Software Systems Demonstrated through Worked P. Rogues 2004.**
- **Requirements Engineering: From System Goals to UML Models to Software Specifications.** A. Lamsweerde 2009.
- **Software Engineering with UML.** B. Unhelkar 2018.

# Many

## Thanks to

- Arnaud Gotlieb, SIMULA Research Lab., Oslo, Norway
- Christine Solnon, CITI, INSA Lyon
- Delphine Longuet, LRI, Paris-Sud ([youtube channel](#))
- Keunhyuk Yeom, Pusan Univ
- Pierre Gérard, Paris 13