

**UNIVERSITE ADVENTISTE COSENDI**

*Faculté de Gestion et Informatique*

*Année Académique 2021-2022*

# Outils et Ateliers Génie Logiciels

© Daniel KAMGA

kamgadaniel2007@yahoo.fr

# Contenu du Cours

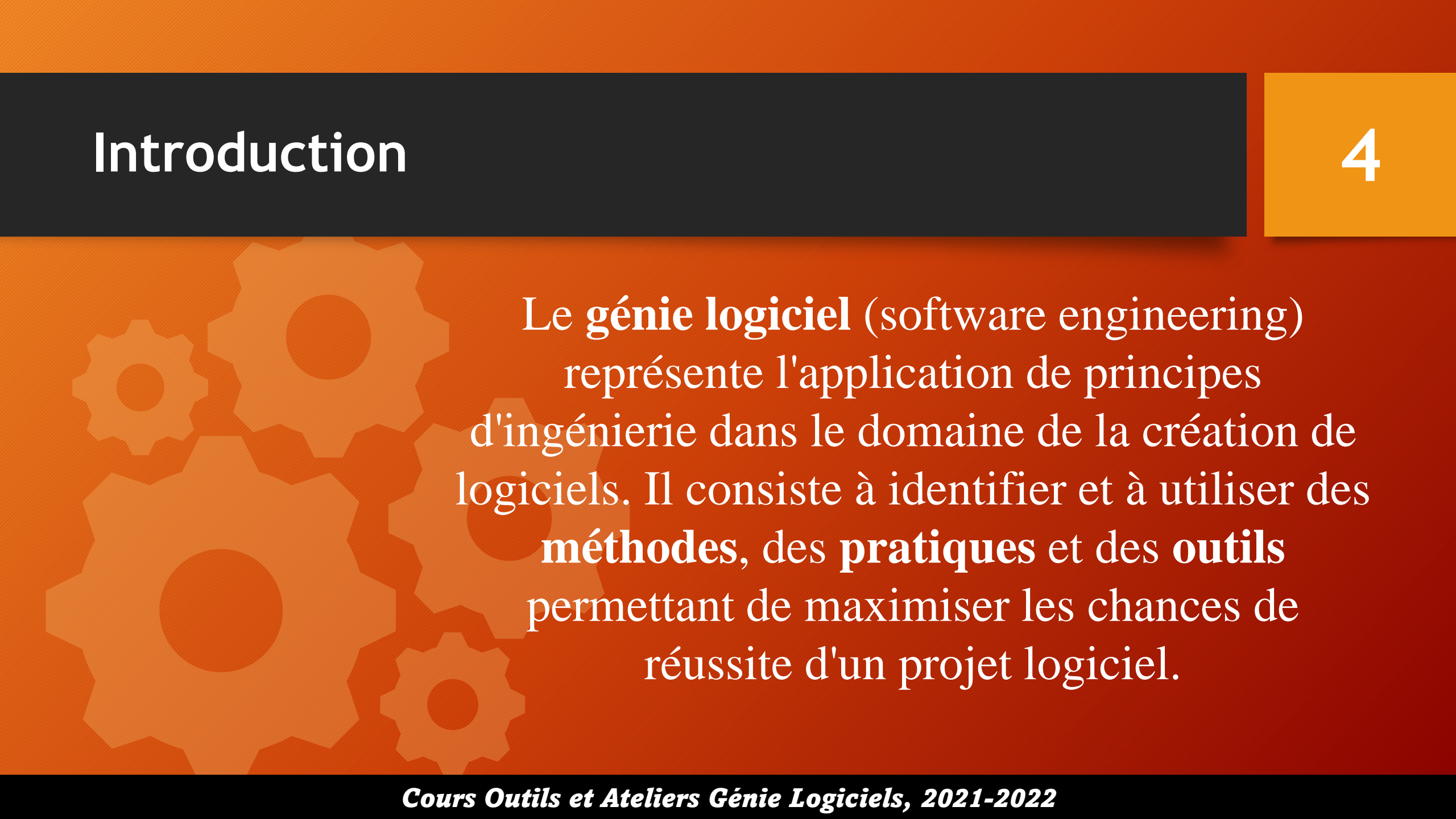
2

- ❑ **Rappel sur le génie logiciel**
- ❑ **Définition et utilité des OAGL**
- ❑ **Classification des OAGL**



# **1. Rappel sur le génie logiciel**

- **Introduction**
- **Définitions**
- **Qualités attendues d'un logiciel**
- **Cycle de vie d'un logiciel**
- **Modèles de cycle de vie d'un logiciel**



Le **génie logiciel** (software engineering) représente l'application de principes d'ingénierie dans le domaine de la création de logiciels. Il consiste à identifier et à utiliser des **méthodes**, des **pratiques** et des **outils** permettant de maximiser les chances de réussite d'un projet logiciel.




# Introduction (con't)

5

Il s'agit d'une science récente dont l'origine remonte aux années 1970. A cette époque, l'augmentation de la puissance matérielle a permis de réaliser des logiciels plus complexes mais souffrant de nouveaux défauts:

- Délais non respectés
- Coûts de production et d'entretien élevé
- Manque de fiabilité et de performances

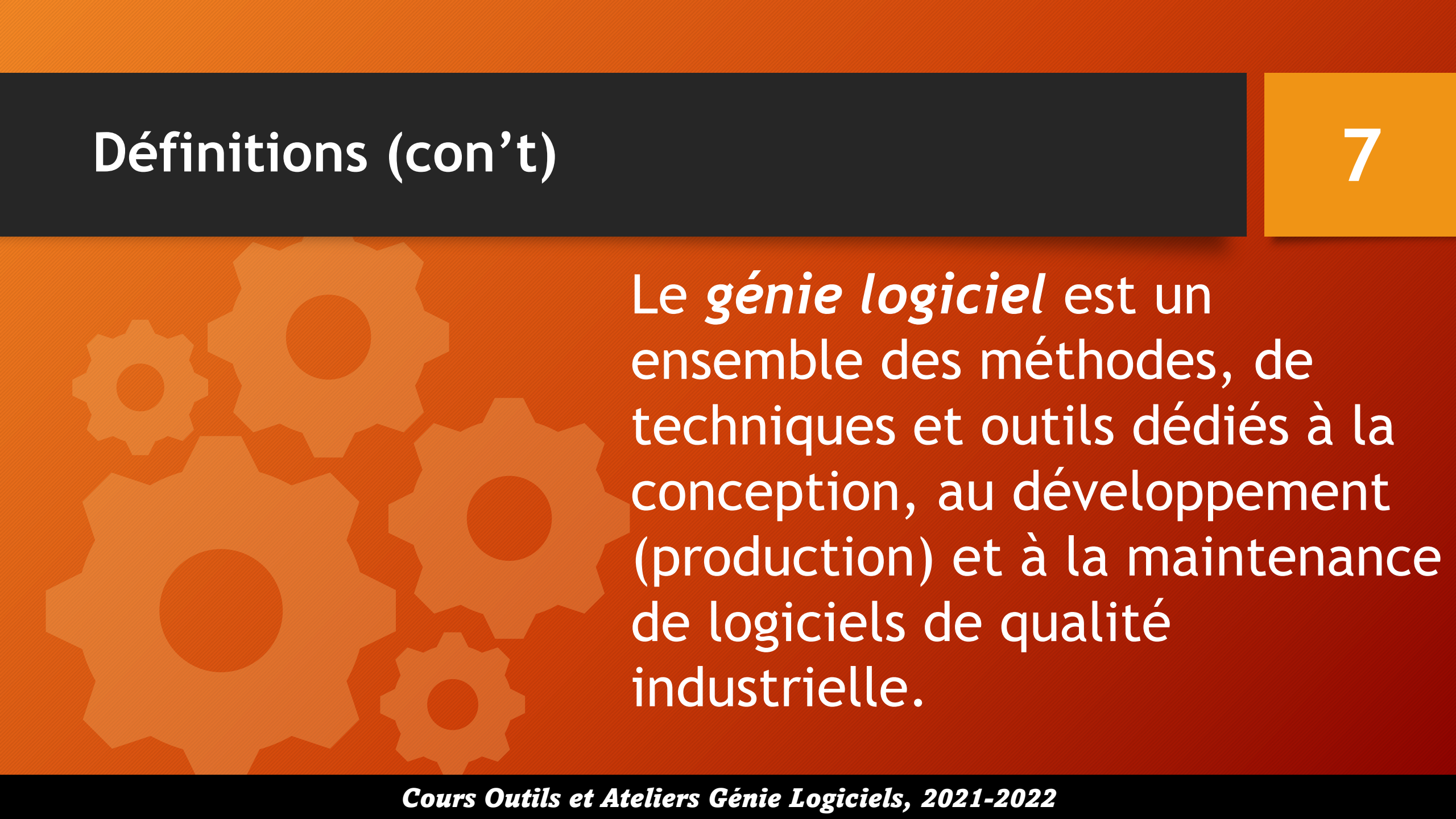


***Un logiciel*** est un ensemble d'entités nécessaires au fonctionnement d'un processus de traitement automatique de l'information. Parmi ces entités, on trouve par exemple : des programmes (en format code source ou exécutables); des documentations d'utilisation; des informations de configuration



# Définitions (con't)

7



Le *génie logiciel* est un ensemble des méthodes, de techniques et outils dédiés à la conception, au développement (production) et à la maintenance de logiciels de qualité industrielle.

# Qualités attendues d'un logiciel

8

- ✓ **Validité** : réponse aux besoins des utilisateurs
- ✓ **Facilité d'utilisation** : prise en main et robustesse
- ✓ **Performance** : temps de réponse, débit, fluidité
- ✓ **Fiabilité** : 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



# Cycle de vie d'un logiciel

9

Le cycle de vie du logiciel comprend généralement les activités suivantes :

- **Définition des besoins (Pourquoi?)**

Production: Cahier des charges, Fonctionnalités attendues, etc...

- **Analyse des besoins / spécification (Quoi?)**

Production: Dossier d'analyse, Ébauche du manuel Utilisateur, etc...

- **Codage et tests unitaires (Comment ?)**

Production: Modules codés et testés, Documentation de chaque module

- **Vérification et validation**

Production: Logiciel testé, Jeu de tests de non-régression, etc...

- **Maintenance**

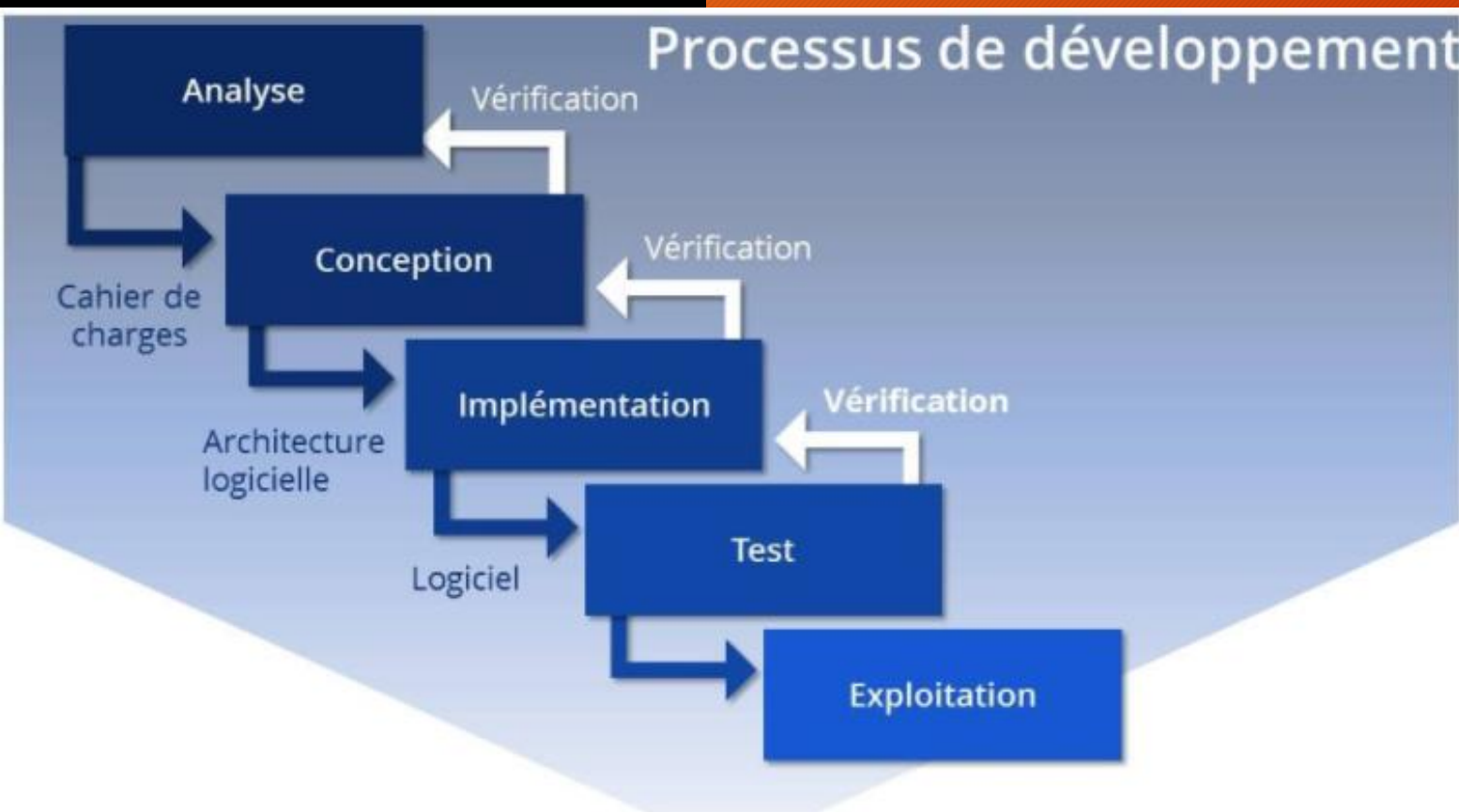
Production: Logiciel corrigé, Mises à jour, patch etc...



# Modèles de cycle de vie d'un logiciel

11

## Modèle en Cascade



Le modèle en cascade (en anglais : *waterfall model*) est un **modèle de gestion linéaire** qui divise les processus de développement en phases de projet successives.

**1. Analyse** : planification, analyse et spécification des besoins

**2. Conception** : conception et spécification du système

**3. Implémentation** : programmation et tests des modules

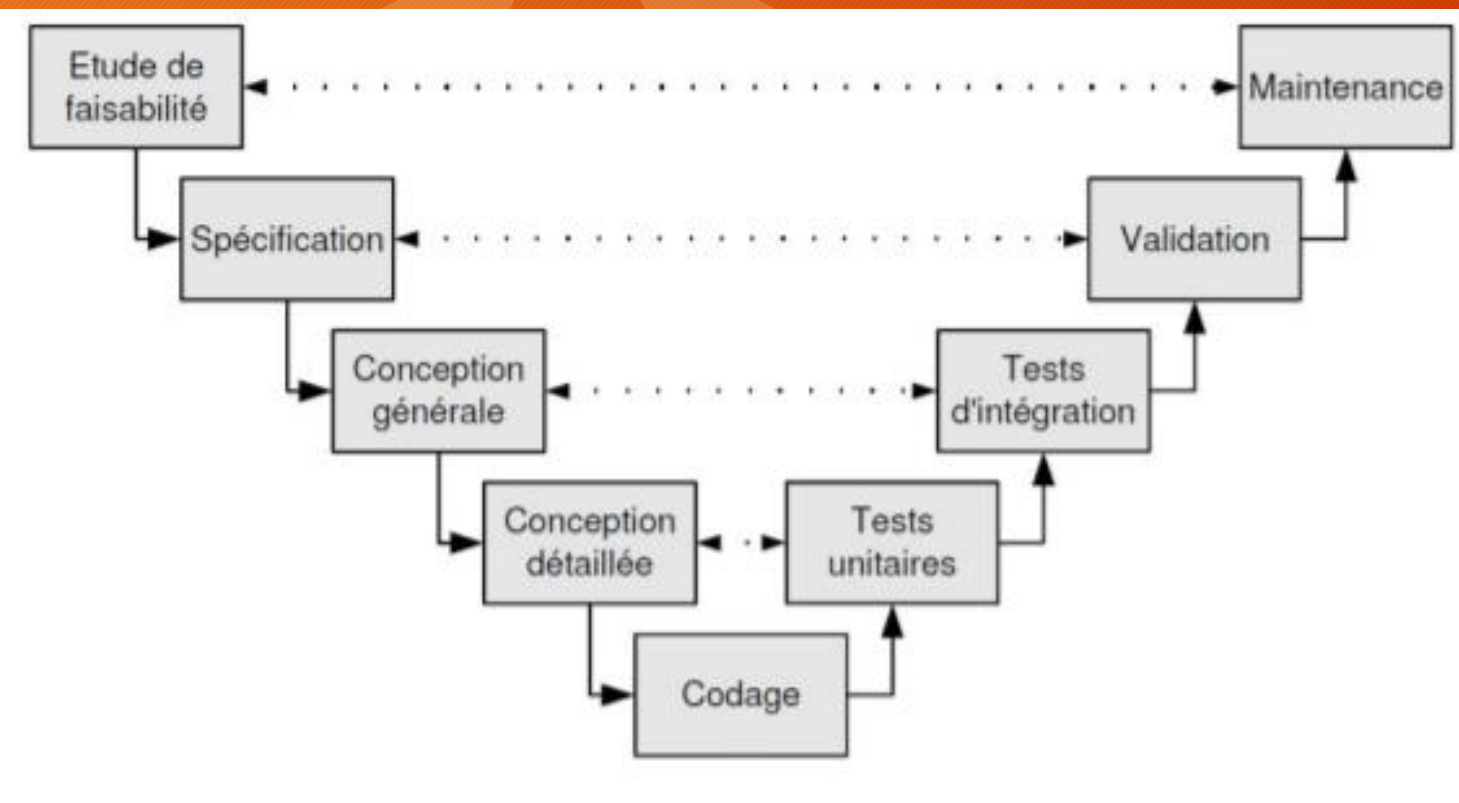
**4. Test** : intégration du système, tests du système et de l'intégration

**5. Exploitation** : livraison, maintenance, amélioration

# Modèles de cycle de vie d'un logiciel

# 12

## Modèle en V



Le cycle en V ou V model en anglais est un modèle utilisé dans différents processus de développement, notamment dans le **développement de logiciels**. le cycle en V définit parallèlement les démarches afférentes à mettre en place en termes d'assurance qualité et détaille la façon dont les différentes phases doivent interagir entre elles.

la particularité de ce modèle de gestion de projet est de combiner une phase de validation pour chaque phase de développement.



## 2. Définition et utilité des OAGL

- **Introduction**
- **Définitions**
- **Utilité OAGL**

*Les OAGLs* sont des outils qui fournissent une assistance automatisée pour le développement de logiciels. L'objectif étant:

- ❑ **La réduction du temps et des coûts de développement logiciel**
- ❑ **L'amélioration de la qualité des systèmes développés.**



# Introduction (con't)

15

Seulement 15 % des applications écrites sont mises en place et fonctionnent ! (selon des enquêtes réalisées aux USA)

**Pourquoi ?**

# Introduction (con't)

16

Ce chiffre s'explique par:

- ☐ Retards de livraisons
- ☐ Dépassements de budgets
- ☐ Technologies en mutation
- ☐ Complexité des applications

D'où le **génie logiciel** et par extension les **outils et ateliers génie logiciels**



# Définitions

17

Un atelier de génie logiciel (noté AGL ou en anglais CASE, pour *Computer Aided Software Environment*) est un ensemble d'outils logiciels structurés au sein d'une même interface, permettant la conception, le développement et le débogage de logiciels.

Un AGL comprend ainsi des outils permettant de:

- modéliser visuellement une application,
- produire du code avec des assistants visuels,
- tester le code (débugueur) produit.

Tout ***logiciel*** pouvant être utilisé à un ***niveau quelconque***  
***dans le processus de production d'un logiciel***



# Définitions (con't)

19

Un outils génie logiciel (noté OGL) est un ensemble d'outils logiciels structurés au sein d'une même interface, permettant la gestion de la production de logiciels.

Pour les besoins de ce cours,  
nous utilisons l'acronyme OAGL  
pour désigner à la fois:

- ☐ **AGL,**
- ☐ **OGL**



Les OAGL visent les objectifs suivant:

- **Faciliter la méthodologie de conception unique:** les OAGL aident l'organisation à standardiser le processus de développement. Ils facilitent également un développement coordonné. L'intégration devient facile à mesure que la méthodologie commune est adoptée
- **Développement rapide d'applications :** améliorer la vitesse et la qualité du développement des logiciels

# Utilité des OAGL (con't)

22

- **Tests** : aide à l'amélioration du processus de test grâce à une vérification automatisée et à une maintenance simplifiée du programme.
- **Gestion de projet** : améliore l'activité de gestion de projet et automatise dans une certaine mesure diverses activités impliquées dans la gestion de projet.
- **Réduire les coûts de maintenance**
- **Augmenter la productivité** : l'automatisation de diverses activités de développement de systèmes et de processus de gestion augmente la productivité de l'équipe de développement



### 3. Classification des OAGL

- Critères de classification
- Avantages et Inconvénients des OAGL
- Recommendations

Il existe une multitude d'OAGL qui peuvent être classifiés selon différents critères.

Nous nous limiterons aux critères suivants:

- **Niveau du cycle de vie du projet**
- **Type de Licence**
- **Domaine d'application spécifique**



## ➤ Niveau du cycle de vie du projet

Ici on distingue deux classes selon la nature des outils intégrés:

- ❖ **Les OAGL de haut niveau (upper-case):** ces OAGL s'intéressent plus particulièrement aux phases d'analyse et de conception du processus logiciel.
- ❖ **Les OAGL de bas niveau (lower-case):** ces ateliers s'intéressent plus particulièrement aux phases d'implémentation et de test du processus logiciel

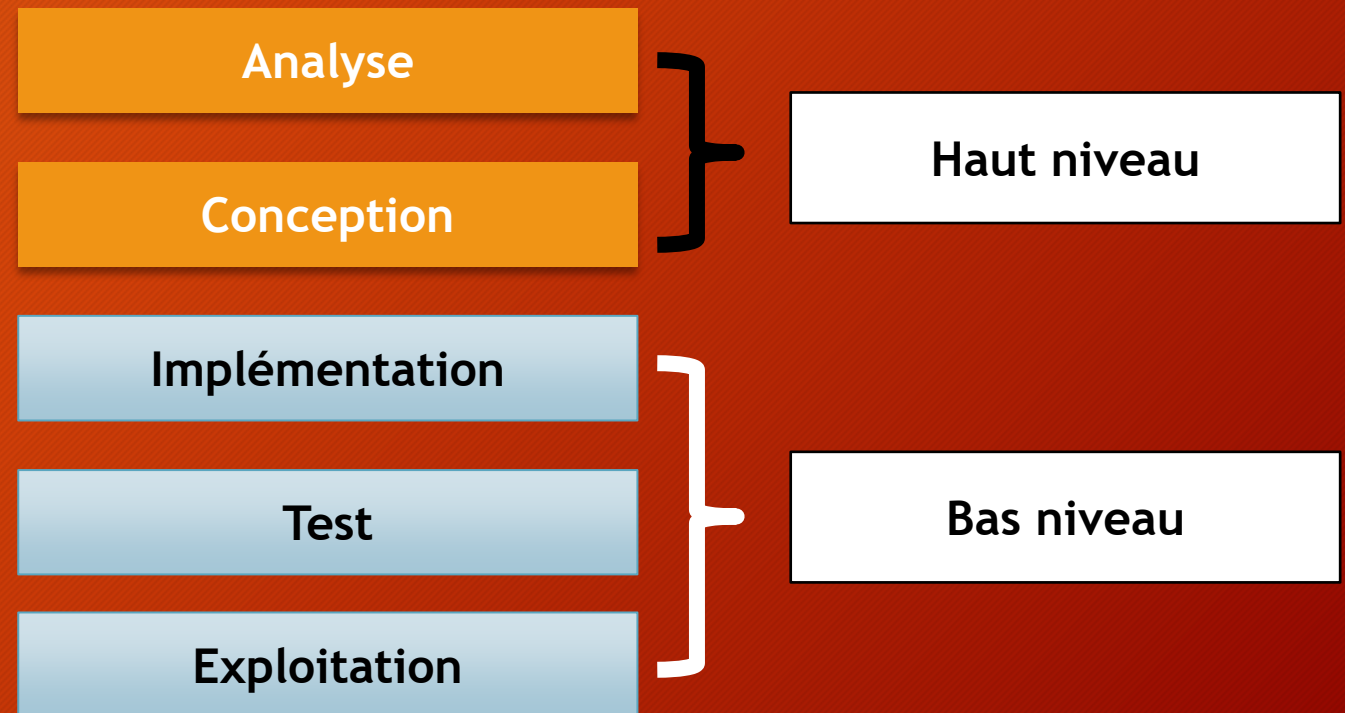
# Critères de classification (con't)

26

## ➤ Niveau du cycle de vie du projet

Exemples: Outils de modélisation, outils de gestion de projets, outils d'analyse, outils de conception etc..

Exemples: Outils de programmation, outils de prototypage, outils de développement web, outils d'assurance qualité etc...





# Critères de classification (con't)

27

## ➤ Type de licence

Exemples: ArgoUML,  
StarUML, Eclipse etc...

### **Licence Libre**

- Open source,
- Non open source,

Exemples: WinDev, Teamwork,  
etc...

### **Licence Payante**

# Critères de classification (con't)

28

## ➤ Domaine d'application spécifique

- **Applications Web:** Django, Symfony, React, Angular etc ...
- **Applications Mobiles:** Android Studio, Ionic, flutter



# Avantages et Inconvénients des OAGL

29

## Avantages

- ☐ Vitesse accrue
- ☐ Précision accrue
- ☐ Aide à la standardisation des notations et des diagrammes
- ☐ Aide à la communication entre les membres de l'équipe de développement
- ☐ Aide à la vérification automatiquement de la qualité des modèles
- ☐ Réduction du temps et des efforts
- ☐ Améliorer la réutilisation des modèles ou des composants de modèles

## Inconvénients

- ☐ L'outil n'est pas une méthode; il aide, mais ne peut pas tout faire.
- ☐ Coûts liés à l'utilisation de l'outil :  
achat + formation.



- Couverture du cycle de vie,
- Flexibilité de changement de l'outils,
- Le coût
  - Acquisition,
  - De prise en main (formation, ...).
- Documentation et aide
  - Manuels, tutoriaux,
  - Support techniques,
  - Communauté des utilisateurs,
  - Forums
- Portabilité (systèmes et plateformes).