Bloc 2-Génie Logiciel (GRP 1)

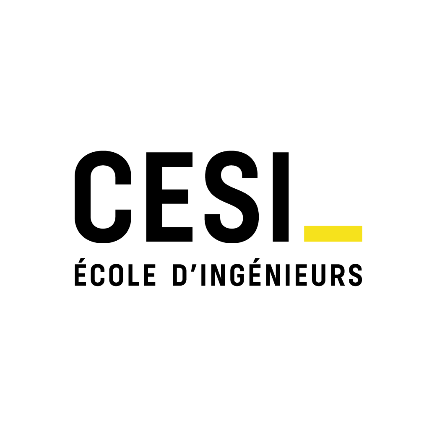
Table des matières

[**Introduction**](#_Toc188871703)

[**ENJEUX**](#_Toc188871704)

[**Les méthodes d'ingénierie des systèmes logiciels**](#_Toc188871705)

[**Les bonnes pratiques**](#_Toc188871706)



# Synthèse

## Introduction

Le génie logiciel, ou software engineering, se définit comme l'application structurée et méthodique des principes d'ingénierie au développement de logiciels. Il vise à concevoir, développer et maintenir des applications répondant à des besoins spécifiques tout en garantissant leur qualité, leur fiabilité et leur évolutivité.

## ENJEUX

Le génie logiciel repose sur des enjeux clés : garantir une qualité élevée avec des logiciels fiables, performants et maintenables ; respecter les délais et budgets grâce à des méthodologies adaptées ; assurer la sécurité des données et la résilience aux cyberattaques ; s’adapter aux évolutions technologiques et réglementaires ; et favoriser une collaboration efficace entre équipes grâce à une communication et une documentation claire.

Un mauvais contrôle des enjeux du génie logiciel peut entraîner des défaillances, une insatisfaction des utilisateurs et des pertes financières. Les dépassements de délais ou de budgets nuisent à la rentabilité, tandis qu’une sécurité insuffisante expose les données à des cyberattaques. Le manque d’adaptabilité ou une collaboration inefficace peut rendre un projet obsolète ou coûteux, compromettant son succès.

## Les méthodes d'ingénierie des systèmes logiciels

Les méthodes d’ingénierie des systèmes logiciels structurent le développement pour répondre aux besoins des projets. Les approches traditionnelles, comme le cycle en V, garantissent une gestion rigoureuse mais manquent de flexibilité face aux changements. Les méthodes agiles, en revanche, privilégient des cycles courts et adaptatifs pour intégrer les retours en continu. L’ingénierie dirigée par les modèles simplifie les systèmes complexes en réduisant les erreurs et en optimisant le développement. Le choix de la méthode dépend des exigences du projet et de son environnement.

## Les bonnes pratiques

* Architecture du logiciel en amont : Séparation du code en plusieurs fichiers.
* Gestion de projet : UML, cycle en V, agile, test (unitaire, régression).
* Code : Choix de l’IDE, de l’environnement, DOCUMENTATION, versioning.
* Partage : Licence du logiciel (APP, auteurs) libre ou non.
* Gestion de version : Bien mettre la version dans les documents, utiliser une forge (GitLab, GitHub, etc.).