ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Chapitre 5
Qualité logicielle

1I2AC1 : Génie logiciel et Conception par objet

Régis Clouard, ENSICAEN - GREYC

« Le débogage est deux fois plus difficile que l'écriture de code.
Donc, si vous écrivez du code de la manière
la plus intelligente possible, vous n'êtes, par définition,
pas assez intelligent pour le déboguer. »
Brian Kernighan

Objectif du chapitre

- Présentation de la notion de qualité logicielle avec un focus particulier sur le point de vue du développeur.
- A l'issue de ce chapitre :
  - Vous serez sensibilisé à la notion de qualité de code.
  - Vous saurez tirer partie des indicateurs de qualité pour renforcer la qualité de vos codes.

Plan du chapitre

1
Notion de qualité logicielle

2
La qualité du point de vue du développeur

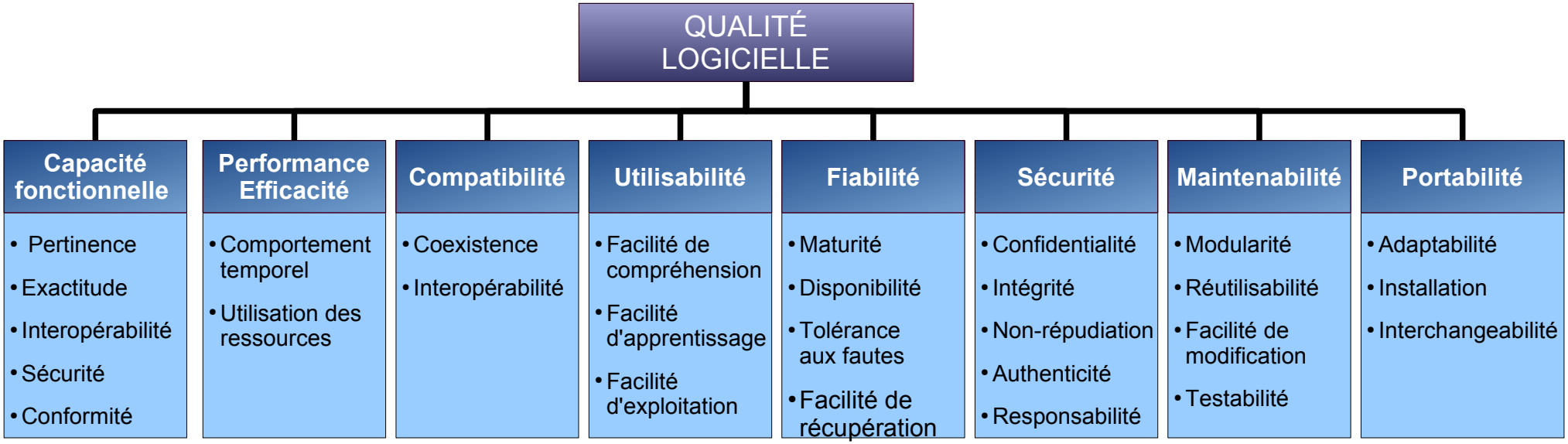
3
Aides au développement de qualité

Définition de la qualité logicielle

- Définition
  - La qualité englobe l'ensemble des caractéristiques d'un logiciel qui affecte sa capacité à satisfaire des besoins exprimés ou implicites en termes de fonctionnalités, délais et coûts.
- Cas particulier du logiciel
  - Pas de définition formelle.
  - Pas de mesure objective.
  - Pas de fiabilité prédictible.
- La qualité d'un logiciel dépend entièrement :
  - de sa construction
  - et des processus utilisés pour son développement.
- Une appréciation globale de la qualité tient autant compte :
  - des facteurs *extérieurs*, directement observables par l'utilisateur,
  - que des facteurs *intérieurs*, observables par les ingénieurs de développement.

Les normes

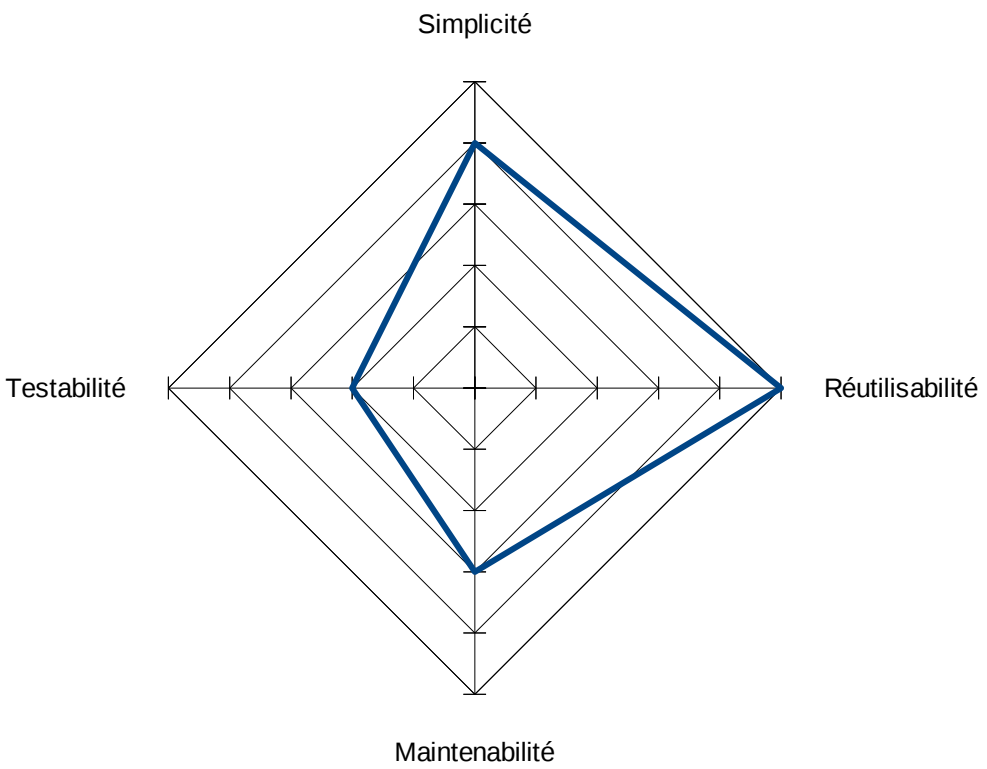
- La qualité du produit du point de vue de sa construction
  - ISO/CEI 25010 : décrite par 8 aspects et 35 caractéristiques.



- La qualité du point de vue du processus de développement
  - La norme ISO/CEI 90003 : fournit des lignes directrices pour l'application d'un système de management de la qualité au développement du logiciel.
  - La norme ISO/CEI 12207 : décrit un modèle pour le processus de cycle de vie du logiciel.

Le point de vue des développeurs

- Dans ce cours, la qualité logicielle du point de vue des ingénieurs de développement (point de vue interne).
  - Elle se restreint à quatre caractéristiques principales portant sur le code :
    1) Simplicité
    2) Testabilité
    3) Maintenabilité
    4) Réutilisabilité



Caractéristiques de la qualité logicielle

- Les caractéristiques recherchées sur le code à produire :
  1) Simplicité (principe agile *KISS* : *Keep It Simple, Stupid*)
     - Mesure le degré d'évidence du code.
     - C'est probablement la qualité qui conditionne toutes les autres.
  2) Testabilité
     - Mesure la facilité à automatiser des tests pour le logiciel.
  3) Maintenabilité
     - Mesure l'effort nécessaire pour corriger ou faire évoluer le logiciel.
  4) Réutilisabilité
     - Mesure l'aptitude d'un logiciel à être réutilisé, en tout ou en partie, dans de nouvelles applications.
     - Une entreprise réutilise beaucoup de ses codes.

Aide à la qualité du code

- Comment maximiser les valeurs de ces caractéristiques ?
  - Des environnements de développement professionnels.
  - Des métriques techniques qui donnent des indices.
  - Respecter des principes et des pratiques.

Environnement de développement

- Le développement agile nécessite des environnements de développement sophistiqués.
  - Environnement de Développement Intégré (p. ex. Intellij)
    - Permet une édition avancée du code.
    - Permet la compilation interactive.
    - Permet le management de code.
  - Logiciel de gestion de versions (p. ex. Git – Voir *gitlab.ecole.ensicaen.fr*).
    - Partage de la production.
    - Suivi de versions.
  - Moteur de production (ie super makefile) (p. ex. Gradle / Maven).
    - Compilation.
    - Déploiement.

Métriques

- En Java, cinq outils d'analyse statistique de la qualité du code sont largement utilisés :
  - **SonarQube** : mesurer la qualité du code source en continu.
  - **Checkstyle** : vérification des conventions et normes de codage.
  - **PMD** : similaire à Checkstyle mais plus focalisé sur les problèmes potentiels de codage comme le code non utilisé ou sous-optimisé, la taille et la complexité du code, et les bonnes pratiques de codage.
  - **FindBugs** : détection des bogues potentiels, des problèmes de performances, ou des mauvaises habitudes de codage.
  - **Cobertura** : outil d'analyse de couverture des tests. Il calcule le pourcentage de code couvert par les tests.
- Les IDE (p.ex. IntelliJ IDEA).
  - Voir menu Analyze.
  - Possibilité d'ajouter les outils précédents sous forme de plugins.

Principes et pratiques

- Les métriques ne sont que des alarmes qui doivent être examinées pour décider si le code doit être changé.
- **Artisanat du code « software craftsmanship »**
  - Approche du développement de logiciels qui met l'accent sur les compétences de codage des développeurs.
  - Il ne suffit pas qu'un logiciel soit fonctionnel, mais il faut qu'il soit bien conçu.
- Principes et pratiques issues des méthodes agiles (principalement eXtreme-Programming).
  - Code propre.
    - Responsabilité collective du code.
  - Test
    - Développement dirigé par les tests.
  - Refactoring
    - Conception simple.

Que retenir de ce chapitre

- La qualité logicielle est impossible à mesurer.
- Il y a bien des normes mais elle n'identifient que des aspects à évaluer sans mesures objectives.
  - Du point de vue des développeurs, les caractéristiques sont principalement :
    - Simplicité, Testabilité, Maintenabilité et Réutilisabilité.
- Les mesures dont on dispose que ne sont que des alarmes.
- Seuls le processus et les principes peuvent mener à la qualité.
  - Processus
    - Développement agile.
  - Principes
    - Code propre et code testé.