ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

05 Chapitre

Qualité logicielle

1I2AC1 : Génie logiciel et Conception orientée 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 de l'ingénieur du développement.
- A l'issue de ce chapitre :
  - Vous vous rendrez compte que, pour le développeur, la qualité logicielle résulte plus d'une démarche artisanale que de l'application de normes.
  - 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

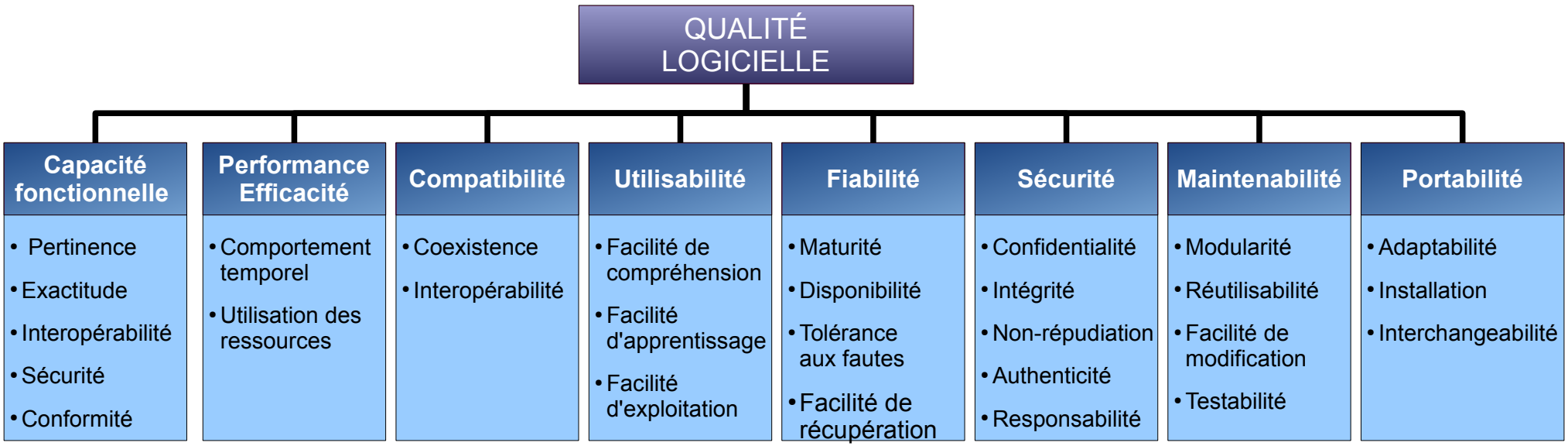
3
Aides au développement de qualité

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 structure et de son organisation.
  - des processus utilisés pour son développement.
- Une appréciation globale de la qualité tient autant compte :
  - des facteurs extérieurs, directement observables par l'utilisateur,
  - des facteurs intérieurs, observables par les ingénieurs de développement.

Les normes

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



Les normes

- La qualité du logiciel du point de vue de sa construction.
  - La norme ISO/CEI 90003 : fournit des lignes directrices pour l'application d'un système de management de la qualité 9001 au développement du logiciel.
  - La norme ISO/CEI 12207 : décrit un modèle pour le processus du cycle de vie du logiciel.
- Bilan
  - Ces normes sont surtout soucieuses de la satisfaction client.
  - Elles se présentent essentiellement comme des facteurs à analyser lors du processus de développement et sur le produit fini.

Le point de vue des développeurs

- Du point de vue des ingénieurs de développement (point de vue interne), la qualité logicielle est envisagée selon quatre caractéristiques principales portant sur le code :
  1) **Simplicité** (principe *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 ses codes.

Au delà des normes

- Pour aider à la qualité logicielle, il faut que le développeur utilise :
  - Un environnement de développement professionnel.
  - Un analyseur de code.
  - Des métriques globales.

Environnement de développement

- Le développement nécessite des environnements de développement sophistiqués.
  - IDE : 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.

Analyseurs de code

- En Java, quatre outils d'analyse statistique de la qualité du code sont largement utilisés :
  - **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.
  - **SonarLint** : similaire à Checkstyle et FindBugs.
- Analyse de code dans IntelliJ IDEA.
  - Voir menu Analyze::inspect code.
  - Il faut le configurer avec ses choix (ceux de l'école : voir la plateforme).
  - Possibilité d'ajouter les outils précédents sous forme de plugins.

Métriques

- En Java, deux outils de calcul de métriques de la qualité du code sont largement utilisés :
  - **SonarQube** : mesurer la qualité du code source en continu.
  - **Cobertura** : outil d'analyse de couverture des tests. Il calcule le pourcentage de code couvert par les tests.
- Métriques dans IntelliJ IDEA.
  - Possibilité d'ajouter les outils précédents sous forme de plugins.

Limite des outils

- Les erreurs détectées par les outils d'analyse de code et les métriques ne sont pas la panacée :
  - Elles ne sont que des alarmes qui doivent être examinées pour décider si le code doit être changé.
  - Elles ne permettent pas de graduer de la qualité du code.
- Il est donc nécessaire en plus d'adopter un comportement volontariste pour rechercher la qualité logicielle.
  - Le développeur doit se considérer comme un artisan (***software craftsman***).

Artisanat du logiciel

- **Software craftsmanship**
  - Approche du développement qui met l'accent sur les compétences de codage du développeur.
  - Il ne suffit pas qu'un logiciel soit efficace, il faut en plus qu'il soit bien conçu.
  - L'efficacité résulte de la qualité du code et pas le contraire.
- Le développement est centré sur le code.
  - Suppression de la documentation annexe au maximum : s'il a besoin de documentation, c'est que le concept n'est pas clair.
  - Code propre : responsabilité collective du code.
  - Code testé : développement dirigé par les tests.
  - Code simple : refonte régulière du code (refactoring).

Que retenir de ce chapitre

- La qualité logicielle est impossible à mesurer.
- Il existe bien des normes mais elles se focalisent essentiellement sur la satisfaction client.
- Ici, nous nous intéressons à la qualité du point de vue des développeurs.
  - Les qualités recherchées selon ce point de vue sont :
    - Simplicité, Testabilité, Maintenabilité et Réutilisabilité.
- Pour cela :
  - Il faut utiliser des environnements de développement sophistiqués qui permettent d'obtenir des critiques du code.
- Mais ce n'est pas suffisant :
  - L'important est surtout d'adhérer à l'idée que le développement logiciel relève de l'artisanat. Le développeur est un artisan au sens premier du terme.