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 des méthodes agiles.
- 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

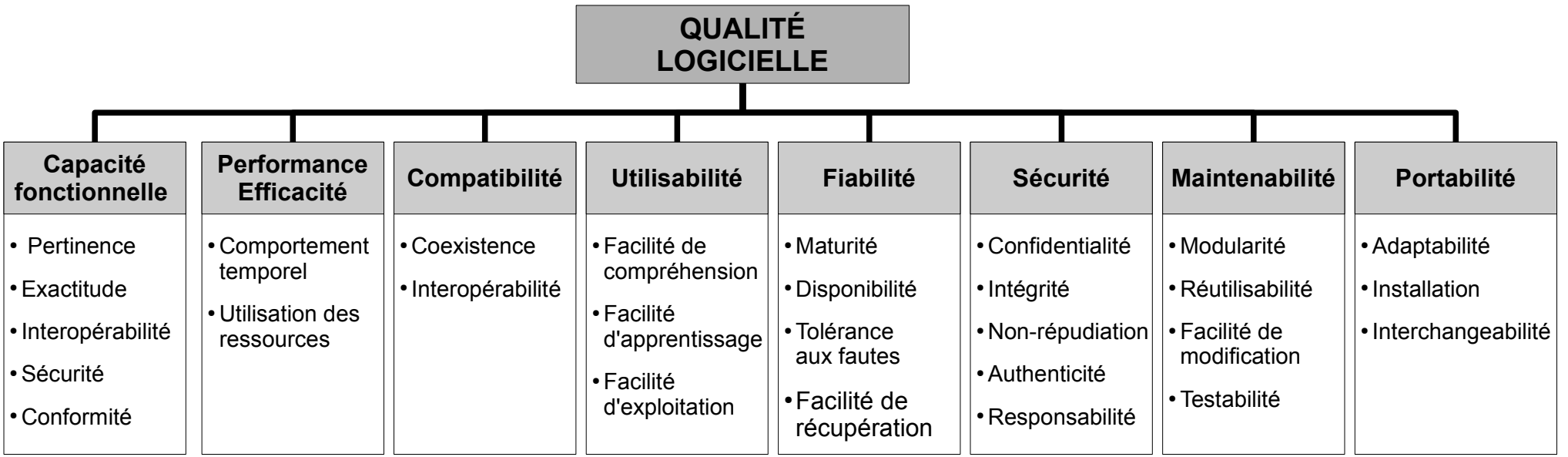
3
Aides au développement de qualité

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.
- 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.
- La qualité logicielle est définie par des normes.

Qualité : facteurs externes

- Normes
  - P. ex. : ISO/CEI 25010 : la qualité décrite par 8 aspects et 35 caractéristiques :



Qualité : facteurs internes

- Normes
  - P. ex. : 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.
  - P. ex. : La norme ISO/CEI 12207 : décrit un modèle pour le processus du cycle de vie du logiciel qui reprend globalement le cycle en V.

Bilan

- Concrètement, ces normes n'aident en rien la production de logiciel de qualité.
  - La norme ISO/CEI 25010 se présente comme une liste de caractéristiques à mesurer sur le produit fini.
    - Mais, en développement logiciel il n'existe pas d'indicateurs objectifs et incontestables pour mesurer la valeur de ces caractéristiques : cela reste empirique.
  - La norme ISO/CEI 12027 se base sur un cycle de développement en V.
    - Mais on sait que ce cycle n'offre pas les meilleures garanties de qualité en développement logiciel.

Point de vue de l'agilité

- Facteurs externes :
  - Collaboration avec le client qui est le juge de la qualité.
- Facteurs internes :
  - Plutôt que le respect de norme, l'agilité considère que la qualité d'un logiciel résulte de la qualité de son code.
    - Il ne suffit pas qu'un logiciel soit efficace, il faut en plus qu'il soit bien conçu.
  - En agilité, le développeur se voit comme un artisan du logiciel.

Artisanat du logiciel (*Software craftsmanship*)

- L'artisanat du logiciel met l'accent sur les compétences de codage des développeurs.
  - Le code est son chef-d'œuvre.
- Les facteurs de qualité du code à privilégier :
  - **Simplicité** (Principe KISS : *Keep it Simply, Stupid*)
    - Mesure le degré d'évidence du code.
  - **Testabilité**
    - Mesure la facilité à automatiser des tests pour le logiciel.
  - **Maintenabilité**
    - Mesure l'effort nécessaire pour corriger ou faire évoluer le logiciel.
  - **Réutilisabilité**
    - Mesure l'aptitude d'un logiciel à être réutilisé en partie pour développer de nouvelles applications.
    - Remarque : Une entreprise réutilise beaucoup ses codes (son patrimoine).

Atelier de l’artisan

- Pour aider à la qualité du code, il faut que l’équipe de développement utilise :
  - Un environnement de développement professionnel.
  - Des outils d’analyse de la propreté du code.

Environnement de développement

- Le développement nécessite des environnements de développement sophistiqués.
  - IDE : Environnement de Développement Intégré (p. ex. Intellij, CLion, Android Studio, etc)
    - 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.
  - Outils d'intégration continue (p. ex. gitlab CI)

Outils d'analyse de la propreté du code

- Exemple d'outils d'analyse de la propreté du code en Java :
  - **Checkstyle** : vérification des règles et conventions 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.
  - **SonarQube** : une combinaison de tous les outils précédents.
- Tous ces outils n'utilisent pas forcément les mêmes métriques. Plus il y a d'outils, plus il y a de chance de détecter des inconsistances.
- Nativement dans l'IDE IntelliJ IDEA :
  - Voir menu Analyze::inspect code.
  - Il faut le configurer avec ses choix (voir la plateforme).
  - 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 la qualité du code.

Que retenir de ce chapitre

- La qualité logicielle est impossible à mesurer.
- Les normes de qualité n'apportent aucune aide pour développer des logiciels de qualité.
- Si l'on s'intéresse à la qualité du point de vue des développeurs, les facteurs de qualité sont :
  - Simplicité, Testabilité, Maintenabilité et Réutilisabilité.
- En l'absence de mesure objective de ces facteurs, l'assurance qualité consiste à :
  - Adopter une posture d'« artisan du code » qui considère que la qualité du logiciel résulte de la qualité du code.
  - Utiliser des environnements de développement sophistiqués qui permettent d'obtenir des critiques de la qualité du code.