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

Définition de la qualité logicielle

- Définition
  - La **qualité** logicielle 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.
  - L'**assurance qualité** (QA) est l'ensemble des activités planifiées et systématiques de toutes les actions nécessaires pour fournir une assurance que la qualité du logiciel est conforme aux exigences et aux attentes établies.
  - Une appréciation globale de la qualité tient compte à la fois de :
    - Facteurs extérieurs directement observables par l'utilisateur
    - Facteurs internes observables par les ingénieurs du développement

La qualité logicielle est régie par des normes

Qualité : normes ISO 9000

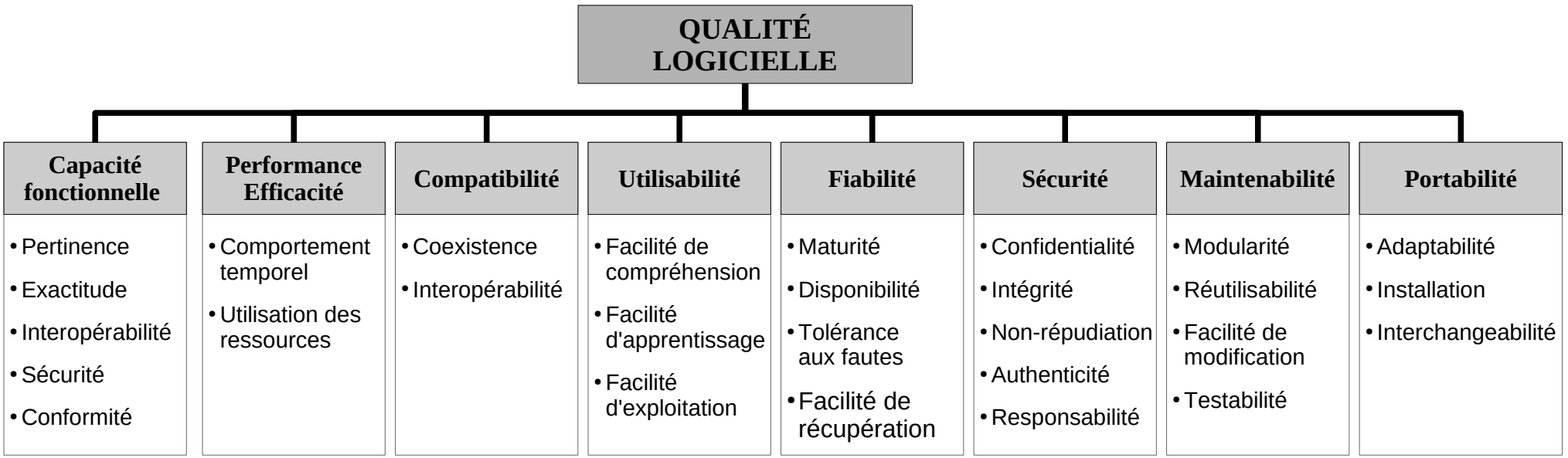
- Constituent un guide pour le management et l'organisation d'un projet (et plus généralement d'une entreprise).
- Donnent des éléments de base pour le pilotage d'un projet et des orientations reconnues comme étant les bonnes pratiques pour satisfaire les clients et s'améliorer.
- Déclinées pour le logiciel

Qualité logicielle : exemples de normes

- **ISO/CEI 12207 : Préconisation pour le cycle de vie du logiciel**
  - Cycle en V
  - Documentation exhaustive pour chaque phase

Qualité logicielle : exemples de normes

- **ISO/CEI 25010 : Liste des indicateurs de qualité**
  - 8 aspects et 35 caractéristiques **mesurables**



Bilan des normes

Inadéquation du cycle vie

- **ISO/CEI 12027**
  - Le cycle de vie en V n'offre pas les meilleures garanties de succès des projets

Inapplicabilité des indicateurs

- **ISO/CEI 25010**
  - Pas de mesure objective pour ces indicateurs (hors performance)
  - Mesure **empirique** des indicateurs → qualité **subjective**

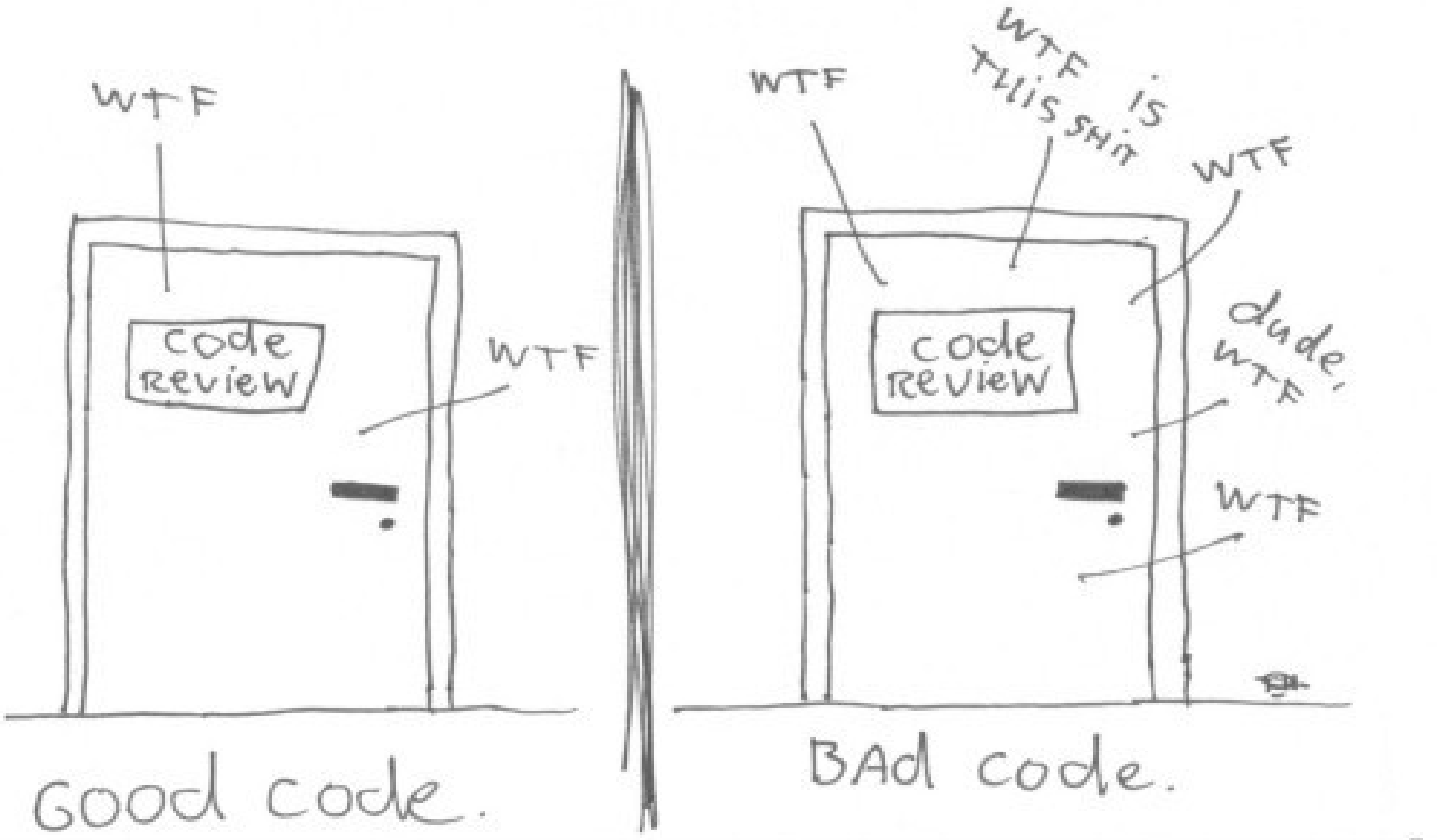
Capacité fonctionnelle	Performance Efficacité	Compatibilité	Utilisabilité	Fiabilité	Sécurité	Maintenabilité	Portabilité
• **Pertinence** • Exactitude • Interopérabilité • Sécurité • Conformité	• Comportement temporel • Utilisation des ressources	• Coexistence • Interopérabilité	• Facilité de compréhension • **Facilité d'apprentissage** • Facilité d'exploitation	• Maturité • Disponibilité • Tolérance aux fautes • Facilité de récupération	• Confidentialité • Intégrité • Non-répudiation • Authenticité • Responsabilité	• **Modularité** • Réutilisabilité • **Facilité de modification** • Testabilité	• **Adaptabilité** • Installation • Interchangeabilité

Inapplicabilité de l'assurance qualité

- Rappel de la définition
  - L'assurance qualité (QA) est un ensemble d'activités planifiées et systématiques de toutes les actions nécessaires pour fournir une assurance que la qualité du logiciel est **conforme** aux **exigences** et aux **attentes** établies.
- Problème en génie logiciel
  - Pas représentation explicite des attentes et des exigences
  - Donc pas d'assurance de conformité possible

The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE

* Saperlipopettes/minute



Le point de vue de l’agilité

Facteurs externes

- Collaboration avec le client qui est le seul juge de la qualité

Facteurs internes

- 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.
  - Le meilleur juge de la qualité est le développeur qui reprend du code.

  - En agilité, le développeur se voit comme un **artisan du logiciel**.

Artisanat du logiciel

- **Software craftsmanship**
  - Approche du développement qui met l'accent sur les compétences de codage du développeur.
    - Le code est un chef-d’œuvre
    - Partage de compétences, humilité

La qualité selon l'artisan du logiciel

- Qualité recherchée pour le code :
  1) **Simplicité** (principe *KISS* : *Keep It Simple, Stupid*)
  2) **Propreté**
  3) **Couverture de test**

L'atelier de l'artisan

- Un environnement de développement professionnel
  - IDE (*IntelliJ*)
  - Gestion de version (*Git*)
  - Moteurs de production (*Makefile (cmake)*, *Gradle*, *Maven*)
  - Outils d'intégration continue (*GitLab CI*, *Jenkins*)
- Des analyseurs de la propreté code
  - *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.
  - *FindBugs* : détection des bogues potentiels, des problèmes de performance, ou des mauvaises habitudes de codage.
  - *SonarQube* : une référence dans le domaine de l'analyse de la qualité de code (une combinaison de tous les outils précédents).

Exemple d’IntelliJ

Que retenir de ce chapitre

- La qualité logicielle est impossible à mesurer
- Il existe bien des normes mais concrètement elles n'apportent rien d'objectif pour produire des logiciels de qualité
- L'agilité considère alors que la qualité d'un logiciel est la résultante de la qualité de son code
  - Il n'y a plus de mesure mais la qualité du code doit être une préoccupation constante
- Pour cela il faut utiliser des environnements de développement sophistiqués qui permettent d'obtenir des critiques du code