Clean as You Code your projects

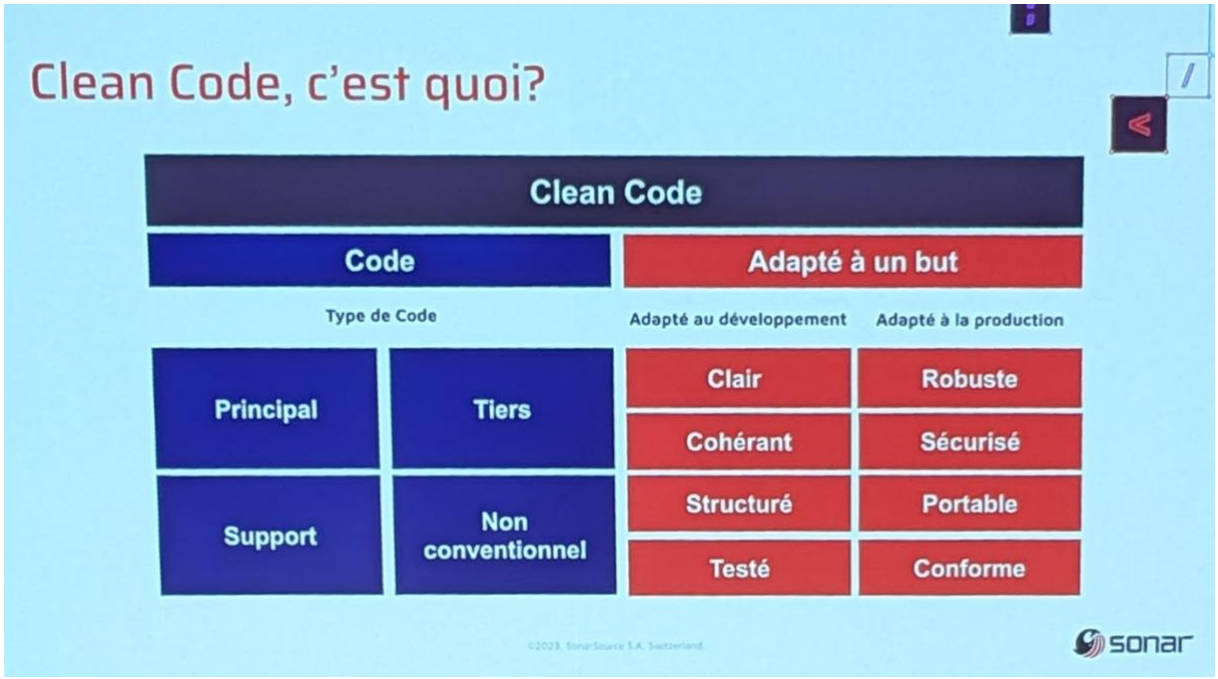
Date : 14 avril 2023
Speakers : Nolwenn Cadic et Marco Comi (SonarSource)
Format : Université (45mn)

Objectif du Clean Code

Ce talk commence par nous relater l'évolution du développement logiciel : à une époque lointaine, on faisait tout soit même, à la main, sans framework, sans automatisation ni TU. L'industrialisation a permis de faire émerger les outils DevOps, le Cloud et les conteneurs. La place du code dans l'entreprise a changé : il est partout. Il est désormais au cœur des entreprises. Il faut en prendre soin et qu'il soit Clean.

Au début de l'aventure Sonar il n'y avait que la qualité du code. Puis vient ensuite la sécurité. Sonar fait évoluer son produit avec l'approche Clean Clode.

Sonar répertorie le code en 4 types : principal, support (TU, tests intégration, scripts de build), tiers (librairies, dépendances, frameworks), code non conventionnel (mathlab, poc, test)



Un code adapté au développement se doit d'être :

1. Clair : pas de code mort
2. Cohérent : suit les conventions de nommage, l'indentation …
3. Structuré : petits modules simples avec peu de dépendances
4. Testé.

Un code adapté à la production doit être :

1. Robuste : et donc fiable
2. Sécurisé : pas de faille
3. Portable
4. Conforme : aux lois comme la RGPD, à l'accessibilité

Nolwenn nous montre l'exemple d'un TU qui teste le mock et pas le code de production.
S'en suit un autre exemple d'un code manipulant un token JWT : la signature du token n'est pas vérifiée
Un 3ième exemple en Python montre un bout de code d'ouverture d'un fichier texte non portable car Python 3 et 4 ne se comporteront pas de la même façon.

Intérêt du Clean Code

1. Minimise l'effort et le cout de l'entretien (40%)
2. Réduit la friction pour les dévs
3. Augmente longitivité du logiciel (x3)
4. Réduit les risques opérationnels, de réputation et de sécurité (90%)

Pourquoi tout le code n'est pas clean ?

1. Apprentissage continu
2. Niveaux différents d'un dév à l'autre
3. Les standards évoluent
4. Vérification des standards fastidieuse
5. Nouvelles fonctionnalités à fournir

Possibilité d'utiliser des analyseurs statiques pour détecter les défauts : par où commencer ? quoi corriger en premier ?

Approche du Cleans as you Code de SonarQube

SonarQube travaille sur l'intégration du Clean Code dans leurs outils.
Exemple d'une application legacy difficile à faire évoluer.
Options possibles :

1. Repartir de 0. A éviter car coûte très cher. Rien ne garantie que la qualité du code de la nouvelle application ne va pas se détériorer.
2. Faire un gros refactoring. Moins radical que l'option 1. A ne pas faire : très long, difficile. Les sprints de refactoring peuvent néanmoins s'envisager. Risque d'introduction de bugs.
3. Clean as You Code : approche utilisée en interne chez Sonar. Règle le problème avant de réparer les conséquences. Analogie avec la fuite d'eau. Passer la serpillère ne résout pas le problème. Il faut mieux commencer à réparer la fuite.

Basé sur le concept de Nouveau Code :

1. Le nouveau code doit être du Clean Code
2. Cela permet de régler le problème avant les conséquences

3. Pas de réécriture : ne bloque pas les évolutions
4. Je suis responsable du code que j'écris. Responsabilise les développeurs.
5. Utilisé avec succès depuis des années chez Sonar

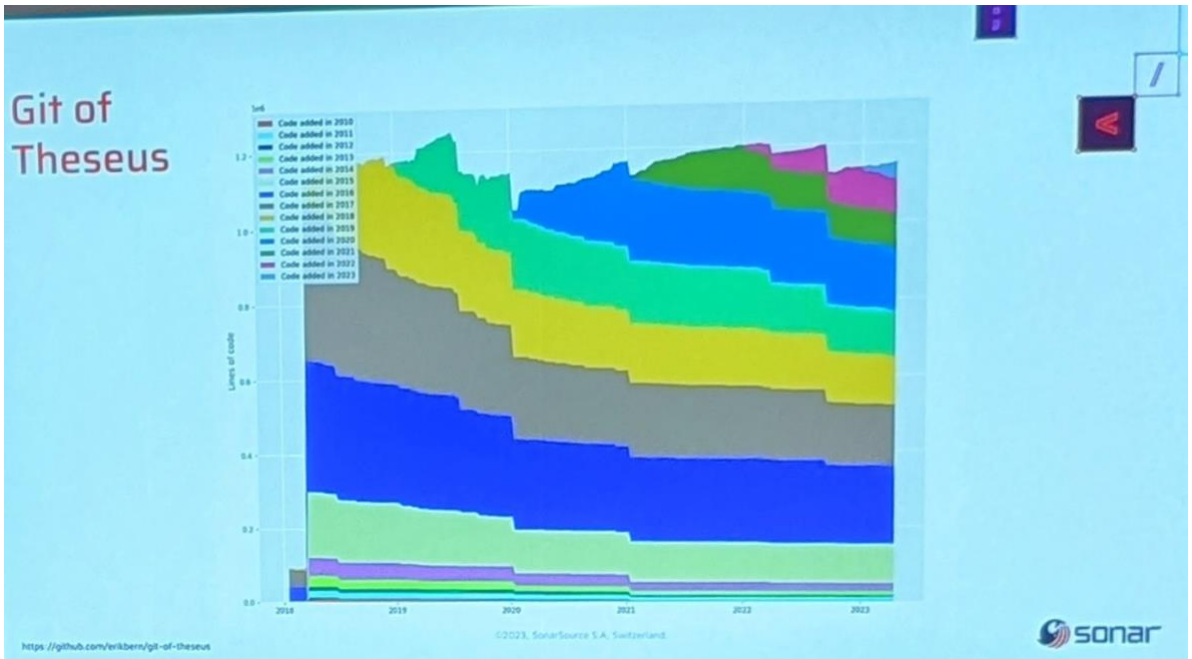
Plus besoin de sprint de refactoring. Aspect pédagogique qui permet d'améliorer le projet au fil de l'eau.

Comment mettre en place Clean Code pour votre projet ?

Une seule règle simple : aucun nouveau code non Clean Code déployé en Production.
Vérification possible via le Quality Gate (niveau de qualité).
Contrôler au plus tôt lorsqu'on écrit du code via SonarLint, supporté par de nombreux IDE. SonarLint montre les problèmes, montre comment corriger et apporte parfois des quick fix.
Seconde ligne de défense au niveau de la Pull Request : utilisation de SonarCloud fonctionnant via GitLab, Azure Devops, GitHub et BitBucket. Pipeline pour ne pas merger les pull request lorsque le Quality Hate échoue.
Le plugin SonarLint est gratuit, OpenSource et forkable.
SonarCloud est gratuit sur les projets Open Source.
L'édition gratuite de SonarQube propose de nombreuses fonctionnalités de base.
Une dernière ligne de défense est mise en place avant le déploiement

Sonar supporte plus de 30 langages et vient avec 5000 règles. Utilisé par 7 millions de dév et 400 000 organisations.
Le profil par défaut SonarWay est adapté à 80% des projets et s'appuie sur le REX de nombreux utilisateurs.
Par défaut le Code Coverage est de 80%. Chez SonarQube ils l'ont augmenté à 90 et même 95%.

Et parce qu'on n'arrête pas de changer le logiciel, on clean aussi le code du passé. En moyenne, en théorie, 20% du code est réécrit chaque année. Au bout de 5 ans, 50% du code est Clean.
Utilisation de Git of Theseus pour vérifier la théorie.
Exemple du code violet écrit en 2016



La théorie se vérifie chez SonarQube.
Au final, combien coûte le Clean Code ? 0 euro. Ecrire du code propre ne coute pas plus cher que d'écrire n'importe quoi.

A retenir

1. Le code source est la clé de l'application
2. Le Clean Code est adapté au développement et à la production
3. Seuls les dévs peuvent avoir un impact sur le code
4. La pratique Clean as you Code aide à améliorer la base de code existante

Questions du public

- De nouvelles features Clean Code vont arrivées dans les prochains mois / années
- Intégration de ChatGPT ? Sonar essaie de limiter aux maximum les faux positifs. Les équipes de R&D s'intéressent au sujet.
- Configuration des règles compliquées : preneur de tous les feedbacks. Ne pas hésiter à remonter ses problèmes.
- Est-ce que Sonar implémente certaines règles Craft présentées dans le livre Clean Code de Robert C. Martin? A priori non. Les speakers n'ont pas la réponse