UTOPIOS
TRANSFORMATION DIGITALE INTELLIGENTE

JAVA CLEAN CODE Partie 7

Les rapports qualimétrique

- Comment mettre en place les rapports de qualimétrie
- Définition de la population cible pour chaque type de rapport
- Mise en relation des métriques fournis et des bonnes pratiques de développement : établir la liste des actions correctives à effectuer
- Comment décliner les règles de développement de l'entreprise dans un contexte du projet ?
- Comment gérer l'évolution dans le temps ?
- Pourquoi et comment ignorer les alertes ?

Les rapports qualimétrique

Comment mettre en place les rapports de qualimétrie

- Pour réaliser un rapport de qualimétrie, on utilise généralement des outils d'analyse de code statique comme SonarQube, PMD ou Checkstyle. Ces outils analysent le code source à la recherche de problèmes potentiels tels que les bugs, les odeurs de code (code smells), les problèmes de sécurité et les problèmes de performance.
- Le rapport généré par ces outils peut être utilisé pour identifier les zones de code qui nécessitent une attention particulière. Il fournit également une vue d'ensemble de la qualité du code de l'application, ce qui permet de prendre des décisions éclairées sur les priorités de refactoring.

Les rapports qualimétrique

Définition de la population cible pour chaque type de rapport

- **Rapports de bugs et de qualité du code** : Ces rapports sont généralement destinés aux développeurs et aux équipes de test. Les développeurs ont besoin de ces informations pour corriger les bugs et améliorer la qualité du code. Les équipes de test peuvent également utiliser ces informations pour cibler leurs efforts de test.
- **Rapports d'architecture** : Ces rapports peuvent être destinés aux architectes logiciels, aux chefs de projet et parfois aux développeurs. Les architectes logiciels peuvent utiliser ces rapports pour examiner la santé globale de l'architecture et identifier les domaines qui nécessitent une refonte ou une attention particulière.
- **Rapports de performance** : Ces rapports sont généralement destinés aux développeurs et aux ingénieurs de performance. Ils peuvent utiliser ces informations pour identifier les goulets d'étranglement et optimiser le code pour une meilleure performance.

Les rapports qualimétrique

Définition de la population cible pour chaque type de rapport

- **Rapports de sécurité** : Ces rapports sont souvent destinés aux responsables de la sécurité, bien que les développeurs aient également besoin de ces informations pour corriger les vulnérabilités. Les responsables de la sécurité peuvent utiliser ces informations pour évaluer les risques et prendre des décisions sur les priorités de correction.
- **Rapports de conformité** : Ces rapports peuvent être destinés à une variété de parties prenantes, y compris la direction, les équipes légales et de conformité, et les auditeurs externes. Ces parties prenantes peuvent utiliser ces informations pour s'assurer que le logiciel est conforme aux diverses exigences réglementaires et normes de l'industrie.

Les rapports qualimétrique

Mise en relation des métriques fournis et des bonnes pratiques de développement

- Lorsque vous analysez les métriques fournies par vos outils d'analyse de code ou de qualité, il est crucial de les mettre en relation avec les bonnes pratiques de développement pour établir une liste d'actions correctives. Ces bonnes pratiques de développement peuvent inclure des principes tels que les principes SOLID, DRY (Don't Repeat Yourself), KISS (Keep It Simple, Stupid), etc., ainsi que les conventions de codage spécifiques à votre entreprise.

1. **Complexité cyclomatique élevée** : Si une méthode ou une classe a une complexité cyclomatique élevée, cela signifie qu'elle peut être difficile à comprendre et à maintenir. Une action corrective possible serait de refactoriser cette méthode ou cette classe pour la simplifier. Vous pouvez diviser une méthode complexe en plusieurs méthodes plus petites, chacune accomplissant une tâche spécifique. Cela est en ligne avec le principe KISS.

Les rapports qualimétrique

Mise en relation des métriques fournis et des bonnes pratiques de développement

2. **Faible couverture de code par les tests** : Si votre couverture de code par les tests est faible, cela signifie que vous risquez de ne pas détecter les bugs ou les régressions. Une action corrective serait d'écrire plus de tests pour couvrir les parties du code qui ne sont pas actuellement testées. Cela est en ligne avec les bonnes pratiques de développement qui préconisent une couverture de code par les tests élevée pour assurer la qualité et la stabilité du code.
3. **Violations des conventions de codage** : Si vos outils d'analyse de code détectent des violations des conventions de codage, cela peut rendre le code difficile à lire et à maintenir. Une action corrective serait de corriger ces violations pour que le code soit conforme aux conventions de codage. Cela est en ligne avec le principe DRY, car un code bien structuré et conforme aux conventions est plus facile à comprendre et évite les redondances.

Les rapports qualimétrique

Mise en relation des métriques fournis et des bonnes pratiques de développement

4. **Code Smells** : Les "Code Smells" sont des indicateurs de problèmes potentiels dans le code qui ne sont pas nécessairement des bugs, mais qui peuvent rendre le code plus difficile à maintenir. Par exemple, un "Code Smell" pourrait être une classe ou une méthode qui est trop longue. Une action corrective serait de refactoriser cette classe ou cette méthode pour la rendre plus petite et plus gérable. Cela est en ligne avec les principes SOLID, en particulier le principe de Responsabilité Unique, qui stipule qu'une classe ou une méthode doit avoir une seule responsabilité.

Les rapports qualimétrique

Comment décliner les règles de développement de l'entreprise dans un contexte du projet ?

- Pour décliner les règles de développement de l'entreprise, il faut d'abord les formaliser. Ces règles peuvent inclure des conventions de codage, des principes d'architecture, des normes de qualité, etc. Elles peuvent être spécifiées dans un guide de style de codage, un ensemble de règles pour un linter, une configuration pour un outil d'analyse de code, etc.
- Pour un projet ex nihilo (c'est-à-dire un projet créé à partir de zéro), ces règles peuvent être appliquées dès le début. Lors de la création du projet, assurez-vous que tous les outils appropriés sont en place pour aider à maintenir ces règles. Par exemple, vous pouvez configurer un linter pour qu'il soit exécuté automatiquement à chaque commit ou pull request. Vous pouvez également utiliser des outils d'analyse de code statique pour détecter les violations de ces règles.
- Pour un projet existant, l'application de nouvelles règles peut être plus complexe. Le code existant peut ne pas suivre ces règles, et il peut ne pas être réaliste ou économique de tout refactoriser immédiatement. Dans ce cas, une approche possible est le "refactoring de garçon scout" : à chaque fois que vous travaillez sur une partie du code, essayez de la laisser dans un meilleur état que vous ne l'avez trouvée. Appliquez les nouvelles règles autant que possible, mais ne vous sentez pas obligé de tout corriger d'un seul coup.

Les rapports qualimétrique

Comment décliner les règles de développement de l'entreprise dans un contexte du projet ? Exemple

- Supposons que votre entreprise a une règle qui stipule que tous les codes doivent suivre le principe SOLID de la programmation orientée objet. De plus, on utilise Checkstyle comme outil d'analyse de code pour appliquer les conventions de codage Java de l'entreprise.

1. **Projet ex nihilo** :

- Supposons que vous démarrez un nouveau projet pour créer une application web de gestion de tâches.
- Dès le début du projet, informez l'équipe de cette règle et expliquez le principe SOLID en détail, en donnant des exemples concrets. Vous pouvez également fournir des ressources pour aider l'équipe à comprendre et à appliquer ce principe.
- Configurez Checkstyle pour qu'il soit exécuté à chaque fois que le code est compilé ou que des pull requests sont créées. Cela garantira que le code suit bien les conventions de codage Java de l'entreprise.
- Faites des revues de code régulières pour vérifier que le code suit le principe SOLID. Si vous trouvez du code qui ne respecte pas ce principe, travaillez avec le développeur pour le refactoriser.

Les rapports qualimétrique

Comment décliner les règles de développement de l'entreprise dans un contexte du projet ? Exemple

2. **Projet existant** :

- Supposons maintenant que vous reprenez un projet existant qui n'a pas suivi le principe SOLID et n'a pas utilisé Checkstyle.
- Commencez par faire une analyse de code avec Checkstyle pour voir où se trouvent les problèmes. Cela vous donnera une idée de l'ampleur du problème.
- Ensuite, discutez avec l'équipe de la manière dont vous allez introduire ces nouvelles règles. Expliquez pourquoi elles sont importantes et comment elles peuvent améliorer la qualité du code.
- Au lieu de refactoriser tout le code d'un coup, vous pouvez adopter l'approche du "boy scout". Chaque fois que vous ou un autre développeur travaillez sur une partie du code, essayez de la laisser dans un meilleur état que vous ne l'avez trouvée. Par exemple, si vous travaillez sur une classe qui ne suit pas le principe SOLID, refactorisez-la pour qu'elle le suive.

Les rapports qualimétrique

Comment décliner les règles de développement de l’entreprise dans un contexte du projet ? Exemple

- Configurez Checkstyle pour qu'il soit exécuté automatiquement à chaque pull request. Cela aidera à prévenir l'introduction de nouveaux problèmes et encouragera les développeurs à suivre les conventions de codage.
- Enfin, faites des revues de code régulières pour vérifier que les nouvelles modifications respectent le principe SOLID et les conventions de codage de l'entreprise. Avec le temps, la qualité du code devrait s'améliorer.

Les rapports qualimétrique

Pourquoi ignorer les alertes

- **Faux positifs** : Parfois, les outils d'analyse de code peuvent générer des alertes qui ne sont pas pertinentes dans le contexte spécifique de votre code. Cela peut se produire lorsque l'outil ne peut pas comprendre entièrement le contexte dans lequel le code est utilisé.
- **Risques acceptables** : Dans certains cas, une alerte peut indiquer un risque que vous êtes prêt à accepter. Par exemple, une alerte peut signaler l'utilisation d'une fonctionnalité obsolète qui n'est pas recommandée pour de nouveaux développements, mais qui est toujours appropriée dans le contexte de votre projet.
- **Problèmes connus non prioritaires** : Il peut y avoir des alertes pour des problèmes connus qui ont été délibérément laissés de côté en raison de contraintes de temps ou de ressources, et qui ne sont pas considérés comme urgents.

Les rapports qualimétrique

Pourquoi ignorer les alertes

Comment ignorer les alertes?

Ignorer une alerte ne signifie pas simplement la supprimer et l'oublier. Au contraire, il est important de documenter pourquoi l'alerte a été ignorée, afin que cette information soit disponible pour les futurs examens ou audits de code.

- **Commenter le code** : Ajoutez des commentaires dans le code pour expliquer pourquoi vous ignorez une alerte spécifique. Certains outils d'analyse de code vous permettent d'ajouter un commentaire spécial qui indique à l'outil d'ignorer l'alerte.
- **Configurer l'outil d'analyse de code** : Vous pouvez souvent configurer l'outil d'analyse de code pour ignorer certaines alertes. Vous devriez documenter ces modifications de configuration et les raisons pour lesquelles elles ont été effectuées.
- **Documenter les décisions** : Gardez une documentation écrite des décisions d'ignorer certaines alertes. Cette documentation peut être dans le code lui-même, dans le système de suivi des bugs, dans le wiki du projet, ou ailleurs. Elle devrait inclure la raison pour laquelle l'alerte est ignorée et, si possible, un plan pour résoudre le problème à l'avenir.