HMIN306 - Évolution et Maintenance de Sytèmes Logiciels

TP1 : Généralités

BEYA NTUMBA Joel

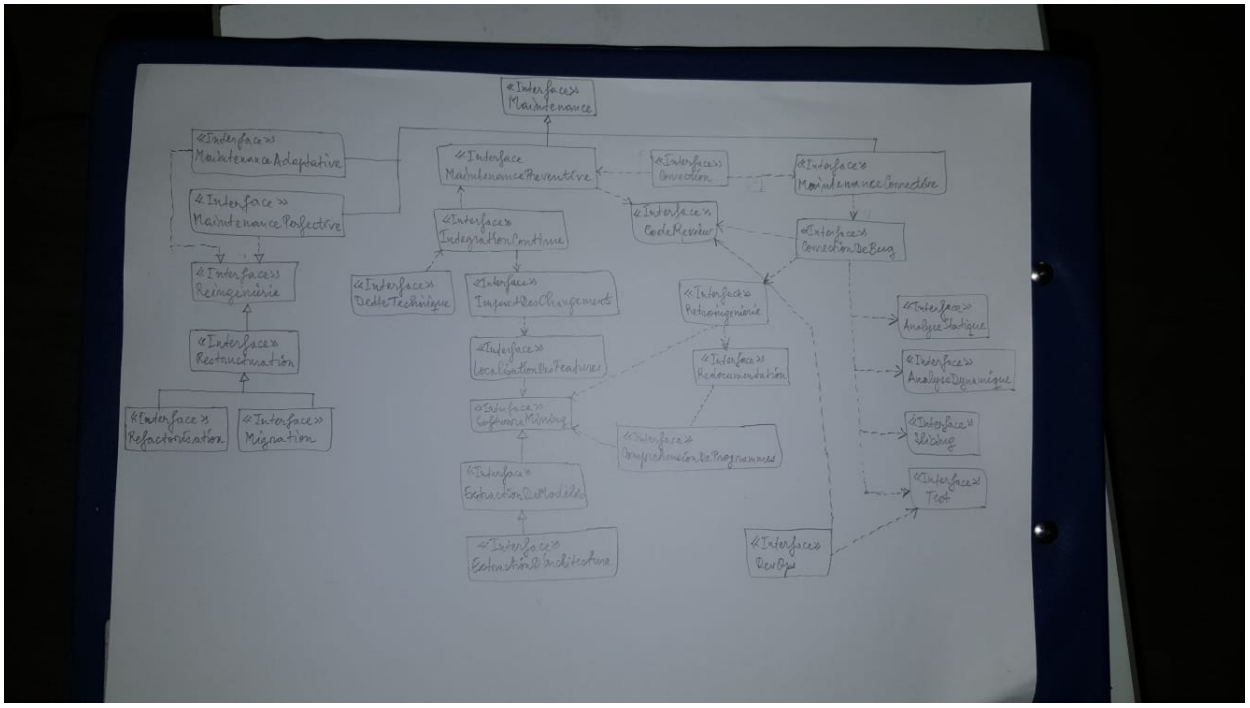
19 Janvier 2020

Master 2 Informatique
Année académique 2019/2020

Table des matières

1 Compréhension des concepts liés à l'évlution et maintenance des logiciels

Réalisation d'un modèle UML montrant les concepts liés au domaine de l'évolution/maintenance de logiciel ainsi que leurs relations :



2 Outils de Maintenance/Évolution

2.1 FindBugs

Un programme qui utilise l'analyse statique pour rechercher des bugs dans le code Java en identifiant des patterns reconnus comme étant des bugs.

2.1.1 Étude du Logiciel

Findbugs procède à un audit de code. Pour ce faire, il analyse le bytecode à la recherche de certains patterns connus. Il ne se limite pas à une recherche par expressions régulières, il essaye de comprendre ce que le programme veut faire.

FindBugs est disponible en tant que petite application graphique. Des plug-ins sont également disponibles pour Netbeans, IntelliJ IDEA et Eclipse.

Une autre option est le plugin FindBugs maven et le plugin Apache Ant

2.1.2 Installation

Prérequis

- FindBugs est un plugin d'Eclipse compatible avec les versions 3.x : (3.0, 3.1, 3.2, ...).
- Si vous avez déjà installé une version de findBugs avant mi-Mai 2006, alors avant toute installation, supprimer cette version :
  Simplement supprimer le répertoire de.tobject.findbugs_0.0.n qui se trouve sous le répertoire plugins du Eclipse.
- Vérifier que la variable d'environnement « JAVA_HOME » est bien existante et correcte.
- Avoir Eclipse $\hat{3}.7$

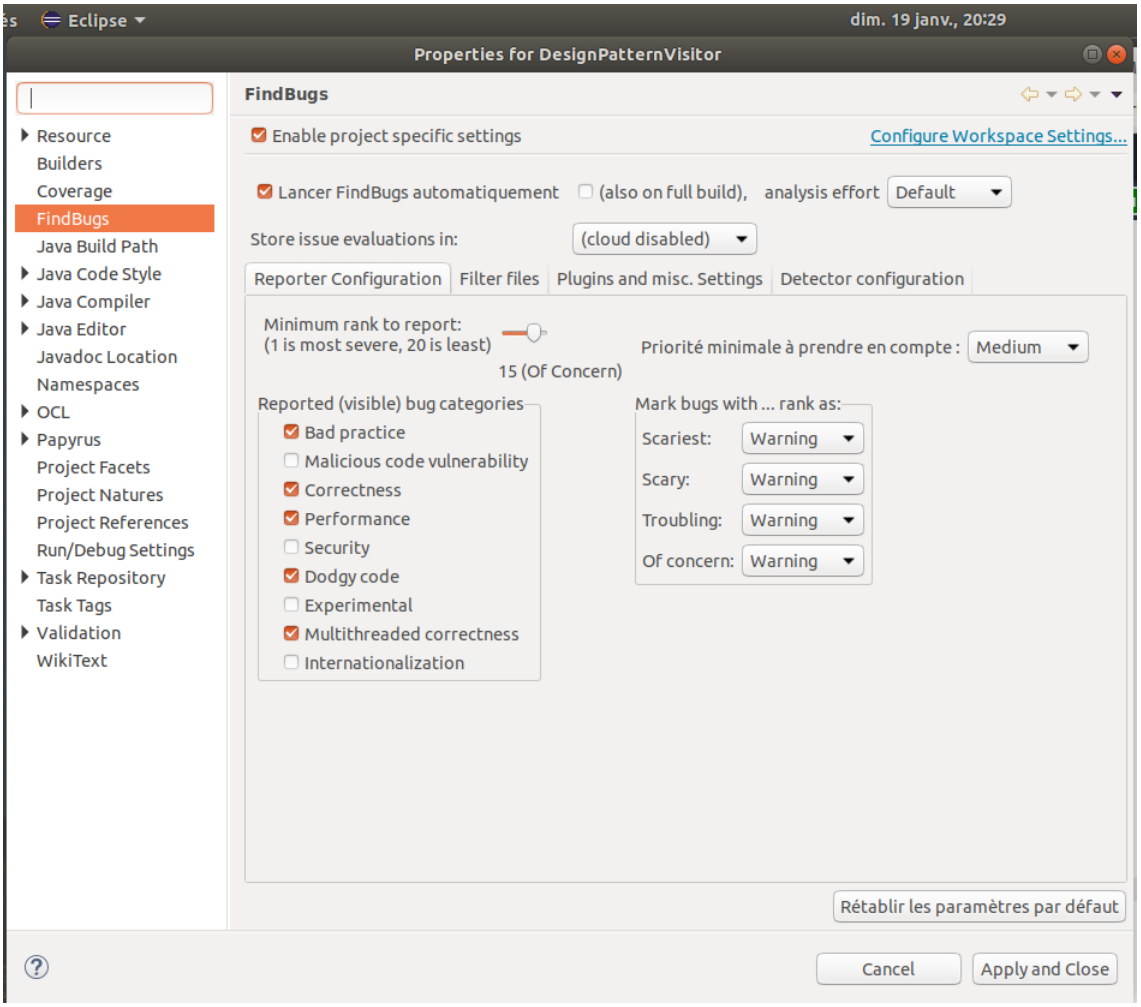
Installation du Plugin sous Eclipse Afin d'installer le plugin FindBugs, suivre les étapes suivantes :

- Dans Eclipse, Cliquer sur **Help -> Install New Software**
- Dans Install, Cliquer sur **Add ->** Puis entrez ceci :
  - **Name :** FindBugs update site
  - **URL :** choisir un parmi :
    - http ://findbugs.cs.umd.edu/eclipse for official releases
    - http ://findbugs.cs.umd.edu/eclipse-candidate for candidate releases and official releases
    - http ://findbugs.cs.umd.edu/eclipse-daily for all releases, including developmental ones
  - Cochez l'option findbugs ainsi apparu dans la recherche, ouis cliquez sur **Next**
  - Accepter les licences **I accept** et cliquer sur **Next**
  - Selectionner **Go ahead and install it anyway** ensuite cliquer sur **Yes** pour redemarer Eclipse ainsi pouvoir l'utiliser

2.1.3 Utilisation

Après installation, on peut definir les propriètés de l'outils

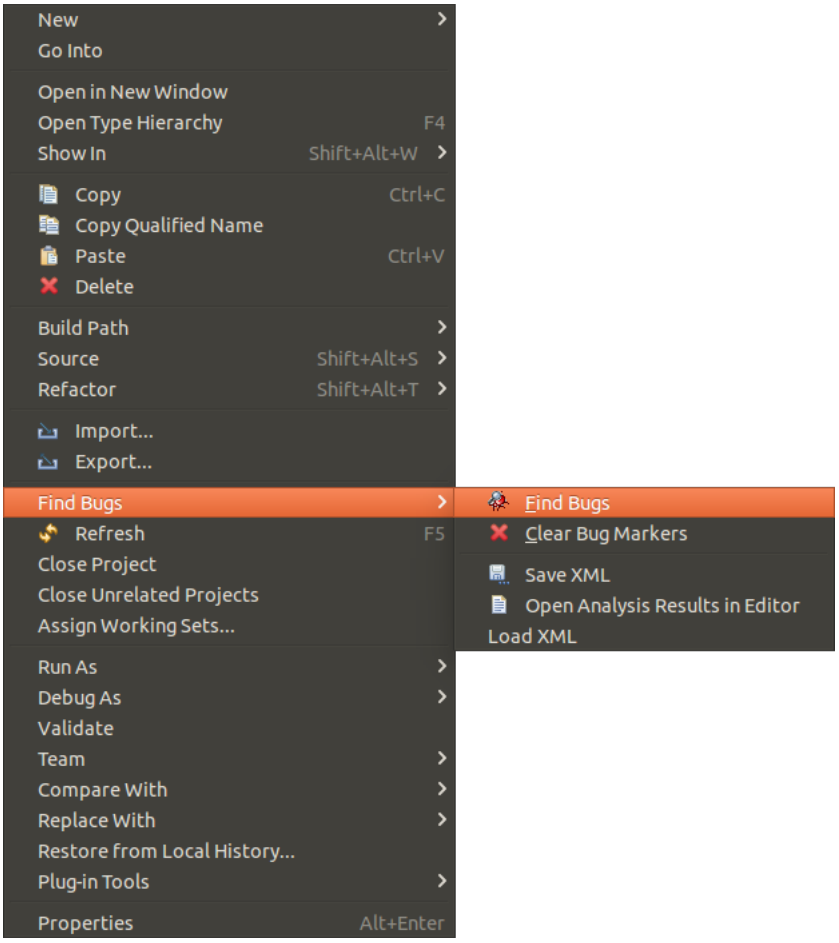
Cliquer sur **File -> Properties -> ...**



Selectionnez vos preferences puis cliquez sur **Apply and Close**

Cliquez ci pour ouvrir une démo Youtube d'utilisation

Ensuite pour l'appliquez sur un projet **Clique droit** sur la racine du projet :



2.1.4 Bilan de l'utilisation

Findbugs aide à éviter le mauvais code. Il y'a toujours des possibilités de bugs sur le code écrit vite fait dans le tas.

Un exemple

```
/* La chaîne est immuable, donc invoquer une méthode sur un objet
immuable, ne mettra pas à jour l'objet */

String str = "Javatips.net" ;
str.trim () ;

/* Le champ statique est accessible et modifiable par un code
malveillant ou par un accident, il doit donc être déclaré comme octet */

statique public final FIND_FLAG = 0 ;
```

2.2 CheckStyle

2.2.1 Étude du Logiciel

Checkstyle est un outil gratuit d'analyse de code source qui aide à améliorer la qualité de votre code en vérifiant avec certaines normes préconfigurées.

Checkstyle est également disponible en tant qu'outil de ligne de commande. Si vous avez un IDE différent de Eclipse, des plug-ins disponibles pour Netbeans, IntelliJ IDEA, etc.

Checkstyle peut vérifier de nombreux aspects de votre code source. Il peut trouver des problèmes de conception de classe, des problèmes de conception de méthode. Il a également la possibilité de vérifier la mise en page du code et les problèmes de formatage.

Pour une liste détaillée des chèques disponibles, veuillez vous référer à la page

2.2.2 Installation

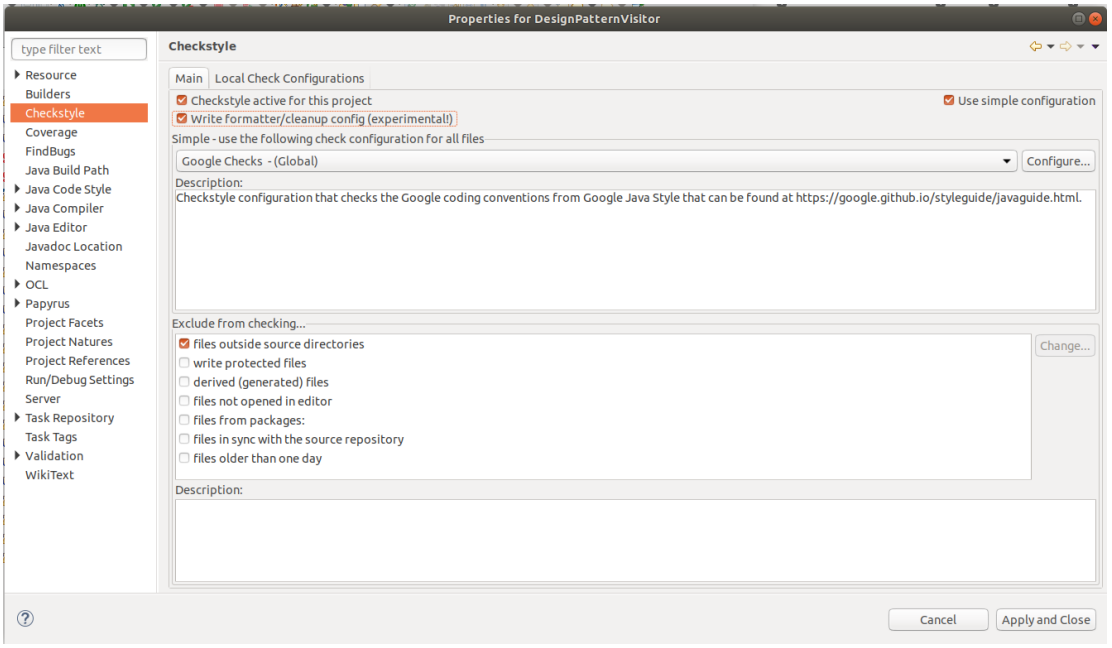
Installation du Plugin sous Eclipse Afin d'installer le plugin Checkstyle, suivre les étapes suivantes :

- Dans Eclipse, Cliquer sur **Help -> Install New Software**
- Dans Install, Cliquer sur **Add ->** Puis entrez ceci :
  - **Name :** Checkstyle
  - **URL :** http ://eclipse-cs.sourceforge.net/update
  - Cochez l'option findbugs ainsi apparu dans la recherche, ouis cliquez sur **Next**
  - Accepter les licences **I accept** et cliquer sur **Next**
  - Selectionner **Go ahead and install it anyway** ensuite cliquer sur **Yes** pour redemarer Eclipse ainsi pouvoir l'utiliser

2.2.3 Utilisation

Après installation, on peut definir les propriètés de l'outils

Cliquer sur **File -> Properties -> ...** Cochez l'appliquer sur le projet que vous utilisez



Selectionnez vos preferences puis cliquez sur **Apply and Close**

2.2.4 Bilan de l'utilisation

Checkstyle est un outil utile pour veiller à ce que le code qu'on produit soit uniforme et qui puisse respectec les normes de codage.

À travers "ce lien" on peut voir l'étendu des possibilités que nous offre cet outil.

Cliquez ci pour ouvrir une démo Youtube d'utilisation

3 Analyse d'approches en Maintenance et Évolution Logiciel

3.1 The Pricey Bill of Technical Debt : When and Whom will it be Paid

Les entreprises de de dévloppements logiciels doivent proposer leurs applications de manière continue et rapide tout en maintenant la qualité de service qu'ils proposent aux clients à court et à long terme.

Cependant, ceci peut être entravé par des limitations d'évolution et des efforts de maintenance élevés dus à des problèmes internes de qualité des logiciels par ce qui est décrit comme une **Dette Technique** ***Technical Deb***.

Bien que d'importants travaux théoriques aient été entrepris pour décrire les effets négatifs de la Dette Technique, ces études ont tendance à avoir une base empirique faible et manquent souvent de données quantitatives.

3.2 Étude de la technique

L'objectif de cette étude est d'estimer le temps perdu, causé par les intérêts de la Dette Technique pendant le cycle de vie des logiciels.

L'étude méné dans cet article examine également comment les praticiens perçoivent et estiment l'impact des conséquences négatives du Dette Technique pendant le processus de développement des logiciels.

3.3 Synthèse de la technique

Les résultats montrent qu'en moyenne, on estime que 36% de tout le temps de développement est gaspillé en raison de la dette technique ; la dette technique liée à la conception architecturale complexe et aux exigences génère l'effet le plus négatif ; et que la plupart du temps est gaspillé pour comprendre et/ou mesurer la dette technique

De plus, l'analyse des rôles professionnels et de l'âge du système logiciel dans l'enquête a révélé que les différents rôles sont affectés différemment et que les conséquences de la dette technique sont également influencées par l'âge du système logiciel.