Table des matières

[ **Identification des projets possédant le plus de bugs ouverts sur XWiki** 2](#_Toc2095011)

[***Protocole*** 2](#_Toc2095012)

[***Résultats*** 2](#_Toc2095013)

[ **Calculs de la sévérité des bugs de certains composants (de XWiki Platform)** 3](#_Toc2095014)

[***Protocole*** 3](#_Toc2095015)

[***Résultats*** 4](#_Toc2095016)

[ **Récupération des sources liées aux tickets Jira (avec l’aide de Jenkins)** 6](#_Toc2095017)

[***Protocole*** 6](#_Toc2095018)

[***Résultats*** 6](#_Toc2095019)

[ **Récupération de deux métriques (complexité et couverture de code)** 7](#_Toc2095020)

[***Protocole*** 7](#_Toc2095021)

[***Résultats*** 7](#_Toc2095022)

[ **Mise en relation entre couverture de code et sensibilité** 8](#_Toc2095023)

[***Protocole*** 8](#_Toc2095024)

[***Résultats*** 8](#_Toc2095025)

# **Identification des projets possédant le plus de bugs ouverts sur XWiki**

## ***Protocole***

* Étape 1 :

|  |
| --- |
| Se rendre sur le lien du dashboard ***Jira*** de *XWiki*.  Lien : <https://jira.xwiki.org/secure/Dashboard.jspa> |

* Étape 2 :

|  |
| --- |
| Aller dans la rubrique « Issues » (située en haut à gauche du dashboard).  Sélectionner dans le menu *Type* : « Bug » puis dans le menu *Status* : « Open ».  Exporter la recherche au format CSV (All fields) avec pour séparateur un « ; » (en haut à droite).  Dans le cas où l’opération précédente ne fonctionnerait pas, vous pouvez utiliser le lien suivant. Cependant, il limite la recherche à 1000 entités.  <https://jira.xwiki.org/sr/jira.issueviews:searchrequest-csv-all-fields/temp/SearchRequest.csv?jqlQuery=issuetype+%3D+Bug+AND+status+%3D+Open&delimiter=;&tempMax=1000> |

* Étape 3 :

|  |
| --- |
| Ouvrir le fichier obtenu précédemment via Excel et retirer (à la main) tous les logs d’erreurs.  On conserve ainsi uniquement les données étant utiles pour cette recherche. |

* Étape 4 :

|  |
| --- |
| Appliquer la formule *NB.SI.ENS* de Excel sur chaque couple « Priorité de bug » - « Projets *XWiki* ».  Exemple de formule : **=NB.SI.ENS(L1:L756; « Minor »; G1:G756; « XWiki Platform »)**  Cette formule permet donc de détecter la quantité de bugs de priorité mineure dans le projet *XWiki* Platform. |

## ***Résultats***

Ce protocole nous permet de récupérer au maximum les 1000 derniers bugs.

Le but de ce protocole est d’identifier dans quels projets se situent les bugs courants ainsi que la priorité associée à chacun de ces bugs.

# **Calculs de la sévérité des bugs de certains composants (de XWiki Platform)**

## ***Protocole***

* Étape 1 :

|  |
| --- |
| Se rendre sur le lien du dashboard ***Jira*** de *XWiki*  Lien : <https://jira.xwiki.org/secure/Dashboard.jspa>  Ensuite, sélectionner le projet *XWiki Platform.*  On arrive sur la page des « issues ».  Ensuite cliquer sur le lien *Components* dans le menu de gauche.  Ou bien se rendre directement sur ce lien :  <https://jira.xwiki.org/projects/XWIKI?selectedItem=com.atlassian.jira.jira-projects-plugin:components-page> |

* Étape 2 :

|  |
| --- |
| Chercher à la main les 10 composants actifs les plus importants en termes d’issues (fastidieux).  Pour chacun d’entre eux :   * Cliquer sur le composant pour accéder à sa page d’issues. * Sélectionner « Bug » dans le menu *Type* de la barre de recherche. * Cliquer sur Export en haut à droite de la page et choisir l’option *CSV* (Current fields). * Choisir le « ; » comme séparateur. |

|  |
| --- |
| Ou bien vous pouvez télécharger directement les fichiers grâce aux liens suivants :   * *Old Core:*   <https://jira.xwiki.org/sr/jira.issueviews:searchrequest-csv-current-fields/temp/SearchRequest.csv?jqlQuery=project+%3D+XWIKI+AND+issuetype+%3D+Bug+AND+component+%3D+%22Old+Core%22&delimiter=;>   * *Web - Templates & Ressources :*   <https://jira.xwiki.org/sr/jira.issueviews:searchrequest-csv-current-fields/temp/SearchRequest.csv?jqlQuery=project+%3D+XWIKI+AND+issuetype+%3D+Bug+AND+component+%3D+%22Web+-+Templates+%26+Resources%22&delimiter=;>   * *Extension :*   <https://jira.xwiki.org/sr/jira.issueviews:searchrequest-csv-current-fields/temp/SearchRequest.csv?jqlQuery=project+%3D+XWIKI+AND+issuetype+%3D+Bug+AND+component+%3D+Extension&delimiter=;>   * *Administration :*   <https://jira.xwiki.org/sr/jira.issueviews:searchrequest-csv-current-fields/temp/SearchRequest.csv?jqlQuery=project+%3D+XWIKI+AND+issuetype+%3D+Bug+AND+component+%3D+Administration&delimiter=;>   * *Flamingo Skin :*   <https://jira.xwiki.org/sr/jira.issueviews:searchrequest-csv-current-fields/temp/SearchRequest.csv?jqlQuery=project+%3D+XWIKI+AND+issuetype+%3D+Bug+AND+component+%3D+%22Flamingo+Skin%22&delimiter=;>   * *WYSIWYG Editor :*   <https://jira.xwiki.org/sr/jira.issueviews:searchrequest-csv-current-fields/temp/SearchRequest.csv?jqlQuery=project+%3D+XWIKI+AND+issuetype+%3D+Bug+AND+component+%3D+%22WYSIWYG+Editor%22&delimiter=;>   * *Storage :*   <https://jira.xwiki.org/sr/jira.issueviews:searchrequest-csv-current-fields/temp/SearchRequest.csv?jqlQuery=project+%3D+XWIKI+AND+issuetype+%3D+Bug+AND+component+%3D+Storage&delimiter=;>   * *Notifications :*   <https://jira.xwiki.org/sr/jira.issueviews:searchrequest-csv-current-fields/temp/SearchRequest.csv?jqlQuery=project+%3D+XWIKI+AND+issuetype+%3D+Bug+AND+component+%3D+Notifications&delimiter=;>   * *Wiki :*   <https://jira.xwiki.org/sr/jira.issueviews:searchrequest-csv-current-fields/temp/SearchRequest.csv?jqlQuery=project+%3D+XWIKI+AND+issuetype+%3D+Bug+AND+component+%3D+Wiki&delimiter=;>   * *Search – Solr :*   <https://jira.xwiki.org/sr/jira.issueviews:searchrequest-csv-current-fields/temp/SearchRequest.csv?jqlQuery=project+%3D+XWIKI+AND+issuetype+%3D+Bug+AND+component+%3D+%22Search+-+Solr%22&delimiter=;> |

* Étape 3 :

|  |
| --- |
| Dans Excel, créer un fichier étant une concaténation de tous les CSV téléchargés précédemment. Il s’agit donc d’ajouter tout le contenu de chacun des 10 fichiers dans un seule fichier au format XLXS afin de pouvoir le traiter avec des formules et d’ainsi créer des graphiques. |

* Étape 4 :

|  |
| --- |
| Désormais, il suffit d’appliquer une formule *NB.SI.ENS* sur chaque priorité d’erreur pour chacun des composants à traiter afin d’obtenir la quantité de bugs correspondant à ces critères.  Exemple de formule : **=NB.SI.ENS(XWiki!G806:G1755;Feuil1!A3)**  Où ici, *XWiki!G806:G1755* corresponds aux bugs du composants **Core** et *Feuil1!A3* est égal à **« Minor »**. |

## ***Résultats***

Ces données peuvent être triées par composant ainsi que par priorité de bug à traiter. Ceci nous permet de réaliser des % de types d’erreurs pour chacun des composants.

À partir de là, il est facile de réaliser un indice de sensibilité en attribuant à chaque type d’erreur un poids et en divisant la somme des ces couples poids-type par le nombre d’erreurs dans le composant.

**Un exemple de formule pour calculer un indice de chaleur :**

*Nombre (n)*

*Poids (p): Trivial (Tr) = 1, Minor (Mi) = 2, Major (Ma) = 4, Critical (Cr) = 8 & Blocker (Bl) = 16*

*((nTr \* pTr) + (nMi \* pMi) + (nMa \* pMa) + (nCr \* pCr) + (nBl \* pBl)) / (nTr + nMi + nMa + nCr + nBl)*

À l’aide de la formule précédente, on calcule l’indice de sensibilité de chacun des 10 composants. Afin de pouvoir faire une comparaison plus efficace des indices calculés précédemment, on calcule également l’indice moyen des 10 composants réunis.

Une fois celui-ci obtenu, il est trivial de créer des pourcentages permettant de voir la différence de sensibilité entre un composant et la moyenne de ceux du projet.

# **Récupération des sources liées aux tickets Jira (avec l’aide de Jenkins)**

## ***Protocole***

* Étape 1 :

|  |
| --- |
| Dans un premier temps, il faut installer **python3** et **pip**.  Ensuite, il faut exécuter la commande suivante: « **pip install urllib3** ». |

* Étape 2 :

|  |
| --- |
| Aller dans le répertoire « **XWiki\_Jenkins** ».  Exécuter le script python **run.py**. |

## ***Résultats***

Le script **run.py** permet de faire des appels à Jenkins via son API. On peut ainsi récupérer la liste de toutes les branches de XWiki présentes sur *Jenkins,* et les ajouter dans un fichier nommé **Resultats\_protocole\_3.csv**.

Ensuite, le script récupère jusqu’aux 20 derniers *builds* de chaque branche, puis les parse afin d’en retirer :

* Les issues *Jira.*
* Les fichiers modifiés liés à cette issue.
* L’ID du commit sur *GitHub*.

# **Récupération de deux métriques (complexité et couverture de code)**

Une fois de plus, nous traitons ici le repository **XWiki Platform**.

## ***Protocole***

* Étape 1 :

|  |
| --- |
| Accéder à la section « *Top risks* » du build **XWiki Platform - Parent POM 11.1-SNAPSHOT** sur Clover.  Lien : <http://maven.xwiki.org/site/clover/20190202/clover-commons+rendering+platform-20190202-0222/top-risks.html>  Faire un copier-coller des données présentes sur la page (de [APTParser](http://maven.xwiki.org/site/clover/20190202/clover-commons+rendering+platform-20190202-0222/org/xwiki/rendering/internal/parser/apt/APTParser.html) à [ZipExplorerPlugin](http://maven.xwiki.org/site/clover/20190202/clover-commons+rendering+platform-20190202-0222/com/xpn/xwiki/plugin/zipexplorer/ZipExplorerPlugin.html)) vers un nouveau fichier Word. |

* Étape 2 :

|  |
| --- |
| Dans le fichier Word, supprimer tous les hyperliens avec un Ctrl-A et un Ctrl-Maj-F9.  Faire un Ctrl-H, cliquer sur le bouton « **Plus > >** », cocher « **Utiliser les caractères génériques** », mettre un *caractère espace* dans le champ **rechercher** puis mettre *^p* dans le champ **remplacer par**.  Sauvegarder sous le format **Document XML Word** avec pour nom : « **data\_format.xml** ». |

* Étape 3 :

|  |
| --- |
| À l’aide d’un éditeur de texte, ouvrir le XML, chercher la balise **<w:body>** et supprimer toutes les lignes précédentes. Chercher la balise **</w:body>** et supprimer toutes les lignes suivantes.  Enfin, supprimer toutes les occurrences de « **w:**» (En remplaçant par une string vide dans l’éditeur de texte). |

* Étape 4 :

|  |
| --- |
| Pour cette étape, il faut installer **python3** et **pip**. Une fois cela effectué, vous pouvez entrer la commande suivante : « **pip install lxml** ».  Mettre les fichiers **data\_format.xml** et **xmlparse.py** dans le même dossier.  Exécuter le script python **xmlparse.py**. |

## ***Résultats***

Après avoir exécuté le protocole ci-dessus, un fichier **Resultats\_protocole\_4.csv** est créé.

Celui-ci contient 3 colonnes : la première contient le nom de toutes les classes présentes dans XWiki Platform, la seconde contient elle la fourchette de couverture de code de la classe en question et enfin la troisième contient une estimation (proche de la réalité) de la complexité de la classe concernée.

# **Mise en relation entre couverture de code et sensibilité**

## ***Protocole***

* Étape 1 :

|  |
| --- |
| Créer un dossier source au niveau du fichier **correlation.py** puis y insérer les fichiers résultats **Resultats\_protocole\_3.csv** & **Resultats\_protocole\_4.csv**. |

* Étape 2 :

|  |
| --- |
| Exécuter le script python **correlation.py**. |

* Étape 3 :

|  |
| --- |
| **Sous Excel :**   * Créer une colonne « **Bugs x Complexity** » qui contiendra la multiplication de **« Bugs Number »** et de **« Estimate Complexity »** pour chaque ligne. * Créer une colonne « **Bugs x Coverage** » qui contiendra la multiplication de **« Bugs Number »** et de la valeur moyenne de **« Coverage »** pour chaque ligne.   *Exemple de valeur moyenne pour [70%-80%] :* ***75%***   * Sur une nouvelle feuille, créer une colonne **« Composants »** regroupant uniquement une seule fois les noms de chaque composant présent du fichier. *Exemple de nom :* ***« oldcore »*** * Créer une colonne **« Nombre de bugs »** qui sera la somme des bugs présents dans un composant à l’aide de la formule sur chaque ligne :   **=SOMME.SI(Colonne «Component»; « Nom du composant »; Colonne « Bugs Number »)**   * Créer une colonne **« Couverture »** qui sera la moyenne de la couverture de code des classes d’un composants sur lequel des bugs sont présents :   **=SOMME.SI(Colonne «Component»; « Nom du composant »; Colonne « Bugs x Coverage** **») / « Nombre de bugs »**   * Créer une colonne **« Complexité »** qui sera la moyenne de la complexité des classes d’un composants sur lequel des bugs sont présents :   **=SOMME.SI(Colonne «Component»; « Nom du composant »; Colonne « Bugs x Complexity** **») / « Nombre de bugs »** |

## ***Résultats***

Le but de ce protocole est de mettre en relation les classes identifiées comme contenant des bugs (cf. résultats de l’expérience 3) avec leur complexité et couverture de code (cf. résultats de l’expérience 4).

Cela nous permet de pouvoir identifier la complexité et la couverture de code moyenne des composants contenants des classes liées à des issues.

Les résultats de cette expérience seront comparés avec ceux obtenus dans l’expérience 2. Cela nous permet de corréler sensibilité et couverture de code et répondre à notre seconde hypothèse.