8_analyse statique des métriques avec Sonar et Eclipse

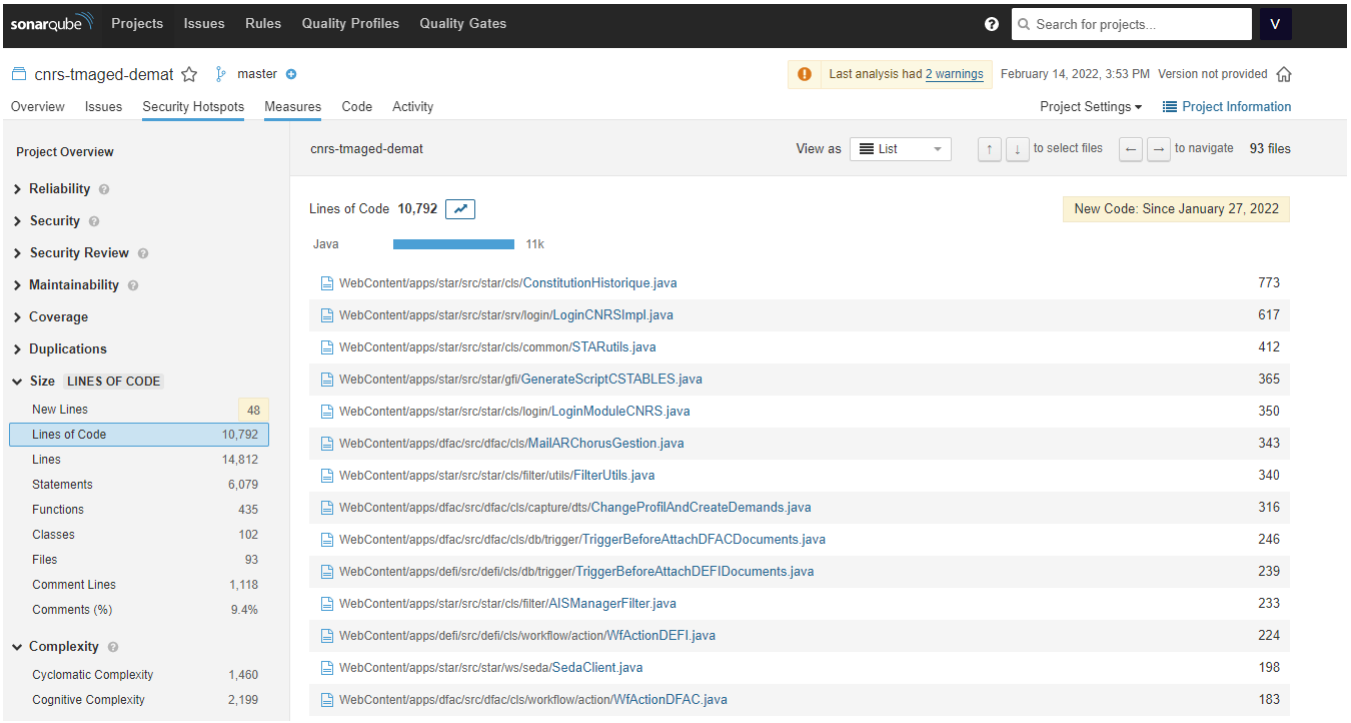
Pour analyser les métriques et la qualité du code, nous utilisons SonarQube : https://sonar.codep.inetum.world/projects

Sonar analyse la qualité du code et donne un rapport sur respect des règles de codage et sévérité.

Il donne aussi les métriques suivantes en classant les classes les plus volumineuses en premier :

- Nombre de classes
- Nombre de méthodes
- Nombre de lignes de code
- Taux de commentaire
- Duplication de code
- Complexité cyclomatique = nombre de « chemins » au travers d'un programme.

Exemple pour Demat, aller dans "Measures" puis dans "Size" pour afficher les détails des métriques :

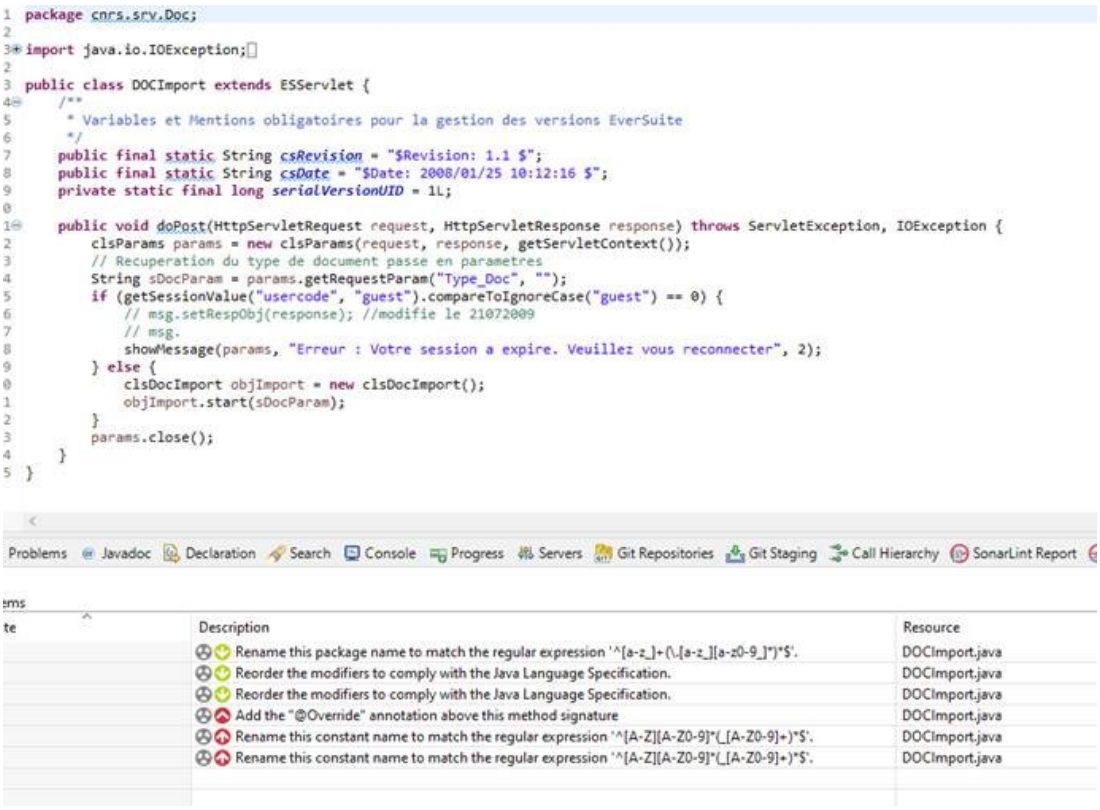


On voit aussi une analyse de la couverture des tests unitaires (par contre pas de TU sur nos projets CNRS actuellement).

On peut aussi installer le plugin Eclipse SonarLint qui permet de souligner les erreurs de style et de conventions de codage.

La documentation en ligne de SonarLint : SonarLint for Eclipse | Code Quality & Security Extension

Exemple sur EVALUATION où nous voyons des erreurs de nommage, d'annotation manquante et d'ordre d'écriture de mots clés :



```java
package cnrs.srv.Doc;

import java.io.IOException;

public class DOCImport extends ESServlet {
    /**
     * Variables et Mentions obligatoires pour la gestion des versions EverSuite
     */
    public final static String csRevision = "$Revision: 1.1 $";
    public final static String csDate = "$Date: 2008/01/25 10:12:16 $";
    private static final long serialVersionUID = 1L;

    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        clsParams params = new clsParams(request, response, getServletContext());
        // Recuperation du type de document passe en parametres
        String sDocParam = params.getRequestParam("Type_Doc", "");
        if (getSessionValue("usercode", "guest").compareToIgnoreCase("guest") == 0) {
            // msg.setRespObj(response); //modifie le 21072009
            // msg.
            showMessage(params, "Erreur : Votre session a expire. Veuillez vous reconnecter", 2);
        } else {
            clsDocImport objImport = new clsDocImport();
            objImport.start(sDocParam);
        }
        params.close();
    }
}
```

Problems | Javadoc | Declaration | Search | Console | Progress | Servers | Git Repositories | Git Staging | Call Hierarchy | SonarLint Report

ems

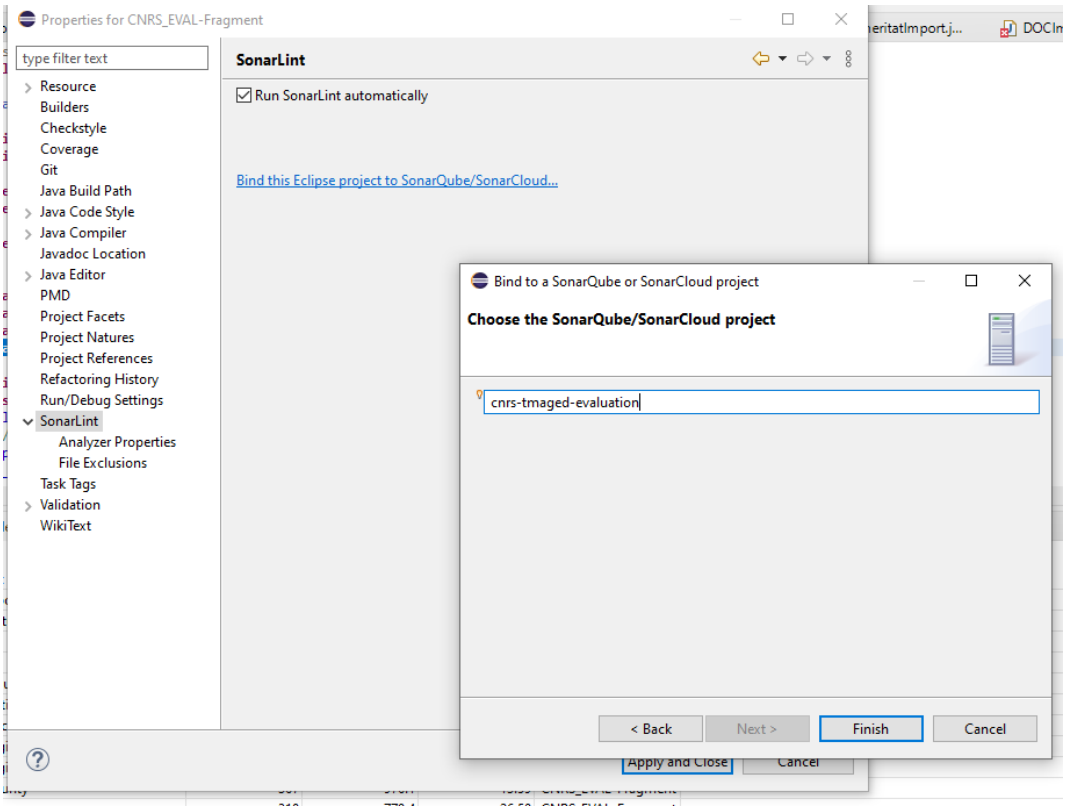
te	Description	Resource
	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.	DOCImport.java
	Reorder the modifiers to comply with the Java Language Specification.	DOCImport.java
	Reorder the modifiers to comply with the Java Language Specification.	DOCImport.java
	Add the "@Override" annotation above this method signature	DOCImport.java
	Rename this constant name to match the regular expression '^[A-Z][A-Z0-9]*(_[A-Z0-9]+)*$'.	DOCImport.java
	Rename this constant name to match the regular expression '^[A-Z][A-Z0-9]*(_[A-Z0-9]+)*$'.	DOCImport.java

Pour installer SonarLint sous Eclipse:

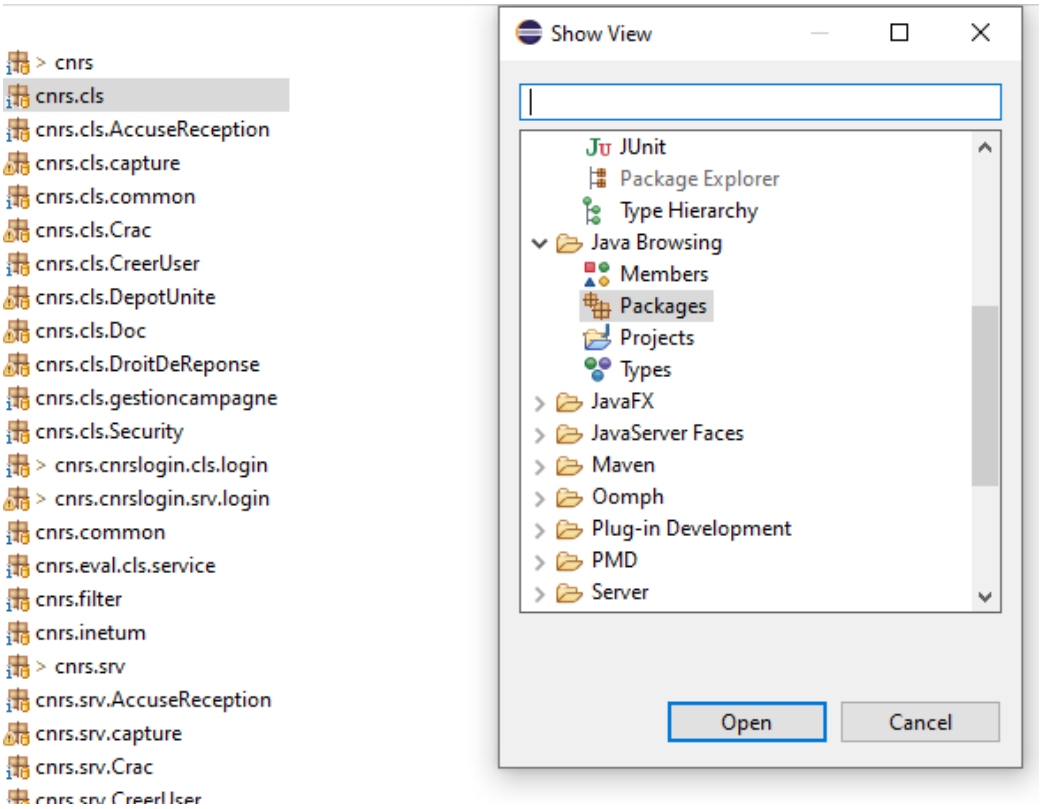
1. aller dans Eclipse Marketplace
2. rechercher SonarLint
3. installer le plugin (accepter la licence et approuver le certificat)

Pour activer ou désactiver SonarLint:

1. click droit sur le projet dans l'explorateur de projet sous Eclipse
2. dans la section SonarLint:
    a. cocher Run Sonar Lint automatically
    b. puis vous pouvez relier SonarLint à SonarQube en précisant l'url SonarQube puis le nom du projet :

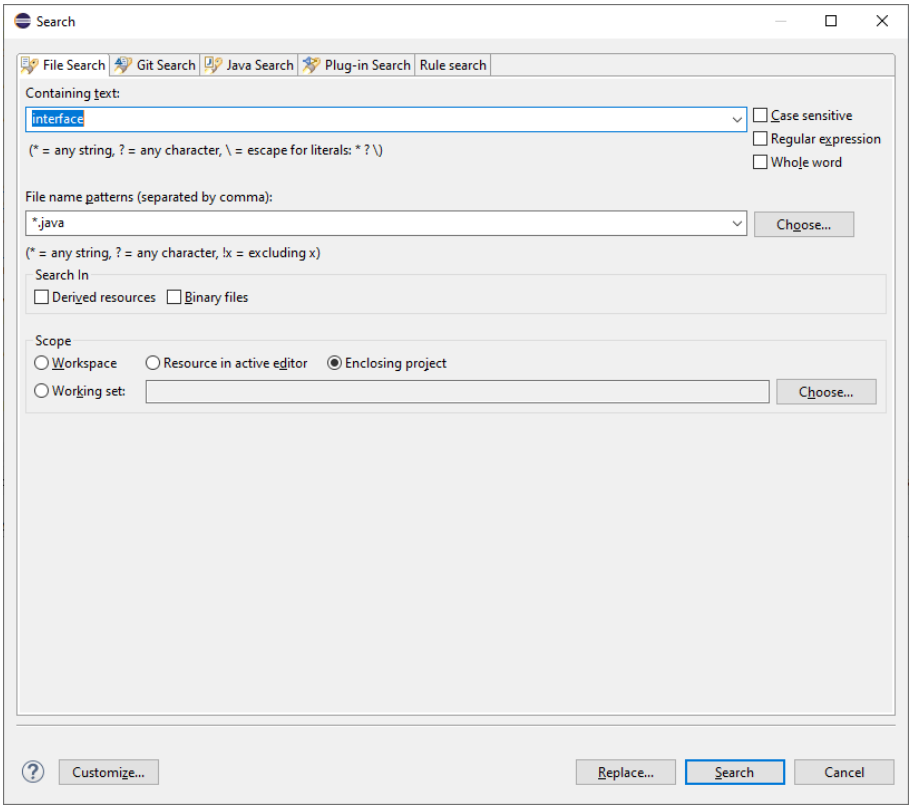


Pour compter le nombre de package du projet sous Eclipse, il faut ouvrir la vue "Packages" sous "Java Browsing" :



Dans la fenêtre de cette vue, nous avons alors la liste de tous les packages Java utilisés dans le projet.

Pour compter le nombre d'interface JAVA du projet sous Eclipse, il faut ouvrir la fonctionnalité "Search" (CTRL-H) puis rechercher "public interface" dans toutes les classes JAVA (*.java) du projet courant :



Récapitulatifs des métriques demandées par le CNRS et l'outil utilisé :

Métrique demandée	Outil pour récupérer la métrique	Commentaire
Nombre de classes	SonarQube	
Nombre de méthodes	SonarQube	
Nombre de lignes de code	SonarQube	
Taux de commentaire	SonarQube	
Duplication de code	SonarQube	
Complexité cyclomatique	SonarQube	
Respect des règles de codage et sévérité	SonarQube + SonarLint	
Nombre d'interfaces	Eclipse	Recherche dans les fichiers java
Nombre de packages	Eclipse	Vue Java Browsing / packages
La couverture de tests unitaires (pourcentage de lignes de code du projet qui est appelé pendant la phase de test)	SonarQube	PAS DE TESTS UNITAIRES SUR NOS PROJETS CNRS
Le taux de succès des tests unitaires	SonarQube	PAS DE TESTS UNITAIRES SUR NOS PROJETS CNRS
LCOM4 (Lack of Cohesion Methods = nombre de composants connectés dans une classe)	SonarQube ? (fonctionnalité enlevée car les résultats n'étaient pas satisfaisant)	A voir si le LCOM5 marchera mieux quand il sera disponible sous SonarQube