

1 Le Bottom-Up

L'approche « Bottom-Up » de CAQS permet de connaître directement les éléments les plus problématiques.

Cette approche contient plusieurs parties :

- La volumétrie
- La répartition des améliorations à apporter
- La répartition des notes par élément
- La recherche
- Le détail pour un élément

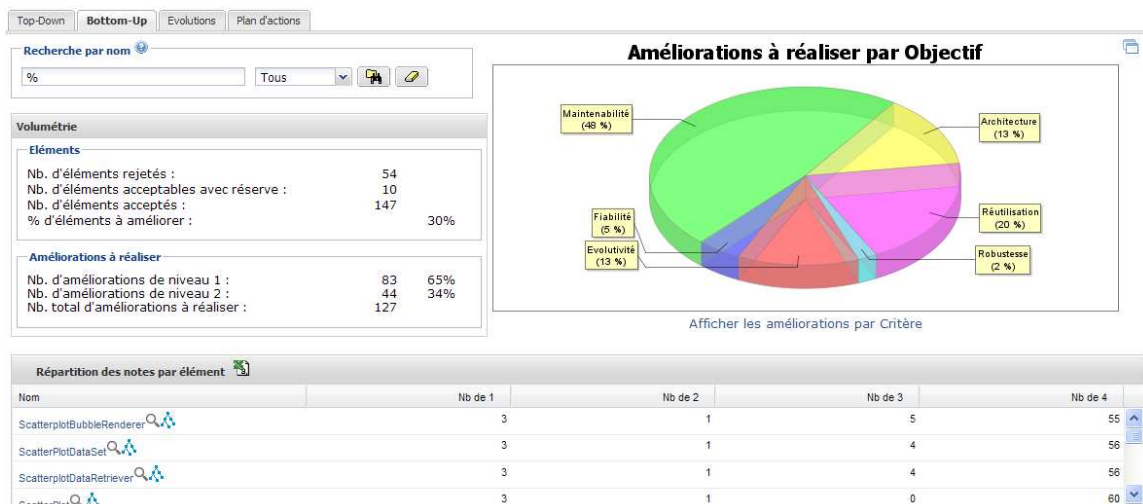


Figure 1. Bottom-Up

Le bouton permet d'exporter ladite liste au format CSV¹.

1.1 La volumétrie

Les informations suivantes sont représentées dans la volumétrie :

- Le **nombre d'éléments rejetés** : cela correspond au nombre d'éléments dont le critère ayant la plus mauvaise note, a une note comprise entre 1 (inclus) et 2 (exclus).
- Le **nombre d'éléments acceptables avec réserve** : cela correspond au nombre d'éléments dont le critère ayant la plus mauvaise note, a une note comprise entre 2 (inclus) et 3 (exclus).
- Le **nombre d'éléments acceptés** : cela correspond au nombre d'éléments dont tous les critères ont été acceptés (note comprise entre 3 (inclus) et 4 (inclus)).

¹ CSV : Comma-Separated Values. Fichier texte présentant les informations par ligne et dont les valeurs sont séparées par des points-virgules. Ce type de fichier est lisible, entre autres, par Microsoft Office® ou « OpenOffice.org Calc ».

- Le **pourcentage d'éléments à améliorer** : cela correspond au pourcentage d'éléments dont au moins un critère a une note comprise entre 1 (inclus) et 3 (exclus).
- Le **nombre d'améliorations de niveau 1** : cela correspond au nombre de critères dont la note est comprise entre 1 (inclus) et 2 (exclus).
- Le **nombre d'améliorations de niveau 2** : cela correspond au nombre de critères dont la note est comprise entre 2 (inclus) et 3 (exclus).
- Le **nombre total d'améliorations à réaliser** est le nombre total de critères dont la note est comprise en 1 (inclus) et 3 (exclus).

1.2 La répartition des améliorations à apporter.


Un graphique permet de connaître la répartition des éléments à améliorer par objectif ou par critère².



Figure 2. Bottom-Up : Répartition des améliorations à apporter.

Chaque objectif (ou critère selon le choix) est représenté par un quartier dans le graphique. L'étiquette d'un « quartier » indique le nom de l'objectif (ou du critère) ainsi que le pourcentage que représentent les défauts concernant cet objectif dans la totalité des améliorations à apporter.


Ainsi, dans l'exemple montré par la Figure 2, 55% des améliorations à apporter concernent l'objectif « Maintenabilité ».

 La population des éléments prise en compte pour le calcul de la répartition des améliorations à apporter, concerne les éléments affichés sur la page de « Bottom-Up », c'est-à-dire les éléments sélectionnés par le filtre de recherche.


² Ce graphique n'apparaît pas quand l'élément sélectionné n'est pas une Entité Applicative.

1.3 La répartition des notes par élément

La répartition des notes par élément, se présente sous la forme d'un tableau. Les éléments y sont représentés du plus « mauvais » au « meilleur », c'est-à-dire de celui qui a besoin du plus de corrections³ à celui qui en a besoin du moins.

 Par défaut, et seulement s'il y a plus de 100 éléments à afficher, seuls les 100 éléments les plus problématiques sont affichés. Un lien « **Afficher tous les éléments** » permet d'afficher tous les éléments. Dans ce cas, un autre lien permet alors de n'afficher que les 100 éléments les plus problématiques.



La première colonne indique le nom « court » de l'élément. Le nom long est visualisable en laissant le pointeur de la souris sur le nom court.

 Le nom court d'un élément est, par exemple dans le cas du langage Java, le nom de la classe, non préfixé par l'arborescence de packages auxquels il appartient et sans les paramètres.

Chacune des autres colonnes est dédiée à une note attribuable. Ainsi, la colonne **Nb de 1** (nombre de 1) indique le nombre de critères ayant une note de 1 pour l'élément.

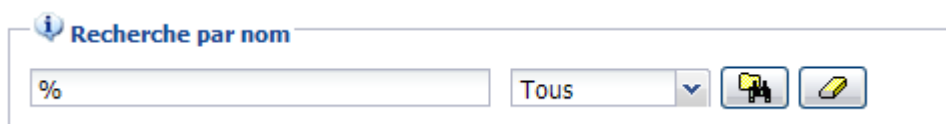
En cliquant sur le nom d'un élément, il est possible de connaître le détail des notes pour celui-ci.

Le tableau listant les éléments est triable par nom ou par nombre d'occurrences d'une note.

De même que pour la synthèse d'un critère, si disponible, le source et l'analyse d'impact peuvent être visualisés via les icônes  et .

1.4 La recherche

Un filtre des résultats affichés en fonction du nom et du type d'élément est disponible. Pour ce faire, il faut indiquer dans la recherche le nom et le type d'élément.



Recherche par nom

% Tous

Figure 3. Filtrer les résultats par nom

Le filtre par type d'élément permet de spécifier quel ensemble d'éléments seront pris en compte durant la recherche.

³ Qui a le plus d'éléments ayant une note de 1


⁴ Section 3, page 7, pour l'analyse d'impact et section 2, page 5, pour l'affichage du code source.

 La recherche se fait toujours sur le nom long de l'élément.

Il existe des caractères spéciaux qui permettent de faciliter la recherche d'éléments.

Ces caractères sont :

- % : il remplace une chaîne de caractère. Ainsi, la chaîne « %B » représente tous les éléments dont le nom se finit par un 'B'. la chaîne « %B% » représente tous les éléments qui contiennent dans leur nom, quel que soit l'endroit, le caractère 'B'. Le caractère '%' peut aussi représenter la chaîne vide.
- _ : le caractère « souligné » remplace un unique caractère. La chaîne « B_C » représente tous les éléments dont le nom contient trois caractères, la première étant 'B' et la dernière 'C'.

 Il est possible de combiner autant de caractères '%' et '_' dans la recherche que nécessaire.

La recherche se faisant sur le nom long, il faut savoir que :

- La recherche de chaîne de caractères se fait aussi sur les noms de packages, en plus du nom de la classe.
- Dans le cas de méthodes, la recherche se fait aussi sur les types de paramètres.

Cela implique que, pour rechercher une méthode particulière, sans indiquer les paramètres, il est obligatoire de finir la requête de recherche par un '%' afin de prendre en compte le fait qu'il y a des paramètres.

Les résultats affichés d'une recherche sont donc les éléments qui contiennent la chaîne de caractères recherchée dans leur nom long (packages, nom de classe, nom de méthodes et types des paramètres).

Les noms des paramètres ne sont pas utilisés.

 La recherche est sensible à la casse !

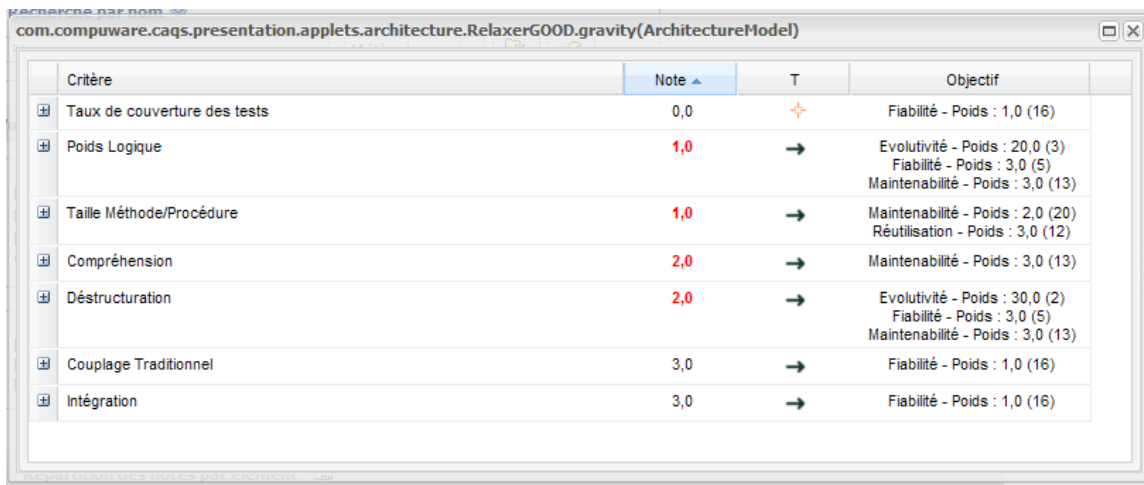
1.5 Le détail pour un élément

En cliquant sur un nom d'élément dans la liste des éléments problématiques du « Bottom-Up », le détail des notes obtenues pour cet élément apparaît.

L'exemple en Figure 4 montre le détail des notes obtenues pour l'élément « gravity ». Cet élément est une méthode car il contient un paramètre, de type ArchitectureModel. La méthode est indiquée comme faisant partie de la classe QCandidate.



Sur l'exemple, nous pouvons voir que la méthode a obtenu une note de 1 pour le critère « Poids Logique ». La colonne **Objectif** donne le nom de l'objectif, suivi du poids du

critère dans le calcul de la note de l'objectif. Le pourcentage entre parenthèses indique la part du critère dans le calcul de la note de l'objectif.




Critère	Note	T	Objectif
Taux de couverture des tests	0,0	✚	Fiabilité - Poids : 1,0 (16)
Poids Logique	1,0	→	Evolutivité - Poids : 20,0 (3) Fiabilité - Poids : 3,0 (5) Maintenabilité - Poids : 3,0 (13)
Taille Méthode/Procédure	1,0	→	Maintenabilité - Poids : 2,0 (20) Réutilisation - Poids : 3,0 (12)
Compréhension	2,0	→	Maintenabilité - Poids : 3,0 (13)
Déstructuration	2,0	→	Evolutivité - Poids : 30,0 (2) Fiabilité - Poids : 3,0 (5) Maintenabilité - Poids : 3,0 (13)
Couplage Traditionnel	3,0	→	Fiabilité - Poids : 1,0 (16)
Intégration	3,0	→	Fiabilité - Poids : 1,0 (16)


Figure 4. Détail des notes obtenues pour un élément

-  La Figure 4 montre un critère ayant obtenu une note égale à « 0 ». Cela signifie que le critère n'a pas été calculé pour l'élément sélectionné.
-  Il peut être utile de rappeler que la manière dont un critère est calculé est totalement indépendante de l'objectif. Pour un même élément, un critère aura une note identique pour tous les objectifs dans lesquels il intervient. Seul le poids dans le calcul de la note de l'objectif diffère.

2 Affichage du code source

L'affichage du code source d'un élément est disponible en cliquant sur le lien  situé à côté du nom de l'élément dans les sections « Bottom-Up⁵ » et « Synthèse d'un critère⁶ ».

Le code source, si disponible, s'affiche alors dans une fenêtre de type « pop-up ».

-  Le code source d'un élément peut ne pas s'afficher si le navigateur (ou une barre d'outils installée sur le navigateur) est configuré pour bloquer tout « pop-up »

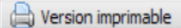
La fenêtre de « pop-up » peut être divisée en deux sections :

- Liste des violations sur l'élément affiché
- Code source de l'élément

⁵ Se référer à la section 1, page 1.

La liste des violations sur l'élément affiché est visualisable sous forme de liste déroulante.


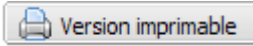
- ⚠ La liste des violations n'est disponible que pour les éléments ayant été analysés avec des outils fournissant le numéro de ligne où la violation a été vue.
- ⚠ Les métriques structurelles (complexité, etc.) ne sont pas représentées dans la liste.

Violations: 171 : Méthode sans Javadoc. 


```
171 final void activate2(Transaction ta, Object a_activate, int a_depth) {
172     beginTopLevelCall();
173     try {
174         stillToActivate(a_activate, a_depth);
175         activate3CheckStill(ta);
176     } catch (Throwable t) {
177         fatalException(t);
178     } finally {
179         endTopLevelCall();
180     }
181 }
182
183 final void activate3CheckStill(Transaction ta) {
184     while (i_stillToActivate != null) {
185
186         // TODO: Optimize! A lightweight int array would be faster.
187
188         Iterator4 i = new Iterator4Impl(i_stillToActivate);
189         i_stillToActivate = null;
190
191         while (i.moveToNext()) {
192             ObjectReference yo = (ObjectReference) i.current();
193
194             i.moveToNext();
195             int depth = ((Integer) i.current()).intValue();
196
197             Object obj = yo.getObject();
198             if (obj == null) {
199                 removeReference(yo);
200             } else {
201                 yo.activate1(ta, obj, depth, i_refreshInsteadOfActivate);
202             }
203         }
204     }
205 }
206
207 public int alignToBlockSize(int length) {
208     return ((length + blockSize - 1) / blockSize) * blockSize;
209 }
```

Figure 5. Affichage du code source d'un élément


Sélectionner une violation dans la liste des violations dirige automatiquement vers la ligne adéquate.

-  Le bouton  permet de générer une version imprimable du code source annoté de l'ensemble des anomalies surlignées en jaune, dans laquelle le surlignage n'apparaît plus et est remplacé par le libellé de l'erreur.

3 Analyse d'impact

L'analyse d'impact pour un élément est disponible en cliquant sur le lien  situé à côté du nom de l'élément dans les sections « Bottom-Up⁷ » et « Synthèse d'un critère⁸ ».

L'analyse d'impact s'affiche dans une fenêtre de type « pop-up ».

 L'analyse d'impact pour un élément peut ne pas s'afficher si le navigateur (ou une barre d'outils installée sur le navigateur) est configuré pour bloquer tout « pop-up »

Elle permet de connaître, rapidement et visuellement, quels éléments seront impactés par une modification effectuée sur un élément donné.

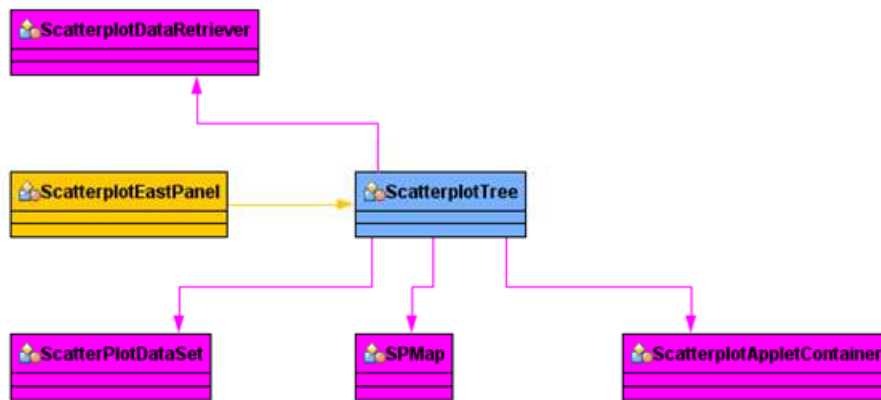



Figure 6. Analyse d'impact

L'élément dont on veut connaître l'impact sur son environnement est représenté en jaune. Les éléments impactés sont en rose. Les éléments impactants sont en orange.

En faisant un clic droit sur un élément autre que l'élément central, et en sélectionnant l'option « Explorer XXX », il est possible d'ajouter au graphique les éléments impactés et impactants relatifs à l'élément sélectionné par ce « clic droit ».

Les liens d'impact sont représentés par des flèches.

 L'élément source d'une flèche impacte l'élément destination.

⁷ Se référer à la section 1.3, page 3.