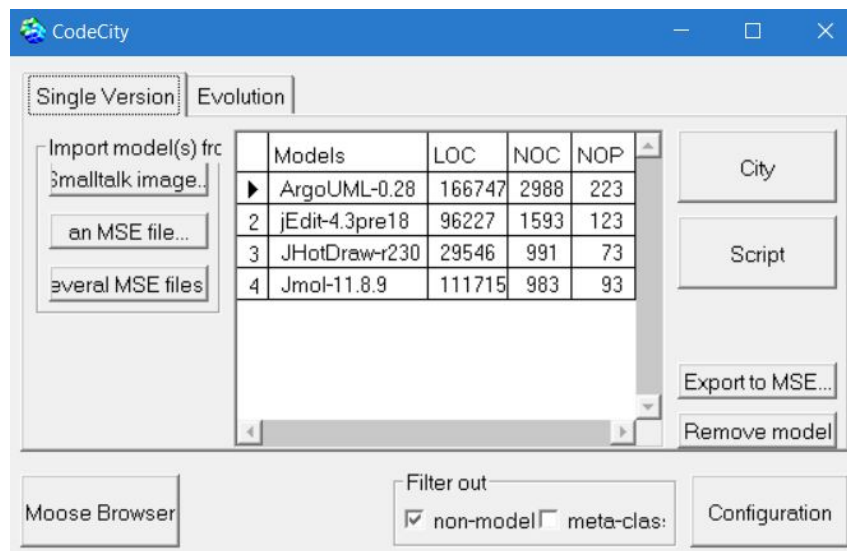


# Tutoriel CodeCity

Thibaut ETIENNE, Fabien FERAUD, Floriant NOVELLON

## Importer un modèle :

Pour importer un modèle que l'on souhaite analyser il faut le passer dans le format de fichier .mse, pour cela il faut utiliser le logiciel iPlasma avec lequel on va exporter notre modèle java en modèle MSE avec comme format FAMIX 2.1 pour que cela soit compatible. Ensuite il suffit de cliquer sur "Import model from an MSE file" dans la partie gauche de la fenêtre ci-dessous.

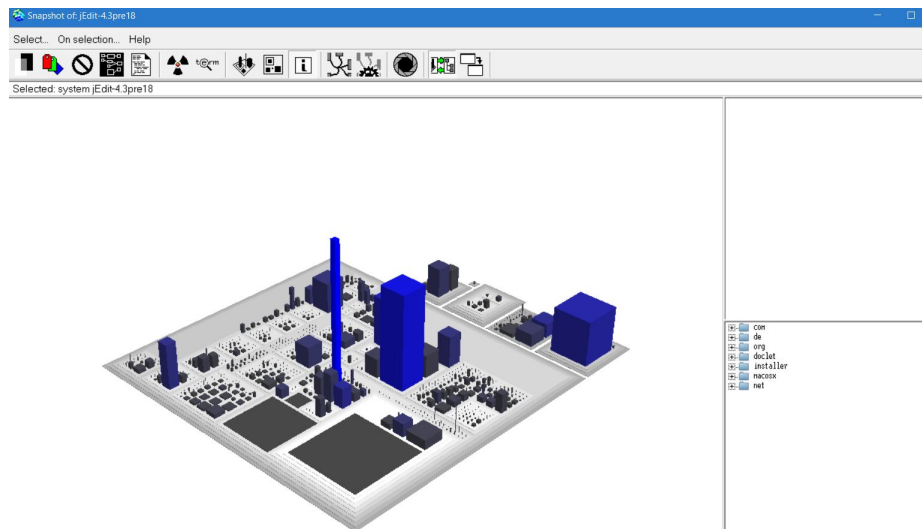


## Analyser un modèle (par ses classes) :

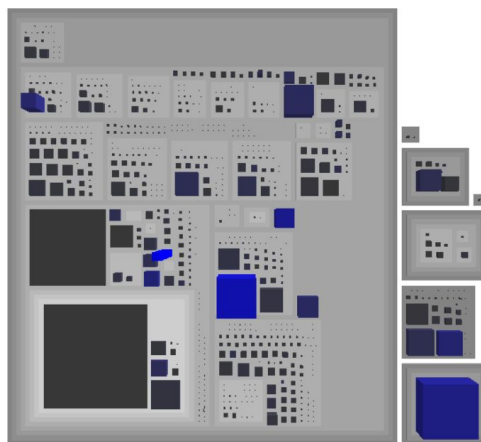
Nous pouvons voir sur cette image que l'on a 4 modèles d'importer, "ArgoUML", "jEdit", "JHotDraw" et "Jmol"

en face de chacune de ces informations nous avons le nombre de lignes de code (LOC), le nombre de classes (NOC) et le nombre de packages (NOP).

Pour analyser plus en détail un modèle il suffit de cliquer dessus (nous allons prendre jEdit) et d'ensuite cliquer sur "City" et cela va ouvrir une nouvelle fenêtre (ci-dessous).



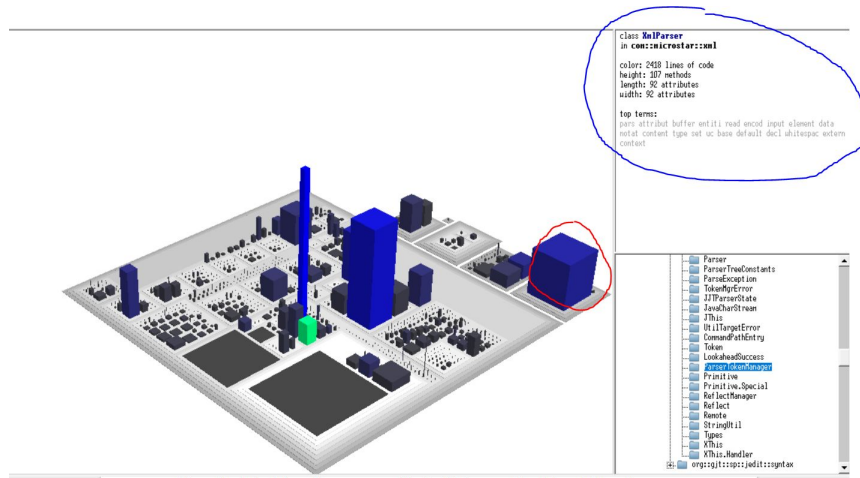
Sur cette fenêtre nous avons une perspective 3D de notre modèle, appelé “perspective isométrique”. Nous pouvons également visionner notre modèle en 2D avec la perspective “top-down” qui ressemble à l’image ci-dessous



On peut ensuite explorer le modèle (avec n’importe quelle perspective) en appuyant sur la touche “w” pour zoomer et “s” pour dé zoomer, et en se déplaçant avec les flèches directionnelles.

Dans ces représentations, chaque “bâtonnet” est une entité logicielle du programme en question (ici jEdit).

Pour obtenir des informations sur ces entités il suffit de passer le pointeur de la souris sur les entités et des informations apparaîtront dans la fenêtre de droite (comme ci-dessous)



On sait donc que l'entité représentée par le cube à droite (entouré en rouge) est une classe nommée "XmlParser", avec 2418 lignes de code, 107 méthodes...

Les différents "blocs" représentent donc des classes, et les "socles" représentent les packages.

En regardant les informations données dans la fenêtre de droite on en apprend un peu plus sur la représentation.

La couleur (colors) indique le nombre de lignes de code dans les classes, plus la couleur est proche du gris, moins il y a de lignes de code. Plus la couleur est bleue, plus il y a de lignes de code.

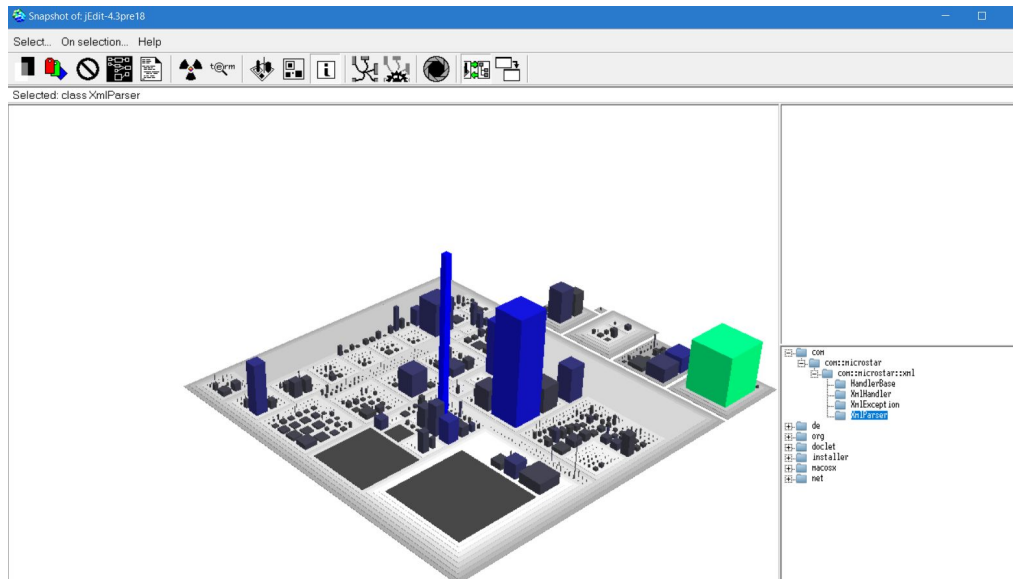
La hauteur (height) indique le nombre de méthodes dans la classe, plus il y en a, plus la hauteur du bloc est grande.

Et enfin, la longueur (length) et la largeur (width) indiquent le nombre d'attributs.

On peut ensuite sélectionner un bloc pour pouvoir l'analyser plus en détail. Pour cela on peut le sélectionner de deux façons différentes :

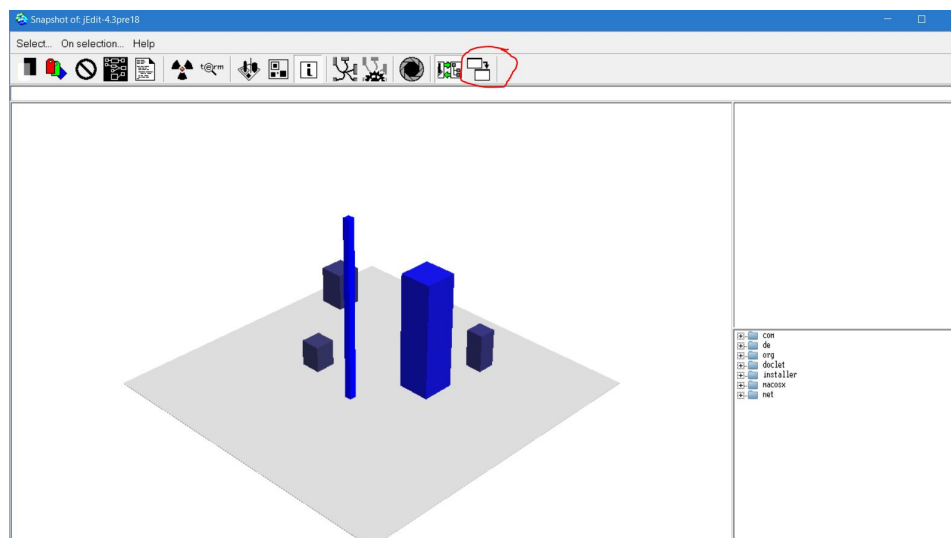
- La première est en cliquant simplement dessus
- La seconde est en faisant une requête (en utilisant le menu "select" en haut de la fenêtre)

Dans les deux cas nous arrivons sur le résultat suivant :

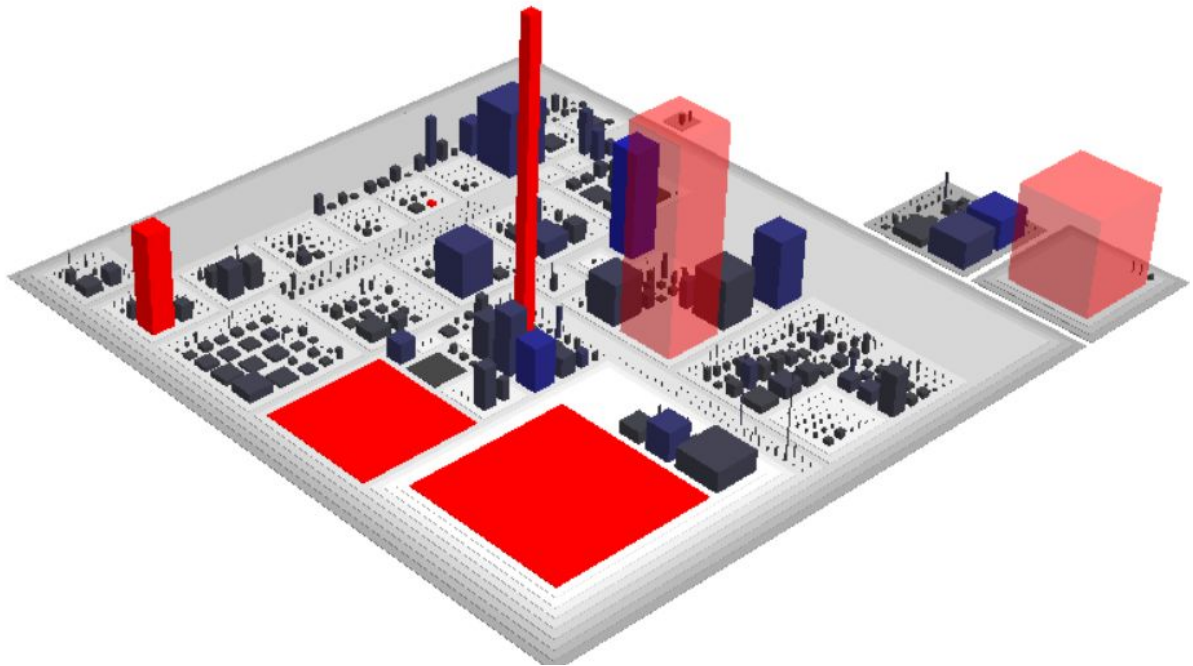



Nous avons donc le détail des éléments de la classe dans la fenêtre en bas à droite. Nous pouvons sélectionner plusieurs classes à la fois en restant appuyer sur la touche ctrl et en cliquant sur les différentes classes que nous souhaitons analyser.

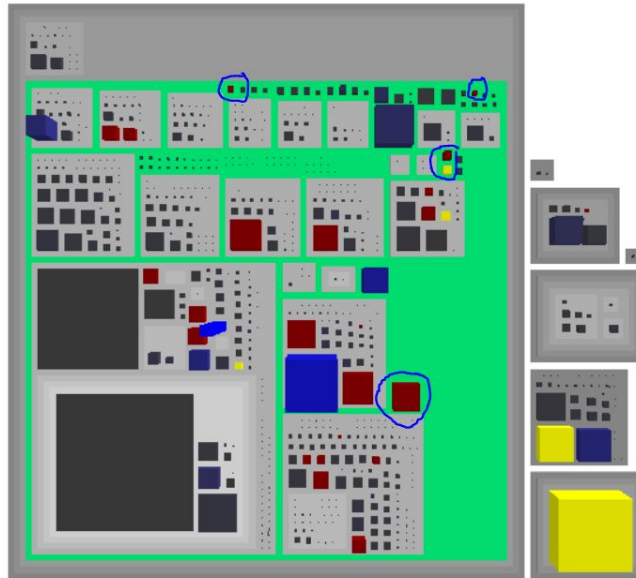
On peut ensuite créer un nouvel affichage à partir des blocs sélectionné en cliquant sur "Isolate sélection" (entouré en rouge ci-dessous) dans la barre d'outils, on obtient par exemple ceci :



Pour une meilleure visibilité on peut ensuite supprimer, colorer, rendre transparents les classes et les packages comme dans l'exemple ci-dessous



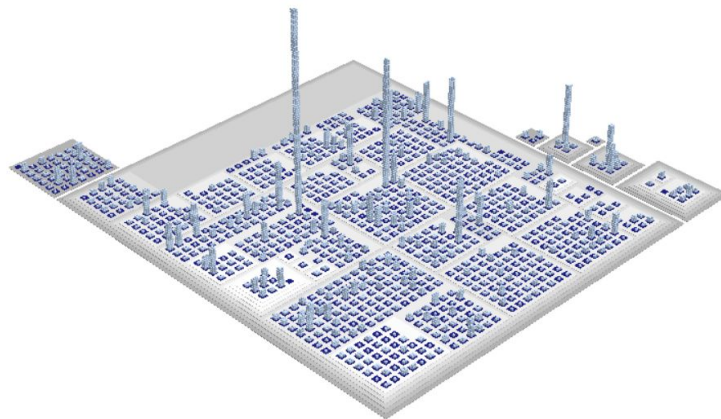
La partie qui est sûrement la plus importante de CodeCity est la visualisation des problèmes de conceptions liées aux classes et aux méthodes. Pour faire cela il suffit simplement de cliquer sur le bouton  dans la barre d'outils et cela affichera la fenêtre "disharmony map". Cette fenêtre va nous permettre de sélectionner les classes et les méthodes ayant des problèmes plus ou moins importants de conception et de les visualiser sur l'affichage, Cela va nous permettre de savoir quelles sont les packages comprenant le plus de ces classes avec des problèmes de conceptions et donc d'identifier les probables futures problèmes liés à notre modèle.



Par exemple nous pouvons voir sur le modèle ci-dessus que le package “org::gjt::sp::jedit” (en bleu-vert) contient 5 classes avec de fort problèmes de conceptions (entouré en bleu).

### **Analyser un modèle (par ses méthodes) :**

Nous pouvons également décider d’analyser le modèle en affichant non plus les classes mais les méthodes, pour cela il suffit de fermer notre modèle, d’aller dans configuration dans le premier panel que nous avons vu durant ce tutoriel, et de sélectionner load → finegrained. Nous aurons donc la visualisation présente ci-dessous.



Nous pourrions donc, sur cette visualisation, appliquer les différentes techniques vues précédemment, comme la suppression, la coloration et également l’analyse détaillée avec la fenêtre “disharmony map”.