

## Tâche 1 : Plan GQM

### Goal :

Analyser la dernière version stable du code de la branche master du JFreeChart pour évaluer sa facilité d'analyse du point de vue du chef du projet.

### Question :

Q1 : Est-ce qu'il y a assez de tests ?

Q2 : Est-ce que les tests sont à jour avec le reste du code ?

Q3 : Est-ce que les tests sont trop complexes ?

Q4 : Est-ce que les tests sont suffisamment documentés ?

### Métriques :

#### Question 1 :

##### TPC (tests par classe) :

TPC nous permettra de calculer le nombre de tests dans chaque fichier test, elle nous sera utile pour évaluer la quantité des tests effectués. Si TPC est relativement faible par rapport à la taille du projet, cela peut indiquer des lacunes dans les tests, ce qui signifie qu'il pourrait y avoir des parties du code non couvertes par les tests. Pour le mesurer nous modifierons le programme t1s que nous avons implémenté lors du tp1 pour nous donner en plus le tassert moyen, ce qui est équivalent à Tpc.

##### TPP (tests par package) :

Nous allons collecter le nombre de test présent dans chaque package que nous, avons, nous évaluerons la moyenne de tous les TPPs. Un seuil pour le nombre de test par package est fixé sur la base de la complexité du projet, une moyenne des TPPs en dessous du seuil sera un indice que le projet n'est pas suffisamment testé. Pour le mesurer nous modifierons le programme t1s que nous avons implémenté lors du tp1.

##### PCC (Pourcentage code couvert) :

Mesure le pourcentage de ligne de code non testées dans chaque package. Si un package a un pourcentage élevé de ligne de code non testées, cela pourrait indiquer que ce package est insuffisamment testées. Un seuil sera fixé, si le PCC est en dessous de ce seuil cela signifiera que le package concerné a besoin d'amélioration sur la couverture de ses tests. Pour le mesurer nous utiliseront le plug-in pour projet maven jacoco.

#### Question2 :

##### AGE (âge d'un fichier) :

Nous permettra de déterminer si nos fichiers tests sont plus vieux que les fichiers des classes qu'ils testent. Un seuil d'un certain nombre de jour est fixé, si la moyenne des fichiers tests sont en dessous du seuil cela signifiera que la majorité des tests sont plus vieux que les fichiers qu'ils tests, et donc ne seront pas considéré à jour. Nous ferons une implémentation de age pour mesurer celui-ci.

##### NCH :

Calcule le nombre de commits dans l'historique d'une classe, pour chaque classe de test cette metrique compte le nombre de commits associés à cette classe depuis le début de son historique. Nous comparerons les valeurs de NCH des classes de tests à celles du reste du code. Si le NCH des classes de tests est significativement inférieur à celui du code source principal, cela peut indiquer que les tests ne

sont pas mis à jour aussi fréquemment que le code source, ce qui pourrait indiquer que le code n'est pas à jour.

PCC (Pourcentage code couvert) :

Mesure le pourcentage de ligne de code non-testé. Un pourcentage élevé nous indiquera que beaucoup de code n'est pas testé, ce qui pourrait suggérer un ajout de lignes de code sans ajout de tests pour les vérifiés, ce qui peut nous indiquer que les tests ne sont plus à jours

Question3 :

Ratio taille code / taille test :

Mesure la taille du code de test en comptant le nombre total de lignes de code dans les classes tests. On mesurera la taille du code source en comptant le nombre total de lignes de code dans le code que nous testons. On fera ensuite le ration Ratio taille code / taille test. Des seuils appropriés en fonction de JFreeChart sont fixés. Un ratio faible peut indiquer que les tests sont proportionnellement plus complexes que le code de la classe testée. Un ratio élevé, indiquera que les tests sont relativement plus simples que le code source. Nous utiliserons l'application cloc pour le mesurer.

NCLOC :

Calcule le nombre de lignes de code non-vides qui ne sont pas des commentaires. Un NCLOC élevé dans les fichiers tests par rapport à la taille de JFreeChart pourrait indiquer que les tests sont trop complexes. Nous utiliserons l'application cloc pour le mesurer.

TCMP :

Mesure ratio entre le nombre de ligne de code et le nombre de tests dans une classe (TLOC/TASSERT). Nous évaluerons le TCMP moyen de toutes les classe de test de jfreechart. Un ratio élevé nous indiquera qu'en moyenne les tests sont trop complexe et un ratio faible nous indiquera que les tests ne sont pas trop complexes. Nous modifierons le programme tls pour obtenir TCMP moyen.

Question4 :

CLOC :

Donnera le nombre de lignes de commentaires. Un seuil sera fixé et si le nombre de commentaire est en dessous de ce seuil, ça nous indiquera que le projet n'est pas suffisamment documenté. Nous utiliserons le programme cloc pour mesurer CLOC.

DC (densité de commentaires) :

Un nombre de commentaires important dans une classe test peut être indicateur de documentation, mais cette métrique ne suffit pas forcément. Nous fixerons un seuil, une DC élevée est un signe que les tests sont potentiellement bien documentés, tandis qu'une DC faible peut indiquer que les tests nécessitent une amélioration de la documentation. Cette métrique sera calculée à l'aide du programme cloc.

Tâche 2 : Mesurer jfreechart

```
github.com/AlDanial/cloc v 1.98 T=2.84 s (360.7 files/s, 99798.8 lines/s)
```

Language	files	blank	comment	code
Java	1023	26594	123888	132539
SUM:	1023	26594	123888	132539























































































Utilisation de cloc dans jfreechart/src

```
github.com/AlDanial/cloc v 1.98 T=1.98 s (335.2 files/s, 106046.3 lines/s)
```

Language	files	blank	comment	code
Java	664	18363	99974	91740
SUM:	664	18363	99974	91740

Utilisation de cloc dans jfreechart/src/main

Lien GitHub:  
<https://github.com/AlexisBoucher/IFT3913>

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cov.	Missed	Cov.	Lines	Missed	Methods	Missed	Classes
org.free.chart.render.xy		34%		28%	1 276	1 846	3 669	6 034	202	564	7	43	
org.free.chart.render.xy		54%		54%	1 469	2 841	2 820	6 109	126	1 212	2	32	
org.free.chart.render.xy		52%		42%	1 046	1 838	2 567	5 978	231	723	3	44	
org.free.chart.render.category		55%		36%	824	1 241	1 911	4 127	101	382	1	25	
org.free.chart.render.xy		0%		0%	282	282	1 172	1 172	140	16	16	16	
org.free.chart.render.xy		0%		9%	442	503	1 235	1 654	147	190	6	12	
org.free.chart.render.xy		59%		59%	374	596	900	2 049	142	280	4	11	
org.free.chart.data.time		64%		48%	474	990	936	2 616	150	503	3	26	
org.free.chart.annotations		56%		52%	227	451	635	1 516	98	251	2	16	
org.free.chart.xy		57%		65%	383	937	597	1 142	180	561	3	40	
org.free.chart.xy		73%		61%	229	455	591	2 407	142	291	0	14	
org.free.chart.xy		61%		62%	220	458	479	1 460	58	210	0	7	
org.free.chart.render		61%		53%	300	603	548	1 513	88	276	3	13	
org.free.chart.plot.compass		27%		33%	164	327	445	678	49	106	0	11	
org.free.chart.plot		7%		7%	148	162	434	482	48	58	3	8	
org.free.chart.labels		52%		45%	199	394	457	1 093	71	215	1	27	
org.free.chart.blocks		67%		67%	147	477	429	1 381	71	116	0	17	
org.free.chart.internal		68%		61%	266	566	404	1 489	75	244	0	14	
org.free.chart.series		58%		54%	157	290	298	799	21	72	0	8	
org.free.chart.line		43%		30%	149	229	338	631	41	98	1	6	
org.free.chart.general		73%		67%	221	523	347	1 255	52	164	2	15	
org.free.chart.render		64%		0%	143	336	368	994	52	163	1	7	
org.free.chart.data.time		17%		24%	61	88	205	294	19	34	0	1	
org.free.chart.ENTITY		44%		33%	137	208	258	468	73	129	2	15	
org.free.chart.pantti		55%		55%	105	212	185	486	57	129	0	5	
org.free.data		80%		75%	139	456	218	1 011	38	195	2	18	
org.free.data		71%		75%	75	75	213	213	47	47	8	9	
org.free.chart.text		56%		56%	152	274	153	722	27	104	9	6	
org.free.chart.seriesImpl		0%		0%	78	78	190	190	19	19	3	3	
org.free.chart.data.flow		41%		33%	65	95	119	206	22	38	1	4	
org.free.chart.data		52%		30%	94	128	142	254	35	54	0	2	
org.free.chart.data		68%		57%	12	105	37	105	8	51	51	4	
org.free.chart.data		77%		66%	83	205	100	437	14	90	4	4	
org.free.chart.data.category		0%		0%	35	35	98	98	31	31	4	4	
org.free.chart.data		0%		0%	15	15	68	68	5	5	1	1	
org.free.chart.data		0%		0%	16	16	59	59	6	6	1	1	
org.free.chart.data		74%		58%	58	278	61	222	22	50	4	4	
org.free.chart.data.time		64%		33%	73	73	53	158	19	48	0	4	
org.free.chart.data		82%		66%	57	126	54	313	12	52	1	8	
org.free.chart.data.resources		32%		n/a	13	18	25	36	13	18	4	6	
org.free.chart.data		73%		79%	12	34	31	103	7	17	1	5	
org.free.chart.data		75%		58%	16	16	77	77	3	3	2	3	
org.free.chart.data		12%		38%	12	38	12	38	12	38	0	10	
Total	93 908 of 206 375	54%	11 406 of 21 087	45%	10 492	18 759	23 704	54 526	3 046	8 140	87	541	

```
Age en moyenne des fichiers du programme : 289.0 jours
Age en moyenne des fichiers tests: 289.0 jours
L'age moyen des test sont plus jeunes de 0.0jour
```

**Total, 39771, 27, 5.294363, 516**

## Utilisation de jacoco dans jfreechart

- Utilisation de JaCoCo pour collecter des données de couverture de code.
- Exécution des tests unitaires JFreeChart avec JaCoCo activé.
- Extraction des données de couverture pour chaque classe et package

- Utilisation de l'outil CLOC (Count Lines of Code) pour collecter les données sur le nombre de ligne par classe.
- Analyse des résultats de CLOC pour obtenir le nombre de méthodes par classe.

- La métrique âge recueille les données sur l'âge moyen des fichiers.
- Les données recueillis ne nous permettent pas de conclure concernant l'état de mise à jour des classes car la date du projet reste celle de l'ajout de JFreeChart à notre projet ou au Git

- Cette métrique recueille les données :
  - Nombre de lignes qui ne sont pas des commentaires dans une classe
  - Nombre d'assertion dans une classe
  - Tcmp : qui suggère des classes complexes selon un seuil
  - Nombre de test par packages
- Les moyennes sont calculées
- Une analyse des résultats est faite en fonction de la complexité du projet JFreeChart

Pour chaque métrique nous exécuterons des programmes appropriés et collecteront les données pertinentes. Les données collectées seront stockées dans des fichiers CSV ou d'autres formats lisibles comme le .txt.

GitHub est utilisé pour fournir l'accès à nos programmes et données collecté, lien du répertoire en tête de page.

### Tâche 3 : Réponses aux questions

**Question 1 :** Pour PCC, nous avons fixé un seuil de 60% pour pouvoir conclure qu'il y a assez de test, après analyse avec l'outil Jacoco nous concluons qu'il pourrait y avoir plus de tests car le pourcentage de couverture est de 54% ce qui est inférieur à 60%. Pour TPP nous avons un seuil à 600 et nous observons une moyenne de 516. Pour TPC nous avons un seuil à 40 et nous observons une moyenne de 27, ce qui nous indique qu'il n'y a pas assez de tests

**Question 2 :** PCC montre qu'il pourrait y avoir plus de test, (que des lignes de code ont été ajoutées après que les tests aient été créés) mais nous ne pouvons pas conclure que les tests sont à jour par rapport au reste du code du projet JFreeChart car les données que nous avons collectées ne sont pas représentatives. Cela est dû au fait que le GitHub n'a pas gardé les dates exactes. Dans les normes nous aurons fixé un seuil de 30 jours d'âge moyen, tous les fichiers tests vieux de 30 jours et plus que la classe correspondante, auraient été classés non à jour. Si plus de 50% des fichiers étaient classés vieux cela indiquerait que les tests ne sont plus à jour. Pour la mesure de NCH nous avons manqué de temps et ne l'avons pas évalué. Puisque nous n'avons qu'une mesure valide nous ne pouvons pas conclure, mais en raison du résultat donné par Jacoco nous pensons que les tests ne sont pas à jour.

**Question 3 :** Nous avons mis un seuil à 40% du code pour la métrique DC, nous observons une densité de code de test de 40 799/132 539 ce qui correspond à un peu plus de 30%. Pour la métrique tcmp nous avons mis un seuil de 10 tloc/tassert. Nous voyons que tcmp moyen est d'environ 5,3. Sur cette base, les tests ne sont pas trop complexes.

**Question 4 :** Sur 81951 lignes recueillies par la métrique Cloc au total dans le projet, nous avons 39771 qui ne sont pas des commentaires donc plus de 50% des lignes sont des commentaires cela peut potentiellement indiquer une densité de commentaire élevée dans le fichier. D'autres analyses sont nécessaires pour donner une conclusion plus précise mais sur la base de cette analyse nous pouvons affirmer que le code est potentiellement assez documenté.