RAPPORT PROJET GENIE LOGICIEL

Username : Assembleur12

NOM ET PRENOM : TIL-LA DJIBRIL SAMUEL

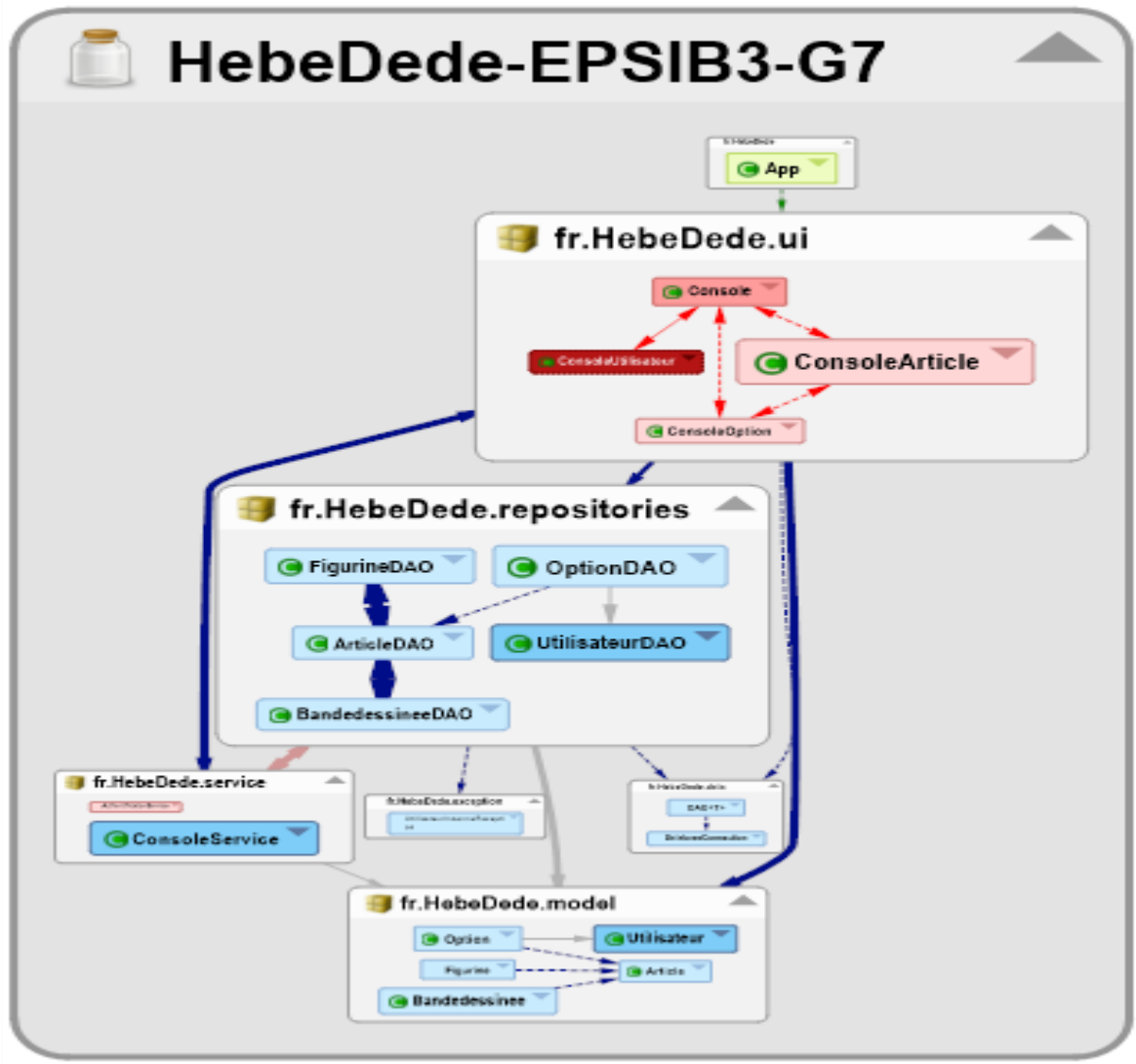

Figure 1

A travers cette figure (figure 1), nous voyons les dépendances entre les classes et les packages à ce niveau nous avons 3 dépendances donc le **couplage est faible** entre les classes d'où la maintenance du système ne sera pas compliquée.

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
Number of Parameters (avg/max per method)		1	1,564	12	/TPE GENIE LOGICIEL/src/fr/HebeDede/model/Bandedessinee.java	Bandedessinee
Number of Static Attributes (avg/max per type)	18	0,783	0,832	3	/TPE GENIE LOGICIEL/src/fr/HebeDede/ui/Console.java	
Efferent Coupling (avg/max per packageFragment)		1,857	1,807	5	/TPE GENIE LOGICIEL/src/fr/HebeDede/repositories	
Specialization Index (avg/max per type)		0,007	0,031	0,154	/TPE GENIE LOGICIEL/src/fr/HebeDede/model/Utilisateur.java	
Number of Classes (avg/max per packageFragment)	23	3,286	1,485	5	/TPE GENIE LOGICIEL/src/fr/HebeDede/model	
Number of Attributes (avg/max per type)	29	1,261	2,191	9	/TPE GENIE LOGICIEL/src/fr/HebeDede/model/Bandedessinee.java	
Abstractness (avg/max per packageFragment)		0,071	0,175	0,5	/TPE GENIE LOGICIEL/src/fr/HebeDede/data	
Normalized Distance (avg/max per packageFragment)		0,586	0,343	1	/TPE GENIE LOGICIEL/src/fr/HebeDede/exception	
Number of Static Methods (avg/max per type)	35	1,522	3,005	12	/TPE GENIE LOGICIEL/src/fr/HebeDede/ui/ConsoleArticle.java	
Number of Interfaces (avg/max per packageFragment)	0	0	0	0	/TPE GENIE LOGICIEL/src/fr/HebeDede	
Total Lines of Code	2162					
Weighted methods per Class (avg/max per type)	505	21,957	23,33	100	/TPE GENIE LOGICIEL/src/fr/HebeDede/ui/ConsoleArticle.java	
Number of Methods (avg/max per type)	108	4,696	5,622	21	/TPE GENIE LOGICIEL/src/fr/HebeDede/model/Bandedessinee.java	
Depth of Inheritance Tree (avg/max per type)		1,652	0,758	3	/TPE GENIE LOGICIEL/src/fr/HebeDede/exception/UserAlreadyExistsE...	
Number of Packages	7					
Instability (avg/max per packageFragment)		0,342	0,34	1	/TPE GENIE LOGICIEL/src/fr/HebeDede	
McCabe Cyclomatic Complexity (avg/max per method)		3,531	3,771	21	/TPE GENIE LOGICIEL/src/fr/HebeDede/ui/ConsoleArticle.java	modifFiche
Nested Block Depth (avg/max per method)		1,825	1,167	5	/TPE GENIE LOGICIEL/src/fr/HebeDede/service/ConsoleService.java	renseigneChampMaxMin...
Lack of Cohesion of Methods (avg/max per type)		0,176	0,31	0,889	/TPE GENIE LOGICIEL/src/fr/HebeDede/model/Bandedessinee.java	
Method Lines of Code (avg/max per method)	1643	11,49	13,819	56	/TPE GENIE LOGICIEL/src/fr/HebeDede/ui/Console.java	selectMenu
Number of Overridden Methods (avg/max per type)	2	0,087	0,408	2	/TPE GENIE LOGICIEL/src/fr/HebeDede/model/Utilisateur.java	
Afferent Coupling (avg/max per packageFragment)		5,143	3,27	10	/TPE GENIE LOGICIEL/src/fr/HebeDede/model	
Number of Children (avg/max per type)	7	0,304	1,081	5	/TPE GENIE LOGICIEL/src/fr/HebeDede/data/DAO.java	

Figure 2

La figure deux nous montre les plusieurs métriques parmi lesquelles nous intéresserons à quelques-uns (McCabe Cyclomatic complexity, Method lines of code ….) puisque La complexité cyclomatique d'une méthode augmente proportionnellement au nombre de points de décision. Une méthode avec une haute complexité cyclomatique est plus difficile à comprendre et à maintenir. Et aussi une complexité cyclomatique trop élevée (supérieure à 30) indique qu'il faut refactoriser la méthode, une complexité cyclomatique inférieure à 30 peut être acceptable si la méthode est suffisament testée. La complexité cyclomatique est liée à la notion de "code coverage", c'est à dire la couverture du code par les tests. Dans l'idéal, une méthode devrait avoir un nombre de tests unitaires égal à sa complexité cyclomatique pour avoir un "code coverage" de 100%. Cela signifie que chaque chemin de la méthode a été testé.