Dossier de Génie Logiciel

Date de rentrée : dernière séance de l'année.

1 Objectifs

- Concevoir et construire une application orientée objets en appliquant des principes de TDD.
- Justifier les choix opérés au niveau de la conception.
- Appliquer les techniques de tests vues au cours.

2 Contexte

Ecrire une application permettant de détecter des classes qualifiées de "God class". Une telle classe a trop de responsabilités et accède généralement directement aux données d'autres classes. La réutilisabilité et la facilité de compréhension de cette classe n'est dès lors pas facile. Une bonne pratique de programmation orientée objet veut que l'on attribue au plus une seule responsabilité à une classe.

Les métriques utilisés pour détecter une "God class" sont(cf. ouvrage Object-oriented Metrics in Practice de Stéphane Ducasse et Michèle Lanza)

- Mesure de la complexité de la classe : Weighted Method Count (WMC) : `http://www.aivosto.com/project/help/pm-oo-ck.html`.
  - La nombre de méthodes dans une classe ou
  - la somme des complexité statique de toutes les méthodes d'une classe. Dans ce dernier cas, la complexité cyclomatique de McCabe peut être utilisé comme mesure de la complexité.

  Une valeur seuil pour un WMC trop élevé est 47.
- Mesure de la cohésion d'une classe : Tight Class Cohesion (TCC)(`http://www.aivosto.com/project/help/pm-oo-cohesion.html`) Le métrique sert à évaluer la cohésion d'une classe. Sa valeur est située en 0 et 1. Il correspond au nombre relatif de pairs de méthodes d'une classe qui accède au moins à un même attribut de cette classe. En dessous de 1/3, on considérera que la classe a peu de cohésion. Votre solution doit permettre l'utilisation d'un autre métrique de cohésion.
- Indicateur du respect de l'encapsulation des autres classes : Access to Foreign Data (ATFD) : représente le nombre de classes extérieures auxquelles la classe accède les attributs(de manière directe ou par les accesseurs). Pas plus que quelques uns(à définir) comme limite.

Les classes que l'on cherche à détecter dans un projet pour lequel on se propose d'étudier le code sont donc celles qui sont trop complexes, avec peu de cohésion et qui accèdent directement aux variables membres des autres classes. Remarque : cela ne signifie pas que la classe détectée est mal conçue mais simplement qu'il y a là un indicateur qui nous invite à pousser la réflexion quand à la conception du code.

3 A faire

- Représentez dans un diagramme de classes les principales classes de votre solution.
- Représentez dans un diagramme approprié les différents packages qui composent votre solution.
- Représentez dans un diagramme de séquence les principales interactions entre les différents composants pour remplir la fonctionnalité principale.

- Justifier vos choix de conception(par exemple par les patrons de conception utilisés).
- Utiliser trois techniques de tests au minimum :
  - ☐ Une technique pour les tests fonctionnels(black box testing)
  - ☐ Une technique pour les tests d'intégration.
  - ☐ Une technique de test se basant sur le code(white box testing).
  - Vous penserez à définir un critère d'adéquation pour les suites de tests développées.
  - Au minimum, un rapport de couverture de code sera fournit.
- Illustrez votre solution avec deux cas concrets.

4 Consignes

- Un rapport est à remettre le jour de la présentation du dossier. Il traitera au minimum des points obligatoires de l'énoncé. La forme est laissée à votre propre convenance.
- Soyez précis dans les points que vous voulez expliquer.
- Le dossier demande de votre part des initiatives, de la recherche et de la critique d'informations,...
- L'élaboration du travail et les choix opérés feront l'objet de discussions pendant les séances de laboratoire.

5 Quelques outils que vous pouvez utiliser

La liste reprise ci-dessous est sensée vous donner quelques références d'outils à utiliser dans le cadre du travail. Il est permis d'en utiliser d'autres.

- un parseur pour le code Java. Exemple `https://code.google.com/p/javaparser/wiki/UsingThisParser`
- un serveur d'intégration. Exemple `http://jenkins-ci.org/`
- des outils pour mettre en oeuvre les tests. Exemples :
  - `http://junit.org/`
  - `http://cobertura.github.io/cobertura/`