Université Toulouse III – Paul sabatier
L2 Informatique

Antoine de Roquemaurel (antoine.de-roquemaurel@univ-tlse3.fr)
Fabrice Valleix (valleix.fabrice@gmail.com)
Groupe 2.2

Dossier

Systèmes d'information et applications Web

Toulouse, le 10 mai 2013

Table des matières

0.1 But du document

C'est une description de haut niveau du produit, c'est-à-dire l'architecture générale du système, en termes de « modules », de sous modules et de leurs interactions. De plus, chaque module doit être décrit (définition des interfaces et des fonctionnalités générales). Ce document doit en premier lieu asseoir la confiance en la finalité et la faisabilité du produit, et, en second lieu, servir de base pour l'estimation des tâches à effectuer et du calendrier de leur réalisation.

Le « Dossier de Conception Préliminaire » doit également mettre en évidence le plan de tests, en termes de besoins de l'utilisateur, et montrer que l'on peut y satisfaire grâce à l'architecture proposée.

0.2 Compilation

La compilation du projet se fait à l'aide de l'utilitaire `Make`, ainsi la simple commande `make` à la racine du projet suffit à compiler le projet.

Cependant, afin de pouvoir compiler le projet, il est indispensable de posséder la bibliothèque *Ncurses* sur sa machine, sinon la compilation ne fonctionnerai pas.
Il est possible de l'installer avec la commande `apt-get install libncurses5-dev` sur les Linux utilisant le gestionnaire de paquet de Debian.

Les tests quant à eux se compile à l'aide de la commande `make test`, cette commande va compiler puis exécuter tous les tests, cependant si vous ne possédez pas *CUnit* sur votre machine, il est également indispensable de taper la commande suivante afin de signaler au système l'emplacement de la bibliothèque.

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:‘pwd‘ /Cunit/lib && export LD_LIBRARY_PATH
```

0.3 Exécution

Afin de tester l'application, deux types de tests sont présent dans le projet.

0.3.1 Les tests fonctionnels

Ces tests sont tous présents dans l'executable `./boggle`. Afin d'appeler les différentes fonctionnalités du programme, il est nécessaire de faire passer un paramètre, celui-ci peut prendre la forme d'une de ces trois chaines de caractères :

`-solveur` Correspond à la version 1 du projet. Afin d'appeler la version 1 de l'application, l'exécutable doit être appelé à l'aide de l'argument `-solveur`

Dans cette version, une grille carrée de la taille demandée par l'utilisateur est génére, en tenant compte de la fréquence des lettres dans la langue Française. Une fois la grille générée, la position d'une case est demandée à l'utilisateur, l'utilisateur entre donc les deux coordonnées, et tous les mots commençant par cette case seront affichés à l'écran.

Attention, les coordonnées de la grille commences à zéro.

`-texte` Afin d'appeler la version 2, l'exécutable doit être appelé à l'aide de l'argument `-text`

Cette version fait appel à la version 1, en effet, au lancement de l'application, il est de nouveau demandé la taille de la grille, ensuite l'intégralité de la grille générer est résolue. Une fois cette étape franchie, l'utilisateur à 3 minutes pour entrer le plus de mots possibles, l'application lui signalant si le mot est accepté ou non, une fois ce temps impartis, la solution est affichée, puis le nombre de points obtenu par le joueur.

`-ncurses` Afin d'appeler la version 3, l'exécutable doit être appelé à l'aide de l'argument `-ncurses`

Cette version suit le même principe que la version précédente, à la différence près qu'elle utilise la bibliothèque *Ncurses*. Ainsi, la saisie des mots se fait dorénavant avec les touches fléchées du clavier, et espace pour ajouter une lettre au mot. Pour proposer le mot surligné, la touche entrée doit être appuyée. Il est également possible de demander le nombre de mots commençant par la case sélectionnée à l'aide de la touche h.

Une fois les 3 minutes écoulées, les mots proposés par l'utilisateur et le nombre de points obtenus sont affichés, il est proposé à l'utilisateur d'afficher la solution complète.

0.3.2 Les tests unitaires

Chapitre 1

Documentation

Chapitre 2

Gestion de projet

Pour ce projet, nous étions deux à travailler dessus, ainsi nous avons utilisé plusieurs techniques afin de se coordonner et de limiter les problèmes. Ceci n'est pas notre premier projet ensemble, notre travail en fut simplifié.

2.0.3 Conventions de codage

2.1 Un outil de gestion de projet : Redmine

FIGURE 2.1 – Affichage des demandes dans Redmine

Pour le projet, nous avons utiliser *Redmine*, une plateforme web de gestion de projet (Cf figure **??**). Elle nous a permis de simplifier le travail, et de ne rien oublier.

En effet, nous pouvons créer des tâches, signaler qu'elles sont en cours/terminés/en tests, leur donner des dates limites, les affecter à une personne etc...Ainsi lorsque l'un de nous commençait une tâche, il le signalait sur le *redmine*, ce qui permettait de tenir au courant son binôme de ses actions et de l'avancée du projet.

2.2 Un logiciel de versionnement : Git

Afin de limiter les problèmes du travail collaboratif, nous avons utilisé un logiciel de versionnement Git. Il a deux intérêt, tout d'abord, nous pouvons travailler à deux en parallèle sur le projet sans se soucier de fusionner notre travail [1].

D'autre part, tous les logs étant enregistrés, nous pouvons savoir qui à fait quoi et quel jour, cela permet de voir également l'avancée du projet.

Enfin, toutes les modification sont stockées sur le serveur, ainsi en cas de problème, il est très facile

1. À condition de ne pas travailler sur deux lignes de code identiques

de revenir à la version précédente ou même de comparer deux versions afin de voir les changements et de comprendre rapidement pourquoi une fonctionnalité à régressé.

2.2.1 Logiciel d'analyse de code : Sonar

2.2.2 Logiciel de gestion de projet : Redmine

2.2.3 Logiciel de versionnement : Git