Université Toulouse III – Paul sabatier
L2 Informatique
Projet tuteuré

Antoine de Roquemaurel
Fabrice Valleix
Groupe 2.2

Dossier de conception préliminaire

Projet de Boggle

Toulouse, le 19 mars 2013

Table des matières

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.

2 Diagramme de décomposition en modules

La description détaillée des différents modules est disponible section 3.

R Afin de ne pas surcharger le schéma, le module `Utile` n'a pas été représenté ici, en effet tous les modules du projet sont susceptibles d'en avoir besoin, de plus ce module ne contient pas de fonctions spécifiques au projet mais des fonctions utiles travaillant sur des types de bases.

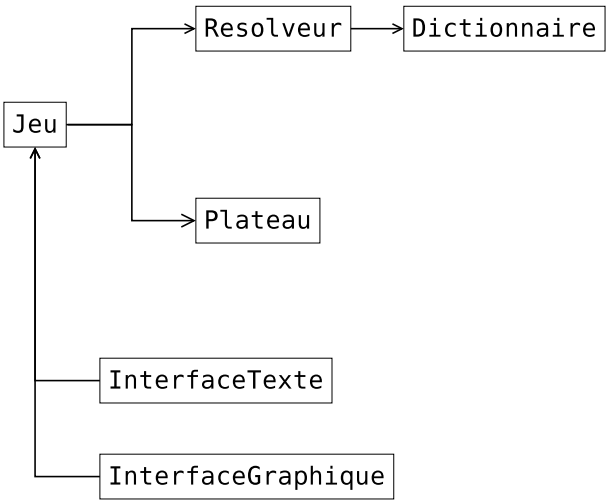

Figure 1 – Diagramme de décomposition en modules

3 Description des différents modules

Un diagramme représentant l'interaction entre les différents modules est disponible section 2.

3.1 Module `Utile`

Rôle	Toutes les fonctions de bases utiles au projet, ces fonctions travaillent sur des types de bases et ne sont pas spécifiques au projet, mais ce module permet de mieux organiser le code.
Type de données	Contient uniquement des traitements
Dépendances	Aucune
Fonctionnalités fournies	La liste sera complété au fur et a mesure du projet en fonction des besoin nécessaires, en voici déjà quelques une : supprimer les accents d'une chaine de caractère, mettre une chaine de caractère en majuscule, n'afficher un message qu'en cas de mode debug, trouver la première chaine de caractère présente dans un tableau, retourner un booléen en fonction d'une certaine probabilité, etc...

Table 1 – Module `Utile`

3.2 Module `Plateau`

Rôle	Gérer la grille de Boggle
Type de données	Tableau à deux dimensions de `char`
Dépendances	`Utile`(3.1)
Fonctionnalités fournies	Générer la grille, Retourner la lettre concernant une case donnée

Table 2 – Module `Plateau`

3.3 Module `Dictionnaire`

Rôle	Gérer le dictionnaire du Boggle
Type de données	Fichier `FILE*` pointant sur le dictionnaire
Dépendances	`Utile`(3.1)
Fonctionnalités fournies	Ouvrir le dictionnaire, parcourir le dictionnaire, dire si un mot est présent dans le dictionnaire ou non.

Table 3 – Module `Dictionnaire`

3.4 Module `Resolveur`

Rôle	Résoudre une grille de Boggle
Type de données	Structure contenant la grille de Boggle, le dictionnaire et un tableau de `char*` avec tous les mots possibles
Dépendances	`Dictionnaire`(3.3), `Plateau`(3.2), `Utile`(3.1)
Fonctionnalités fournies	Résoudre la grille, signaler si un mot est présent dans la grille, retourner la liste des mots de la grille commençant par une lettre.

Table 4 – Module `Resolveur`

3.5 Module Jeu

Rôle	Jouer au Boggle
Type de données	Structure contenant le Plateau et le Résolveur
Dépendances	Plateau(3.2), Resolveur(3.4), Utile(3.1)
Fonctionnalités fournies	Proposer une lettre, Lancer le compte à rebours, Signaler si un mot proposé est correct, retourner le nombre de point obtenus, signaler si le joueur à gagner le jeu ou non

Table 5 – Module Jeu

3.6 Module InterfaceTexte

Rôle	Afficher et permettre de jouer au Boggle en mode texte
Type de données	Jeu
Dépendances	Jeu(3.5), Utile(3.1)
Fonctionnalités fournies	Toutes les fonctions d'affichage et de saisie

Table 6 – Module InterfaceTexte

3.7 Module InterfaceGraphique

Rôle	Afficher et permettre de jouer au Boggle en mode semi graphique
Type de données	Jeu(3.5)
Dépendances	Jeu, bibliothèque externe ncurses, Utile
Fonctionnalités fournies	Toutes les fonctions d'affichage et de saisie

Table 7 – Module InterfaceGraphique

4 Répartition des tâches entre chaque membre

Un module sera toujours affecté à un membre du groupe, celui-ci sera en charge de vérifier que le module avance dans le temps impartis, et de s'occuper de l'intégration. Chacune des tâches seront affecté à un membre du groupe qui devra implémenter la tâche dans les délais prévus.

Le module Utile ne possède personne qui lui est assigné, en effet ce module se remplira en fonction de l'avancement des autres modules, et sera donc développé par les deux membres du binôme tout au long du projet.

L2 - Projet (Boggle) - Demandes

#	Tâche parente	Tracker	Début	Échéance	Sujet	Assigné à
L2 - Projet (Boggle) (28)						
250		Document	03-03-2013		Conception	
276		Module	16-03-2013	30-04-2013	Utile	
277		Module	16-03-2013	23-03-2013	Plateau	Fabrice Valleix
284	Module #277: Plateau	Tache	16-03-2013	23-03-2013	Créer plateau	Antoine de Roquemaurel
285	Module #277: Plateau	Tache	19-03-2013	23-03-2013	Générer grille de Boggle en fonction de la taille	Fabrice Valleix
278		Module	16-03-2013	20-03-2013	Dictionnaire	Fabrice Valleix
286	Module #278: Dictionnaire	Tache	16-03-2013	17-03-2013	Créer dictionnaire	Antoine de Roquemaurel
287	Module #278: Dictionnaire	Tache	16-03-2013	20-03-2013	mot est dans dictionnaire	Antoine de Roquemaurel
288	Module #278: Dictionnaire	Tache	16-03-2013	19-03-2013	Obtenir probabilité d'apparition d'une lettre	Fabrice Valleix
279		Module	21-03-2013	10-04-2013	Resolveur	Antoine de Roquemaurel
299	Module #279: Resolveur	Tache	21-03-2013	05-04-2013	Résoudre une grille	Antoine de Roquemaurel
300	Module #279: Resolveur	Tache	05-04-2013	08-04-2013	mot est dans la grille	Fabrice Valleix
301	Module #279: Resolveur	Tache	06-04-2013	10-04-2013	Nombre de mots commençant par une séquence de lettre	Fabrice Valleix
280		Module	10-04-2013	16-04-2013	Jeu	Fabrice Valleix
290	Module #280: Jeu	Tache	10-04-2013	16-04-2013	lancer timer	Fabrice Valleix
297	Module #280: Jeu	Tache	10-04-2013	16-04-2013	Valider un mot	Fabrice Valleix
298	Module #280: Jeu	Tache	10-04-2013	14-04-2013	Retourner le nombre de points obtenus	Antoine de Roquemaurel
281		Module	16-04-2013	26-04-2013	InterfaceTexte	Fabrice Valleix
293	Module #281: InterfaceTexte	Tache	23-04-2013	26-04-2013	Permettre la saisie d'un mot	Fabrice Valleix
294	Module #281: InterfaceTexte	Tache	23-04-2013	26-04-2013	Afficher nombre de points	Fabrice Valleix
303	Module #281: InterfaceTexte	Tache	16-04-2013	23-04-2013	Interface générale	Fabrice Valleix
282		Module	16-04-2013	30-04-2013	InterfaceGraphique	Antoine de Roquemaurel
291	Module #282: InterfaceGraphique	Tache	23-04-2013	27-04-2013	Afficher solution	Antoine de Roquemaurel
292	Module #282: InterfaceGraphique	Tache	23-04-2013	26-04-2013	Afficher aide	Antoine de Roquemaurel
295	Module #282: InterfaceGraphique	Tache	23-04-2013	26-04-2013	Afficher nombre de points	Antoine de Roquemaurel
296	Module #282: InterfaceGraphique	Tache	23-04-2013	30-04-2013	Sélectionner lettre pour saisie	Antoine de Roquemaurel
302	Module #282: InterfaceGraphique	Tache	16-04-2013	23-04-2013	Interface générale	Antoine de Roquemaurel
283		Technical Story	19-04-2013	30-04-2013	Refactoring	Antoine de Roquemaurel

Figure 2 – Liste des tâches et leur répartition

5 Calendrier de réalisation des tâches

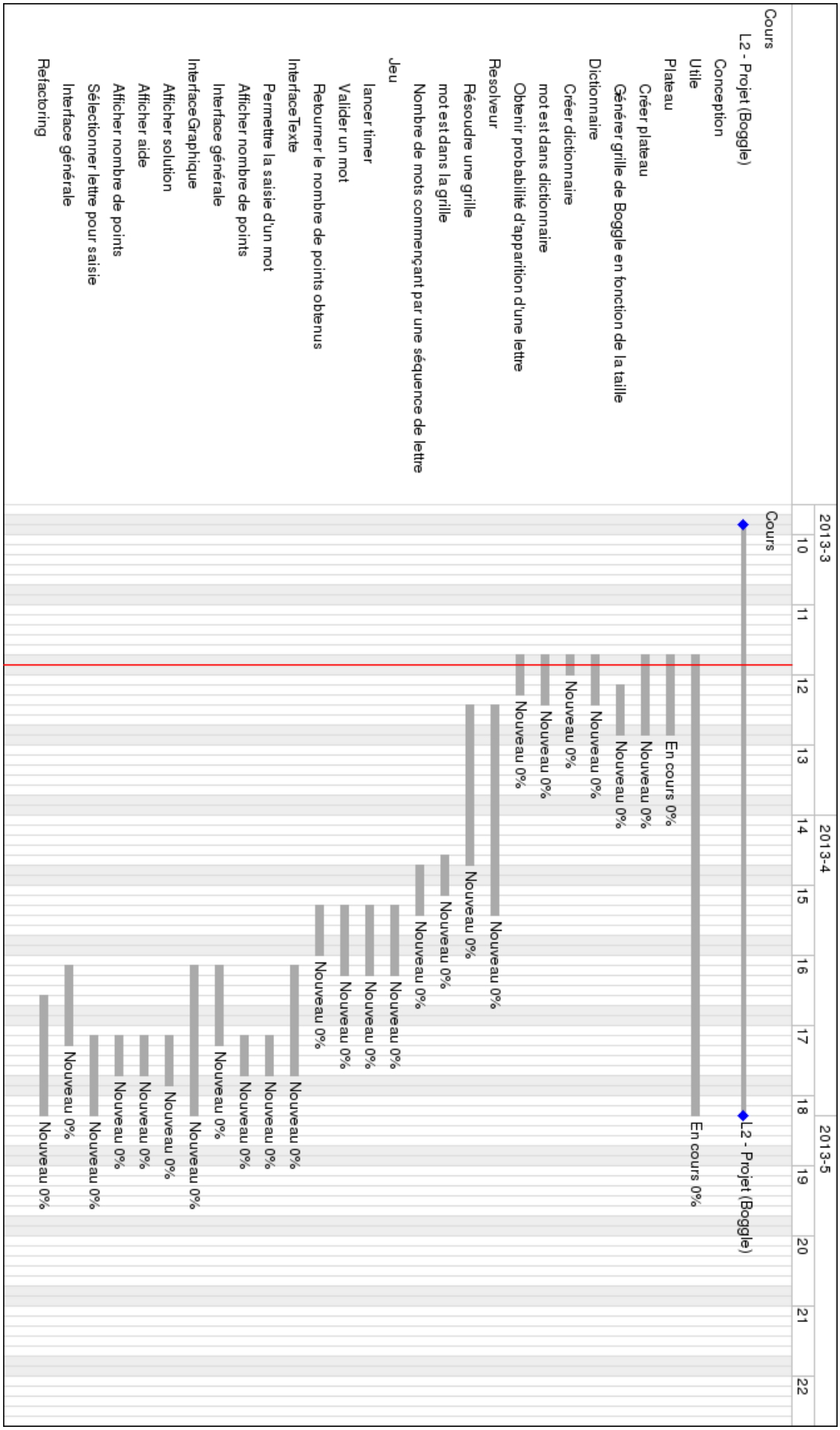

Figure 3 – Diagramme de Gantt

6 Plan de tests

Chaque fonction de chaque module sera testée à l'aide de tests unitaire, s'appuyant sur la bibliothèque `cunit`. Chacun des tests unitaire au pour but de tester un et un seul cas, mais l'ensemble des tests unitaires d'un module devront avoir passer en revue tous les cas possibles d'appel d'une fonction : Que ce soit un cas nominal, ou un cas d'erreur, ainsi chacune des lignes de code auront été testé, si ce n'est pas le cas, du code mort aura été détecté.

Une fois que chaque fonction aura passé les tests unitaires, nous allons tester les modules séparément afin de vérifier que toutes les fonctions fonctionnent bien entre elle, ces tests seront différents en fonctions des modules.

6.1 Plateau

Un test fonctionnel aura lieu afin de tester le module, pour cela, nous appellerons la fonction de génération de grille et vérifierons qu'elle a bien génére en fonction des tailles que nous lui donnons.

Ce test s'effectuera tout d'abord dans son fonctionnement nominal, pour toutes les tailles que l'utilisateur est susceptible de rentrer (de 2 à 15, cette taille étant fixée dans les spécifications), ensuite un test s'effectuera sur des valeurs alternatives : nombre négatif, flottants, supérieur à 15 etc. . .et nous vérifierons que les erreurs sont bien gérés.

6.2 Dictionnaire

Afin de tester le dictionnaire, les tests unitaires suffiront, en effet, il faudrait vérifier que la fonction qui dit si un mot est présent dans le dictionnaire ou bon est correcte.

6.3 Résolveur

Afin de tester le Résolveur, nous essayerons avec des grilles prédéfinis de tailles variables, et vérifierons que le résolveur retourne bien tous les mots présent dans ces grilles, ceci sans erreurs. Le résolveur dépendant du dictionnaire, nous devons avoir testé préalablement le dictionnaire afin de pouvoir tester ce module.

Afin de vérifier tous les cas possibles, nous utiliserons le plus de grilles possible, tout d'abord des grilles classiques de taille 4×4, avec le plus de mots possibles présent dans la grille. Ensuite nous testerons le résolveur sur des grilles de taille 15×15 afin de vérifier qu'il n'est pas trop lent par rapport à ce que nous avions énoncé dans les spécifications. Également, nous lui ferons passer une grille ou aucun mot n'est possible dans la grille afin de vérifier que dans ce cas là, le module fonctionne correctement.

6.4 Jeu

Afin de tester ce module, une interface est indispensable, ainsi nous nous en tiendrons aux tests unitaires, afin de vérifier que toutes les fonctions du module fonctionnent, ensuite ce module sera testé via l'intermédiaire de l'interface en mode texte.

6.5 InterfaceGraphique et InterfaceTexte

Ces deux modules sont les interfaces du jeu, il est difficile de faire des tests automatisé pour les interfaces, de plus ces deux modules sont intimement liés au contrôleur, `Jeu`, ainsi nous ne pourrons pas tester les interfaces séparément des autres modules, ces deux interfaces seront donc testé à l'aide d'un test fonctionnel lors d'une partie de Boggle. Ce test s'effectuera avec une grille prédéfinie.

A Annexes

A.1 Table des figures

A.2 Liste des tableaux