
JABBOID
Coscujuela
Jalabert
Ladegaillerie
Rizzo
Plan d'assurance
qualite

Tableau des Validations

Nom	Statut	Date	Signature
Tony Coscujuela	Validé	29/08/2014	TC

Tableau des Révisions

Nom	Date de lecture	Version	Signature
Rizzo Damien	13/08/2014	0.6	
Jalabert Steven	13/08/2014	0.7	
Tony Coscujuela	15/08/2014	0.8	

Table des matières

Plan d'Assurance Qualité

I- Documents applicables

Les différents documents auxquels s'applique ce plan d'assurance qualité sont les suivants :

- Le manuel d'installation du logiciel,
- Le manuel d'utilisation du logiciel,
- Le cahier des charges,
- Le plan d'assurance qualité,
- Le document de spécification.

II- Objet du document

Ce document représente la version préliminaire du Plan d'Assurance Qualité. Il sera revu et complété au démarrage de la conception. Le présent document s'applique aux prestations réalisées par notre groupe dans le cadre du projet Jabboid.

Le Plan d'Assurance Qualité (PAQ) traduit notre engagement quant à la politique d'assurance qualité applicable à notre prestation dans ce projet. Il s'applique depuis le démarrage du projet jusqu'à sa clôture. Ce document est consultable par chaque membre de l'équipe en tant que document à mettre en application. Il est utilisé pour apporter la visibilité sur le travail mis en œuvre pour réaliser les prestations et sur les méthodes appliquées pour obtenir le niveau de qualité répondant aux exigences fixées. Le Plan d'assurance qualité décrit l'organisation et les processus mis en place afin de répondre au cahier des charges.

III- Objectifs qualité

Le PAQ décrit :

- La présentation générale et le processus d'élaboration et de mise à jour du Plan d'Assurance Qualité,
- Les documents applicables,
- Le glossaire des principaux termes ainsi que les abréviations utilisées,
- La méthodologie applicable pour la réalisation des prestations,
- Le processus de gestion de la qualité.

La mesure de la qualité tout au long du projet sera concentrée sur 4 exigences :

Fonctionnement, Ergonomie, Vérification et Fiabilité.

	Engagements qualité	Actions qualité	Résultats attendus
FONCTIONNEMENT	Garantir le fonctionnement avec optimisations des ressources.	Vérifier le fonctionnement sur n'importe quelle machine du client correspondant au standard défini par le client.	Le logiciel est capable de fonctionner sur la machine la moins puissante du client.
ERGONOMIE	Garantir la facilité d'utilisation du logiciel et de l'IHM par les utilisateurs finaux.	Faire tester par une dizaine de personnes le logiciel avant la livraison.	± 80% de satisfaction sur : • Ergonomie • Facilité d'utilisation • Facilité d'apprentissage
VÉRIFICATION	Garantir le bon fonctionnement de chaque module.	Tester chaque étape de développement correspondant à chaque fonction et module.	Un test par fonction/module avec fichier de résultat associé.
FIABILITÉ	Garantir la fiabilité du système.	Livrer une version totalement fonctionnelle au client.	Aucune anomalie bloquante recensée dans les versions diffusées auprès des utilisateurs.

IV- Modifications et dérogations au PAQL

En cas de problème de qualité, n'ayant pas été prévus dans le cas du PAQL, nous nous réservons la possibilité de mettre ce dernier à jour en changeant de version et de le faire réviser par les différents membres du projet.

En cas ou une dérogation au PAQL serait inévitable, nous joignons ici, un tableau de dérogation :

Nom	Date	Description	Permanente ou Temporaire	Signatures
JALABERT Steven	14/04/2014	Ajout de mesures de qualité/ plan GED / infos docs	Permanente	SJ

V- Intervenants, rôles et responsabilités

COSCUJUELA Tony : Responsable Qualité

JALABERT Steven : Chef de Projet

LADEGAILLERIE Stéphane : Responsable Documentation

RIZZO Damien : Responsable Développement

VI- Contraintes qualité

Voici les contraintes qualité concernant l'aspect programmation du projet :

— Commentaires // pour une ligne, /*… */ pour plusieurs lignes

— Commentaires, nom de variables, méthodes en anglais

— Constantes en majuscules
Ex : MYCONST

— Les noms des variables/méthodes privées seront en minuscules, tandis que la première lettre d'une variable/méthode publique sera en majuscule. Et les variables composées de plusieurs mots auront la première lettre de chaque mot en majuscule.
Ex : Private int example, Public int DoExample(), plusGrandNombrePremier.

— Le nom des variables doit contenir le type de celle-ci.
Ex: int iCpt, string strFoo, float fMean.

— Les variables, méthodes, etc., doivent avoir le nom le plus descriptif possible. Les mots doivent être écrits en entier plutôt qu'avec des abréviations, sauf dans le cas de certains mots clés très courants comme Nbr, Max ou Min
Ex : Utiliser Counter au lieu de i, PositionX au lieu de x, et surtout pas de variables Temp, Toto, FooBar...

— Les accolades doivent être placées au début du bloc comme ci-dessous :

```
If (someCondition)
{
        DoSomething() ;
}
```

— Les indentations doivent être des caractères de tabulation et sont obligatoires.

— Les lignes doivent avoir une largeur maximale de 120 caractères de façon à faciliter la lecture et l'impression du code.

— Les primitives : if, else, while, for et do doivent obligatoirement être suivies d'un bloc entre accolades, même s'ils ne contiennent qu'une instruction ou qu'ils sont vides.

— On doit utiliser la méthode permissivité minimale. Ne devront éviter les variables globales, les données membres publiques, etc. Les variables globales doivent être utilisées le moins possible et, le cas échéant, doivent être encapsulées dans un seul objet global.

— Les fichiers et les méthodes devront inclure des entêtes tels que décrits ci-dessous. (Compatible JAVADOC) :

ENTÊTE DE MÉTHODES:

```
/**
 * Validates a chess move.
 *
 * Use {@link #doMove(int, int, int, int)} to move a piece.
 *
 * @param theFromFile file from which a piece is being moved
 * @param theFromRank rank from which a piece is being moved
 * @param theToFile   file to which a piece is being moved
 * @param theToRank   rank to which a piece is being moved
 * @return            true if the move is valid, otherwise false
 */
boolean isValidMove(int theFromFile, int theFromRank, int theToFile, int
theToRank)
{
    ...
}
```

ENTÊTE DE CLASSES:

```
/**
 * @author      Firstname Lastname <address @ example.com>
 * @author      Firstname Lastname <address @ example.com>
 * @version     1.6                 (current version number of program)
 * @since       2010-03-31      (the version of the package this class
 * was first added to).
 */
public class Test
{
    // class body
}
```

VII- Terminologie

Les termes suivants seront les termes spécifiques utilisés dans tous les documents produits lors de ce projet :

CdC : Cahier des Charges

CRR : Compte Rendu de Réunion

DC : Dossier de Conception

DS : Dossier de Spécifications

PAQL : Plan d'Assurance Qualité Logicielle

PAQ : Plan d'Assurance Qualité

PDL : Plan de Développement Logiciel

CV : Cycle de Vie

CP : Chef de Projet

RQ : Responsable Qualité

BDD : Base De Données

DB : Database

SVN : Subversion

XMPP : Extensible Messaging and Presence Protocol

SDK : Software Development Kit

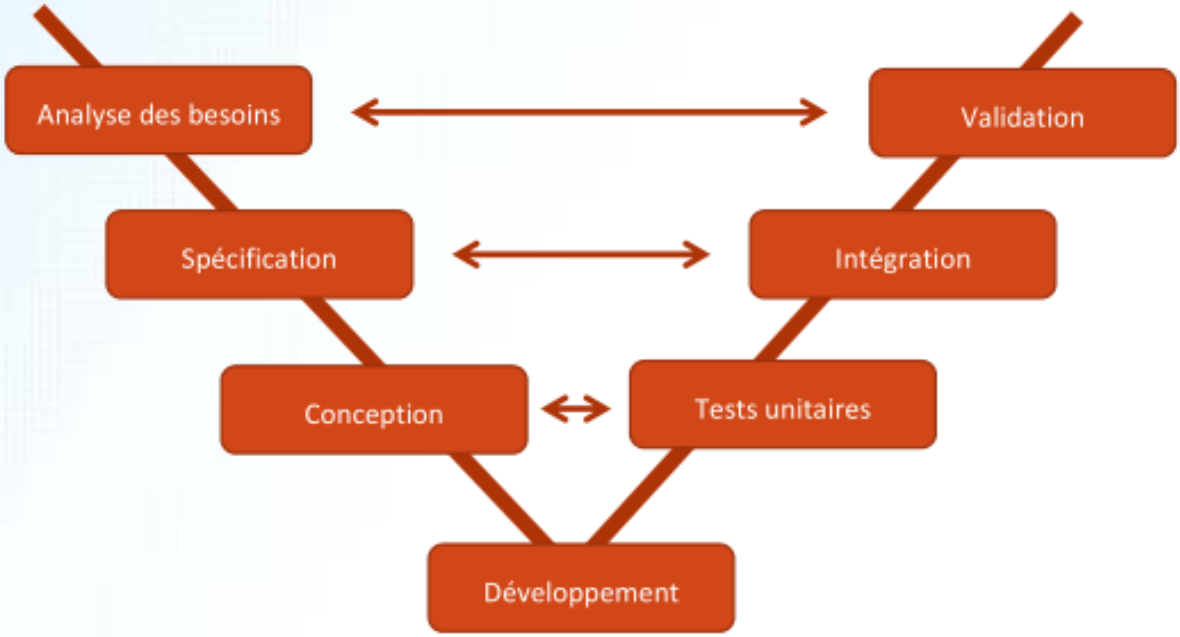
VM : Virtual Machine

VIII- Cycle de développement

Le cycle de développement mis en place dans le cadre du projet est un cycle de développement en V.

Il permet, en cas d'anomalie, de limiter un retour aux étapes précédentes.

Les phases de la partie montante doivent renvoyer de l'information sur les phases en vis-à-vis lorsque des défauts sont détectés, afin d'améliorer le logiciel.



Analyse des besoins :

Cette étape permet de prendre en compte les besoins du client ainsi que son cahier des charges, elle décrit l'existant, les attentes et les exigences générales exprimés par le client.
Elle permet aussi de définir un planning prévisionnel, de mettre en place un environnement de programmation ainsi qu'une démarche qualité propre au sujet.

Spécifications :

Cette étape établit la spécification technique des besoins, elle définit un ensemble de fonctionnalités ainsi qu'un aperçu de l'interface homme-machine du programme final. Ces informations sont regroupées au sein du dossier de spécification.

Conception :

Cette étape met en place l'organisation des différents éléments du système défini dans le dossier de spécification et des relations entre ces éléments.
Dans le cadre de ce projet, la structure du système informatique sera représentée sous forme de graphiques UML tels que les diagrammes de classe, les diagrammes de séquence et les cas d'utilisation.

Développement :

C'est l'ensemble des activités liées à la création du logiciel. Cette étape correspond à la rédaction du code source du logiciel en rapport avec le dossier de spécification et le dossier de conception UML.

Tests unitaires :

Cette étape, en regard avec l'étape de conception permet de s'assurer du fonctionnement correct d'une partie du logiciel, elle permet de confronter une réalisation à sa spécification, et de s'assurer qu'elle fonctionne correctement en toute circonstance.
En cas d'anomalie ou de différence avec les résultats attendus, le programme sera modifié pour coller au mieux aux spécifications initiales.

Intégration :

Une fois que les développeurs ont chacun validé leurs développements lors des tests unitaires, cette étape va regrouper leurs modifications dans le cadre d'une livraison. Les tests d'intégration ont pour but de valider le fait que toutes les parties développées indépendamment fonctionnent bien ensemble de façon cohérente.

Validation :

Cette étape permet de vérifier si toutes les exigences client décrites dans le document de spécification sont respectées. Elle permet la livraison du logiciel final.

IX- Documentation produite

À l'issue de ce projet, plusieurs documents auront été produits.
Certains devront être livrés avec le produit fini, certains seront disponibles si le client les souhaite et certains ne serviront qu'à l'équipe de développement.
Le tableau suivant précise pour chaque document son statut.

Document	Statut
Cahier des charges	Livrable
Plan d'assurance qualité logicielle	Livrable
Dossier de spécifications	Livrable
Dossier de conception	Consultable
Plan de développement logiciel	Privé
Plans de tests	Livrable
Jeu de tests	Livrable
Manuel utilisateur	Livrable
Manuel d'installation	Livrable

- Description des différents statuts :
  - **Livrable** : dois être fourni au client
  - **Consultable** : peut être consulté par le client
  - **Privé** : destiné à l'équipe du projet uniquement

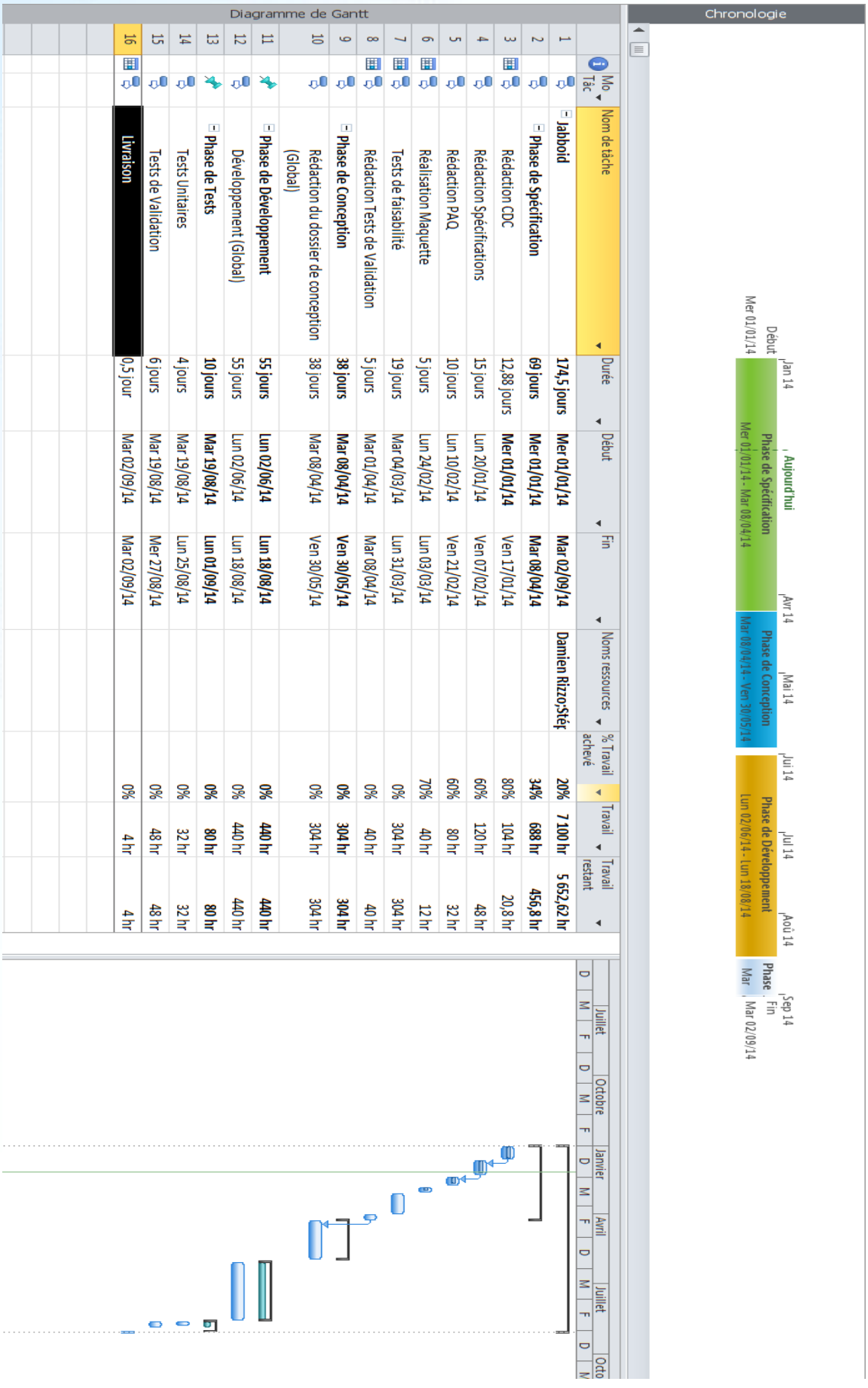
La lecture croisée sera effectuée au minimum par 2 membres de l'équipe de développement (l'un rédige, l'autre relit).

L'auteur d'un document assure la réalisation des corrections proposées par le relecteur et gère le changement des numéros de versions.

L'enchaînement des tâches est le suivant :

- Création du document
- Diffusion aux relecteurs potentiels avec la fiche de relecture associée
- Intégration des modifications par l'auteur et modification éventuelle du numéro de version en fonction des modifications effectuées
- Validation finale, après plusieurs cycles de diffusion/correction, effectuée par le responsable qualité Tony COSCUJUELA

X- Planning


Chronologie
Début
Mer 01/01/14
Jan 14
Aujourd'hui
Avr 14
Mai 14
Jui 14
Jul 14
Aoû 14
Sep 14
Phase de Spécification
Mer 01/01/14 - Mar 08/04/14
Phase de Conception
Mar 08/04/14 - Ven 30/05/14
Phase de Développement
Lun 02/06/14 - Lun 18/08/14
Phase
Mar
Fin
Mar 02/09/14
Diagramme de Gantt
Mo Tâc
Nom de tâche
Durée
Début
Fin
Noms ressources
% Travail achevé
Travail
Travail restant
1 Jabboid 174,5 jours Mer 01/01/14 Mar 02/09/14 20% 7 100 hr 5 652,62 hr
2 Phase de Spécification 69 jours Mer 01/01/14 Mar 08/04/14 Damien Rizzo,Stép 34% 688 hr 456,8 hr
3 Rédaction CDC 12,88 jours Mer 01/01/14 Ven 17/01/14 80% 104 hr 20,8 hr
4 Rédaction Spécifications 15 jours Lun 20/01/14 Ven 07/02/14 60% 120 hr 48 hr
5 Rédaction PAQ 10 jours Lun 10/02/14 Ven 21/02/14 60% 80 hr 32 hr
6 Réalisation Maquette 5 jours Lun 24/02/14 Lun 03/03/14 70% 40 hr 12 hr
7 Tests de faisabilité 19 jours Mar 04/03/14 Lun 31/03/14 0% 304 hr 304 hr
8 Rédaction Tests de Validation 5 jours Mar 01/04/14 Mar 08/04/14 0% 40 hr 40 hr
9 Phase de Conception 38 jours Mar 08/04/14 Ven 30/05/14 0% 304 hr 304 hr
10 Rédaction du dossier de conception (Global) 38 jours Mar 08/04/14 Ven 30/05/14 0% 304 hr 304 hr
11 Phase de Développement 55 jours Lun 02/06/14 Lun 18/08/14 0% 440 hr 440 hr
12 Développement (Global) 55 jours Lun 02/06/14 Lun 18/08/14 0% 440 hr 440 hr
13 Phase de Tests 10 jours Mar 19/08/14 Lun 01/09/14 0% 80 hr 80 hr
14 Tests Unitaires 4 jours Mar 19/08/14 Lun 25/08/14 0% 32 hr 32 hr
15 Tests de Validation 6 jours Mar 19/08/14 Mer 27/08/14 0% 48 hr 48 hr
16 Livraison 0,5 jour Mar 02/09/14 Mar 02/09/14 0% 4 hr 4 hr
Juillet
Octobre
Janvier
Avril
Juillet

XI- Gestion de configuration

SVN / Google Code Issue Tracker / Google Code Wiki / Squash Test Manager

XII- Gestion de la documentation

Pour gérer le stockage de la documentation que nous allons créer durant ce projet, nous utiliserons le Google Drive. Ce Google Drive contient, un dossier documentation, organisé de la manière suivante :

Documents : dois contenir les diverses documentations, qui auront chacun leurs propres dossiers (spécifications, conception, codage…) avec pour chacun deux sous-dossiers, releases et ressources avec le dossier releases qui contient tout ce qui a été envoyé au client, avec son numéro de version.

Ressources : contiendra lui toutes les ressources nécessaires au document comme les images, les versions bêta ou des fichiers temporaires. Peut contenir divers documents (docs api, SDK…)

XIII- Organisation des données

Pour gérer les données, nous utiliserons aussi le Google Drive. Ce Google Drive sera organisé de la manière suivante :

4 dossiers principaux : Référence, Démo, GED et Travail qui auront chacun une arborescence propre.

Démo :

Contiendra la démo à faire passer le jour de l'oral.

Référence :

Contiendront les divers documents (spécifications, conception, codage…) dans leur état final.

GED :

Contiendra des ressources nécessaires externes au projet, comme des applications et leurs licences, environnement de développement, environnement de travail, environnement de tests, divers tutoriels/aides/documents provenant d'internet.

Travail :

Développement :

Dois contenir tout ce que la personne a réalisé (codage, documents, manuels, etc. : TOUT) de façon organisée. Organisée de manière à avoir un dossier par membre.

XIV- Organisation du travail en équipe

L'organisation du travail en équipe passe par l'utilisation décrit plus haut comme le Google Drive et le SVN.

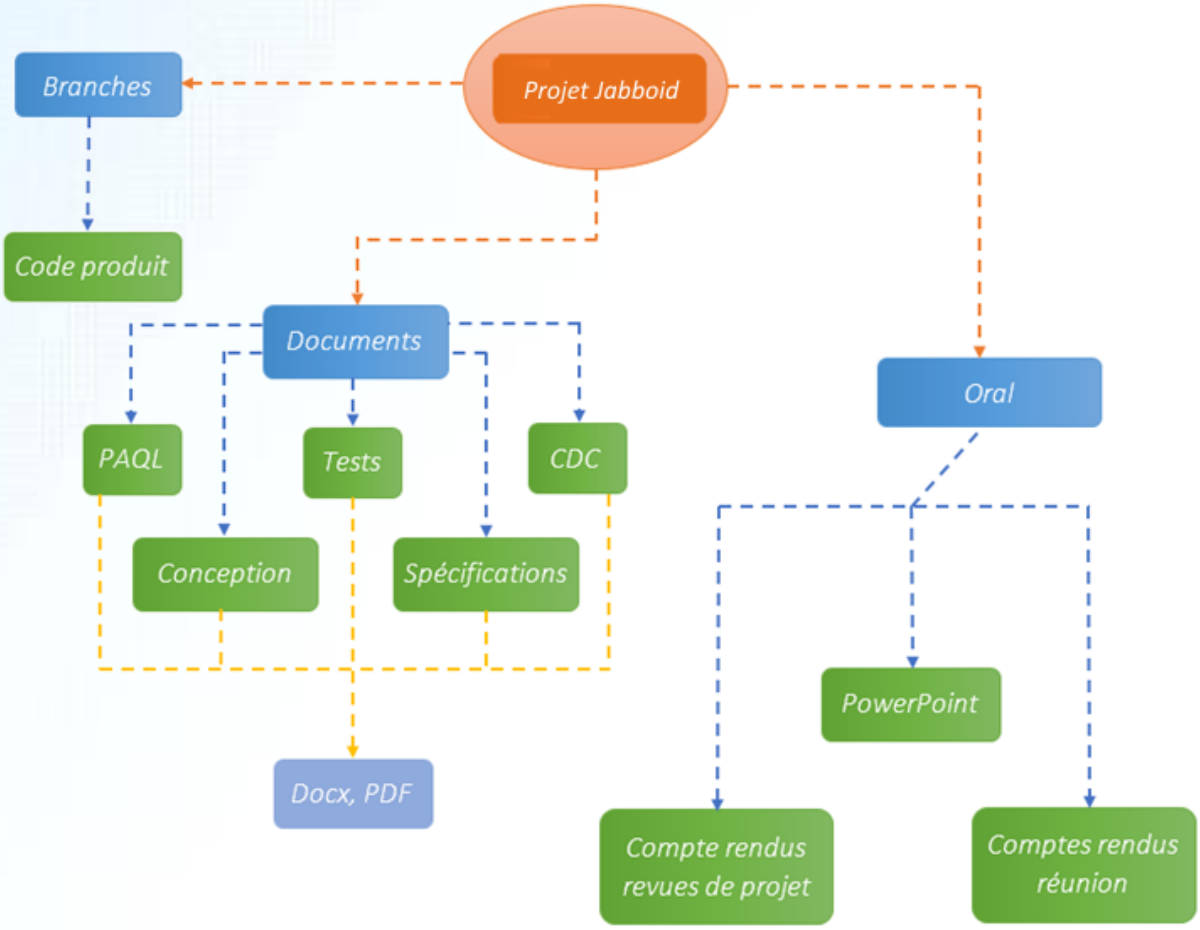
Mais aussi par des réunions régulières, notamment dans le cadre de nos créneaux gestion de projet, mais aussi durant des réunions plus ou moins formelles grâce à l'outil Skype, qui nous permettent de rester en contact.

Il sera également mis à disposition de chaque membre un accès au serveur Jabber et une copie de ce serveur sur une machine virtuelle, pour y effectuer des tests sans risques pour le serveur « en production ».

XV- Sauvegarde et Archivage

Pour la sauvegarde des différents documents, comme nous l'avons expliqué plus haut, nous utiliserons Google Drive.

Par contre pour l'aspect programmation du projet nous utiliserons un SVN : http://code.google.com/p/jabboid/ ou chaque membre du projet dispose d'un accès utilisateur.



XVI- Annexes

1) Modèle de rapport de réunion

Rapport de Réunion

[Meeting Title]	
Date du jour	[Meeting Time]
Organisée par	
Type	
Animateur	
Scribe	
Participants	

[Agenda Topic]	
Discussion	
Conclusions	

Actions	Responsables	Deadline

Jabboid – Le lundi 5 mai 2014

2) Modèle de page de garde