Université Paul Sabatier

Master Intelligence Artificielle et
Reconnaissance des Formes
Master Robotique : Décision et Commande

Plan de Développement Qualité

Navigation Autonome de Robot Mobile

Auteur :
Thibaut Aghnatios
Marine Bouchet
Bruno Dato
Tristan Klempla
Thibault Lagoute

Tuteur :
Frédéric Lerasle
Michaël Lauer
Michel Taix

LAAS-CNRS

7 novembre 2016

Suivi du document

Nom du document	Version Majeure	Version Majeure	Date de création	Dernière version
PDDQ	A	0	27/10/2016	09/11/2016

Auteurs du document

Rédaction	Intégration	Relecture	Validation Interne
Equipe	Marine Bouchet	Bruno Dato Thibaut Aghnatios	Bruno Dato Thibaut Aghnatios

Validation du document

Validation	Nom	Date	Visa

Liste de diffusion

Le plan de développement qualité est diffusé à l'ensemble des clients et des intervenants externes aux projets.

Historiques de révision

Version	Modification apportée	Auteur	Date
A.0	Création et intégration du document	Marine Bouchet	07/11/2016
A.0	Ajout de la section **??**	Bruno Dato	09/11/2016

Table des matières

1 Introduction

La structure et le contenu de ce Plan de Développement Qualité de Projet ont été élaborés dans le cadre du projet « Navigation autonome de robots mobiles » proposé aux étudiants de deuxième année du Master « Intelligence Artificielle et Reconnaissance des Formes » et « Robotique : Décision et Commande » de l'Université Paul Sabatier à Toulouse.

Il a pour objectif la définition et la description des différentes dispositions à mettre en œuvre pour un développement optimal du projet afin d'en assurer la qualité et d'atteindre les résultats attendus. Plus précisément, sont déterminés, d'un commun accord :

— l'organisation globale du projet
— le plan de gestion et de développement du projet
— les droits et les devoirs de chaque partie prenante
— la répartition des responsabilités entre les organismes dans la structure
— les plans de développement et de gestion du projet
— les outils qui seront adoptés

Après acceptation par le client, ce plan deviendra le document contractuel applicable en matière de gestion et d'assurance qualité entre le Titulaire et le Client durant la durée totale du projet. La responsable qualité s'assurera qu'il est effectivement appliqué. Il ne pourra subir aucune modification sans accord préalable du Client. En cas de divergence entre les exigences et le plan qualité, les exigences s'appliqueront en priorité.

2 Présentation du projet

2.1 Contexte

2.1.1 Master IARF et RODECO

Le master « Intelligence Artificielle et Reconnaissance des Formes » (IARF) a comme objectif de former des professionnels de haut niveau capables de concevoir des solutions à des problèmes complexes utilisant des méthodes avancées de représentation et de traitement de l'information, faisant appel à des techniques d'intelligence artificielle (IA) et de reconnaissance des formes (RF) et d'apprentissage automatique, appliqués notamment au traitement d'images et à la robotique.

Le master « Robotique : décision et commande » (RODECO) a pour vocation de promouvoir des connaissances dans le domaine de l'automatique par des enseignements avancés autour de la robotique, de l'informatique et de la commande des systèmes. Ces compétences permettent d'aborder des problématiques très actuelles comme la robotique industrielle haute performance où les aspects commande sont fondamentaux et la robotique de service où la décision et la perception tiennent une place essentielle. Suivant ce raisonnement, deux blocs de spécialisation sont proposés en M2 :

Robotique et décision qui propose un renforcement des aspects « informatique » (intelligence artificielle, reconnaissance des formes, dialogue homme/machine), vision par ordinateur et robotique mobile. Cette spécialisation donne les compétences nécessaires pour appréhender le domaine de la robotique de service ;

Robotique et commande qui se focalise sur le développement et l'implantation de commandes avancées pour la robotique. Cette spécialisation donne donc les compétences nécessaires pour élaborer des solutions évoluées de contrôle/commande pour la réalisation de tâches robotique haute performance.

Au cours de cette deuxième année, les étudiants acquièrent une double compétence en Automatique et Informatique et les capacités requises pour modéliser, analyser, concevoir et réaliser des systèmes automatiques complexes, autonomes et/ou embarqués où sont impliqués la perception (capteurs), l'analyse (traitement de signal, audio, image, vidéo), le raisonnement et la décision (incertitude, reconnaissance de formes, contraintes) et de l'action (commande, robotique).

2.1.2 Contexte du projet

Le projet de Master 2 permet de mettre en commun et en pratique les connaissances acquises dans ces trois parcours dans un but commun. En l'occurrence, sur le projet « Navigation autonome de robots mobiles », Tristan et Marine font parti du parcours IARF, Thibaut et Bruno sont issus de la spécialité Décision de RODECO et enfin, Thibault, de la spécialité Commande.

Il se déroule tout le long de l'année par tranche, nommées W, de 3, 4 et 4 semaines, en alternance avec les blocs de cours B.

B1	W1	B2	W2	B3	W3
5 sem.	3 sem. 17/10 - 14/11	5 sem.	4 sem. 02/01 - 27/01	5 sem.	4 sem. 01/03 - ? ?/03

2.1.3 Entrées

☑ Cahier des charges
☑ Documentations

2.1.4 Sorties

L'objectif du projet est de répondre à un cahier des charges divisé en 4 étapes incrémentales. Le projet étant emmené à être réutilisé par la suite pour d'autres étudiants, la qualité du produit est primordiale.

☐ Un code propre, fonctionnel et surtout facilement réutilisable
☐ Documentations
☐ Robot opérationnel qui effectue les tâches demandées
☐ Manuel Utilisateur

2.1.5 Limites

— Incrémentation des solutions : difficulté d'anticiper les solutions et problèmes des étapes suivantes
— Cours de vision 2D et de navigations en B2 et vision 3D en B3

2.2 Parties-prenantes

2.2.1 MOE

Thibaut Aghnatios
thibaut.aghnatios@laposte.net
Spécialité Décision

Tristan Klempka
klempka.tristan@gmail.com
Spécialité IARF

Marine Bouchet
bouchetmarinee@gmail.com
Spécialité IARF

Thibault Lagoute
lagoute.31@gmail.com
Spécialité Commande

Bruno Dato
bruno.dato.meneses@gmail.com
Spécialité Décision

2.2.2 MOA

Frédéric Lerasle
lerasle@laas.fr
Equipe RAP, LAAS - CNRS

Michel Taix
taix@laas.fr
Equipe GEPETO, LAAS - CNRS

Michel Lauer
michael.lauer@laas.fr
Equipe TSF, LAAS - CNRS

2.2.3 Intervenants externes

Julien Vanderstraeten
julien.vanderstraeten.ups@gmail.com
Coach

Cyril Briand
briand@laas.fr
Coach

2.3 Contraintes

La réalisation du projet est régie par certaines contraintes citées ci-dessous :

- Robot TurtleBot
- Planning du Master et ses jalons
- Connaissances partielles selon les périodes de projet

3 Organisation du projet

3.1 Cycle de développement

Le développement du projet se fera selon Scrum, une méthode Agile dédiée à la gestion de projet. Cette méthode, basée sur les stratégies itératives et incrémentales, permettra de produire, à la fin de chaque « sprint », un résultat achevé et validé, pour avoir une version fonctionnelle à tout moment. Cette démarche correspond bien au cahier des charges où les étapes sont à la fois itératives et incrémentales.

Pour chaque sprint, la période initiale permettra de faire le point sur les éléments techniques du départ du projet et d'établir une liste des points à préciser ou à compléter, à savoir les charges de travail et le calendrier associé. Une nouvelle version du Plan de Qualité et du Plan de Développement adaptée à l'étape en cours sera donc produite. Chaque sprint donne lieu à une phase de codage et une phase de tests.

Scrum se différencie des autres méthodes de développement par ses avantages qui font de ce procédé une réponse à certains problèmes fréquemment rencontrés dans le développement logiciel. Pour éviter l'effet tunnel, la communication restera permanente entre les membres de l'équipe mais aussi entre l'équipe et le client, permettant une meilleure coopération à l'intérieur de l'équipe Scrum.

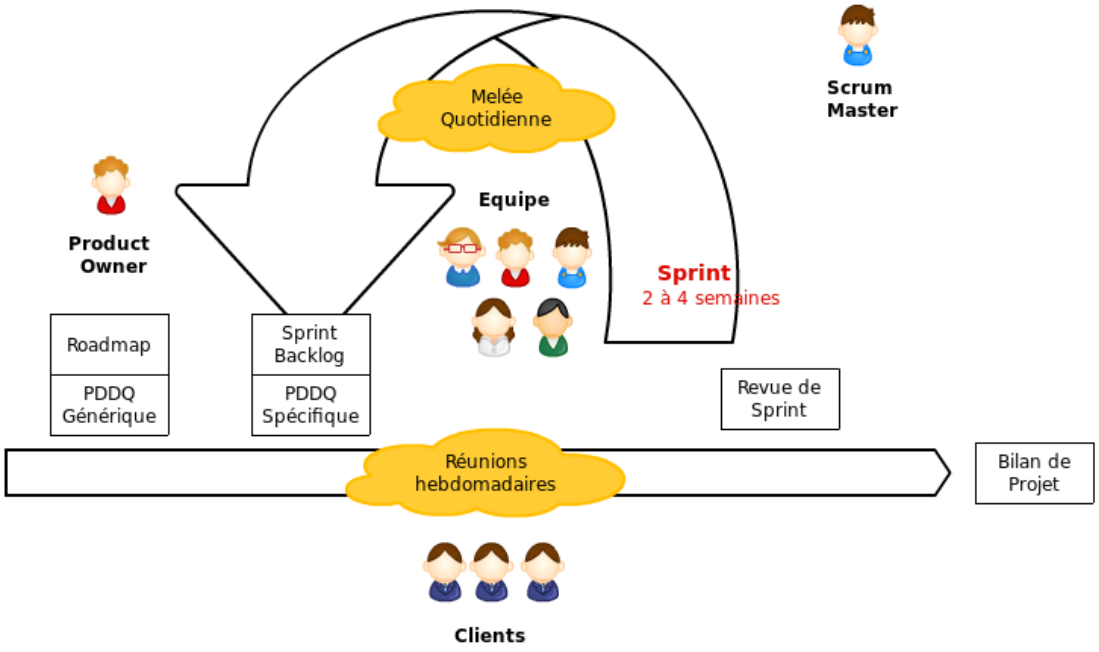


FIGURE 1 – Schéma Méthode Scrum

3.2 Sprints

Le projet est divisé en 4 étapes définies par le client :

Etape 1 Réaliser une tâche robotique permettant au robot d'atteindre une balle de couleur (détecter une balle de couleur, localisation dans le repère courant du robot, aller à la balle).

Etape 2 Réaliser une tâche de navigation dans un environnement supposé connu et sans obstacles pour atteindre un amer final en se déplaçant d'amer en amer (QR code : amer 2D) afin de ne pas se perdre et en optimisant un critère (distance, temps, commande,....).

Etape 3 Réaliser une tâche de navigation dans un environnement supposé connu et sans obstacles inconnus pour atteindre un amer final en se déplaçant d'amer en amer (3D) afin de ne pas se perdre et en optimisant un critère (distance, temps, commande,....).

Etape 4 Réaliser une tâche de balayage grâce à une modélisation par carte laser.

La première étape a pour objectif de prendre en main les différentes ressources à notre disposition. Il n'y a donc pas de réelle étape de spécification et de conception contrairement aux étapes suivantes. Celles-ci apparaitront dans les futures version de ce document.

A la fin et entre chaque étape, une étude des limites et des problématiques de la prochaine étape est demandée.

4 Arbre produit

Un arbre produit, ou *Product Breakdown Structure* donne une liste exhaustive des différents livrables du projet, de manière hiérarchisée. Il permet d'avoir une vue claire des différentes fonctionnalités à développer, de décrire l'architecture logicielle et matérielle du produit à livrer, et d'en prévoir les coûts. Dans le cadre de notre projet de navigation, l'aspect budgétaire ne sera pas abordé puisqu'il s'agit d'un projet dans le cadre de la formation. La figure suivante présente l'arbre produit du projet de navigation. Les parties encadrées correspondent aux nœuds de l'arbre, c'est à dire les groupes de fonctions à développer, tandis que les parties non encadrées correspondent aux feuilles, autrement dit aux fonctions elles-mêmes. L'arbre produit de notre projet de navigation est décomposé en cinq parties :

Simulateur ROS : concerne les différents éléments à utiliser sous ce simulateur, à savoir la modélisation de l'environnement, les différents obstacles, le calcul des trajectoires de notre robot et l'intégration des blocs au cours des différentes étapes.

Perception : décrit l'ensemble des fonctionnalités permettant l'identification à partir d'une image. On distingue ainsi 2 groupes : la perception de l'environnement et la détection des obstacles. La détection des obstacles tient compte des différentes méthodes utilisées au cours de ce projet : le nuage de points, l'étude laser, les différentes méthodes de segmentation, etc. . .

Localisation : décrit les fonctions nécessaires à la navigation autonome du robot. Il devra permettre au robot de calculer sa position et déterminer son orientation grâce à des repères.

Décision : décrit les fonctions nécessaires aux différentes décisions que le robot doit prendre comme pour la génération de la trajectoire (en choisissant la trajectoire la plus courte et la plus rapide à prendre selon l'orientation du robot) et les décisions pour les différents évènements possibles.

Action : Concerne toutes les fonctions correspondant au domaine de la commande robotisée permettant le respect des consignes (atteindre une balle de couleur, un amer, . . .), la réalisation des mouvements, et les contraintes cinématiques. On parlera aussi de l'odométrie.

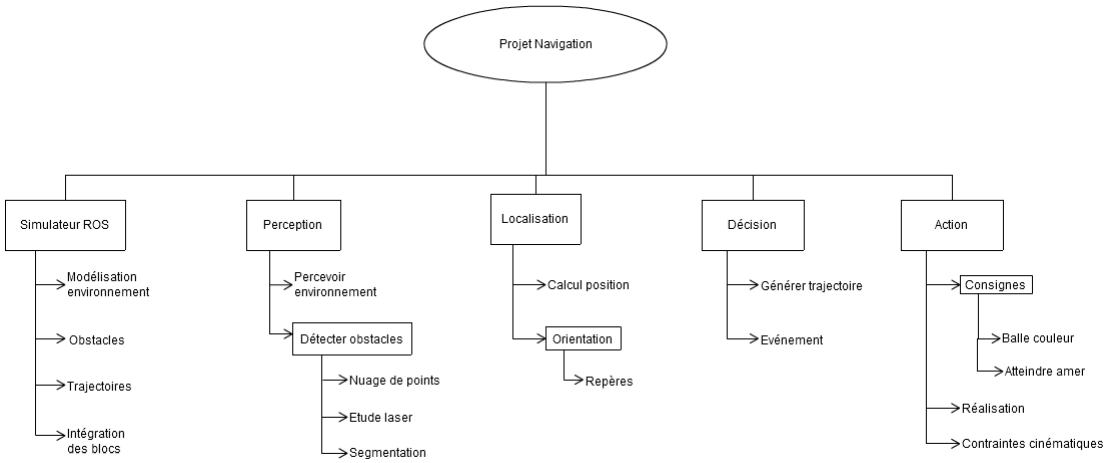


Figure 2 – Arbre Produit

5 RoadMap

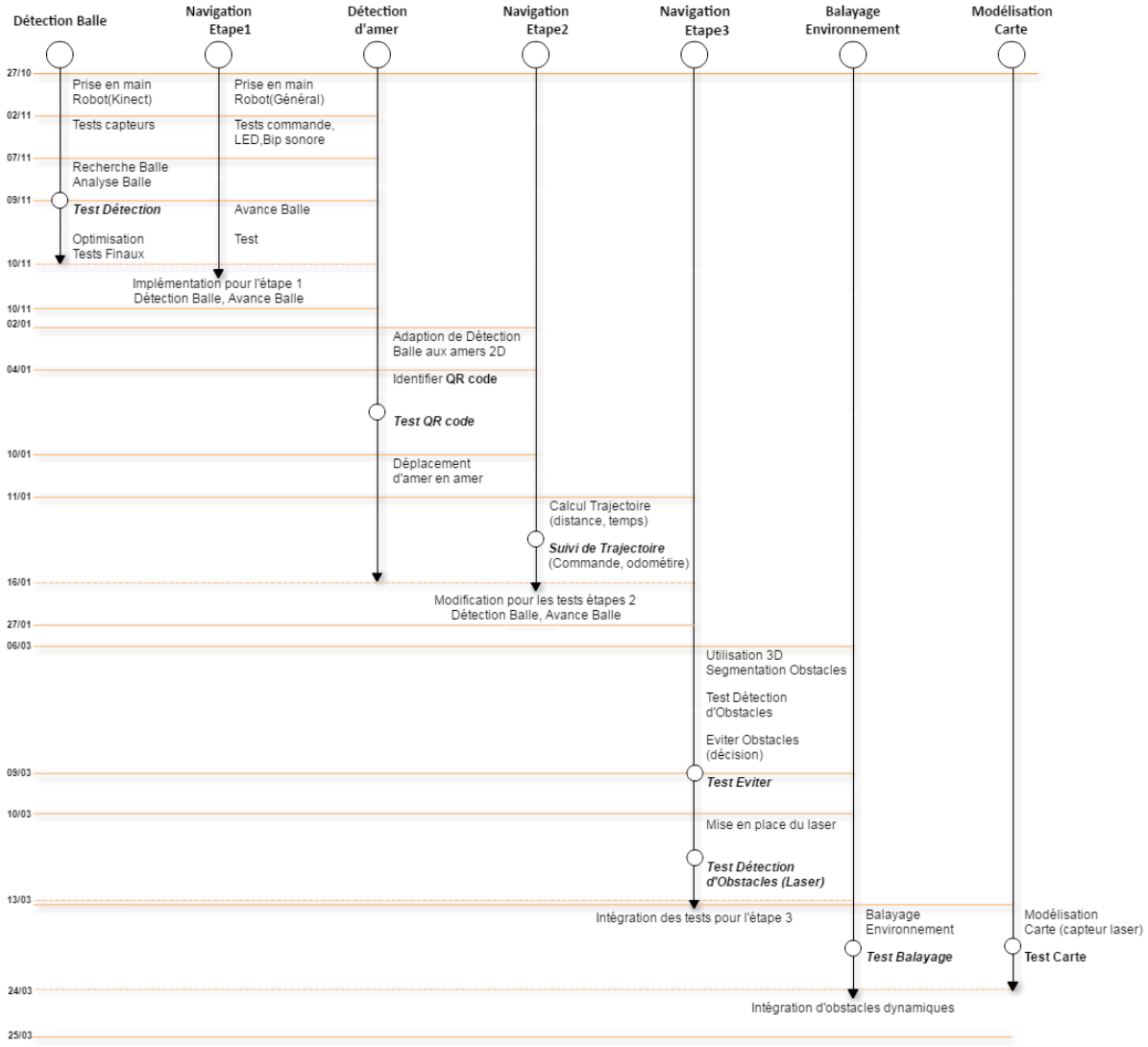


FIGURE 3 – Roadmap

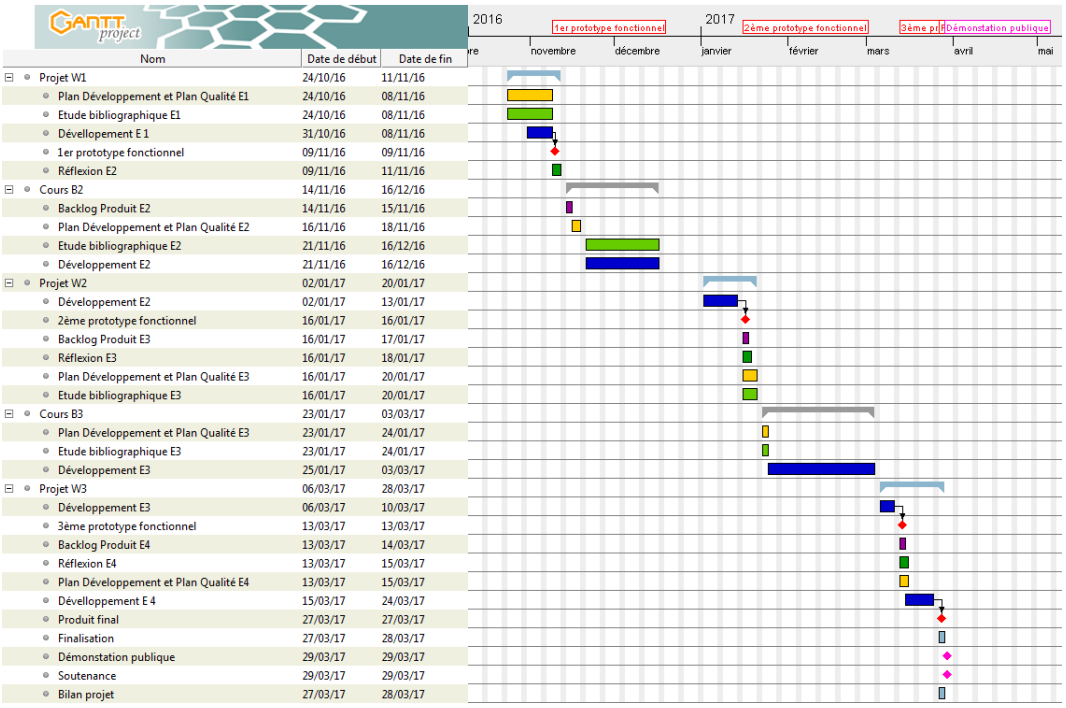
6 Planning

6.1 Lôt de tâches

Un lot de tâches, ou *Work Breakdown Structure*, est une décomposition hiérarchique des tâches à effectuer dans un projet. Il permet à la fois d'avoir une représentation visuelle du travail restant à réaliser, mais aussi d'évaluer rapidement l'avancement de chacune des tâches.

La figure **??** représente nos lots de tâches pour l'ensemble du projet, nous en avons identifié six.

6.2 Planning général



Nom	Date de début	Date de fin
Projet W1	24/10/16	11/11/16
Plan Développement et Plan Qualité E1	24/10/16	08/11/16
Etude bibliographique E1	24/10/16	08/11/16
Dévellopement E 1	31/10/16	08/11/16
1er prototype fonctionnel	09/11/16	09/11/16
Réflexion E2	09/11/16	11/11/16
Cours B2	14/11/16	16/12/16
Backlog Produit E2	14/11/16	15/11/16
Plan Développement et Plan Qualité E2	16/11/16	18/11/16
Etude bibliographique E2	21/11/16	16/12/16
Développement E2	21/11/16	16/12/16
Projet W2	02/01/17	20/01/17
Développement E2	02/01/17	13/01/17
2ème prototype fonctionnel	16/01/17	16/01/17
Backlog Produit E3	16/01/17	17/01/17
Réflexion E3	16/01/17	18/01/17
Plan Développement et Plan Qualité E3	16/01/17	20/01/17
Etude bibliographique E3	16/01/17	20/01/17
Cours B3	23/01/17	03/03/17
Plan Développement et Plan Qualité E3	23/01/17	24/01/17
Etude bibliographique E3	23/01/17	24/01/17
Développement E3	25/01/17	03/03/17
Projet W3	06/03/17	28/03/17
Développement E3	06/03/17	10/03/17
3ème prototype fonctionnel	13/03/17	13/03/17
Backlog Produit E4	13/03/17	14/03/17
Réflexion E4	13/03/17	15/03/17
Plan Développement et Plan Qualité E4	13/03/17	15/03/17
Dévelloppement E 4	15/03/17	24/03/17
Produit final	27/03/17	27/03/17
Finalisation	27/03/17	28/03/17
Démonstation publique	29/03/17	29/03/17
Soutenance	29/03/17	29/03/17
Bilan projet	27/03/17	28/03/17

FIGURE 4 – Gantt du projet

6.3 Livrables

6.3.1 Pour le client

W1 du 17/10 au 14/11 :
- Plan de Développement et de Qualité V1 : 8/11
- Code de l'étape 1 et Manuel Utilisateur correspondant : 8/11

W2 du 02/01 au 27/01 :
- Etat de l'Art de l'étape 2 : à définir avant le W2
- Plan V2 : 02/03
- Backlog de l'étape 2 : 02/03
- Plan de Développement et de Qualité V2 : à définir avant le W2
- Code V2 et Manuel Utilisateur V2 : à définir lors du W2
- Cahier de recettes V2 : à définir avant le W2
- Plan de Développement et de Qualité V3 : à définir avant l'étape 3

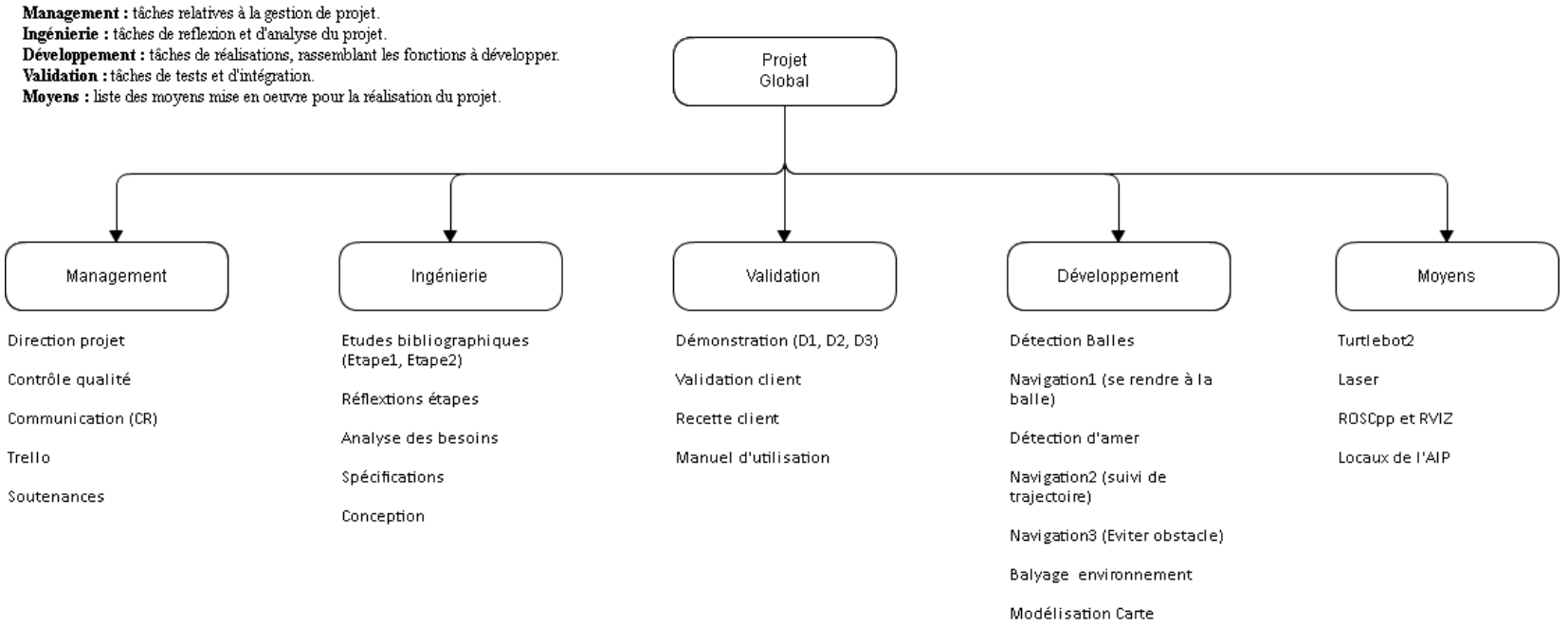


FIGURE 5 – Lot de tâches

— Backlog de l'étape 3 : à définir avant l'étape 3

W3 du 01/03 au 31/03 :

— Etat de l'Art de l'étape 3 : à définir avant l'étape 3
— Code V3 et Manuel Utilisateur V3 : à définir lors du W3
— Cahier de recettes V3 : à définir avant le W3
— Backlog : à définir avant l'épate 4
— Plan de Développement et de Qualité V4 : à définir avant l'étape 4
— Code V4 et Manuel Utilisateur V4 : à définir lors de l'étape 4
— Bilan de projet : à définir avant le W3
— Cahier de recettes : à définir avant le W3

Tout au long du projet :

— Compte-rendus de réunions

6.3.2 Pour les intervenants externes

— Compte-rendus de réunions
— Plan de Développement Qualité

6.4 Planning du W1

On peut voir sur la figure **??** le planning de la première période de projet W1.

6.5 Conclusion du projet

Ce plan qualité vise à assurer que les dispositions prises par l'équipe pour obtenir la qualité du logiciel définie en accord avec les clients soient respectées. Avec le même objectif, le responsable qualité, soutenu par toute l'équipe, sera en charge de vérifier que les engagements pris dans le présent document auront été appliqués tout au long de l'avancement de ce projet.

Les membres de l'équipe projet sont tenus de se conformer aux dispositions décrites dans le plan d'assurance et de contrôle qualité. Le non-respect des prescriptions du Plan de Qualité constaté donne lieu à un plan d'action curatif pour corriger les effets du dysfonctionnement et éventuellement à un plan d'action préventif pour éviter que celui-ci ne se reproduise. Ce dernier plan peut entraîner une modification du plan qualité. L'utilisation de ce plan doit permettre un total succès du projet :

Succès du produit : avoir un produit final qui satisfait les besoins des utilisateurs, qui a le niveau de qualité requis (tests, code propre et commenté) et qui puisse être évolutif (générique) et réutilisable (documentations)

Succès de la démarche : les demandes du client satisfaites dans les délais, les méthodes établies et le planning bien respecté

Le projet et les résultats obtenus seront décrits dans le bilan et exprimés lors de la présentation orale et de la démonstration publique.

FIGURE 6 – Gannt du W1

7 Assurance qualité

7.1 Organisation interne

L'organisation de notre projet suit une configuration Agile. Nous avons réparti les rôles dans l'équipe afin de respecter cette rganisation :

- Scrum Master (Tristan Klempka) : Responsable du processus de développement
- Product Owner (Bruno Dato) : Il représente le client au sein de notre équipe. Il dirige l'ordre de développement des fonctionnalités en maximisant la satisfaction du client.
- Équipe de développement : Ensemble des membres du projet

En ce qui concerne la communication interne au groupe, des mêlées sont organisées par le Scrum Master tous les jours vers 10h. Ces réunion sont courtes et ne doivent pas dépasser les 15 minutes. Chaque membre du groupe fait le point sur ce qu'il a effectué la veille, ce sur quoi il travaillera le jour de la réunion et des éventuels obstacles liés à ses activités

7.2 Organisation externe

La communication avec les clients se fera premièrement par courrier électronique. Les clients pourront alors répondre à toute l'équipe via l'adresse mail du groupe mise à leur disposition. Deuxièmement, des réunions hebdomadaires seront également prévues afin de tenir informés les intervenants sur l'avancement du projet. Des réunions supplémentaires pourront également être organisées si cela s'avère nécessaire. Chaque réunion sera prévue au moins 2 jours à l'avance et donnera lieu à un compte rendu de réunion. Ces comptes rendus seront envoyés par courriel, dont l'objet sera identifié (voir 7.4), et ils seront rédigés à partir des formulaires fournis en Annexe. Ces documents seront soumis à l'approbation dans les trois jours. De plus, toutes les informations relatives au projet, quelle que soit leur nature seront accessibles immédiatement dans nos espaces de travail (voir 7.3).

7.3 Outils

7.3.1 Développement

Système d'exploitation : Ubuntu 14.04 LTS et ROS

Git : Logiciel libre de gestion de versions. Nous l'utilisons pour la gestion de nos sources mais aussi pour gérer la rédaction collaborative de nos documents.

Github : Hébergeur de notre dépôt Git. L'interface Web fourni également un accès à des outils de gestion du code. Une liste d'Issues qui sera utilisé pour discuter des modifications effectuée dans les codes et un système de *Pull Request* qui nous permet de valider et d'intégrer des modifications. Un wiki est également accessible via cette interface.
[lien vers le dépôt GitHub]

7.3.2 Documentation

Unified Modeling Language (UML) : Ce langage va nous permettre de modéliser les différents composants et le déroulement de nos programmes. La modélisation y est graphique et permet de comprendre et de communiquer simplement sur le fonctionnement d'une application complexe.

Google Drive : Système de stockage et de partage en ligne de fichiers. Il permet d'archiver et d'avoir accès aux versions finales des documents rédigés et utilisés durant le projet.
[lien vers le dépôt Google Drive]

Cacoo : Service en ligne de création et de partage de diagramme.

LaTeX - TexMaker : LaTeX est un langage de rédaction de document. Il permet la rédaction de document et l'intégration de plusieurs parties facilement. Nous avons décidé de choisir une distribution LaTeX commune pour s'assurer que les documents produits soient homogènes (encodage, caractères spéciaux... etc)

7.3.3 Espace de travail

GanttProject : Logiciel libre de gestion de projet. Il nous permet de modéliser dans le temps l'ensemble des tâches liée au projet ainsi que leur relation entre elles.

Trello : Outil de gestion de projet en mode « Tableau de tâches ». Il nous permet d'organiser la distribution et la réalisation des tâches. Il sert également comme vecteur d'information sur la gestion du projet en général (liens des outils, documents importants... etc.).
[lien vers le tableau Trello]

Placker : Outil de gestion de projet en mode diagrammes de Gantt. Il permet de gérer les tâches d'un tableau Trello à l'aide d'un diagramme de Gantt. Il est utilisé pour surveiller l'avancement des tâches de plus bas niveau.

Google Mail : Service de messagerie électronique. Il est utilisé pour l'ensemble de nos échanges internes ou externes au projet.

Google Agenda : Service en ligne d'agenda. Utilisé pour la prise de rendez-vous et la mise en place des réunions internes et externes au groupe de projet.

7.4 Standards

7.4.1 Gestion des documents

Chaque document produit par le groupe projet devra comporter une page de garde reprenant les éléments nécessaires à leur suivi :

- Informations générales de suivi :
  - Nom du document
  - Version
  - Date de création
  - Date de modification
- Auteurs du document
- Auteur et date de la validation du document pour la version en cours.
- Historique de révision

Processus de production des documents :

- Rédaction des parties du document par l'ensemble des auteurs concernés ;
- Ajout des rédactions par *commit* sur le dépôt Git.
- Fusion et résolution des conflits. Ajout des corrections si nécessaire ;
- Validation finale d'une version du document. *Tag* de version sur le dépôt Git ;
- Envoi ou diffusion du document aux destinataires concernés ;
- Archivage du document.

Il peut être nécessaire de répéter plusieurs fois la partie rédaction, ajout et fusion afin d'obtenir une version du document convenable.

7.4.2 Objet des e-mails

Pour des raisons de suivi, l'ensemble des e-mails est envoyé à plusieurs destinataires chacun travaillant dans des domaines différents. Pour faciliter la recherche et l'identification du contenu des e-mails, nous avons décidé de respecter la forme d'objet suivante :

```
[Projet NAV - Thème] Objet du mail
```

7.4.3 Documents

Les documents fournis au client sont au format PDF et un format LaTeX est ajouté lorsqu'une validation du client est requise. Les noms des documents doivent respecter la forme suivante :

```
[Projet Navigation Autonome] Nom du document Version
```

7.4.4 Entête des fichiers

L'entête de chaque fichier source doit être complétée afin d'expliquer leur comportement. Elle doit être mise à jour au fur et à mesure des évolutions du fichier qu'elle décrit. Chaque entête doit permettre de comprendre facilement et rapidement les fonctionnalités codées. Voici une liste non exhaustive des éléments à renseigner :

- Méthode/Fonction
  - `D:` Descriptif
  - `A:` Auteur(s)
  - `E:` Description des paramètres
  - `S:` Donnée(s) renvoyée(s)
  - `R:` Donnée renvoyée
  - `F:` Exceptions et/ou code(s) d'erreur(s) renvoyé(es)
- Classe
  - Descriptif
  - Auteur(s)

Il est recommandé de commenter au maximum le code afin de faciliter la phase de développement du projet.

7.4.5 Nommage

Les fichiers suivent le nommage de leur classe respective. Pour les fichiers C++ c'est le suffixe .cpp qui sera utilisé. Pour les headers C++ le suffixe .hpp sera utilisé.

Classes :

- Première lettre en majuscule
- Mélange de minuscule, majuscule.
- Première lettre de chaque mot en majuscule
- Donner des noms simples et descriptifs
- Éviter les acronymes
- N'utiliser que `[a-z]` `[A-Z]` et `[0-9]`

Variables :

- Première lettre en minuscule
- Mélange de minuscule, majuscule avec la première lettre de chaque mot en majuscule
- Donner des noms simples et descriptifs
- N'utiliser que `[a-z]` `[A-Z]` et `[0-9]`

Constantes :

- Tout en majuscule
- Séparer les mots par des underscore
- Donner des noms simples et descriptifs
- N'utiliser que `[A-Z]` et `[0-9]`

7.4.6 Découpage du code

Il est important que le code soit suffisamment découpé. La taille des fichiers ne devrait pas excéder 1000 lignes et chaque fonction (ou méthode) ne devrait pas dépasser la centaine de lignes au grand maximum. Les headers et les fichiers source séparent les déclaration des définitions.

L'indentation doit être strictement respectée et doit correspondre à un espacement de 4 caractères. La taille des lignes ne doit pas dépasser 80 caractères.

7.4.7 Workflow

Voici une description du *workflow* utilisé pour réaliser les tâches de développement de notre projet :

- La branche *master* contient uniquement un système complet en état de fonctionner. Elle est protégée en écriture par le Scrum Master.

— La branche *develop* est notre branche de travail par défaut.
— Pour travailler sur une nouveauté, on travaille directement sur la branche *develop* ou on créé une nouvelle branche à partir de la branche *develop* lorsque la fonctionnalité développée demande au moins un jour de travail.
— Lorsque la branche de travail est prête, on ouvre une *Pull Request* pour demander une intégration à la branche master au Scrum Master.
— Validation et/ou corrections des modifications. Une fois que cela est fait on fusionne dans la branche *master*.
— Déploiement du code de la branche *master* sur le système.

Il est également important d'alimenter le *wiki* lorsqu'un membre du groupe estime qu'une information importante concernant les développements doit être partagée aux utilisateurs et/ou aux membres du projet. La forme de rédaction est libre mais chaque page doit être référencée avec un titre précis et explicite sur le contenu qu'elle présente.

Il est conseillé de respecter le format suivant pour les préfixes des *commit* sur le dépôt Git :
— `ADD` : Ajout de fichiers, de méthodes, d'une configuration, d'un texte ... etc...
— `ENH` : Amélioration d'un élément déjà présent
— `RM` : Suppression d'un fichier, d'une méthode, d'une classe, d'un texte ... etc...
— `BUG` : Correction d'un bug.
— `TEST` : Relatif aux tests.
— `DOC` : Relatif à la documentation.

Il est nécessaire que l'ensemble des membres du projet utilisent la liste d'*Issues* pour pouvoir effectuer et archiver les différents discussions sur le code du projet. Il est possible d'ouvrir et de fermer ces *issues*. Lorsqu'un *commit* est en rapport avec une *issue* il doit être relié avec celui-ci (i.e. `BUG : add method close #12` ce commit ferme automatiquement l'*issue* numéro 12).

8 Analyse des risques

L'analyse des risques projet se doit d'être menée dès le tout début de la phase de lancement, afin d'identifier au plus tôt et de la manière la plus exhaustive possible les éléments qui pourraient avoir une influence négative sur le déroulement du projet. Cette analyse est destinée à évoluer tout au long de notre projet. Les risques seront synthétisés et classés dans un tableau avec pour chacun sa définition, sa probabilité, sa gravité, ses causes et effets, ainsi que les actions préventives et correctives à mettre en oeuvre pour y pallier. On définira ainsi la criticité de chaque risque.

Définition	Probabilité	Gravité	Cause(s)	Effet(s)	Actions préventives (P) et correctives (C)		Criticité
Product Owner ou Scrum Master absent (longue durée)	Faible (1)	Importante (3)	Problème de santé, abandon de la formation/du projet	Retard de livraison important	(P) Mettre en place une équipe qui ne dépend pas que d'une seule personne (C) Changement des rôles de Product Owner ou Scrum Master		Importante (3)
Ne pas finir le projet à temps ou mauvaise estimation des charges	Moyenne (2)	Importante (3)	Mauvaise estimation, mauvaix planning	Insatisfaction du client	(P) Planifier, estimer la charge de travail et faire le point toutes les semaines (P) Ajuster la planification en fonction		Importante (3)
Indisponibilité de certains membres de l'équipe durant une partie du projet	Faible (1)	Importante (3)	Problème de santé, abandon de la formation/du projet	Retard de livraison important	(P) Tenir la documentation (techniques, compte-rendu, ...) à jour (C) Essayer d'adapter le projet aux membres du groupe restant		Importante (3)
Conflit avec l'encadrant	Faible (1)	Importante (3)	Mauvaise entente, pas de communication	Mauvaise condition de travail	(P) Communiquer (C) Contacter les supérieurs afin d'exposer le problème		Importante (3)
Ne pas aboutir le travail de recherche	Faible (1)	Moyenne (2)	Travail de recherche incomplet	Insatisfaction du client	(P) Se concentrer sur moins d'algorithmes (P) Etre le plus générique possible dans la conception pour que l'interface soit la plus flexible et réutilisable possible		Moyenne (2)
Ne pas répondre aux demandes du client, manque d'informations	Moyenne (2)	Importante (3)	Mauvaise communication, propositions trop ambitieuses, mauvaise gestion du temps, ...	Insatisfaction du client	(P) Communiquer sans cesse avec le client en cas de doute (P) Faire des maquettes, des prototypes en définissant clairement leurs objectifs en expliquant ce qu'il s'agit de démontrer, et quand terminer ces activités (P) Spécifier clairement les exigences et lister l'ensemble des tests avant le développement		Importante (3)
Obtenir un résultat trop peu intuitif	Moyenne (2)	Moyenne (2)	Complexité de la solution proposée	Difficulté pour réutiliser le travail effectué	(P) Test par des utilisateurs extérieurs au projet		Moyenne (2)
Modifications non contrôlées, mauvais choix technique, sous-estimation des capacités d'infrastructure	Moyenne (2)	Moyenne (2)	Propositions effectuées trop ambitieuses	Retard de livraison important	(P) Faire régulièrement des tests, suivre ses résultats (P) Analyse d'impact (P) Versioning fréquent pour avoir toutes les versions antérieures d'un fichier à tout moment		Moyenne (2)
Perte de code source	Faible (1)	Importante (3)	Crash du PC, perte des clés USB	Retard de livraison important	(P) Faire des copies sur d'autres supports		Importante (3)

FIGURE 7 – Analyse de risques

9 Tests et Validation

Scénario Etape 1 : Détection d'une balle de couleur et approche du TurtleBot
Pré-requis : Environnement propre, lumineux, et une balle de couleur de diamètre 10.5 cm.
Le listing des différents tests de l'étape 1 est le suivant :

ID	Démarche	Comportement attendu
1	Lancement du robot	Bip sonore indiquant que le robot est lancé
2	Modification des couleurs LED	Changement de la couleur de la LED
3	Essai des differents bip sonores	Bips sonores différents
4	Demande d'avance du robot (translation)	Avance du robot en visuel
5	Demande de rotation du robot	Rotation du robot en visuel
6	Lancement de la caméra	Caméra visible à l'écran
7	Lancement de la détection d'une balle	Détecte la balle et renvoie la couleur à l'écran
8	Lancement du programme permettant la détection d'une balle et l'approche vers cette dernière	Bip sonore indiquant que le robot est lancé, détection de la balle (rotation tant qu'elle n'est pas détectée), avance vers la balle et bip de fin une fois la balle atteinte.

FIGURE 8 – Les différents tests de l'étape 1

À partir de notre liste de tests, nous intégrons au fur et à mesure les fonctionnalités nécessaire à la résolution de l'objectif de l'étape 1 : détecter une balle de couleur dans l'environnement du robot et s'y approcher. Pour la validation des tests, elle se fera lors de la démonstration du prototype avec le client. Un cahier de recette sera effectué par la suite.

Table des figures

ANNEXE

Compte-rendu de réunion numéro [n]

Date : [jour jj mois aaaa]

Objectif(s)	[liste]
Participants	[liste]
Auteur	[auteur]

Sujets abordés

[Titre 1]

[Titre 2]

Décision(s) prise(s)

Prochaine réunion :