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 :
Hugo Brefel
Sylvain Guillaume
Luc Rubio
Salah Eddine Ghamri
Pierre Beauhaire

Tuteur :
Frédéric Lerasle
Michaël Lauer
Michel Taix

LAAS-CNRS

Mars 2018

Suivi du document

Nom du document	Version Majeure	Date de création	Dernière version
PDDQ	3.0	20/03/2018	20/03/2018

Auteurs du document

Rédaction	Intégration	Relecture	Validation Interne
Équipe	Hugo Brefel Sylvain Guillaume Luc Rubio Salah Eddine Ghamri Pierre Beauhaire	Hugo Brefel Sylvain Guillaume Luc Rubio Salah Eddine Ghamri Pierre Beauhaire	Hugo Brefel Sylvain Guillaume Luc Rubio Salah Eddine Ghamri Pierre Beauhaire

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 au projet.

Historiques de révision

Version	Modification apportée	Auteur	Date
3.0	Création et intégration du document	Pierre Beauhaire Luc Rubio	20/03/2018

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. Le 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), 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 », Hugo, Sylvain et Pierre font partie du parcours IARF, Luc est issu de la spécialité Décision de RODECO et enfin, Salah Eddine, de la spécialité Commande.

Le projet est décomposé tout le long de l'année en tranches W, de 3 et 4 semaines, en alternance avec des blocs de cours B.

Cours1	Release1	Cours2	Release2	Cours3	Release3
6 sem.	3 sem. 23/10 - 10/11	5 sem.	3 sem. 18/12 - 22/12 10/01 - 19/01	6 sem.	3 sem. 12/03 - 31/03

2.1.3 Objectif

Actuellement le robot peut se déplacer dans une salle et estimer à peu près sa position sur une carte préétablie en utilisant des amers (AR-code). Les objectifs du projet vont, dans un premier temps, être de pouvoir asservir le robot sur l'AR-code, d'améliorer l'estimation de la localisation du robot via le filtre de Kalman et qu'il puisse réaliser des déplacements dans un environnement comprenant plusieurs salles. Nous allons également nous intéresser à la façon dont la carte représentant l'environnement du robot est créée (jusqu'ici elle était préétablie) et allons essayer de rendre la création et la mise à jour de la carte dynamique. Ceci notamment pour que le robot soit capable de détecter des obstacles qui ne sont pas sur la carte initiale de l'environnement.

2.1.4 Entrées

- ☑ Cahier des charges
- ☑ Documentations

2.1.5 Sorties

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

- ☐ Code propre, fonctionnel et surtout facilement réutilisable
- ☐ Documentation
- ☐ Robot opérationnel qui effectue les tâches demandées
- ☐ Manuel Utilisateur
- ☐ Manuel Développeur

2.1.6 Limites

- Incrémentation des solutions : difficulté d'anticiper les solutions et problèmes des étapes suivantes
- Cours de vision 2D et de navigations en Bloc2 et vision 3D en Bloc3

2.2 Parties-prenantes

2.2.1 MOE

Hugo Brefel
brefel.hugo@gmail.com
Spécialité IARF

Pierre Beauhaire
beauhaire.pierre@gmail.com
Spécialité IARF

Sylvain Guillaume
sylvain31g@free.fr
Spécialité IARF

Luc Rubio
luc.rubio.lr@gmail.com
Spécialité Décision

Salah Eddine Ghamri
salaheddineghamri@gmail.com
Spécialité Commande

2.2.2 MOA

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

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

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

2.2.3 Intervenants externes

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

2.3.1 Exigences

Les clients ont spécifié plusieurs exigences :
- Se déplacer dans un environnement connu à l'avance par le robot, composé de différentes salles.
- Utiliser les amers dans le but de se localiser.
- Éviter les obstacles dynamiques.

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.

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.

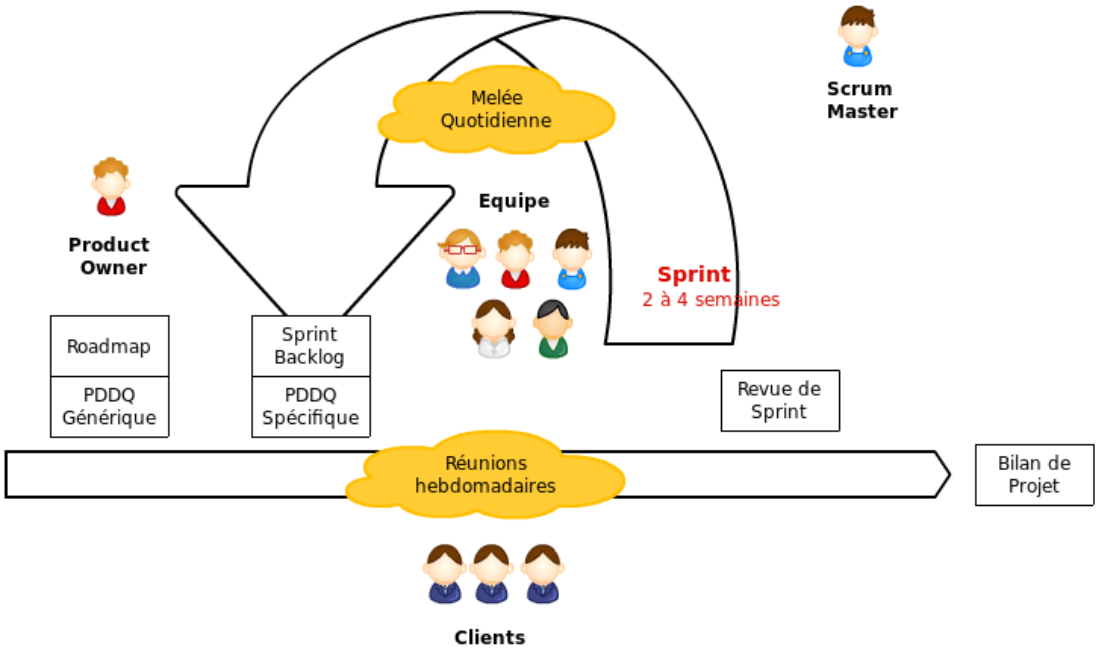


FIGURE 1 – Schéma Méthode Scrum

3.2 Sprints

Le projet est divisé en 3 étapes définies par le client :

Étape 1 Se familiariser avec l'environnement de travail ROS et prendre en main les travaux déjà effectués par l'équipe précédente.

Étape 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,....).

Étape 3 Réaliser une tâche de navigation dans un environnement supposé connu et avec obstacles statiques pour atteindre une position finale 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,...).

La première étape a pour objectif de prendre en main les différentes ressources à notre disposition. Nous avons également étudié les différentes erreurs qui pouvaient apparaître lors de l'utilisation d'un Turtlebot : erreurs sur l'odométrie lors du mouvement du robot en ligne droite ou en rotation, erreurs sur la détection des amers ; tout cela dans le but d'obtenir une quantification des erreurs.

La deuxième étape a pour but la réalisation de la navigation d'un TurtleBot d'un point A à un point B dans un environnement connu, sans obstacles. La rencontre d'amers permet au robot de compenser ses erreurs de trajectoire et ainsi de se relocaliser sur la carte.

La troisième étape a pour but la réalisation de la navigation d'un TurtleBot d'un point A à un point B dans un environnement connu, avec obstacles statiques. La rencontre d'amers permet au robot de compenser ses erreurs de trajectoire et ainsi de se relocaliser sur la carte. Une mise à jour de la carte sous RViz est également effectuée en temps réel.

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 :

Perception : Décrit l'ensemble des fonctionnalités permettant l'identification à partir d'une image. Plusieurs fonctions interviennent. Le robot va chercher à détecter des amers. Un traitement est également appliqué sur la carte de l'environnement : une érosion pour enlever le bruit sur l'image, une dilatation pour compenser l'érosion... La détection des obstacles devra tenir 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. Le robot estime sa position grâce à la reconnaissance des amers et à la triangulation du filtre de Kalman.

Décision : Décrit les fonctions nécessaires aux différentes décisions que le robot doit prendre pour naviguer jusqu'au point but. Le robot calcule un graphe de connexité reliant les points du nuage de points. Il détermine ensuite le chemin avec le coût le plus intéressant jusqu'à la position finale. Afin d'avoir une trajectoire continue, des courbes de Bézier sont calculées pour lisser la trajectoire.

Action : Concerne toutes les fonctions correspondant au domaine de la commande robotisée permettant le respect des consignes (atteindre un amer...), la réalisation des mouvements, et les contraintes cinématiques.

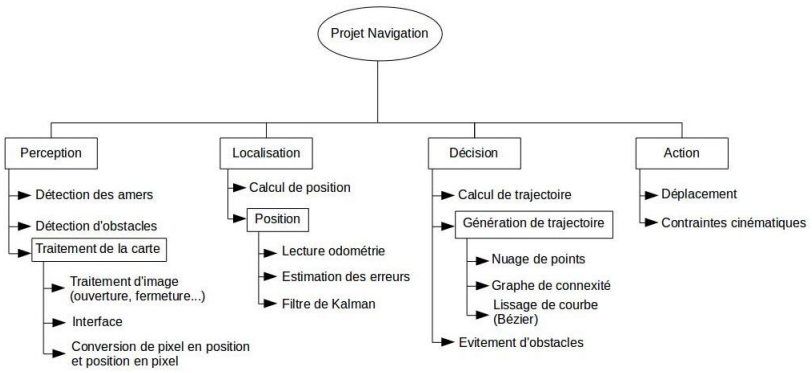


FIGURE 2 – Arbre Produit

5 RoadMap

	Release 1	Release 2	Release 3
étude du travail existant	-Apprentissage de la librairie ROS -Étude des limites des ressources existante - Test	- Révision du code existant	/
Traitement de carte	/	- Interface - Traitement d'image - Conversion de position	- Interface - Ajout d'obstacle
Détection d'obstacle	/	- Étude des capteur de la Turtlebot	- Création de fonction d'évitement
Visualisation d'amers	/	- Étude des limites de vision de la Kinect - Détection des zones de visibilité des AR code - Positionnement - Asservissement - Filtre de Kalman	- Asservissement - Détection d'amer et relocalisation en mouvement - Finition filtre de Kalman
Conception de trajectoires	/	- PRM - A* dans un graphe cyclique - Béziers - Détection de collision	- Intégration de la visualisation et détection d'obstacle - Suiveur de trajectoire (commande)
Management	- Gestion du trello - Gestion du GitHub - Prise de rendez-vous avec les clients	- Gestion du trello - Gestion du GitHub - Prise de rendez-vous avec les clients	- Gestion du trello - Gestion du GitHub - Prise de rendez-vous avec les clients
Documentation	-Réalisation d'un PDDQ -Réalisation d'une roadmap -Réalisation d'une analyse de risque -Réalisation d'un arbre produit -Rapports de réunions -Présentation	- Amélioration du PDDQ - Amélioration de la roadmap - Rapports de réunions - Manuel d'utilisation - Manuel de développement - État de l'art	-Amélioration du PDDQ -Manuel d'utilisation -Manuel de développement -Livrables (flyer, poster, code, slides)
Test	-Test du code existant	- Test unitaire - Test d'intégration - Validation	-Test unitaire -Test d'intégration -Validation
État de l'art	Début de recherche d'article	Livraison de l'état de l'art	/

FIGURE 3 – Roadmap

6 Planning

6.1 Backlog

Un *backlog* est une liste de fonctionnalités ou de tâches jugées nécessaires et suffisantes pour la réalisation satisfaisante du 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.

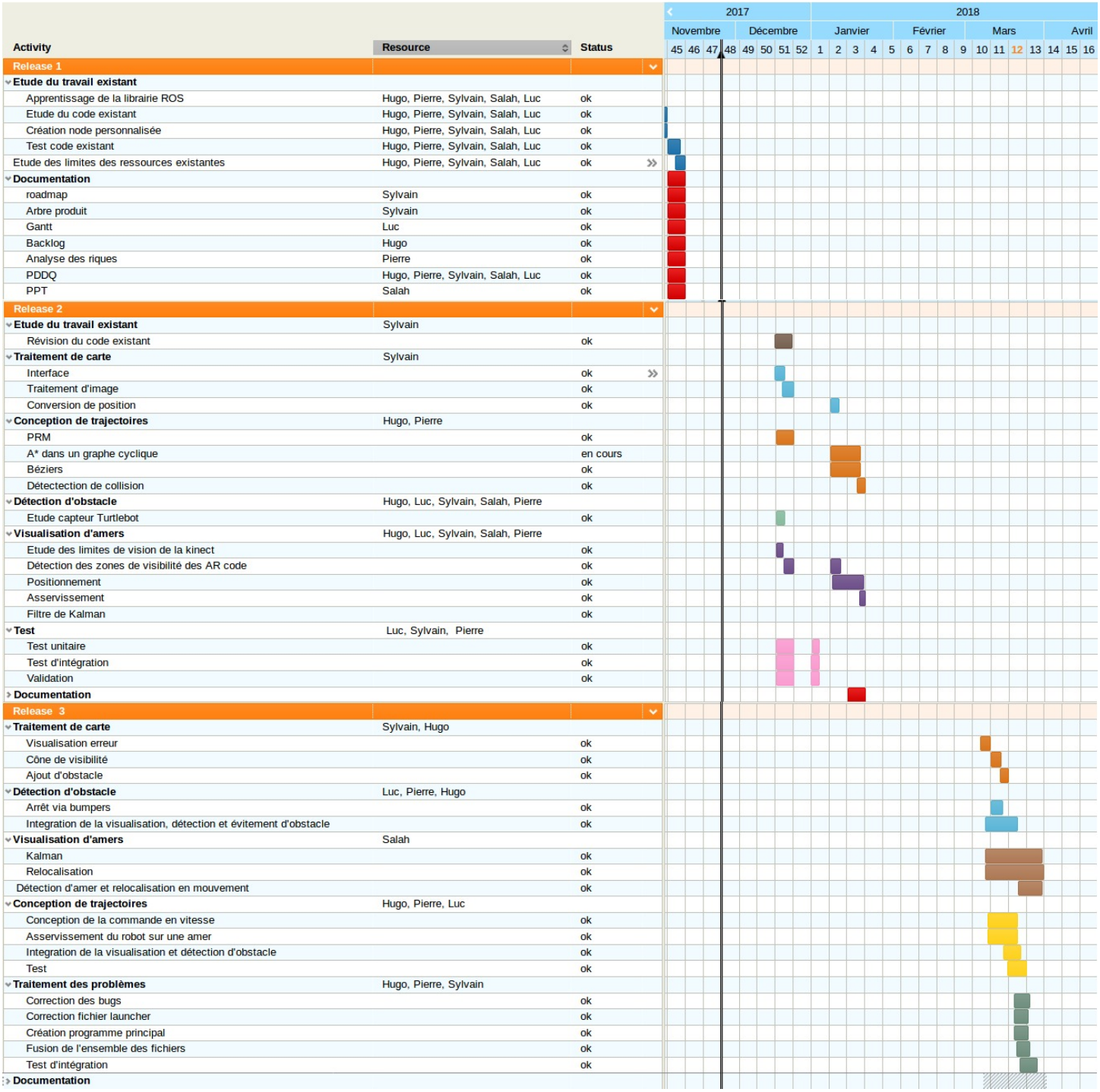
Release 1		
Sprint n°	**Exigence**	**Priorité de l'exigence**
Sprint 1	Prise en main du fonctionnement de ROS	1
	Prise en main du fonctionnement de Turtlebot sous ROS	2
Sprint 2	Prise en main du code existant	1
Sprint 3	Tester les limites du code déjà existant	2
	Réaliser les livrables de fin de première release	1

Release 2		
Sprint n°	**Exigence**	**Priorité de l'exigence**
Sprint 1	Remaniement du code existant	1
	Réalisation de tests de précision sur le code existant	2
Sprint 2	Début de la conception de trajectoire	1
Sprint 3	Fin de la conception de trajectoire	2
	Visualisation d'amer	2
	Traitement de cartes	2
	Réaliser les livrables de fin de deuxième release	1

Release 3		
Sprint n°	**Exigence**	**Priorité de l'exigence**
Sprint 1	Recherche d'amer/Asservissement	2
	Correction des divers bugs existants	2
	Remaniement du fichier launcher	1
Sprint 2	Arrêt via bumpers	2
	Localisation (Kalman)	2
	Détection d'obstacle	2
	Fusion/tests d'intégrations	2
	Création du flyer	1
	Création du poster	1
Sprint 3	Réaliser les livrables de fin de troisième release	1
	Préparation de la journée du département	1
	Préparation de la présentation flash	1

FIGURE 4 – Backlog des Release1, Release2 et Release3

6.2 Planning général



Activity	Resource	Status
Release 1		
Etude du travail existant		
Apprentissage de la librairie ROS	Hugo, Pierre, Sylvain, Salah, Luc	ok
Etude du code existant	Hugo, Pierre, Sylvain, Salah, Luc	ok
Création node personnalisée	Hugo, Pierre, Sylvain, Salah, Luc	ok
Test code existant	Hugo, Pierre, Sylvain, Salah, Luc	ok
Etude des limites des ressources existantes	Hugo, Pierre, Sylvain, Salah, Luc	ok
Documentation		
roadmap	Sylvain	ok
Arbre produit	Sylvain	ok
Gantt	Luc	ok
Backlog	Hugo	ok
Analyse des riques	Pierre	ok
PDDQ	Hugo, Pierre, Sylvain, Salah, Luc	ok
PPT	Salah	ok
Release 2		
Etude du travail existant	Sylvain	
Révision du code existant		ok
Traitement de carte	Sylvain	
Interface		ok
Traitement d'image		ok
Conversion de position		ok
Conception de trajectoires	Hugo, Pierre	
PRM		ok
A* dans un graphe cyclique		en cours
Béziers		ok
Détectection de collision		ok
Détection d'obstacle	Hugo, Luc, Sylvain, Salah, Pierre	
Etude capteur Turtlebot		ok
Visualisation d'amers	Hugo, Luc, Sylvain, Salah, Pierre	
Etude des limites de vision de la kinect		ok
Détection des zones de visibilité des AR code		ok
Positionnement		ok
Asservissement		ok
Filtre de Kalman		ok
Test	Luc, Sylvain, Pierre	
Test unitaire		ok
Test d'intégration		ok
Validation		ok
Documentation		
Release 3		
Traitement de carte	Sylvain, Hugo	
Visualisation erreur		ok
Cône de visibilité		ok
Ajout d'obstacle		ok
Détection d'obstacle	Luc, Pierre, Hugo	
Arrêt via bumpers		ok
Integration de la visualisation, détection et évitement d'obstacle		ok
Visualisation d'amers	Salah	
Kalman		ok
Relocalisation		ok
Détection d'amer et relocalisation en mouvement		ok
Conception de trajectoires	Hugo, Pierre, Luc	
Conception de la commande en vitesse		ok
Asservissement du robot sur une amer		ok
Integration de la visualisation et détection d'obstacle		ok
Test		ok
Traitement des problèmes	Hugo, Pierre, Sylvain	
Correction des bugs		ok
Correction fichier launcher		ok
Création programme principal		ok
Fusion de l'ensemble des fichiers		ok
Test d'intégration		ok
Documentation		

FIGURE 5 – Gantt du projet

6.3 Livrables

6.3.1 Pour le client

Release1 du 23/10 au 10/11 :

— Backlog de Release1
— Plan de Développement et de Qualité V1 : 16/11/2017

Release2 du 18/12 au 19/01 :
— Backlog de Release2
— État de l'Art : à la fin du Release2
— Plan de Développement et de Qualité V2 : 19/01/2018

Release3 du 12/03 au 31/03 :
— Flyer
— Poster
— Backlog de Release3
— Plan de Développement et de Qualité V3 : 20/03/2018
— Code et Manuel Utilisateur : fin de Release3
— Manuel Développeur : fin de Release3
— Présentation et démonstration publique : fin de Release3

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 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 de développement qualité. Le non-respect des prescriptions du plan de développement 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 : satisfaire les demandes du client dans les délais. Respecter les méthodes établies et le planning.

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

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 organisation :

- Scrum Master (Pierre Beauhaire) : Il est responsable du processus de développement.
- Product Owner (Hugo Brefel) : Il représente le client au sein de notre équipe. Il dirige l'ordre de développement des fonctionnalités en maximisant la satisfaction du client.
- Responsable Technique (Sylvain Guillaume) : Il veille au respect des spécifications et des contraintes de mise en œuvre.
- Responsable Qualité (Luc Rubio) : Il effectue les tests unitaires et les tests de validation. Il est responsable de la qualité du projet, et doit donc veiller au respect des exigences.
- 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 sur les éventuels obstacles liés à ses activités.

7.2 Organisation externe

La communication avec les clients se fera généralement par courrier électronique. Les clients pourront alors répondre à l'équipe via l'adresse mail du groupe mise à leur disposition. Des réunions 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, 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 fournit également un accès à des outils de gestion du code. Une liste d'Issues sera utilisé pour discuter des modifications effectuées dans les codes. Un wiki est également accessible via cette interface.
[lien vers le dépôt GitHub]

7.3.2 Documentation

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

Tom's Planner : Logiciel 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]

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

7.4.4 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]`

7.4.5 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.

7.4.6 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.
- La branche *test* permet d'effectuer des tests sur notre code, sans déteriorer la version actuelle.
- La branche *Version 3-4-5* sont des sauvegardes de version.
- 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 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.

Risque	Probabilité	Gravité	Cause	Effet	Action Corrective (C) ou Préventive (P)
Spécification vague ou pas claire	Faible	Majeure	Mauvaise analyse et mauvaise identification des besoins client	Travail réalisé pas en adéquation avec les attentes du client	- Établir un document de spécification et vérifier avec le client (P) - Établir une revue de spécification (C)
Indisponibilité du matériel (Turtlebot, accès à l'AIP…) ou des plateformes (site ROS...)	Moyenne	Majeure	Turtlebot utilisée pour un cours, AIP fermée, instabilité des serveurs…	Blocage de l'avancement du projet	- Planifier une nouvelle organisation (P) - Effectuer une sauvegarde locale des tutoriels ROS (P)
Inaptitude d'un membre	Moyenne	Majeure	Mauvaise affectation et distribution des rôles	Rendu de mauvaise qualité	- Affecter les tâches en fonction des qualités de chaque membre de l'équipe (P)
Indisponibilité de certains membres de l'équipe durant une partie du projet	Faible	Majeure	Problème de santé / Abandon de la formation	Retard dans le développement et la livraison	- Mettre en place une équipe qui ne dépend pas que d'une seule personne (P) - Tenir la documentation (Manuels développeur, utilisateur, compte-rendus…) à jour (P) - Changer les rôles clés (Product Owner, Scrum Master...) (C) - Adapter le projet aux membres restants du groupe (C)
Indisponibilité des encadrants	Faible	Moyenne	Problème de calendrier	Mauvaises conditions de travail	- Communiquer (P)
Conflit avec les encadrants	Faible	Majeure	Mauvaise entente, pas de communication	Mauvaises conditions de travail	- Communiquer, fixer des rendez-vous longtemps à l'avance (P) - Contacter les supérieurs afin d'exposer le problème (C)
Retard du projet ou mauvaise estimation des charges	Moyenne	Majeure	Mauvais planning, mauvaise estimation du travail à effectuer	Insatisfaction du client	- Planifier le travail à effectuer, estimer la charge adéquate et faire le point chaque semaine (P) - Ajuster la planification (P)
Modifications non contrôlées, mauvais choix techniques, sous-estimation des capacités d'infrastructure	Moyenne	Moyenne	Propositions effectuées trop ambitieuses	Retard de livraison important	- Faire régulièrement des tests (P) - Analyser l'impact d'une nouvelle version (P) - Garder disponible les anciennes versions du projet (P)
Perte du code source	Faible	Majeure	Crash du PC, perte de clés USB	Retard de livraison important	- Faire des copies sur plusieurs supports indépendants (P)
Travail de recherche non abouti	Faible	Moyenne	Travail de recherche incomplet, manque de temps…	Insatisfaction du client	- Se concentrer sur moins d'algorithmes (P)
Travail réalisé pas en adéquation avec les demandes du client	Faible	Majeure	Mauvaise communication, mauvaise spécification des besoins client, Mauvaise gestion du temps, propositions trop ambitieuses…	Insatisfaction du client	- Communiquer avec le client (P) - Définir clairement les objectifs et une date limite pour la fin de chaque activité (P) - Spécifier clairement les exigences et lister l'ensemble des tests avant de commencer le développement (P)
Obtenir un résultat trop peu intuitif	Moyenne	Moyenne	Complexité de la solution proposée	Difficulté pour réutiliser le travail effectué	- Faire tester par des utilisateurs extérieurs au projet (P)
Problème d'environnement (luminosité trop importante pour le robot)	Moyenne	Majeure	Localisation de la salle de travail	Mauvaise vision du robot	- Installer des rideaux (C) - Effectuer les mouvements du robot quand la lumière n'est pas aveuglante (P) - S'assurer que tous les objets de la scène soient visibles (P)
Incapacité de se déplacer (pour le robot)	Moyenne	Majeure	Roues du robot trop petites ou en mauvais état, sol en mauvais état	Le robot ne se déplace pas, et/ou les données renvoyées par ses capteurs sont erronées	- Tester les déplacements du robot dans différentes salles et sur des sols différents (P)
Amers mal placés	Moyenne	Majeure	Amers mal placés	Impossibilité de se localiser	- Placer les amers de sorte que le robot puisse en voir au moins un s'il effectue un tour sur lui-même (P)
Vitesse du robot trop importante	Faible	Majeure	Mauvaise implémentation du déplacement du robot	Risque de dégâts matériels et temps d'arrêt possiblement important	- Mettre un seuil maximal en vitesse lors du déplacement du robot (P)
Erreur de relocalisation	Faible	Majeure	Mauvaise implémentation	Risque de collision avec l'environnement	- Minimiser l'erreur de relocalisation grâce à la mise en place de solutions (estimation de l'erreur, filtre de Kalman...)
Détection d'obstacles défectueuse	Faible	Majeure	Mauvaise implémentation de la détection d'obstacles	Collision avec une personne ou avec un obstacle, risque de dégâts matériels ou physiques	- S'assurer que la détection et l'évitement d'obstacles sont au point avant d'effectuer des déplacements (P)
Bumpers du robot défectueux	Faible	Majeure	Robot en mauvais état	Collision avec l'environnement non reconnue par le robot, aucun arrêt du robot	- S'assurer que les bumpers soient fonctionnels (P) - Changer les bumpers du robot (C)
Perte totale de la position du robot	Faible	Majeure	Aucun amers en vue, caméra ou roue du robot non fonctionnelle	Impossibilité d'effectuer le trajet prévu, possibilité de collision avec l'environnement	- Placer les amers de manière judicieuse (P) - Tester la caméra et les roues du robot avant utilisation (P)

FIGURE 6 – Analyse des risques

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 :