UNIVERSITÉ DE NANTES

Miage

OutDoorGate+RileyRover
Spécification d'exigences logicielles

Version 1.0
2020-12-10

Pascal ANDRE

Historique des modifications du document

Date	Version	Description	Auteur
<aaaa-mm-jj>	<x.x>	<détails>	<nom>
modèle	1.0	Projet de conception	Gerson Sunyé, Pascal André
Exigences	1.0	Projet de conception 2019	Pascal André

Table des matières

Spécification d'exigences logicielles

1. Introduction

Ce document énumère les exigences du projet « OutDoorGate ». Il suit la norme IEEE 830-1998.

1.1 Objectif du document

Ce document a pour objectif de décrire les exigences fonctionnelles et non-fonctionnelles du projet « OutDoorGate+RileyRover ».
On s'attachera à décrire le comportement utilisateur attendu pour chaque aspect du système au travers de use cases et de diagrammes de séquences fournis en annexe.

1.2 Portée du document

Ce document s'applique seulement au développement de composants « métier » du système. Il ne concerne pas l'interface utilisateur.

1.3 Définitions, acronymes et abréviations

IHM – Interface Homme-Machine
REST - Representational State Transfer
Lejos LeJOS, Java for Lego Mindstorms / EV3

1.4 Références

https://ev3.univ-nantes.fr
http://www.lejos.org/ev3.php

1.5 Vue d'ensemble

Voir la présentation du cas d'étude.

2. Description générale

2.1 Perspectives du produit

2.1.1 Interfaces système

Le système ne communiquera pas avec d'autres systèmes que ceux indiqués dans la description du cas d'étude.

2.1.2 Interfaces utilisateurs

Le système se compose des composants des deux sous-systèmes avec pour chacun une interface utilisateur pour la partie pilotage déployée sur une tablette android.

2.1.3 Interfaces matérielles

L'interface matérielle du prototype est celle supportée par Lejos pour le contrôle d'automates EV3.

2.1.4 Interfaces logicielles

Le système doit être divisé en deux sous-systèmes et un composant d'intégration. Tous les composants disposent d'interfaces logicielles claires et simples.

2.1.5 Interfaces de communication

Les deux sous-systèmes communiquent selon un protocole qu'il conviendra de définir soigneusement. Les communications des sous-systèmes sont définies dans leur document d'exigences fonctionnelles.
La mise en œuvre des communications utilisera une communication wifi ou bluetooth.

2.1.6 Contraintes de mémoire

Le système doit pouvoir s'exécuter correctement sur l'EV3 fourni et sur une tablette disposant d'un 1Go de mémoire vive.
Le pilotage est une application mobile restreinte exigeant peu de ressources.

2.2 Fonctions du produit

Le système doit permettre l'émulation d'un fonctionnement automatique et sécurisé de la porte de clôture.

2.3 Caractéristiques des utilisateurs

Les utilisateurs du système seront des développeurs débutants ou expérimentés.

2.4 Contraintes

Le système doit permettre de manipuler le portail en toute sécurité. On reprendra les exigences non-fonctionnelles des sous-systèmes qu'on s'attachera à remplir. On complète avec :.

- Le portail ne peut être connecté qu'à un seul véhicule à la fois.
- L'état des dispositifs est cohérent avec celui des contrôleurs (cohérence des capteurs) et des affichages.
  - L'affichage du véhicule est cohérent avec celui du portail.
  - Si le portail est ouvert, le véhicule le perçoit comme tel.
  - etc.
- Lorsqu'une présence est détectée en fermeture, le portail se bloque. Lorsqu'il reprend son mouvement, aucun obstacle ne doit être détecté.

2.5 Hypothèses et dépendances

On ne traite que de la partie logicielle. On ne tient pas compte du fonctionnement en mode manuel, lorsque le contrôleur est déconnecté.

2.6 Exigences reportées

Les versions futures du système proposeront des dispositifs supplémentaires paramétrables à la fois pour chacun des sous-systèmes et leur intégration

- Un détecteur de présence empêche le portail de se refermer en cas d'obstruction.
- Lorsqu'un obstacle est détecté en mouvement, le portail se bloque. Lorsqu'elle reprend son mouvement, aucun obstacle ne doit être détecté.
- Ajout de temporisations (contraintes temporelles). Par exemple, le portail "en attente" se referme au bout 2 minutes…
- Le portail se met en demi-ouverture.
- En cas de rupture de communication avec le véhicule (pas de nouvelle action au bout d'un certain temps), le portail se met automatiquement en mode autonome.
- En cas de dysfonctionnement d'un dispositif, le portail se met automatiquement en mode manuel, par exemple si un capteur n'est plus actif (ne répond plus).

Elles comprendront l'utilisation d'un mécanisme de persistance de données de connexion et d'identification ainsi que différentes interfaces utilisateur : web, IHM classique, etc.
Pour le pilotage et l'utilisation distante, plusieurs modes (et/ou) applications seront proposées, notamment une application fournissant un mode d'accès mobile.
Elles permettront aussi l'accès distant à travers une interface REST pour une connexion avec d'autres dispositifs de domotique (éclairage, alarme…).

3. Exigences spécifiques

3.1 Fonctionnalités

3.1.1 Gestion de la configuration

La gestion de la configuration comprend le paramétrage, le lancement et l'arrêt de l'installation contrôlée.

3.1.2 Gestion des accès

La gestion des accès est l'ensemble des traitements de la base de données relatifs aux usagers, véhicules et télécommandes, et à leur type : saisie, mise à jour, consultation et suppression.

3.1.3 Gestion des mouvements

La gestion des mouvements est l'ensemble des traitements relatifs aux actions contrôlées du portail et du véhicule.

3.1.4 La gestion des Logs

La gestion des logs mémorise l'ensemble des événements relatifs au fonctionnement normal, mais aussi aux données d'accès et aux défaillances et pannes

3.1.5 Monitoring

Le monitoring est l'ensemble des traitements relatifs au pilotage et au suivi des événements.

3.1.6 Gestion des incidents

La gestion des incidents est l'ensemble des traitements de la base de données relatifs aux interruptions de mouvement et aux défaillances et pannes matérielles ou logicielles.

3.1.7 Autres

On place ici divers traitements utilitaires ou secondaires.

3.2 Spécification des cas d'utilisation

Les cas d'utilisation sont décrits implicitement par un document fourni en annexe. Une représentation par cars d'utilisation UML est possible.

3.3 Exigences supplémentaires

3.3.1 Utilisabilité

Aucune exigence d'utilisabilité.

3.3.2 Fiabilité

Le système se met automatiquement en mode manuel en cas de panne ou de dysfonctionnement
Un contrôle de fiabilité externe au contrôleur global est fortement recommandé.

3.3.2.1 Tests unitaires

Chaque méthode appartenant aux interfaces de composants doit être testée par au moins un test unitaire.

3.3.3 Performance

La réponse à une action de la télécommande prend moins de 2 dixièmes de seconde..

3.3.3.1 Fermeture de la porte.

La porte ne peut rester ouverte plus de 15 heures.

3.3.4 Maintenabilité

Le changement de paramétrage ne doit pas bloquer le fonctionnement plus de 10 minutes.

3.3.4.1 Conventions de code Java

Le code source Java doit respecter le style proposé par Google :

- https://google-styleguide.googlecode.com/svn/trunk/javaguide.html

3.3.4.2 Conventions de code Python

Le code source Python doit respecter le style proposé par Google :

- http://google-styleguide.googlecode.com/svn/trunk/pyguide.html

3.3.4.3 Conventions de code Scala

Le code source Scala doit respecter le style proposé par la communauté Scala :

- http://docs.scala-lang.org/style/

3.3.4.4 Conventions de code Ruby

Le code source Ruby doit respecter le style proposé par le guide de style Ruby :

- https://github.com/bbatsov/ruby-style-guide

4. Contraintes de conception

4.1 Langage de spécification

Le langage de modélisation est UML accompagné d'OCL..

4.2 Langage de programmation

EV3 dispose d'un langage visuel pour mettre en œuvre le contrôle d'applications EV3, si ce langage est choisi il faudra bien mettre en évidence les règles de passage de la conception à ce langage.

Dans l'optique d'un développement logiciel nous préconisons une mise en œuvre de l'application dans un des langages de programmation à objets acceptés pour les Lego Minstrome EV3, Lejos (Java pour Lego) ou Python.

4.3 Langage de conception

Les diagrammes utilisés comme support à la conception doivent respecter la norme UML (version > 2.4) et le langage d'expression de contraintes OCL (version > 2.4).

4.4 Outils de construction

L'utilisation de Maven (version > 2.0) est souhaitée. Prévoir les itérations suivantes.

4.5 Outils de développement

L'utilisation d'un environnement de développement (IDE) compatible avec Maven, comme IntelliJ IDEA ou NetBeans, est fortement souvent recommandée, mais dans notre cas, Eclipse est préférable du fait de la présence de plugins associés à Lejos.

4.6 Bibliothèques et composants logiciels

Les tests unitaires doivent utiliser JUnit (version > 5.0) ou son équivalent pour les autres langages de programmation.

5. Sécurité

Le système doit permettre de manipuler la porte en toute sécurité (voir document du cas d'étude pour un échantillon de contraintes extra-fonctionnelles).
Pour le prototype, aucune exigence de sécurité n'est exigée de l'application ni de ses données, mais en prévoir l'obligation pour les versions suivantes.

6. Exigences de documentation utilisateur et d'aide en ligne

Aucune documentation utilisateur n'est demandée.

7. Normes applicables

Les documents du projet seront présentés conformément à la norme imposée pour ce projet (voir le document LS2N:T1.5:NT:A:1.1).
Les documents du projet seront numérotés selon la nomenclature proposée dans le même document LS2N:T1.5:NT:A:1.1.

Aucune norme additionnelle ne s'applique au système dans la version actuelle mais l'intégration aux normes NF et EU sera mise en place ultérieurement.

8. Classification des exigences fonctionnelles

Le tableau suivant fixe l'ordre a priori des exigences fonctionnelles.
Les priorités pourront être revues lors des itérations et annoncées dans les documents de conception.

Code	Fonctionnalité	Priorité
UC1	Lancement, arrêt des sous-systèmes	1
UC1.1	Fonctionnement de base du véhicule	1
UC1.2	Fonctionnement de base du portail	1
UC1.3	Paramétrage des accès, ajout de véhicule, configuration	2
UC2	Lancement, arrêt du système global	1
UC3	Intégration niveau 1	2
UC3.1	Sécurité niveau 1 (tests lumière, présence)	2
UC3.2	Sécurité niveau 1 (tests télécommande)	2
UC4	Monitoring	2
UC4.1	Monitoring - Afficher l'état du système	3
UC4.2	Monitoring - historique	4
UC4.3	Monitoring - filtrage du log	6
UC5.1	Incidents - dysfonctionnement capteurs	5
UC5.2	Incidents - Liste des anomalies	4
UC5.3	Incidents - suivi des pannes	7
UC6.1	Accès concurrents	6
UC6.2	Evénements aléatoires, déclenchement de pannes	6
UC6.3	Gestion distante des usagers	7
UC6.4	Gestion des télécommandes	7
UC6.5	Gestion des véhicules	8

9. Annexes

Sous-système OutDoorGate + Lot sprint précédent (étude de cas, exigences, sources et documentation)
Sous-système RileyRover+ Lot sprint précédent (étude de cas, exigences, sources et documentation)

Description de l'étude de cas OutDoorGate+RileyRover
Exercice 2.8, pages 83-86 du livre :

1. Pascal André and Alain Vailly. Exercices corrigés en UML ; Passeport pour une maîtrise de la notation, volume 5 of Collection Technosup. Editions Ellipses, 2003. ISBN 2-7298-1725-5.

Documentation de projet

- LS2N:T1.5:NT:A:1.1 (norme documents)
- LS2N:T1.1:DC:A:1.0 (sujet de projet)
- LS2N:T1.1:DC:B:1.0 (étude de cas)
- LS2N:T1.1:DC:B:1.0 (Spécification d'exigences logicielles)
- LS2N:T1.4:DD:A:1.0 (dossier de conception générale)