UNIVERSITÉ DE NANTES

Miage

DomoDoor
Spécification d'exigences logicielles

Version 1.0
2017-12-12

Pascal ANDRE

Historique des modifications du document

Date	Version	Description	Auteur
<aaaa-mm-jj>	<x.x>	<détails>	<nom>
modèle	1.0	Projet 2014-2015	Gerson Sunyé, Pascal André
modèle	1.1	Projet 2016-2017	Pascal André

Table des matières

Spécification d'exigences logicielles

1. Introduction

Ce document énumère les exigences du projet « DomoDoor ». 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 « DomoDoor ».
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 Home-Machine
REST - **Representational State Transfer**

1.4 Références

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.

2.1.2 Interfaces utilisateurs

Le système se résume à des composants sans interface utilisateur.

2.1.3 Interfaces matérielles

Aucune interface matérielle n'est exigée.

2.1.4 Interfaces logicielles

Le système doit être divisé en composants disposant d'interfaces logicielles claires et simples.

2.1.5 Interfaces de communication

Le système ne communiquera avec aucun autre système ou serveur autre que les dispositifs physiques et la télécommande.

2.1.6 Contraintes de mémoire

Le système doit pouvoir s'exécuter correctement sur un ordinateur personnel disposant d'un 1Go de mémoire vive.

2.2 Fonctions du produit

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

2.3 Caractéristiques des utilisateurs

Les utilisateurs du système seront des développeurs débutants ou expérimentés.

2.4 Contraintes

Le système doit au moins permettre de manipuler la porte en toute sécurité.

- La porte ne peut être à lafois ouverte totalement et fermée totalement.
- L'état des dispositifs est cohérent avec celui du contrôleur (cohérence des capteurs).
  - Si la porte est ouverte, le contrôleur la perçoit comme tel.
  - Si la porte est ouverte ou fermée, le moteur est à l'arrêt.
  - Si la porte est en ouverture, le moteur est en tirage.
  - Si la porte est en fermeture, le moteur est en poussée.
  - etc.
- En cas de dysfonctionnement d'un dispositif, la porte se met automatiquement en mode manuel, par exemple si un capteur n'est plus actif (ne répond plus).

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

- Un détecteur de présence empêche la porte de se refermer en cas d'obstruction. Lorsqu'un obstacle est détecté en mouvement, la porte se bloque. Lorsqu'elle reprend son mouvement, aucun obstacle ne doit être détecté.
- La porte "en attente" se referme au bout 2 minutes.
- La porte se met en demi-ouverture.

Elles comprendront l'utilisation d'un mécanisme de persistance de données 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 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 de la porte.

3.1.4 La gestion des incidents

La gestion des incidents est l'ensemble des traitements de la base de données relatifs aux défaillances et pannes
Monitoring
Le monitoring est l'ensemble des traitements relatifs au pilotage et au suivi des événements.

3.1.5 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.

3.1.6 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

La spécification est écrite en UML et OCL (voir Langage de conception) et les preuves mises en œuvre avec un outil de vérification OCL.

Dans une itération future, une spécification plus formelle sera développée en Kmelia avec l'outil COSTO pour mettre en place une ligne de produit logiciel sous forme de chaîne de transformation.

4.2 Langage de programmation

De manière générale, l'application doit être mise en œuvre dans un des langages de programmation suivants : Java, Smalltalk, Scala, Groovy, Ruby ou Python.

Dans les premières itérations la mise en œuvre se fait en utilisant l'environnement Lego Mindstorm. :

- LEGO® MINDSTORMS ®R Education EV3 https://education.lego.com/en-us/downloads/mindstorms-ev3
- Les applications seront développées en Java avec Lejos https://lejos.sourceforge.io/ev3.php.

On pourra aussi utiliser
L'appli Appli EV3 Programmer (tablette) ou le logiciel logiciel EV3
https://www.lego.com/fr-fr/mindstorms/about-ev3

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.

4.5 Outils de développement

L'utilisation d'un environnement de développement (IDE) compatible avec Maven, comme IntelliJ IDEA ou NetBeans, est fortement recommandée, mais Eclipse est accepté.

Toutefois on devra choisir un IDE compatible avec avec Lejos https://lejos.sourceforge.io/ev3.php.

4.6 Bibliothèques et composants logiciels

Les tests unitaires doivent utiliser JUnit (version > 4.10) 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

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

Code	Fonctionnalité	Priorité
UC1.1	Lancement	1
UC1.2	Vérifier les conditions de bon fonctionnement	2
UC1.3	Passer en mode manuel	1
UC2.1	Ajout d'un usager	1
UC2.2	Ajout d'une télécommande	1
UC2.3	Suppression d'unusager	3
UC2.4	Suppression d'une télécommande	3
UC3.1	Afficher l'état du système	2
UC3.2	Gérér les mouvements de la porte	2
UC3.3	Afficher l'historique	4
UC3.4	Liste des anomalies	4
UC3.5	Filtrage du log	5
UC4	Gestion des incidents	6
UC5	Gestion des traitements divers	7

9. Annexes

Description de l'étude de cas DomoDoor
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.