UNIVERSITÉ DE NANTES

Miage

Lego-Workshop
Spécification d'exigences logicielles

Version 1.0

2020-12-13

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 2020	Pascal André

Sommaire

1. Introduction 5
   - 1.1 Objectif du document 5
   - 1.2 Portée du document 5
   - 1.3 Définitions, acronymes et abréviations 5
   - 1.4 Références 5
   - 1.5 Vue d'ensemble 5

2. Description générale 5
   - 2.1 Perspectives du produit 5
     - 2.1.1 Interfaces système 5
     - 2.1.2 Interfaces utilisateurs 5
     - 2.1.3 Interfaces matérielles 5
     - 2.1.4 Interfaces logicielles 5
     - 2.1.5 Interfaces de communication 5
     - 2.1.6 Contraintes de mémoire 6
   - 2.2 Fonctions du produit 6
   - 2.3 Caractéristiques des utilisateurs 6
   - 2.4 Contraintes 6
   - 2.5 Hypothèses et dépendances 6
   - 2.6 Exigences reportées 6

3. Exigences spécifiques 7
   - 3.1 Fonctionnalités 7
     - 3.1.1 Gestion de la configuration 7
     - 3.1.2 Gestion des accès 7
     - 3.1.3 Gestion des mouvements 7
     - 3.1.4 La gestion des incidents 7
     - 3.1.5 Gestion des incidents 7
     - 3.1.6 Autres **Erreur ! Signet non défini.**
   - 3.2 Spécification des cas d'utilisation 7
   - 3.3 Exigences supplémentaires 8
     - 3.3.1 Utilisabilité 8
     - 3.3.2 Fiabilité 8
     - 3.3.3 Performance 8
     - 3.3.4 Maintenabilité 8

4. Contraintes de conception 8
   - 4.1 Langage de spécification 8
   - 4.2 Langage de programmation 8
   - 4.3 Langage de conception 9
   - 4.4 Outils de construction 9
   - 4.5 Outils de développement 9
   - 4.6 Bibliothèques et composants logiciels 9

5. Sécurité 9

6. Exigences de documentation utilisateur et d'aide en ligne 9

7. Normes applicables 9

8. Classification des exigences fonctionnelles 9

9. Annexes 11

Spécification d'exigences logicielles

1. Introduction

Ce document énumère les exigences du projet « Lego-Workshop ». 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 « Lego-Workshop ».
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

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 résume à des composants avec une interface utilisateur pour la partie télécommande 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 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.
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 au moins permettre de manipuler les pièces en toute sécurité.

- Chaque poste (trieur de couleur, bras manipulateur…) ne réalise qu'une opération à la fois.
- Les postes exécutent uniquement les ordres d'un seul opérateur auquel ils sont connectés.
- L'état des dispositifs est cohérent avec celui du contrôleur (cohérence des capteurs).
  - Si le poste exécute une opération, le contrôleur la perçoit comme tel.
  - Si le poste est à l'arrêt, tous les moteurs sont à l'arrêt.
  - Si le poste est en panne, tous les moteurs sont à l'arrêt.
  - etc.
- Lorsqu'une présence est détectée, le poste 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

- Un détecteur de présence empêche le poste de toucher un obstacle en cas d'obstruction.
- Lorsqu'un obstacle est détecté en mouvement, le poste se bloque. Lorsqu'elle reprend son mouvement, aucun obstacle ne doit être détecté.
- Ajout de temporisations (contraintes temporelles). Par exemple, le poste "en attente" s'arrête au bout d'un temps en minutes paramétré par l'opérateur.
- Un poste non asservi à un opérateur depuis 5 minutes à la fin de la dernière opération effectuée se met "en attente".
- En cas de rupture de communication avec l'interface de pilotage (pas de nouvelle action au bout d'un certain temps), le poste se met automatiquement en mode manuel.
- En cas de dysfonctionnement d'un dispositif, le poste 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 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

Plusieurs cas d'usage standard sont esquissés qui devront être spécifiés précisément dans l'analyse fonctionnelle. On établira à chaque fois un scénario nominal et des scénarios d'exception. Ces scénarios sont des points d'entrée pour les phases de tests de recette de l'application.

3.1.1 Gestion du mode autonome

Le poste de travail est installé avec un programme qu'il effectue en pleine autonomie. Cela varie en fonction du poste de travail. Par exemple, on trie une pièce de chaque couleur ou on déplace une pièce avec le bras et on la remet à la même place.

3.1.2 Gestion de la configuration

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

3.1.3 Gestion des accès

La gestion des accès est l'ensemble des traitements de la base de données relatifs aux usagers et interfaces de pilotage, et à leur type : saisie, mise à jour, consultation et suppression. Dans une version simple, on demandera seulement un mot de passe, stocké dans un fichier de configuration sécurisé de l'interface de pilotage.

3.1.4 Gestion des opérations

La gestion des opérations est l'ensemble des traitements relatifs aux actions contrôlées du poste vis-à-vis des ordres reçus pour les gammes de fabrication.

3.1.5 Monitoring

Le monitoring est l'ensemble des traitements relatifs au pilotage et au suivi des événements.
La gestion des événements permet de tracer le déroulement les processus.

3.1.6 Gestion des incidents

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

3.1.7 Gestion des accès concurrents

La concurrence est intimement liée au parallélisme d'activité. On prend en compte ici le traitement des verrous d'accès aux ressources et composants de base.

3.1.8 Sécurité

On place ici divers traitements relatifs à la fiabilité, la sécurité des personnes, la sécurité des données et des communications, et à l'évitement de failles de sécurité.

3.1.9 Autres

On place ici divers traitements utilitaires ou secondaires ou dans les perspectives d'évolution.

3.2 Spécification des cas d'utilisation

Les cas d'utilisation sont décrits implicitement par un document fourni en annexe sur le cas d'étude. Une représentation par cas d'utilisation UML est à réaliser. On considèrera les lots de fonctionnalités ci-dessus comme des cas d'utilisation naturels.

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, Léjos (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 le poste 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 UC	Fonctionnalité	Priorité
UC1	Fonctionnement autonome du poste (ou ressource)	
UC1.1	Fonctionnement nominal des postes (trieur, bras articulé)	1
UC1.2	Exceptions	1
UC2	Gestion de la configuration	
UC2.1	Paramétrage, lancement	2
UC2.2	Arrêt des postes de travail (sous-systèmes)	2
UC3	Gestion des accès	
UC3.1	Ajout d'interface de pilotage, accès par mot de passe "câblé"	3
UC3.2	Gestion des accès - un seul opérateur à la fois	3
UC3.3	Configuration des accès (administrateur, opérateur) BD	8
UC3.4	Sécurisation des accès	8
UC4	Gestion des opérations	
UC4.1	Description et lancement d'ordres de fabrications	4
UC4.2	Gestion complète d'une gamme de fabrication	7
UC5	Monitoring	
UC5.1	Informations sur l'état du poste piloté	3
UC5.2	Informations sur l'état du processus	7
UC5.3	Monitoring - Afficher l'état du système	3
UC5.4	Monitoring - historique	5
UC5.6	Monitoring - filtrage du log	7
UC6	Gestion des incidents	
UC6.1	Incidents - dysfonctionnement capteurs	5
UC6.2	Incidents - Liste des anomalies	7
UC6.3	Incidents - suivi des pannes	8
UC6.4	Evénements aléatoires, déclenchement de pannes	9
UC7	Concurrence	
UC6.1	Gestion des accès concurrents	10
UC6.2	Gestion des accès prioritaires	11
UC8	Sécurité	
UC8.1	Sécurité niveau 1 (protection des personnes)	5
UC8.2	Sécurité niveau 2 (accès)	6
UC8.3	Sécurité niveau 3 (communications)	6
UC8.4	Sécurité niveau 4 protection des données	10
UC9	Autres	
UC9.1	intégration des deux postes de travail pilote	20
UC9.2	Gestion d'atelier (intégration des postes)	20
UC9.3	Gestion des interfaces de pilotage multi-postes	30
UC9.4	Gestion des postes de travail	30
UC9.5	Gestion du transport	25
UC9.6	Gestion des gammes de fabrication de l'atelier	40
UC9.7	Reporting de fonctionnement global	50
UC9.8	Gestion distante des usagers (backoffice)	60
UC9.9	Logistique	70

9. Annexes

Description de l'étude de cas Lego-Workshop
Référence : P. André and A. Vailly. : Développement de logiciel avec UML2 et OCL Cours et exercices corrigés, Editions Ellipses - 12/2013, 384 pages, ISBN: 9782729883539

Exemples de modélisations logiques :

- Exercice 2.8, pages 83-86 du livre :
- Exercice 2.8, pages 87-93 du livre :
- Cas d'étude Ascenseur , pages 209-226 du livre :

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)