Feuille de travaux pratiques n^o 1
Projet DomoDoor

Etude de cas

Ce document présente le cas d'étude et une première analyse en UML. On s'intéresse à un système de porte automatique de garage d'un immeuble.

1 Le cas d'étude en bref

Le cas porte sur un équipement domotique simplifié, la gestion d'une porte de garage en considérant deux dispositifs, un *client* sous forme de télécommande ou de smartphone et un *serveur*, le logiciel de contrôle de la porte, qui pilote les différents dispositifs physique. Une description est donnée en annexe.

Il s'agit de concevoir et mettre en œuvre un logiciel de gestion de portes, dont l'analyse du domaine et les spécifications des exigences logicielles (fonctionnelles et non fonctionnelles) sont fournies.

2 Description du fonctionnement et des composants

Ce système comprend une porte qui monte ou descend, un moteur pour actionner la porte (tirer, pousser) et des capteurs pour recueillir des informations (contact porte ouverte, contact porte fermée). Les capteurs sont tous similaires, chacun signale à son contrôleur qu'un contact a eu lieu. L'usager dispose d'une télécommande pour piloter la porte avec simplement deux boutons : ouverture, fermeture.

Le principe de fonctionnement du système est le suivant. Supposons la porte fermée. L'usager enclenche l'ouverture de la porte en appuyant sur le bouton `Ouverture` de sa télécommande. Il peut arrêter l'ouverture en réappuyant sur le bouton `Ouverture`, le moteur s'arrête. Dans le cas contraire la porte s'ouvre complètement et déclenche un capteur `porte ouverte` qui entraîne l'arrêt du moteur. Appuyer sur le bouton `Fermeture` entraîne la fermeture de la porte si elle est ouverte (partiellement ou complètement).

On peut arrêter la fermeture en réappuyant sur le bouton `Fermeture`, le moteur s'arrête. Dans le cas contraire la porte se ferme complètement et déclenche un capteur `porte fermée` qui entraîne l'arrêt du moteur. À tout moment, si quelqu'un appuie sur un bouton d'arrêt d'urgence situé sur le mur alors la porte se bloque. Pour la réactiver (dans le même état) on tourne une clé privée dans une serrure située sur le mur.

2.1 Télécommande

La télécommande dispose de deux boutons, `Ouverture` et `Fermeture`, sur lesquels l'utilisateur peut appuyer. Une pression sur le bouton `Ouverture` déclenchera l'ouverture de la porte ou l'arrêt de cette ouverture. Le choix entre ces deux activités sera fait par le contrôleur en fonction de l'état de la porte. La télécommande, seule, ne peut pas décider de la marche à suivre.

Le comportement du bouton de fermeture est similaire. La télécommande, lorsqu'elle est activée, réagit à deux événements (appui sur le bouton d'ouverture ou appui sur le bouton de fermeture) et, à chaque fois, se *contente* de signaler au contrôleur qu'il y a eu pression sur le bouton. Son comportement est modélisé dans la figure 1. La variable `ctrl` représente le contrôleur.

Les événements `ouverture` et `fermeture` induisent l'exécution, par la télécommande, d'une action du contrôleur. Cela se traduit par l'envoi d'un message au contrôleur pour lui demander d'exécuter son action `ouvrir`.

Dans ce diagramme, appel de procédure (*exécute l'opération ouvrir*) et procédure (`ouvrir`) se mélangent. Nous aurions dû écrire `demandeOuverture / ouvrir()`, la procédure `ouvrir()` étant une opération de la classe `Télécommande`. Cette opération envoyant un message au contrôleur. Nous pouvons alors écrire `demandeOuverture / ctrl.ouvre()` dans la figure 2.

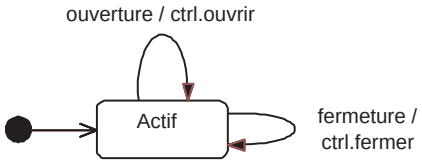

Figure 1 : *Diagramme Etats-transitions de la télécommande - Porte*

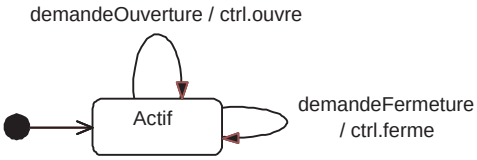

Figure 2 : *Diagramme Etats-transitions de la télécommande - version améliorée - Porte*

2.2 Capteur

Le capteur signale à son contrôleur lorsque le contact a lieu. Il se comporte donc de façon comparable à la télécommande et nous pouvons modéliser son comportement par la figure 3.

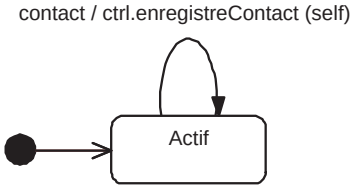

Figure 3 : *Diagramme Etats-transitions de la télécommande - Porte - Capteur*

2.3 Moteur

Le moteur actionne la porte. Soit il pousse la porte, soit il la tire... soit il est à l'arrêt. Nous avons donc trois états, `En poussée`, `En tirée`, `Arrêt`. Le résultat de ces actions est mentionné en filigrane dans le texte. Lorsque le moteur pousse, la porte se referme ; lorsqu'il tire, la porte s'ouvre. Nous supposons que le moteur réalise trois actions : `pousser`, `tirer` et `arrêter`. Ces actions sont exécutées lorsque le contrôleur lui demande de le faire. Son comportement est alors celui de la figure 4.

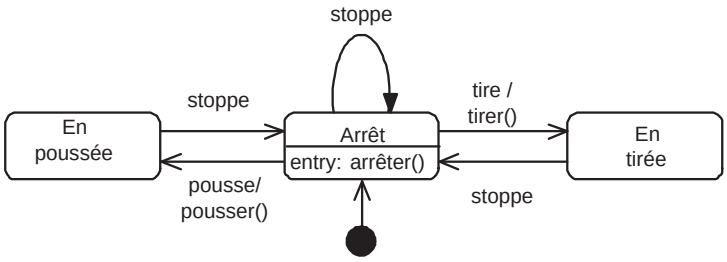

Figure 4 : *Diagramme Etats-transitions du moteur - Porte*

2.4 Porte

La modélisation du comportement de la porte va faire appel à plusieurs notions. Un historique est nécessaire pour gérer la reprise du fonctionnement après blocage. Une porte bloquée à mi-hauteur pendant son ouverture doit, après déblocage, reprendre son mouvement. Au niveau macroscopique, la porte est en service ou `bloquée`.

Lorsqu'elle est en service, elle peut être dans un des trois états suivants : fermée, ouverte ou en mouvement. Elle se bloque dès que le contrôleur le décide. Lors de la reprise, elle reprend son activité là où elle s'était arrêtée. Cet arrêt dure jusqu'à ce qu'une seconde pression sur la télécommande soit détectée, ce qui aura pour effet de relancer le mouvement. La porte peut donc être dans six états distincts, `PorteOuverte`, `PorteFermée`, `PorteBloquée`, `PorteArrêtée`, `EnOuverture` et `EnFermeture`. L'état `Mouvement` est en fait un super-état (il regroupe `EnOuverture`, `PorteArrêtée` et `EnFermeture`) et le diagramme final est un automate hiérarchique. Le comportement de la porte peut être décrit par la figure 5.

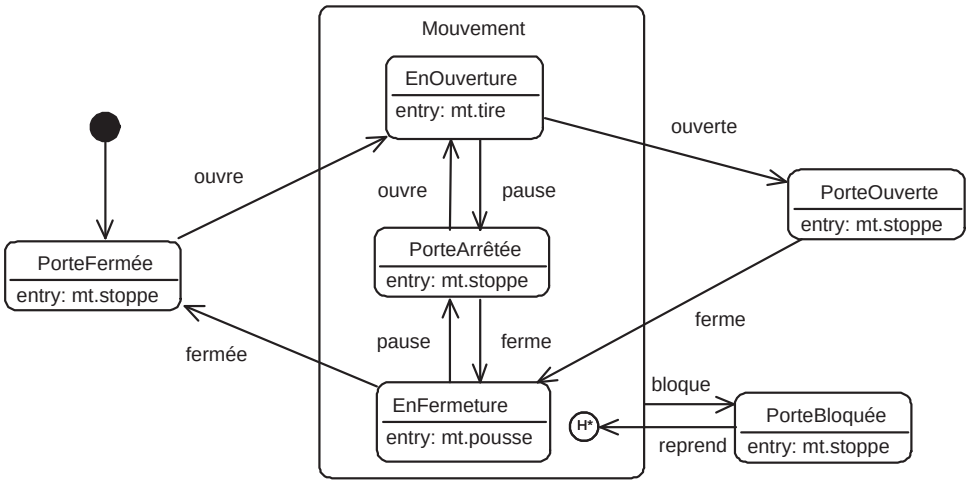

Figure 5 : *Diagramme Etats-transitions de la porte - Porte*

La variable `mt` correspond au moteur qui commande la porte.

2.5 Contrôleur

Le contrôleur reçoit les signaux des capteurs et de la télécommande et commande la porte. Son rôle est d'analyser ces signaux et de décider, en fonction de ceux-ci, ce que doit faire la porte. Pour décrire ceci, nous choisissons d'adopter la même structure que celle utilisée pour décrire le comportement de la porte. Les états du contrôleur correspondent, en fait, aux différentes étapes dans son fonctionnement. Nous avons un état `Mouvement`, un état `Fermée`, un état `PorteOuverte` et un état `Urgence` et, au niveau inférieur, des états `EnOuverture`, `EnFermeture` et `EnAttente`.

Le diagramme d'activités de la figure 6 décrit ce comportement.

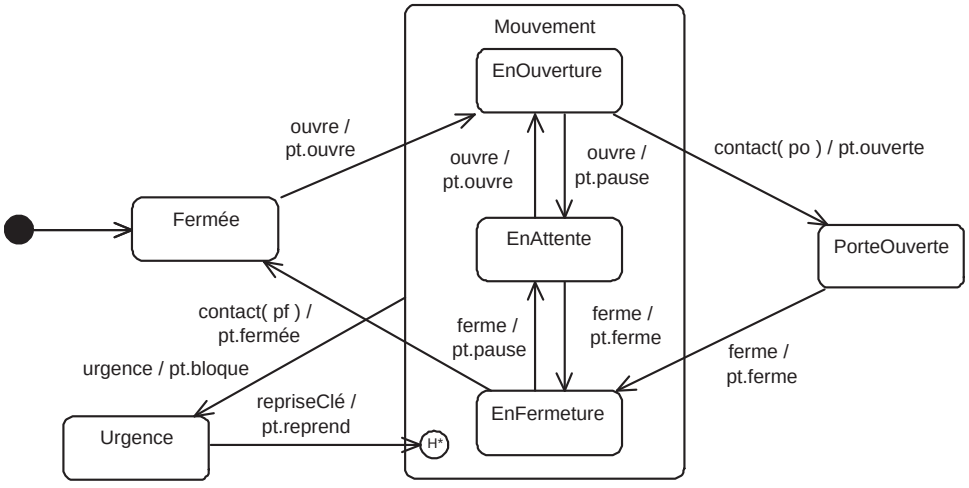

Figure 6 : *Diagramme Etats-transitions du contrôleur - Porte*

La variable `pt` représente la porte.

2.6 Système global

Le système en fonctionnement comprend des dispositifs matériels et logiciels.

2.7 Structure

La structure matérielle est constituée d'un contrôleur, d'un moteur, d'une porte, d'une télécommande et de deux capteurs (un pour détecter la porte ouverte et un pour détecter la porte fermée). Dans une version abstraite du diagramme de classes, figure 7, nous avons uniquement des opérations.

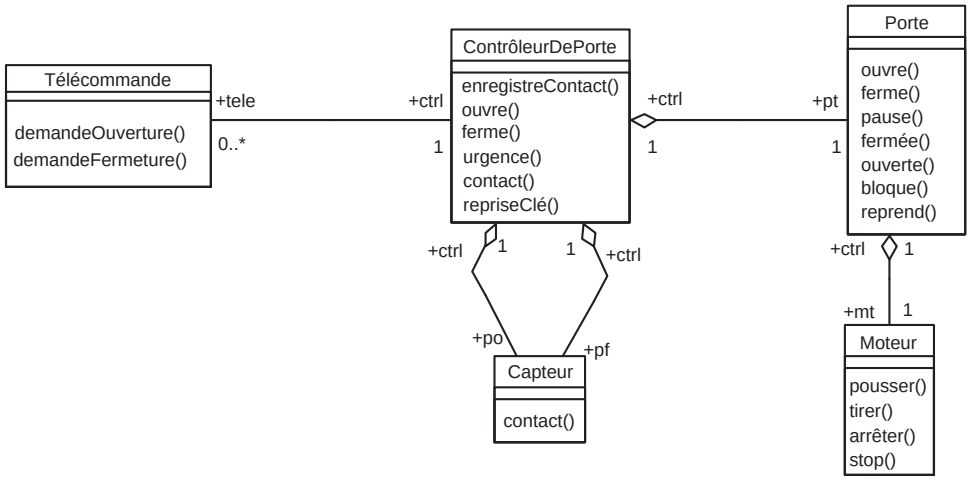

Figure 7 : *Diagramme de classes - Porte*

Nous pourrons, par la suite (durant la phase de conception) ajouter éventuellement un attribut `état` dans chacune des classes ayant un comportement dynamique (i.e. représenté par un diagramme états-transitions), cet attribut servant de support d'enregistrement de l'état courant. Cela risque toutefois de faire double emploi avec un attribut du métamodèle. De la même façon,nous pourrions ajouter un attribut pour gérer l'historique.

2.8 Protocole de communication

Les dispositifs physique échangent avec le contrôleur directement. On supposera un système simple d'échanges de messages en circuit fermé.

La télécommande communique à distance avec le contrôleur pour commander l'ouverture et la fermeture des portes.

On définira un protocole sécurisé vérifiant l'authenticité de la télécommande pour ces échanges.

3 Exigences, contraintes et hypothèses

Nous précisons ici quelques exigences du système.

3.1 Hypothèses et restrictions

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

3.2 Exigences fonctionnelles

Le système doit permettre de manipuler la porte via son contrôleur. Ces exigences fonctionnelles sont classées par priorité.

1. Gestion de configuration : lancement, arrêt.
   (a) Initialiser le mode automatique, paramétrer

i. Vérifier l'état des dispositifs physique (capteurs et actionneurs).
ii. Initialiser chaque dispositif.
iii. Initialiser le mode automatique du contrôleur.
iv. Vérifier la cohérence globale (assertions).

(b) Passer en mode Manuel.

2. Gestion des accès
   (a) Enregistrer un usager pour l'administration du système
   (b) Modifier, supprimer les d'informations d'un usager
   (il y a toujours au moins un administrateur du système).
   (c) Enregistrer une télécommande
   (d) Supprimer une télécommande
3. Monitoring
   (a) Afficher l'état du système.
      i. Etat du contrôleur
      ii. Etat des dispositifs
   (b) Gérer les mouvements
      i. Ouverture partielle ou totale
      ii. Fermeture partielle ou totale
   (c) Gestion d'un historique (Log)
      i. Liste des commandes et des états du contrôleur
      ii. Liste des anomalies et changements de modes (urgence/normal)
4. Gestion des incidents
   (a) Passer en mode Urgence
   (b) Revenir en mode normal
   (c) Prise en compte des interruptions de mouvements et des opérations de reprise.
   (d) Suivi des pannes
5. Divers.

3.3 Extra-Fonctionnelle

Le système doit permettre de manipuler la porte en toute sécurité.
— La porte ne peut-être à la fois 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).

3.4 Evolutions à prévoir

On peut imaginer des facilités 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.

3.5 Exigences techniques

La mise en œuvre se fait en utilisant l'environnement Lego Mindstorm.

LEGO® MINDSTORMS® 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.

A titre d'expérimentation, on pourra aussi utiliser L ?appli Appli EV3 Programmer (tablette) ou le logiciel logiciel EV3 https://www.lego.com/fr-fr/mindstorms/about-ev3.