Dussart Jessy
Larose Quentin

Projet FSM

Contrôleur de machine à boisson

Table des matières

Introduction

Dans ce projet, nous avons implémenté en intégralité la partie MVP du projet en ayant pris quelque liberté d'un point de vue conception sur les points non explicite du sujet. Nous avons ajouté à cela toutes les extensions proposées ce qui comprend donc la soupe, l'iced tea, l'implémentation d'une barre de progression ainsi que la détection des gobelets dans l'éventualité où notre utilisateur voudrait apporter sa propre tasse afin de faire des économies et de sauver la planète.

Implémentation

- MVP

Pour le MVP, nous avons implémenté les points 1 à 17 du sujet en répondant au mieux à la consigne et en apportant notre point de vue personnel lorsque c'était nécessaire sur les parties non couvertes par le sujet.

De plus, pour des raisons évidentes de simulation, les actions avec un temps d'exécution très cours et n'étant pas essentielle pour la partie FSM de notre projet ont été considéré immédiate, le chauffage de l'eau, le versement de l'eau pour les boissons ainsi que le refroidissement des boissons (dans le cas de l'iced tea) ont été entièrement simulé du côté du code métier (le JAVA) et enfin la gestion des stocks à elle aussi été simulé côté JAVA car pour nous avons imaginé que les stocks sont gérés par des capteurs directement dans la machine à café.

Toujours pour des raisons de simulation, le système de fidélité qui apparaît lorsque l'ont payé par carte bleu demandera le « hash » de la carte par le biais d'un pop-up. Si le hash entré n'est pas connu, un nouvel utilisateur sera créé et s'il est connu la fidélisation s'effectuera sur cet utilisateur. Pour faciliter les essais, un utilisateur avec le hash 'test' a été créé au préalable avec 8 commandes déjà effectuées. Si de la monnaie est déjà insérée et l'utilisateur

décide finalement de payer par carte, l'argent fourni par pièce lui sera intégralement rendu.

- Extensions

Pour les extensions, la soupe et l'iced tea sont implémenté, la barre de progression nous montre l'avancement de la FSM étape par étape de manière arbitraire et pour les gobelets, l'utilisateur peut choisir de mettre ou pas ça propre tasse tant que la préparation n'a pas commencé. Une fois la préparation démarrée il n'est plus possible d'interagir avec la machine à café jusqu'à la fin du processus.

Choix de Conception

- Gestion des évents

Dans notre FSM, nous avons choisi d'avoir une exécution CycleBased car nous avons plusieurs états qui se suivent sans pour autant avoir d'évènement entrant pour les déclencher. En effet quand nous avions commencé avec une exécution EventDriven, nous avons remarqué que toutes les transitions avec comme déclencheur « always » ne se déclenchent par conséquent pas et nous nous trouvions avec une machine totalement bloquée.

Ensuite, nous avons décidé de séparer notre FSM en plusieurs régions qui agissent en simultané et envoient des internal event à la région principale qui gère les différentes phases de la machine. Ça nous a donc permis d'alléger notre FSM. Dans ces régions, nous avons placé les éléments de notre FSM qui devaient être réalisés en parallèle du la partie principale de notre FSM. Nous avons donc séparé la sélection des boissons, l'ajout d'argent ainsi que le système de remise à zéro du corps de la machine. Il a fallu faire bien attention à ce que les régions soit implémenté dans le bonne ordre car certaines d'entre elles agissent sur les autres, il faut donc par exemple que les systèmes de remise à zéro s'exécute avant le système de

préparation des boissons des boissons, sinon la remise à zéro sera ignoré par le système.

- Préparation des boissons

Pour représenter la préparation des boissons, nous avions plusieurs possibilités :

- Représenter toutes les boissons et leurs processus de préparation indépendamment les uns des autres, avec donc des duplications d'actions comme « verser l'eau » ou « mettre un gobelet » par exemple qui sont des actions récurrentes pour plusieurs boissons.
- Factoriser au maximum les étapes communes aux différentes préparations afin d'avoir un axe commun à toutes les boissons.

Nous avons opté pour la deuxième solution car on l'a trouvée plus simple et moins lourde du point de vue de la FSM. En plus de ça, elle n'est pas exclusive car nous pouvons très bien rajouter au tout début de l'étape de préparation un nouvel embranchement complètement séparé dans l'éventualité où nous aurions à faire une préparation avec aucune action en commun avec les autres préparations. Nos états orthogonaux nous permettent aussi de faire différentes actions en simultané ce qui est un réel plus étant donné que l'utilisateur doit patienter lors de la préparation de la boisson. Il a fallu rajouter à cela quelques synchronisations pour rassembler les états simultanés. A cause de l'implémentation des synchronisations dans Yakindu, il n'est pas possible de mettre des conditions sur les transitions allant vers une synchronisation. Nous avons donc dû rajouter des étapes de synchronisation « sync » pour pouvoir tester la fin de nos étapes. Pour les options, nous les avons ajoutées dans la partie commune à toutes les préparations de notre FSM et nous avons ajouté quelques conditions pour les exécuter uniquement lors de la préparation des boissons pour lesquelles elles sont disponibles. Cette vérification supplémentaire n'est cependant pas essentielle car notre code métier JAVA bloque la sélection des options non disponibles pour la boisson actuellement sélectionnée.

V &V

Notre travail en V&V a été de réaliser notre FSM avec l'outil LTSA. Pour ce faire, nous avons décomposé notre FSM en différents fichiers LTSA :

drinkManagement.lts

paymentManagement.lts

machineManagement.lts

step2.lts

step3.lts

step4.lts

ActivityManagement.lts

L'activityManagement permet d'envoyer des internal events à notre région principale, c'est pourquoi notre fichier.lts est composé de peu d'états. En effet, on retrouve une transition sur un même état qui représente l'userAction et qui remet à jour le compteur de secondes, et une transition resetMachineInternal pour reset les données de la commande si aucune action n'est faite au bout de 45 secondes.

Malheureusement, l'ActivityManagement ne dispose pas d'assez d'états et de transitions pour effectuer des vérifications de propriétés.

PaymentManagement.lts

De même la PaymentManagement ne dispose que d'un état et d'une transition étant donné qu'elle ne fait qu'émettre des internal events. Un état Payment ainsi qu'une transition addMoney lorsque l'utilisateur ajoute de l'argent.

MachineManagement.lts

Le fichier MachineManagemen représente l'exécution de la partie principale de la FSM en ne détaillant pas les étapes de la préparation.

Step1.lts

Le fichier step2 permet de représenter l'étape 1 de la préparation qui comprend le chauffage de l'eau ainsi que le placement du coffee pod, du tea bag ou du gobelet en fonction de la boisson.

Step2.lts

Le fichier step2 permet de représenter l'étape 2 de préparation des boissons qui consiste en l'ajout des ingrédients des recettes et l'ajout de gobelets si aucun récipient n'est placé. Les propriétés implémentées dans ce fichier vérifient la liveness, on vérifie que l'on arrive bien dans l'état désiré une fois une action faite. De plus, la safety est également vérifiée puisqu'on s'assure que l'on n'arrive pas dans un état où une transition n'est pas censée arriver. Par exemple la transition step2Expresso est censée nous amener sur la transition putBeans, on vérifie cela et la liveness est vérifiée, cependant, pour vérifier la safety, il faut s'assurer que la step2Cofee n'arrive pas sur la transition putBeans.

Step3.lts

Dans ce fichier également, les propriétés vérifient à la fois la safety et la liveness. On vérifie bien que les transitions mènent uniquement aux transitions auxquelles elles devraient accéder et vice-versa, les conditions sont testées pour qu'elles ne mènent pas aux transitions inaccessibles depuis leurs états. L'étape représentée est l'ajout de sucre et différentes étapes propres à chaque boisson comme le retrait du sachet de thé pour les thés.

Step4.lts

Enfin le fichier step4 possède lui aussi des propriétés vérifiant la safety et la liveness, pour s'assurer que c'est le cas on pose donc des questions de safety et liveness au système. Ce fichier correspond à l'ouverture et la fermeture des portes de sécurité pour l'iced tea mais également à l'ajout de lait puis enfin à la récupération du récipient.

Prise de recul

Nous n'avons pas rencontré de réel problème lors de la réalisation de notre FSM mis à part la fonction de Yakindu sur la gestion des événements et sur le fonctionnement des synchronisations. Bien que nos choix ne puissent pas être qualifiés de meilleurs que d'autres, ils sont toutefois tous cohérents par rapport aux autres et notre machine correspond au comportement voulu dans les exigences du sujet. Plusieurs points auraient pu être fait différemment mais on ne peut pas dire qu'ils y auraient eu un réel apport pour notre machine.

Il aurait toutefois pu être intéressant de trouver un moyen de faire avancer la barre de progression de manière plus progressive en fonction du temps que prennent les différentes actions plutôt que de lui faire faire des sauts en fonction de l'étape en cours comme c'est le cas actuellement.