Dussart Jessy
Larose Quentin

Projet FSM

Contrôleur de machine à boisson

Introduction

Ce rapport qui fait lieu de notre projet réalisé dans le cadre du module FSM est décomposé en plusieurs parties. Tout d'abord nous parlerons des différents éléments implémentés au sein de notre projet, le MVP et les différentes options implémentées, ensuite nous parlerons des choix de conception effectués au sein de notre et de notre FSM. Enfin, nous finirons par présenter notre produit V&V ainsi que les différentes pistes d'exploration possibles vis-à-vis de notre projet.

Membres de l'équipe

Les membres de l'équipe sont représentés de la manière suivante : (Nom Prénom)

- Dussart Jessy
- Larose Quentin

Éléments implémentés (MVP + Extensions)

MVP

Dans cette partie nous verrons les différents éléments implémentés dans notre projet. Tout d'abord, le MVP et ses fonctionnalités ont été implémentées comme suivent :

- L'utilisateur peut choisir de payer avant ou après la sélection de la boisson.

- Sans action utilisateur au bout de 45 secondes, le choix préalable de boisson est abandonné. L'utilisateur peut également annuler sa commande à tout moment avant la préparation. Dans ce cas, le paiement est abandonné et l'utilisateur est remboursé si le montant était en liquide.

- Si le paiement est effectué avant, alors sans action utilisateur au bout de 45 secondes le paiement est abandonné.

- Les choix de paramètres de la boisson sont également réinitialisés après 45 secondes sans action de la part de l'utilisateur.

- L'utilisateur peut payer par NFC, ses informations sont sauvegardées et son 11ème achat est gratuit dans la limite de la valeur moyenne des 10 premiers achats.

- Les doses pour chaque ingrédient sont stockées et gérées pour chaque boisson.

- Les différents paramètres de boisson (sugar/spices, size, temperature) sont disponibles pour l'utilisateur.

Boissons et options de base

Les différentes boissons et leurs options de base ont également été implémentées avec leurs prix respectifs comme suit (cf. Annexes) :

Option (Prix) Drink (Prix)	Nuage de lait (+0.10€)	Sirop d'érable (+0.10€)	Glace vanille mixée (+0.60€)
Café (0.35€)	Compatible	Compatible	Compatible
Expresso (0.50€)	Compatible	Compatible	Compatible
Thé (0.40€)	Compatible	Compatible	Incompatible

Les cases vertes représentent les boissons pour lesquelles l'option est disponible et implémentée et les cases rouges représentent l'incompatibilité entre la boisson et l'option qui n'est donc pas sélectionnable pour la boisson désirée.

Extensions

Dans ce projet, nous avons pu implémentées toutes les extensions proposées, parmi celles-ci nous retrouvons :

- Gestion de la soupe
- Gestion de l'iced tea
- Gestion de l'avancement de la préparation
- Détection des gobelets

Extensions boissons et options

Viennent donc s'ajouter les boissons suivantes avec leurs prix et options respectives représentées comme précédemment :

Option (Prix) Drink (Prix)	Croûtons (+0.30€)	Sirop d'érable (+0.10€)
Soupe (0.75€)	Compatible	Incompatible
Iced Tea (0.50€ -> 0.75€)	Incompatible	Compatible

Les cases vertes représentent les boissons pour lesquelles l'option est disponible et implémentée et les cases rouges représentent l'incompatibilité entre la boisson et l'option qui n'est donc pas sélectionnable pour la boisson désirée.

Extension Soupe

L'extension de la soupe permet de sélectionner une nouvelle boisson appelée "Soupe" au prix de 75 centimes avec une possibilité d'option "Croûtons" au prix de 30 centimes. Le choix de la boisson soupe sur notre produit implique différents changements, l'utilisateur ne choisit plus une quantité de sucre mais désormais un niveau d'assaisonnement d'épices allant de 0 (Pas assaisonné) à 4 (très assaisonné). Cette option est, tout comme les options de base, dépendante des stocks disponibles dans la machine.

Extension Iced Tea

L'extension Iced Tea propose une boisson "Iced Tea" au prix de 50 centimes si la taille sélectionnée est "medium", cependant, le prix augmente à 75 centimes si la taille désirée est dite "grande". Pour cette boisson, comme pour toutes les boissons pouvant être sucrées, une option "Sirop d'érable" au prix de 10 centimes est disponible et remplace la quantité de sucre à choisir. De plus ce n'est plus une température de type chaleur qui est à choisir mais plutôt un niveau de "fraicheur" de la boisson variant de 20°C (moins frais) à 3°C (plus frais).

Extension gestion de l'avancement de la préparation

Une barre de progression est mise à disposition de l'utilisateur pour qu'il puisse suivre la progression lors de la préparation de sa commande (pas avant ni après la préparation donc). A chaque étape d'une boisson, la barre de progression augmente d'un certain montant. Par exemple, lorsque l'eau a fini de chauffer, la barre de progression va augmenter, etc. La progression est donc unique pour chaque préparation et va varier en fonction de la boisson et des options choisies.

Extension détection des gobelets

Pour lutter contre le gaspillage, une option permettant à l'utilisateur d'utiliser son propre récipient est disponible. En effet, pour inciter l'utilisateur à être éco-responsable, la machine et notre extension offrent même une réduction de 10 centimes sur la commande en cours. Cela résulte également par la non utilisation d'un gobelet polluant lors de la réalisation des boissons. L'utilisateur peut mettre et récupérer son récipient autant qu'il le veut tant que la commande n'est pas en préparation. Une fois en phase de préparation l'utilisateur ne peut plus récupérer son récipient, il lui faut attendre la fin de la préparation pour le faire.

Panel final des boissons et options après ajout des extensions

Option (Prix) Drink (Prix)	Nuage de lait (+0.10€)	Sirop d'érable (+0.10€)	Glace vanille mixée (+0.60€)	Croûtons (+0.30€)
Café (0.35€)	Compatible	Compatible	Compatible	Incompatible
Expresso (0.50€)	Compatible	Compatible	Compatible	Incompatible
Thé (0.40€)	Compatible	Compatible	Incompatible	Incompatible
Soupe (0.75€)	Incompatible	Incompatible	Incompatible	Compatible
Iced Tea (0.50€ -> 0.75€)	Incompatible	Compatible	Incompatible	Incompatible

Les cases vertes représentent les boissons pour lesquelles l'option est disponible et implémentée et les cases rouges représentent l'incompatibilité entre la boisson et l'option qui n'est donc pas sélectionnable pour la boisson désirée.

Choix de conception

Nous allons désormais aborder les différents choix de conception que nous avons réalisés au sein de notre projet, tout d'abord nous verrons les choix de conception qui ont été faits à travers notre code puis les choix faits pour notre FSM.

Code

Dans cette première partie nous aborderons les choix faits dans le code pour fournir certaines fonctionnalités de telle ou telle manière.

Paiement NFC

Un des choix que nous avons fait dans notre code et qui ne figure donc pas sur la FSM est que le paiement NFC est prioritaire sur le paiement en liquide. Par exemple, si l'on insère de l'argent en liquide puis que l'on paie par NFC, l'intégralité de l'argent inséré en liquide sera rendue et on débitera donc l'argent de la commande sur le compte du client.

Gestion récipient

Un autre des choix que nous avons fait est de laisser l'utilisateur placer son récipient (cf. Extension détection des gobelets) ou de le retirer librement tant que la boisson n'est pas en phase de préparation ou de nettoyage. En effet, nous pensons que cela fait plus réaliste et que l'utilisateur ne veut pas être restreint à placer sa tasse et être bloqué, l'erreur est humaine et donc nous adaptons notre système en conséquence.

FSM

Gestion des events

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 avions remarqué que toutes les transitions avec comme déclencheur « always » ne se déclenchaient 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 réalisé 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 certaine 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 unes 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 la trouvait 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 rajouté 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.

Implémentation de la progressBar

En ce qui concerne notre progressBar, nous l'utilisions au début dans le code et la faisions progresser en fonction des appels de fonction, cependant cela donnait beaucoup de responsabilité au code et peu de représentation au travers de la FSM. Nous avons donc préféré faire progresser la barre à l'intérieur de la préparation des boissons et suivre la progression à l'intérieur de cette région.

Une autre solution possible aurait été de calculer le temps de réalisation d'une dite boisson et de ses options, et de faire progresser la barre en temps réel en fonction du temps écoulé depuis le début de la préparation et du temps total nécessaire à la réalisation de celle-ci. Cependant, cette solution repose sur un calcul de temps dépendant de plusieurs éléments et pour chaque calcul, il aurait fallu savoir le temps nécessaire à chaque boisson et option. C'est pourquoi nous pensons que notre solution est plus simpliste et pas forcément moins bonne puisque la barre de progression affiche clairement les différentes étapes de la préparation de la commande. Un des inconvénients de notre solution de progression par étape est peut-être l'attente nécessaire à la mise à jour de la progression, en effet, pour que la barre de progression s'actualise il faut attendre la fin de l'étape courante ce qui peut être un bémol comparé à la solution vue précédemment et fonctionnant en temps réel.

Présentation 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 donnée 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 gobelet 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 et pistes d'explorations

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 réalisés différemment mais on ne peut pas dire qu'il y aurait 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.

Enfin, même si cela n'était pas demandé, nous aurions pu fournir une architecture plus propre pour discerner les différentes responsabilités des éléments du projet et mieux découper ce dernier.