Devoir de spécification et validation du logiciel

Université Lille1 — Master 1 informatique, S2
UFR IEEA — Première session SVL -3h- 2010-2011 — FIL

Durée totale : **3h**. Tous documents autorisés. Nombre de pages : **3**.
Barème indicatif : moitié test, moitié model-checking.

Exercice 1 : Test

On souhaite développer et tester un jeu de société pour enfants : le « voleur de carottes », marque Haba. C'est un jeu coopératif qui peut se jouer avec 1 à 6 joueurs : pour simplifier on jouera avec un seul joueur et on laissera de côté certains détails.

Le jeu se joue avec un plateau de jeu, une figurine de lapin, 7 figurines de carottes et un dé à symboles. Le but du jeu est d'avoir récolté 4 carottes avant que le lapin en ait volé 4. Le plateau de jeu contient 12 cases disposées en anneau. L'une des cases représente un lapin. Chaque case est percée d'un ou deux trous, chaque trou permettant de piquer une carotte dans le plateau. Chaque case est réservée soit à la plantation des carottes, soit à la récolte des carottes quand elles ont poussé [1]. On ne s'intéressera pas ici à la répartition des cases « pour plantation » ou « pour récolte » sur l'anneau.

Initialement, on place le lapin sur la case qui le représente sur le plateau. Les carottes sont placées dans la réserve du joueur, prêtes à être plantées.

À chaque tour de jeu, le joueur lance le dé, et applique l'action correspondant au symbole indiqué par le dé :

carotte (2 faces sur le dé) Si la réserve n'est pas vide, alors le joueur prend une carotte de la réserve, et la plante dans n'importe quel trou libre d'une case « pour plantation » (un trou n'est pas libre si le lapin est dessus, ou si une carotte est déjà plantée dedans). Si la réserve est vide, alors le joueur récolte n'importe quelle carotte d'une case « pour récolte » de son choix, et la met dans son tas de récolte.

un lapin Le joueur avance le lapin d'une case dans le sens des aiguilles d'une montre. Si le lapin arrive sur une case contenant une ou deux carottes, il les vole ! On enlève les carottes du plateau et on les met dans la récolte du lapin.

deux lapins Le joueur avance le lapin de 2 cases. Le lapin vole toutes les carottes des 2 cases sur lesquelles il avance.

arrosoir Le joueur choisit n'importe quelle carotte d'une case « pour plantation » pour la mettre dans n'importe quel trou visible d'une case « pour récolte ». S'il n'y a pas de carotte dans les cases « pour plantation », on ne fait rien.

jardinier Le joueur peut décider de ce qu'il va faire :
- s'il reste des carottes dans la réserve, il a le choix entre planter et récolter une carotte ;
- sinon il a le choix entre arroser une carotte ou en récolter une.

Le jeu se termine quand l'une des récoltes (celle du gagnant : joueur ou lapin) atteint 4 carottes.

Vous pouvez répondre aux questions dans l'ordre qui vous convient. À vous d'écrire les tests les plus simples possibles en réfléchissant notamment aux objets du domaine, à la répartition des responsabilités et à l'architecture de votre application. Tous les tests demandés sont des **tests unitaires en isolation écrits en JUnit4**, qui utilisent Mockito quand c'est nécessaire. **La présentation devra être soignée**.

Q 1.1 : Donner, à l'exclusion de toute autre chose :
- un diagramme UML de votre application listant les signatures des méthodes pour chaque classe ou interface Java utilisée **dans vos tests** (optique TDD).
- des cas de tests pour les fonctionnalités ou scénarios suivants, regroupés dans les classes de test qui vous semblent pertinentes au regard de votre conception :
  1. test du comportement d'un tour de jeu ;
  2. test du critère de la fin du jeu ;
  3. test de l'organisation physique du plateau en anneau ;
  4. test de l'application du symbole « un lapin » ;
  5. test de l'application du symbole « carotte ».

□

1. Une répartition en deux plate-bandes et un code couleur non abordés dans ce sujet permettent de s'y retrouver graphiquement. De plus les trous « pour plantation » sont larges et laissent peu dépasser la carotte, les trous « pour récolte » sont plus petits et donnent l'impression que la carotte a poussé.

Exercice 2 : Model-checking

On s'intéresse aux propriétés d'un site de commande de pizzas en ligne. L'enchaînement standard est une boucle autour des écrans html suivants : page d'accueil, page de choix de la taille de la pizza, page de choix des ingrédients, page de paiement. On considère un site de commande de base ($site_1$) et 2 variantes ($site_2$ et $site_3$).

On utilisera les variables propositionnelles de l'ensemble $\mathcal{P} = \{$ `accueil`, `taille`, `ingredients`, `paiement` $\}$, qui représentent respectivement la page d'accueil, de choix de la taille, de choix des ingrédients, et de paiement. Vous pouvez utiliser aussi une version courte style $\mathcal{P} = \{$ `a`, `t`, `i`, `p` $\}$ du moment que ça reste clair.

Les sites à étudier sont les suivants :

$site_1$ Ce site de base possède un unique comportement : il enchaîne la page d'accueil, puis celle de choix de la taille, puis celle de choix des ingrédients qui permet de choisir tous les ingrédients d'un coup, puis celle de paiement, avant de revenir à celle d'accueil.

$site_2$ **Même fonctionnement que** $site_1$, mais la page de choix des ingrédients est rechargée pour chaque ingrédient choisi : on enchaîne autant de fois cette page que nécessaire.

$site_3$ **Même fonctionnement que** $site_2$, mais il est toujours possible de revenir à la page précédente, sauf depuis la page d'accueil.

Dans les modèles de site, vous annoterez chaque état avec l'ensemble des variables propositionnelles de $\mathcal{P}$ vraies en cet état, les autres étant par convention fausses dans cet état. Comme pour les trois sites chaque variable est vraie dans un unique état, vous n'avez pas besoin de donner un nom aux états. Par exemple, l'état dans lequel `accueil` est vrai peut s'appeler `accueil`.

Les questions 2.2, 2.3, 2.5 et 2.7 dépendent de la question 2.1.

Q 2. 1 : Donner trois automates $\mathcal{A}_1$, $\mathcal{A}_2$ et $\mathcal{A}_3$ qui modélisent respectivement les comportements des sites $site_1$, $site_2$ et $site_3$. □

Q 2. 2 : Donner un processus Spin qui a le même comportement que $\mathcal{A}_1$. □

Q 2. 3 : Donner :
- l'unique exécution de $\mathcal{A}_1$;
- deux exécutions de $\mathcal{A}_2$ qui n'appartiennent pas à $\mathcal{A}_1$;
- deux exécutions de $\mathcal{A}_3$ qui n'appartiennent ni à $\mathcal{A}_1$, ni à $\mathcal{A}_2$.

□

Q 2. 4 : Exprimer en LTL les propriétés suivantes :
- ϕ_1 : le site propose initialement une page d'accueil ;
- ϕ_2 : toute page de choix de taille sera immédiatement suivie d'une page de choix d'ingrédients ;
- ϕ_3 : il est faux que toute page de choix d'ingrédient sera suivie par une page de paiement ;
- ϕ_4 : le site revient infiniment souvent à la page d'accueil ;
- ϕ_5 : le site restera infiniment longtemps sur la page de choix d'ingrédients ;
- ϕ_6 : le site permettra de choisir des ingrédients ;
- ϕ_7 : on ne peut jamais choisir à la fois la taille et les ingrédients de la pizza ;
- ϕ_8 : une fois que le site a permis de choisir des ingrédients, il permet de payer, et entre-temps ne permet que de choisir des ingrédients.

□

Q 2.5 : Pour chaque formule donnée à la question 2.4 et chaque modèle de site, dites si l'automate satisfait la formule et justifier **brièvement** votre réponse [2]. Même si vous n'êtes pas sûr de votre automate, vous pouvez intuitivement deviner la réponse à partir de la description informelle des sites. □

2. Si votre réponse est positive, donner juste assez d'explications pour me faire comprendre que vous n'avez pas répondu au hasard. Attention à ne pas perdre trop de temps en explications inutiles.

Q 2.6 : Exprimer en CTL* les formules suivantes :
- ϕ_9 : à n'importe quel moment il est possible d'atteindre la page de paiement ;
- ϕ_{10} : il est possible que le site finisse par rester infiniment longtemps sur la page de choix d'ingrédients.

□

Q 2.7 : Pour chaque formule donnée à la question 2.6 et chaque modèle de site, dites si l'automate satisfait la formule et justifier **brièvement** votre réponse. □

Q 2.8 : Parmi les formules ϕ_1 à ϕ_{10}, donner :
- une formule de sûreté ;
- une formule de vivacité ;
- une formule d'atteignabilité ;
- une formule d'équité.

□