GLO-3004 Spécification et vérification formelle de logiciels **Hiver 2018**

Projet 1 : étape 2 conception et développement en Java (15%) : *manufacture de confiture*

À remettre avant 23h55, le mercredi **28 mars 2018**. **Travail en équipe de 2 à 4**.

Il s'agit de concevoir et développer en Java le mini-système de contrôle dans une confiturerie selon la question 9, la plus complète, de l'énoncé de l'étape 1, annexe A. Une spécification suggérée est fournie.

La première étape consistait en partie à spécifier en FSP ce système et la deuxième étape consiste à le développer en Java.

Ce cours va faire partie des cours éligibles au prix Pierre Ardouin. L'équipe sélectionnée dans ce cours gagne le prix de ce cours et fera partie d'une autre sélection avec les équipes gagnantes des autres cours. Un comité choisira alors le meilleur travail et c'est ce dernier qui gagnera le prix Pierre Ardouin.

Pour être sélectionnée comme équipe gagnante, l'équipe doit en premier lieu répondre à toutes les exigences demandées dans les 2 énoncés successifs mais aussi d'aller un peu plus loin dans la réalisation que ça soit en spécifiant et développant d'autres propriétés que celles demandées ou bien en réalisant un développement qui intègre une interface personne-machine, par exemple ou d'autres fonctionnalités. Une idée d'extension serait par exemple d'ajouter un autre traitement que subissent les bocaux de confiture comme le nettoyage à haute température avant l'usage ou bien de différencier le traitement subi par chaque type de bocal ou toute autre idée qui rendrait le système plus réaliste.

Les équipes peuvent prendre 100% des notes sans toutefois être choisies. Il faut que l'équipe se démarque par une réalisation pour pouvoir être choisie.

Travail demandé

Livrable 1 (90 points) Il s'agit de développer en Java la spécification de la question 9 que vous avez effectuée durant l'étape 1. La spécification **doit être développée en utilisant les threads de Java pour représenter les processus parallèles spécifiés en FSP**. Il faudrait pouvoir modifier les paramètres déterminant le nombre de bocaux. Aussi, les priorités peuvent être associés aux threads en Java, comme dans certains exemples illustrant la famine fournis au début de la session. Vous devez afficher d'une façon ou d'une autre, dans la console, ou dans l'interface, la trace d'exécution. Le développement doit obligatoirement s'effectuer sous l'IDE Eclipse.

Livrable 2 (10 points) Il s'agit d'un fichier texte **LisezMoi** expliquant comment on exécute votre code, comment on effectue les simulations, comment interpréter les sorties, s'il n'y a pas d'interface personne-machine, et comment on modifie les paramètres.

La répartition des notes n'est donnée qu'à titre indicatif. C'est la cohérence globale du système et sa conformité par rapport aux besoins et aux spécifications qui sont évaluées ainsi que la clarté du code et la facilité de son exécution.

Consignes à suivre obligatoirement pour la remise

- Vous devez fournir un fichier .zip dans lequel vous donnez le ou les fichiers de votre développement Java. Ces fichiers doivent comporter **des commentaires aidant la compréhension et faisant le lien avec les spécifications**. Il faut livrer **obligatoirement** le fichier **LisezMoi** dont le contenu est spécifié ci-dessus.
- Ne m'envoyez pas vos fichiers par courriel. Sauf force majeure.

Autres contraintes : Vous devez respecter les contraintes suivantes.

- Ce travail se fait obligatoirement en équipe. Un questionnaire permettant de faire l'appréciation de la participation au travail d'équipe sera utilisé.
- Vous devez prendre toutes les mesures nécessaires pour éviter le plagiat. Si vous proposez des solutions trop semblables à celles d'une autre équipe ou à des solutions trouvées sur internet, des sanctions pourraient être prises conformément au règlement départemental et de l'Université Laval.

Bon travail.

A Énoncé étape 1

Il s'agit de spécifier en FSP un mini-système de contrôle dans une confiturerie. Ce système doit permettre l'automatisation du remplissage de bocaux par de la confiture puis leur étiquetage.

Les bocaux circulent sur des tapis roulants, quand un bocal arrive au niveau d'une valve celle-ci s'ouvre et remplit le bocal de confiture et un mécanisme le ferme puis le bocal est étiqueté. Pour simplifier, nous considérons que le remplissage et la fermeture du bocal se font en une seule action. Évidemment, le bocal ne peut se remplir que si la valve s'ouvre, et l'étiquette ne peut être collée que si le mécanisme d'étiquetage est prêt. Ainsi les processus `Bocal, Valve` et `Etiquetage` sont modélisés comme suit. Le système élémentaire est spécifié par le processus `Confiturerie0`.

```
const N = 3
range R = 1..N
range R0 = 0..N
set  S = {a,b}
const  T = 2
range RT = 1..T
const V = 2
range RV = 1..V
const E = 2
range RE = 1..E

Bocal=  ( ouvreValve -> remplit  ->  fermeValve ->
          commenceEtiquetage -> etiquette -> termineEtiquetage -> Bocal).

Valve= (ouvreValve -> fermeValve  -> Valve).

Etiquetage = (commenceEtiquetage -> termineEtiquetage -> Etiquetage).

||Confiturerie0 = (Bocal  ||  Valve || Etiquetage).
```

La spécification de ce système va s'effectuer progressivement à travers les questions de ce travail. Notez que la suggestion de noms des processus n'est pas obligatoire, mais cela aiderait énormément durant la phase de correction. Avant chaque partie de la spécification, précisez clairement dans un commentaire en entête, de cette partie, à quelle question vous êtes en train de répondre. Même ci ce n'est pas précisé, dans chacune des questions vous pouvez adapter des processus déjà spécifiés pour les besoins de la question, il vaut mieux dans ce cas nommer différemment ces processus modifiés pour que le tout puisse se compiler aisément.

Travail demandé

Q1 (5 points) Donnez la spécification en FSP d'un processus `Confiturerie1` qui représente la mise en parallèle de deux types de bocaux `a` et `b` qui se partagent les ressources `Valve` et `Etiquetage`. Ces dernières ne peuvent servir qu'un bocal à la fois. Vérifier l'absence de blocage et le progrès.

Q2 (5 points) Donnez la spécification en FSP d'un processus `Confiturerie2` qui représente la mise en parallèle de `N` bocaux de chacun des types `a` et `b` qui se partagent toujours les ressources `Valve` et `Etiquetage`. Ces dernières ne peuvent servir qu'un bocal à la fois. Vérifier l'absence de blocage et le progrès.

Q3 (15 points) Dans cette question, nous supposons que nous avons `V` valves et `E` mécanismes d'étiquetage et que nous avons toujours `N` processus de chacun des types de bocaux `a` et `b`. Spécifiez deux propriétés `PValve` et `PEtiquetage` qui vérifient qu'il y a au maximum `V` bocaux qui utilise des valves en même temps et `E` bocaux qui utilisent le mécanisme d'étiquetage en même temps. Spécifiez un processus `Confiturerie3` qui consiste en la composition parallèle de `N` processus de chacun des types de bocaux `a` et `b` et des deux propriétés `PValve` et `PEtiquetage`.

Vérifier si les propriétés `PValve` et `PEtiquetage` sont satisfaites ou non et expliquez le résultat.

Q4 (10 points) En vous inspirant des propriétés spécifiées dans la question précédente, spécifiez en FSP deux processus `CValve` et `CEtiquetage` dont l'objectif est de contrôler le nombre de bocaux utilisant chacune des ressources, valve et mécanisme d'étiquetage. Spécifiez un processus `Confiturerie4` qui consiste en la composition parallèle de `N` processus de chacun des types de bocaux `a` et `b` , des deux propriétés `PValve` et `PEtiquetage` et des deux processus `CValve` et `CEtiquetage` . Vérifier si les propriétés `PValve` et `PEtiquetage` sont satisfaites ou non et expliquez le résultat.

Q5 (15 points) Nous voulons nous assurer que les bocaux d'un même type sont servis dans l'ordre d'arrivée, premier arrivé, premier servi. Pour ce faire, spécifiez 8 processus qui garantissent pour chaque type que, l'ouverture de la valve pour les processus s'effectue sans dépassement c'est-à-dire dans l'ordre 1,2,... `N`, 1, 2, ...et de la même manière la fermeture de la valve, le début et la fin d'étiquetage. Inspirez-vous de l'exemple du pont à une seule voie pour spécifier 8 processus garantissant cet ordre de fonctionnement pour les bocaux d'un même type. Appelez ces processus, `PasDepasseOuvreValveA` , `PasDepasseFermeValveA` , `PasDepasseOuvreValveB`, `PasDepasseFermeValveB`, `PasDepasseCommenceEtiquetageA` , `PasDepasseTermineEtiquetageA`, `PasDepasseCommenceEtiquetageB` et `PasDepasseTermineEtiquetageB`

Spécifiez un processus `Confiturerie5` qui compose les mêmes processus que `Confiturerie4` en plus de ceux spécifiés dans cette question.

Vérifier si les propriétés `PValve` et `PEtiquetage` sont satisfaites ou non.

Q6 (5 points) Cette question a pour objectif de tester la robustesse du système. Pour ce faire rajouter des priorités de cette façon

```
||Confiturerie6 =
       Confiturerie5 << {[i:R].a.{ouvreValve, remplit, fermeValve,
        commenceEtiquetage, etiquette, termineEtiquetage}}.
```

Vérifiez bien que le progrès fait défaut au processus `Confiturerie6`. Expliquez pourquoi, en commentaire avec entête Q6.

Q7 (20 points) Pour remédier au défaut de progrès, une idée serait d'alterner l'usage des valves et de l'étiquetage entre les types de bocaux `a` et `b`, tout en gardant le non dépassement spécifié dans la question 5. Pour réaliser ce fonctionnement, spécifiez 4 processus `OrdreOuvreValve`, `OrdreFermeValve`, `OrdreCommenceEtiquetage`, et `OrdreTermineEtiquetage`

Le processus `OrdreOuvreValve`, spécifie un fonctionnement comme suit :
`1.a.ouvreValve, 1.b.ouvreValve, 2.a.ouvreValve, 2.b.ouvreValve`, ...,
`N.a.ouvreValve, N.b.ouvrValve, 1.a.ouvreValve, 1.b.ouvreValve, ...`
Les autres processus fonctionnent de façon similaire avec respectivement les actions de fermeture de valve, de début et de fin d'étiquetage.

Spécifiez un processus `Confiturerie7` qui compose en parallèle les mêmes processus que `Confiturerie5` en plus de ceux spécifiés dans cette question. Spécifiez les mêmes priorités que dans la question 6.

Vérifiez l'absence de blocage et la satisfaction de progrès.

Q8 (20 points) La spécification précédente engendre un fonctionnement coûteux dû à l'alternance entre les types de bocaux, en plus du fait qu'on profite moins de la parallélisation.

Pour remédier à cela, nous proposons de faire passer les bocaux d'un même type les uns après les autres et puis laisser la place aux bocaux de l'autre type.

D'abord, un bocal qui demande l'accès à une ressource, fait d'abord une requête. Cela nous donne la spécification suivante :

```
BocalT=
    ( requeteValve -> ouvreValve -> remplit ->  fermeValve ->
     requeteEtiquetage -> commenceEtiquetage ->
     etiquette -> termineEtiquetage -> BocalT).
```

À la place des processus d'ordre de la question précédente spécifiez deux processus contrôlant l'accès aux ressources, `ControleValve` et `ControleEtiquetage` qui au lieu d'alterner l'accès d'un type à l'autre à chaque étape, accordent cet accès à au maximum `N` bocaux d'un type avant de donner le tour à l'autre type. Il faut faire attention à ne pas faire attendre les bocaux d'un type donné si ceux de l'autre type n'ont pas fait de requête d'accès. Inspirez-vous de l'exemple du pont pour spécifier ces deux processus.

Spécifiez un processus `Confiturerie8` qui compose en parallèle les processus spécifiés dans cette question avec les mêmes que la question 5, en remplaçant `Bocal` par `BocalT` avec les priorités suivantes :

```
<<  {[i:R].a.{requeteValve, ouvreValve, remplit, fermeValve, requeteEtiquetage,
        commenceEtiquetage, etiquette, termineEtiquetage},
        [i:R].b.{requeteValve,requeteEtiquetage}}.
```

Vérifiez l'absence de blocage et la satisfaction de progrès.

Q9 (5 points) Pour rendre le système un peu plus réaliste on propose un système dans lequel des ruptures de fruits peuvent avoir lieu, ainsi nous pouvons avoir une rupture de confiture qu'on met dans l'un ou l'autre des types de bocaux.

Nous pouvons spécifier les bocaux et le ravitaillement de cette façon.

```
BocalR=  ( requeteValve -> ouvreValve -> remplit ->  fermeValve ->  B1 |
           rupture -> finRupture -> B0),
B0 = (requeteValve -> ouvreValve -> remplit ->  fermeValve ->  B1),
```

```
B1 = (requeteEtiquetage -> commenceEtiquetage -> etiquette ->
    termineEtiquetage -> BocalR).

Ravitaillement = ([i:R][j:S].rupture -> [i][j].approvisionnement ->
           [i][j].finRupture -> Ravitaillement).
```

Spécifiez un processus `Confiture9` avec les mêmes processus que ceux de la question précédente en remplaçant `BocalT` par `BoclalR || Ravitaillement`. Adaptez, si nécessaire les processus de contrôle. Et considérez les priorités sjuivantes

```
<< {[i:R].a.{requeteValve, ouvreValve, remplit, fermeValve, requeteEtiquetage,
                                    commenceEtiquetage, etiquette, termineEtiquetage},
                                  [i:R].b.{requeteValve,requeteEtiquetage},
                                  [i:R][j:S].{ rupture,finRupture,approvisionnement}}.
```

Vérifiez l'absence de blocage et la satisfaction de progrès.

Consignes à suivre obligatoirement pour la remise : Vous devez remettre par intranet les documents suivants dans un .zip. votre fichier .lts .
Le fichier .lts doit contenir les numéros de matricule et les noms de tous les membres de l'équipe. (perte de 5% des points sinon),.

Ne m'envoyez pas vos fichiers par courriel. Sauf force majeure.

Autres contraintes : Vous devez respecter les contraintes suivantes.

- Ce travail se fait obligatoirement en équipe.
- Vous devez prendre toutes les mesures nécessaires pour éviter le plagiat. Si vous proposez des solutions trop semblables à celles d'une autre équipe ou à des solutions trouvées sur internet, des sanctions pourraient être prises conformément au règlement départemental et de l'Université Laval.

Bon travail