IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

FR-PdT – Plan de test du Fil Rouge

UE – IDL

Identification du document

Nom du document	Plan de test d'un simulateur de Réseau de Petri		
Référence du projet	MAPD Fil Rouge		
Référence du document	FR-PdT	Version :	0.2
Préparé par	A. Beugnard	Date :	21 octobre 2022

Historique des changements

Version	ID demande	Date	Modifié par	Description
0.1	NA	21 juin 2021	A. Beugnard	Plan initial
0.2	NA	18 août 2021	J. Mallet	Relecture

Distribution et validation

Nom	Rôle	Entitié	RACI*	Date validation
Bach, Jean-Christophe	Consultant	RUE IDL	RAC	
Segarra, Mayte	Cheffe	MAPD-D	RA	
Mallet, Julien	EC	MAPD-A	RA	
Martinez, Salvador	EC	MAPD-D	RA	
Dagnat, Fabien	RTAF	IMT Atlantique	I	
Cousin, Éric	AE	IMT Atlantique	CI	

*RACI : **R**éalisation, **A**pprobation, **C**onsultation, **I**nformation

Table des matières

1 Introduction

Ce plan de test concerne un logiciel d'animation de réseau de Petri (RdP). Les réseaux de Petri sont des graphes bipartites (Place, Transition) qui permettent la modélisation de processus, de calculs ou de protocoles divers à des fins de vérification.

Ce plan est constitué en trois parties :

1. Le test de la construction du réseau de Petri
2. Le test de l'activation du réseau de Petri
3. Le test d'intégration à l'éditeur graphique

1.1 Objectifs

Le but premier de ce plan de test est d'assurer une couverture *raisonnable* des fonctions afin d'assurer un bon fonctionnement des animations.

Les choix (et variantes) d'implantations sont nombreux, mais le fonctionnement doit correspondre aux attentes d'un réseau de Petri.

⚠Vous pouvez avoir développé d'autres fonctions. Dans un cadre industriel *c'est une mauvaise pratique que de s'écarter du strict cahier des charges et de son plan de test.* Dans le cadre pédagogique de ce projet, il est de votre *responsabilité* d'étendre les tests aux fonctions que vous ajoutez.

Il s'agit essentiellement de tests fonctionnels. On pourra envisager plus tard des tests non fonctionnels comme des évaluations de performances, des capacité de passage à l'échelle ou des évaluations de maintenabilité ou d'évolutivité.

1.2 Tâches

Pour chaque test, les tâches sont les suivantes :

- Réaliser une fonction de test (méthode qui s'appelle `testQuelqueChose`) ;
- Réaliser un scénario de test qui appelle une ou plusieurs fonctions de test (dans un `main` ou une autre méthode) ;
- Exécuter le test, jusqu'à obtention du résultat attendu ;
- Produire un compte-rendu de test (commentaires associés à la fonction test par exemple).

Le compte-rendu peut (doit ?) être rédigé au fur et à mesure des exécutions et des résultats observés.

2 Éléments à tester

On teste les fonctions utilisateurs.

Les fonctions sur lesquelles s'appuient les fonctions utilisateurs n'apparaissent pas puisqu'elles dépendent des choix d'implantation. C'est à vous de vous assurer de leur bon fonctionnement au travers des fonctions utilisateurs, ou par des tests unitaires spécifiques qui n'apparaissent pas ici.

Les cas de test sont identifiés à partir de la connaissance des experts.

En cas d'identification de nouveaux cas, il *faut* transmettre une demande d'évolution du plan de test à leurs rédacteurs.

Nous ne traitons pas les aspects non-fonctionnels dans ce plan de test.

3 Stratégie de test

Les tests sont effectués de manière itérative, comme le développement.

On va du plus simple au plus complexe.

On s'autorise le passage par des états incomplets du programme pour construire petit à petit le programme. Par exemple, on s'autorisera à créer un arc qui relie une place à une transition sans

utiliser une instance de RdP. Puis quand cela fonctionnera, on n'oubliera pas d'intégrer tous les objets nécessaires (en fonction de vos choix de conception) à une instance de RdP.

⚠Certains utilisent le mot *mock*[1] pour désigner du code incomplet qui simule un comportement déterminé et permet de tester d'autres parties du code.

Les tests simples déjà réalisés doivent rester valides quand le système se complexifie. On s'assure de la *non régression* en ré-exécutant les tests souvent. On peut aussi les cumuler, en les intégrant les uns dans les autres.

Tous les tests donnent lieu à la création d'une fonction de test. Ainsi, pour le test de l'affichage d'une transition (AT), on créera une fonction de test `testAffichageTransition`.

⚠Plus probablement, on créera une famille de fonctions de test `testAffichageTransition` pour des transitions différentes (isolée, uniquement avec des entrées, uniquement avec des sorties, etc.).

⚠Toutes les fonctions de test sont préfixées par `test`.

⚠Toutes les fonctions de test sont organisées dans des classes dédiées au test.

⚠On n'efface pas des tests passés avec succès ; les tests sont accumulés pour être re-exécutés et asurer la non-régression.

3.1 Tests unitaires

Les tests unitaires constituent l'essentiel des tests de ce document. Ils concernent les fonctions utilisateurs. Chaque fonction devra posséder un test pour :

- mettre en évidence son fonctionnement normal. Plusieurs cas peuvent être parfois nécessaires ;
- mettre en évidence des situations d'erreurs qui doivent être détectées. Mauvais paramètres, mais aussi appel dans un contexte incorrect (arcs doublés par exemple).

Un test de fonction est dit *réussi* lorsque toutes les réponses attendues sont observées.

Vous écrirez au moins un test par fonction.

La couverture des tests devrait être $> 98\%$. Les lignes non couvertes devront être identifiées et l'absence de test justifiée[2]. Vous utiliserez Emma pour calculer les couvertures de test.

⚠Une couverture de test de 100% ne garantit pas que tous les chemins possibles soient traités. Chaque ligne peut avoir été exécutée, sans que tous les cas aient été couverts.

En cas d'échec d'un test, vous signalerez par un commentaire associé au test, le résultat observé, puis après correction, la correction réalisée et le nouveau résultat obtenu.

```
/*
* Trace des tests de la fonction AT1
*
* 25/09/2021 : le nom ne s'affiche pas ; correction de Transition>toString()
* 25/09/2021 : affichage correct
*/
```

Le responsable des tests unitaires est le membre du binôme dont la seconde lettre du prénom est la plus grande, puis la troisième, etc.

1. `https://fr.wikipedia.org/wiki/Mock_(programmation_orientée_objet)`
2. En principe tout le code écrit devrait être utilisé.

Les fonctions testées et leurs tests unitaires seront écrits en même temps par les deux membres du binôme.

3.2 Tests de système et d'intégration

Les tests d'intégration concernent l'intégration de votre projet avec l'éditeur de réseau de Petri PNEditor.

Le responsable des tests d'intégration est le membre du binôme dont la seconde lettre du prénom est la plus petite, puis la troisième, etc. [3]

Les tests d'intégration sont des tests d'utilisation interactifs de l'application. Ils ne seront pas automatisés.

Plusieurs scénarios sont décrits, le résultat de chacune des étapes sera enregistré dans un document qui sera livré avec le code.

3.3 Tests de performance, de volume et de stress

Ne s'applique pas à ce projet.

Définition : Indiquez ce que vous comprenez des tests de stress pour votre projet.

Participants : Qui effectuera les tests sous contrainte dans le cadre de votre projet ? Énumérez les personnes qui seront responsables de cette activité.

Méthodologie : Décrivez comment les tests de performance et de stress seront menés. Qui écrira les scripts de test pour les tests, quelle sera la séquence des événements pour les tests de performance et de stress, et comment l'activité de test se déroulera-t-elle ?

3.4 Tests de sécurité et de reprise

Ne s'applique pas à ce projet.

3.5 Test d'acceptation par l'utilisateur

Ne s'applique pas à ce projet.

Se rapproche du test d'intégration, mais devrait être réalisé par des utilisateurs ayant besoin d'animation de réseau de Petri.

Définition : L'objectif du test d'acceptation est de confirmer que le système est prêt pour une utilisation opérationnelle. Pendant le test d'acceptation, les utilisateurs finaux (clients) du système comparent le système à ses exigences initiales.

Participants : Qui sera responsable du test d'acceptation par l'utilisateur ? Indiquez le nom des personnes et leur responsabilité.

Méthodologie : Décrire comment les tests d'acceptation par les utilisateurs seront menés. Qui écrira les scripts de test pour les tests, quelle sera la séquence des événements des tests d'acceptation par l'utilisateur, et comment l'activité de test se déroulera-t-elle ?

3. Bref, l'autre.

3.6 Tests par lots

Ne s'applique pas à ce projet.

3.7 Test de régression automatisé

Les tests unitaires seront organisés de telle façon que la non-régression sera assurée.

Une fonction devra collecter l'ensemble des tests réalisés et être exécutée avant chaque commit.

3.8 Test de la documentation

Ne s'applique pas à ce projet.

3.9 Béta-test

Le beta test sera réalisé lors de l'évaluation.

4 Exigences sur l'environment

Les tests s'effectuent dans l'environnement de développement Eclipse. Ils s'appuient sur les outils :

- de développement classiques (le test, c'est du code)
- Emma. Pour mesurer la couverture de code

5 Planification du test

Les activités de tests se déroulent en parallèle des activités de développement. Les créneaux suivants y sont plus spécialement dédiés :

- 12 octobre 13h45 (introduction aux tests, cours)
- 13 octobre 9h30-12h15 (séance pratique)

6 Procédures de contrôle

Si une erreur ou un manque est détecté dans ce document, merci d'en informer par courriel les responsables des UE IDL et MAPD.

7 Fonctions/caractéristiques à tester

Voici toutes les fonctionnalités du logiciel et les combinaisons de fonctionnalités du logiciel qui seront testées.

Les conditions facultatives (fac) sont laissées à votre choix, mais *explicitez et expliquez* quel choix vous avez fait.

Les fonctions de création et d'affichage de RdP peuvent être développées et testées en parallèle.

Afficher un graphe peut être compliqué. Une version simple consiste à afficher qui connait qui. Dans ce cas, donner des noms aux objets permet de mieux lire les informations.

- Le RdP est composé de places et transitions.
- Chaque place connait des transitions via des arcs.
- Chaque transition connait des places via des arcs.
- Chaque arc (quelque soit sa nature) connait une place et une transition

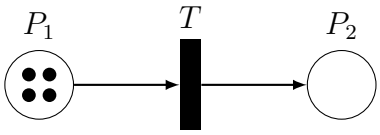
⚠Attention, les tests sont présentés comme des tests de fonction. Une fonction peut être réalisée par une méthode d'instance ou par une méthode de classe. C'est à vous de choisir. La colonne « Entrées » des tableaux suivants peut donc faire référence à l'instance courante (`this`).

Les tests qui concernent des cas d'erreur apparaissent en rouge.

7.1 Affichage dans la console

L'affichage se fait dans la console.

Par exemple, pour le réseau suivant :




```
Réseau de Petri
  2 places
  1 transition
  2 arcs
 Liste des places :
  1 : place avec 4 jetons, 1 arc simple sortant, 0 arc simple entrant
  2 : place avec 0 jetons, 0 arc simple sortant, 1 arc simple entrant
 Liste des transitions
  1 : transition, 1 arc entrant, 1 arc sortant
 Liste des arcs :
  1 : arc simple poids 1 (place avec 4 jetons vers transition)
  2 : arc simple poids 1 (transition vers place avec 0 jetons)
```

Adaptez cet exemple aux choix que vous avez faits.

Vous pouvez enrichir si besoin par un nom que vous auriez associé aux places et transitions.

Les tests suivants sont présentés par ordre de difficulté.

Fonction	Entrées	Cas	code	Résultat attendu
Afficher une transition	une transition	isolée	AT1	affiche la transition
		arc(s) entrant(s)	AT2	
		arc(s) sortants(s)	AT3	
		arc(s) entrant(s) et sortant(s)	AT4	

Afficher une place	une place	isolée, 0 jetons	AP1	affiche la place.
		arc(s) entrant(s), 2 jetons	AP2	
		arc(s) sortants(s), 4 jetons	AP3	
		arc(s) entrant(s) et sortant(s), 1 jeton	AP4	
Afficher une place	une place	isolée, -1 jetons	APe	affiche la place et signale l'erreur sur les jetons
Afficher un arc	un arc	poids>0, une place, une transition	AA1	affiche l'arc
Afficher un arc		poids <0	AAe1	affiche l'arc et signale l'erreur de poids
		les extrémités sont deux places ou deux transitions	AAe2	affiche l'arc et signale l'erreur de connexion
Afficher un RdP	un RdP		AR1	affiche le RdP
Afficher un RdP	un RdP	arcs doublés*	ARe	affiche le RdP et signale l'erreur

* Voir plus loin, tous les arcs doublés (arcs distincts partant d'une même place et arrivant à une même transition ou partant d'une même transition et arrivant à une même place) ne sont pas nécessairement cause d'erreur. Cette vérification peut être compliquée...

Pour chaque affichage, plusieurs cas doivent être traités.

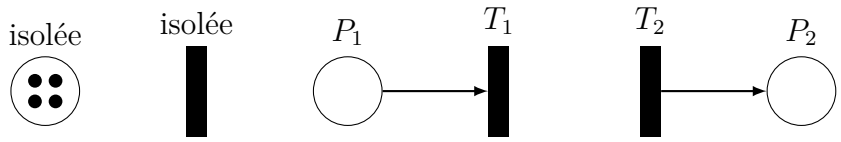
⚠Seuls les cas des arcs simples sont traités. Il faudrait compléter pour les arcs zéros et les arcs videurs.

⚠On peut décider d'afficher des cas impossibles (jeton négatif par exemple) ; c'est de la *responsabilité* du constructeur que d'interdire ce cas.

⚠On peut signaler un message d'erreur lorsqu'une anomalie est détectée lors de l'affichage. On utilise alors `System.err` et non pas `System.out` pour faire les `println()`.

7.2 Création du réseau de Petri

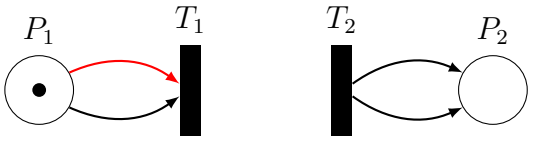
Éléments du réseau de Petri



⚠Les fonctions à tester suivantes n'affichent rien. Elles créent des objets et les retournent à la fonction de test. C'est la fonction de test qui affiche afin que les testeurs puissent observer les résultats.

Fonction	Entrées	Cas	code	Résultat attendu
Créer un réseau de Petri vide	Aucune		CR	un RdP est créé vide
Créer une transition	Aucune/un nom	(fac) nom unique	CT	une transition est créée (sans lien)
Créer une place	Aucune/un nom/jetons	(fac) nom unique, jetons ≥ 0	CP	une place est créée (sans lien)
Créer une place	Aucune/un nom/ jetons	jetons < 0	CPe	aucune place n'est créée. Erreur
Ajouter jetons	une place, N	N < 0	CAJ0	...
Ajouter jetons	une place (J jetons), N	N ≥ 0	CAJ1	...
Enlever jetons	...	...	...	...
...	...	...	...	...
...	...	...	...	...

On peut faire en sorte que *par construction* il soit impossible de créer un arc PP ou un arc TT. Il suffit de ne pas fournir les constructeurs... [4]



⚠Les arcs doublés se comportent-ils de la même façon qu'un arc unique avec un nombre de jetons équivalent ? [5]

Faire un choix concernant les arcs doublés et adapter les tests.

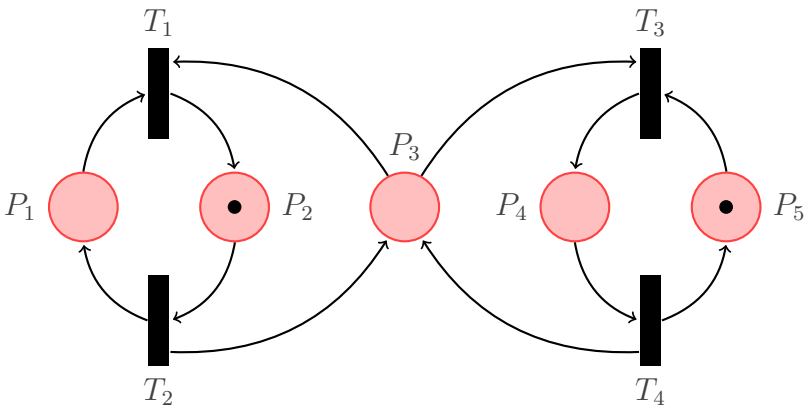
Fonction	Entrées	Cas	code	Résultat attendu
...	...	...	...	...

4. À condition d'avoir distingué les types P et T.

5. Dans le cas de gauche, il faut décider s'il y a assez de jetons. Le faire en deux fois risque de conduire à une mauvaise décision. On a $1 \geq 1$ deux fois mais pas $1 \geq 1+1$. Dans le cas de droite, il n'y a pas de décision, juste une application de la décision. $2 = 2 \times 1$! Quand on s'aperçoit du doublon dans le cas de gauche, on pourrait mettre à jour le poids...

Assemblage dans un RdP

Le scénario de test consiste à créer le réseau de Petri suivant (On peut l'appeler $Mutex$) :



Cet exemple couvre les cas où les transitions et les places ont des entrées et sorties simples ou multiples.

Des opérations inutiles sont introduites pour s'assurer qu'elles sont bien rejetées (si vous avez fait ce choix). Le réseau s'affiche après chaque opération pour s'assurer de sa construction.

Plusieurs variantes autour de ce scénario sont possibles selon vos choix de réalisation. Vous pouvez avoir introduit une fonction qui change le nom d'une place ou d'une transition par exemple.

Fonction	Entrées	Étape	Résultat attendu
Créer un réseau de Petri vide PN	Aucune	CR1	un RdP est créé vide
Créer P_1	PN, "P1", 0	CP1	PN contient P_1
Créer P_1	PN, "P1", 0	CP1	Rejet, PN contient P_1
Créer P_2	PN, "P2", 1	CP2	PN contient P_1 et P_2
Créer T_1	PN, "T1"	CT1	PN contient P_1, P_2, T_1
Créer T_1	PN, "T1"	CT1	Rejet, PN contient P_1, P_2, T_1
$P_1 \to T_1$	PN, $P1$, $T1$, poids $= 1$	CPT1	PN contient $P_1, P_2, T_1, P_1 \to T_1$
$P_1 \to T_1$	PN, $P1$, $T1$, poids $= 1$	CPT1	Rejet, PN contient $P_1, P_2, T_1, P_1 \to T_1$
$T_1 \to P_2$	PN, $T1$, $P2$, poids $= 1$	CTP1	PN contient $P_1, P_2, T_1, P_1 \to T_1, T_1 \to P_2$
$T_1 \to P_2$	PN, $T1$, $P2$, poids $= 1$	CTP1	Rejet, PN contient $P_1, P_2, T_1, P_1 \to T_1, T_1 \to P_2$
Créer T_2	PN, "T2"	CT2	PN contient $P_1, P_2, T_1, P_1 \to T_1, T_1 \to P_2, T_2$
$P_2 \to T_2$	PN, $P2$, $T2$, poids $= 1$	CPT2	PN contient $P_1, P_2, T_1, P_1 \to T_1, T_1 \to P_2, T_2, P_2 \to T_2$
$T_2 \to P_1$	PN, $T2$, $P1$, poids $= 1$	CTP2	PN contient $P_1, P_2, T_1, P_1 \to T_1, T_1 \to P_2, T_2, P_2 \to T_2, T_2 \to P_1$
Créer P_4	PN, "P4", 0	CP4	PN contient $P_1, P_2, T_1, P_1 \to T_1, T_1 \to P_2, T_2, P_2 \to T_2, T_2 \to P_1, P_4$
Créer P_5	PN, "P5", 1	CP5	PN contient $P_1, P_2, T_1, P_1 \to T_1, T_1 \to P_2, T_2, P_2 \to T_2, T_2 \to P_1, P_4, P_5$

Créer T_3	PN, "T3"	CT3	PN contient $P_1, P_2, T_1, P_1 \rightarrow T_1, T_1 \rightarrow P_2, T_2, P_2 \rightarrow T_2, T_2 \rightarrow P_1, P_4, P_5, T_3$
$P_5 \rightarrow T_3$	PN, $P5$, $T3$, poids $= 1$	CPT3	PN contient $P_1, P_2, T_1, P_1 \rightarrow T_1, T_1 \rightarrow P_2, T_2, P_2 \rightarrow T_2, T_2 \rightarrow P_1, P_4, P_5, T_3, P_5 \rightarrow T_3$
$T_3 \rightarrow P_4$	PN, $T3$, $P4$, poids $= 1$	CTP3	PN contient $P_1, P_2, T_1, P_1 \rightarrow T_1, T_1 \rightarrow P_2, T_2, P_2 \rightarrow T_2, T_2 \rightarrow P_1, P_4, P_5, T_3, P_5 \rightarrow T_3, T_3 \rightarrow P_4$
Créer T_4	PN, "T4"	CT4	PN contient $P_1, P_2, T_1, P_1 \rightarrow T_1, T_1 \rightarrow P_2, T_2, P_2 \rightarrow T_2, T_2 \rightarrow P_1, P_4, P_5, T_3, P_5 \rightarrow T_3, T_3 \rightarrow P_4, T_4$
$P_4 \rightarrow T_4$	PN, $P4$, $T4$, poids $= 1$	CPT4	PN contient $P_1, P_2, T_1, P_1 \rightarrow T_1, T_1 \rightarrow P_2, T_2, P_2 \rightarrow T_2, T_2 \rightarrow P_1, P_4, P_5, T_3, P_5 \rightarrow T_3, T_3 \rightarrow P_4, T_4, P_4 \rightarrow T_4$
$T_4 \rightarrow P_5$	PN, $T4$, $P5$, poids $= 1$	CTP4	PN contient $P_1, P_2, T_1, P_1 \rightarrow T_1, T_1 \rightarrow P_2, T_2, P_2 \rightarrow T_2, T_2 \rightarrow P_1, P_4, P_5, T_3, P_5 \rightarrow T_3, T_3 \rightarrow P_4, T_4, P_4 \rightarrow T_4, T_4 \rightarrow P_5$

Connexion des deux...

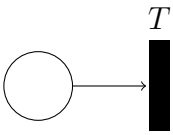
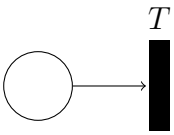
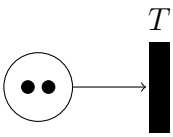
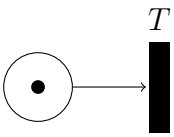
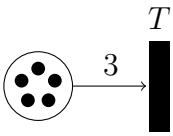
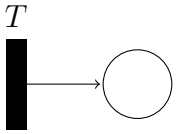
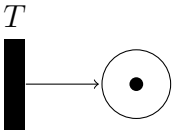
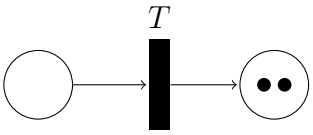
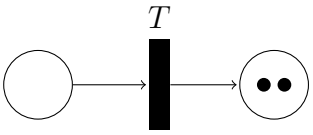
Fonction	Entrées	Étape	Résultat attendu
Créer P_3	PN, "P3", -2	CP5	Rejet, PN contient $P_1, P_2, T_1, P_1 \rightarrow T_1, T_1 \rightarrow P_2, T_2, P_2 \rightarrow T_2, T_2 \rightarrow P_1, P_4, P_5, T_3, P_5 \rightarrow T_3, T_3 \rightarrow P_4, T_4, P_4 \rightarrow T_4, T_4 \rightarrow P_5$
Créer P_3	PN, "P3", 0	CP5	PN contient $P_1, P_2, T_1, P_1 \rightarrow T_1, T_1 \rightarrow P_2, T_2, P_2 \rightarrow T_2, T_2 \rightarrow P_1, P_4, P_5, T_3, P_5 \rightarrow T_3, T_3 \rightarrow P_4, T_4, P_4 \rightarrow T_4, T_4 \rightarrow P_5, P_3$
$P_3 \rightarrow T_1$	PN, $P3$, $T1$, poids $=$ -1	CPT5	Rejet, PN contient $P_1, P_2, T_1, P_1 \rightarrow T_1, T_1 \rightarrow P_2, T_2, P_2 \rightarrow T_2, T_2 \rightarrow P_1, P_4, P_5, T_3, P_5 \rightarrow T_3, T_3 \rightarrow P_4, T_4, P_4 \rightarrow T_4, T_4 \rightarrow P_5, P_3$
$P_3 \rightarrow T_1$	PN, $P3$, $T1$, poids $= 1$	CPT5	PN contient $P_1, P_2, T_1, P_1 \rightarrow T_1, T_1 \rightarrow P_2, T_2, P_2 \rightarrow T_2, T_2 \rightarrow P_1, P_4, P_5, T_3, P_5 \rightarrow T_3, T_3 \rightarrow P_4, T_4, P_4 \rightarrow T_4, T_4 \rightarrow P_5, P_3, P_3 \rightarrow T_1$
$P_3 \rightarrow T_3$	PN, $P3$, $T3$, poids $= 1$	CPT6	PN contient $P_1, P_2, T_1, P_1 \rightarrow T_1, T_1 \rightarrow P_2, T_2, P_2 \rightarrow T_2, T_2 \rightarrow P_1, P_4, P_5, T_3, P_5 \rightarrow T_3, T_3 \rightarrow P_4, T_4, P_4 \rightarrow T_4, T_4 \rightarrow P_5, P_3, P_3 \rightarrow T_1, P_3 \rightarrow T_3$
$T_2 \rightarrow P_3$	PN, $T2$, $P3$, poids $= 1$	CTP5	PN contient $P_1, P_2, T_1, P_1 \rightarrow T_1, T_1 \rightarrow P_2, T_2, P_2 \rightarrow T_2, T_2 \rightarrow P_1, P_4, P_5, T_3, P_5 \rightarrow T_3, T_3 \rightarrow P_4, T_4, P_4 \rightarrow T_4, T_4 \rightarrow P_5, P_3, P_3 \rightarrow T_1, P_3 \rightarrow T_3, T_2 \rightarrow P_3$

$T_4 \to P_3$	PN, $T4$, $P3$, poids $= 1$	CTP6	PN contient $P_1, P_2, T_1, P_1 \to T_1, T_1 \to P_2, T_2, P_2 \to T_2, T_2 \to P_1, P_4, P_5, T_3, P_5 \to T_3, T_3 \to P_4, T_4, P_4 \to T_4, T_4 \to P_5, P_3, P_3 \to T_1, P_3 \to T_3, T_2 \to P_3, T_4 \to P_3$

7.3 Activation du réseau de Petri

Pour tester l'activation, voici des scénarios montrant l'état avant puis après. La fonction est toujours celle activant la transition T.

Le premier tableau concerne des transitions simples.

Avant	code	Après
	RI	
	RD0	
	RD1	
	RD2	...
	RG0	
...	...	...
	RM0	
...	...	...

Le second tableau concerne des transitions à entrées multiples.

Avant	code	Après
...	...	...

À partir du réseau de Petri assemblé page 10, on peut observer le déplacement des jetons avec des séquences de tirage de transition.

Les tirages en rouge ne changent pas l'état.

T1,T3,T2,T1,T2,T1,T3,T2,T3,T4,T3,T1,T2,T4...

On constate que les états P_2 et P_4 ne peuvent jamais posséder un jeton simultanément. C'est une exclusion mutuelle, qui sera abordée en UE CONC.

7.4 Intégration à l'interface

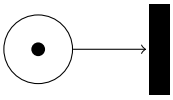
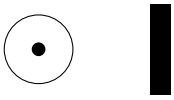
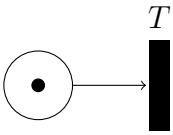
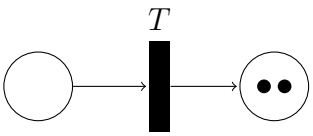
Les tests sont réalisés *interactivement* avec le PNEditor.

Les tests réalisés précédemment par programmation, peuvent être réalisés via l'interface graphique.

- CP1 : créer une place (vide)
- CT1 : créer une transition
- RI : activer une transition isolée ; pas d'erreur, rien ne change
- CPT1 : créer un lien P-T
- RD0 : activer une transition ; pas d'erreur, rien ne change
- CAJ1 : ajouter un jeton à la place ; un jeton s'affiche (1) [2 fois]
- RD1 : activer une transition ; un jeton est consommé, il en reste un
- etc.

7.5 Suppression d'éléments

⚠La suppression d'une place ou d'une transition conduit à la suppression des arcs reliés à cette place ou cette transition.

Avant	code	Après
	ST0	
	SP0	
	SPT0	
	ST1	...
	ST2	...
...	...	...

8 Fonctions/caractéristiques à ne pas tester

On ne s'intéresse pas aux propriétés non fonctionnelles dans ce plan de test.

⚠Les propriétés non-fonctionnelles sont pourtant essentielles dans vos choix de conception et doivent être utilisées pour les justifier.

Concernant la performance, qui n'est pas critique à l'échelle où l'on se place, nous vous demandons toutefois de faire attention aux itérations. Attention au choix de vos structures de données et des opérations que vous leurs appliquez. Faire un `get(i)` sur une liste chainée est en $O(n)$...

9 Ressources et responsabilités

Ne s'applique pas à ce projet compte tenu de sa taille.

Spécifiez les membres du personnel qui sont impliqués dans le projet de test et quels seront leurs rôles (par exemple, Mary Brown (utilisateur) compile les cas de test pour les tests d'acceptation). Identifiez les groupes responsables de la gestion, de la conception, de la préparation, de l'exécution et de la résolution des activités de test ainsi que des problèmes connexes. Identifiez également les groupes responsables de la fourniture de l'environnement de test. Ces groupes peuvent comprendre des développeurs, des testeurs, du personnel d'exploitation, des services de test, etc.

10 Livrables

Les documents principaux sont :

- Plan de test (ce document)
- Cas de test (inclus dans ce document)
- Rapports d'incidents de test (tracés dans le code)
- Rapports de synthèse des tests (en commentaire dans le code qui regroupe l'ensemble des tests)

L'organisation des livrables est spécifique à ce projet, compte tenu de sa taille, de sa durée et de son organisation.

11 Arrêt / Critère de fin

La fin du projet.

12 Critères de reprise

Ne s'applique pas à ce projet.

13 Dépendances

Ne s'applique pas à ce projet

Identifier les contraintes significatives des tests, telles que la disponibilité des éléments de test, la disponibilité des ressources de test et les délais.

14 Risques

Ne s'applique pas à ce projet

Identifier les hypothèses à haut risque du plan de test. Spécifiez des plans d'urgence pour chacune d'entre elles (par exemple, un retard dans la livraison des éléments de test pourrait nécessiter une augmentation des horaires de nuit pour respecter la date de livraison).

15 Outils

Les tests s'effectuent dans l'environnement de développement Eclipse. Ils s'appuient sur les outils :

- de développement classiques (le test, c'est du code)
- Emma. Pour mesurer la couverture de code

Nous n'utilisons pas d'outils d'intégration continue, ni de suivi de bogues.

16 Documentation

NA.