

Ingénierie des protocoles

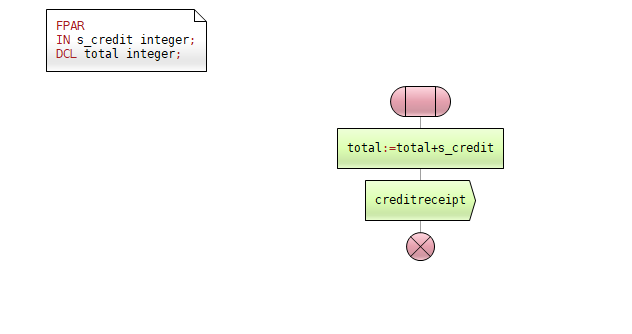
Nom : BIBI

Prénom : Thilleli

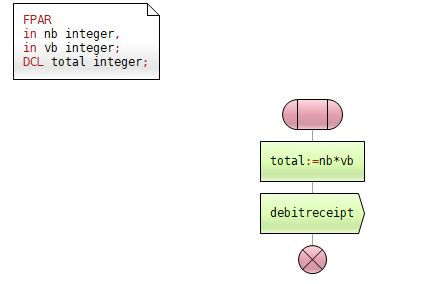
Numéro étudiant : 21965002

1. Modélisation :
2. Procédures :

* Creditproc : crédite un compte bancaire usager, elle renvoie le signal *creditreceipt* au process.



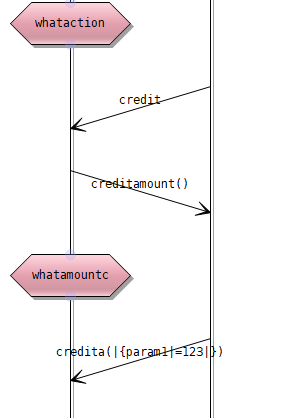
* Debitproc : débite une somme elle renvoie *debitreceipt*



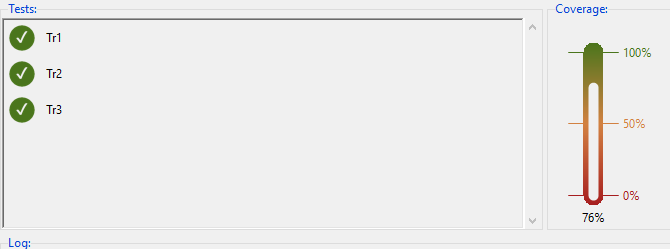
* Trace MSC qui correspond à une séquence de test selon la stratégie UIO pour le comportement demandé : cela revient à faire :

*TSUIO(whataction,credit/creditamount,whatamountc) :*

En arrivant à *whataction* dans le process on demande de créditer notre compte(signal : credit), et là on reçoit *creditamount* et on arrive sur l’état whatamountc comme le montre la capture suivante :



* Les trois traces qui couvrent 65%  de la spécification :



Tr1 et Tr2 sont les deux traces demandées, Tr3 c’est une troisième trace.

**Tr1 :** je crédite un montant correspondant à mon numéro d’étudiant, et cela après *insertcard* et *insertcode* . pour cela je passe par *whataction* puis je choisi *credit* , on arrive donc sur l’état *whatamountc*  et de là on passe par la procédure *creditproc* et enfin *cancelop* et *okcancel* pour revenir à *idle.*

***Tr2***: comme dans Tr1 je fais d’abord *insertcard* et *insertcode* , en arrivant sur *whataction* je choisis debit, puis *a50* (pour le type de billet), et je renseigne le paramètre numberbanknotes et enfin *cancelop* et *okcancel* pour revenir à *idle.*

**Tr3** :après *insertcard* je fais rentrer un code faux puis le code correct, et donc je couvre les deux cas qu’on a dans la spécification par rapport au code.

Je fais ensuite 4 fois *debit* avec *a10,a20,a50,a100* et là aussi je couvre les 4 cas qu’on peut avoir sur *debit*.

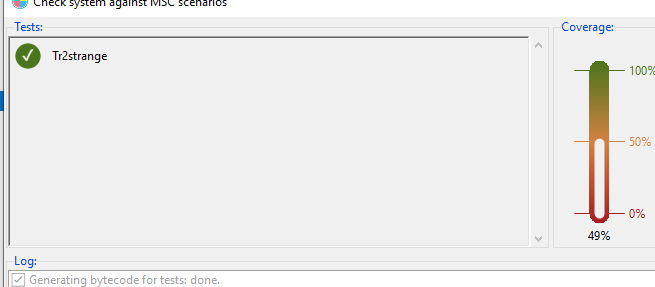
En regroupant les trois traces on aura donc essayé le cas où le code et faux ou le code et vrai, exécuté les procédures *creditproc* et *debitproc* avec les 4 types de billets possibles, avec ça j’ai réussi à couvrir 76% de la spécification.

Il existe encore des comportements dans la spécification que mes traces ne couvrent pas comme

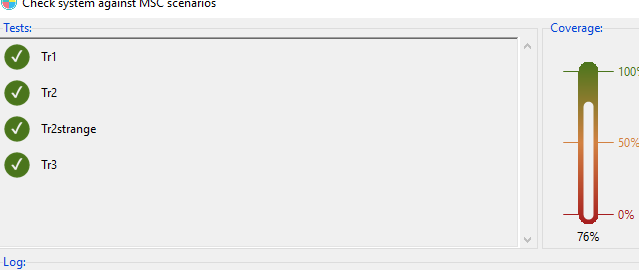
le cas où l’utilisateur ne rentre pas de code et donc le temps d’attente (tempo) sera écoulé, et le cas où on donne un code faux trois fois.

1. Avec une trace récupérée d’un camarade  j’ai eu un verdict *PASS* dans les deux cas:

* Check sur la Trace seule :



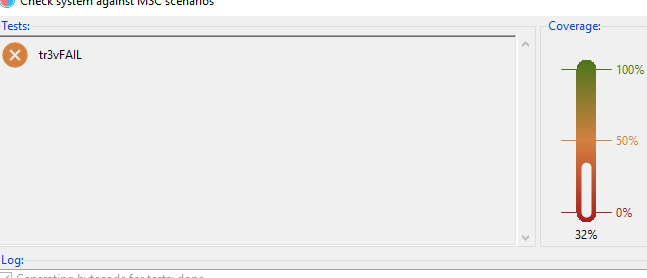
* Check sur la trace avec mes propres traces :



Ce qui est normal : puisque on a tous la même spécification sauf les deux procédures, et donc la spécification de mon camarade, sauf les deux procédures, correspond à ma spécification de plus Si on regarde sa trace on trouve que le comportement pour lequel cette trace est faite est présent et respecte ma spécification, et elle contient **aucun comportement** qui n’est pas présent dans ma spécification d’où les *PASS*.

1. tr3vFAIL :

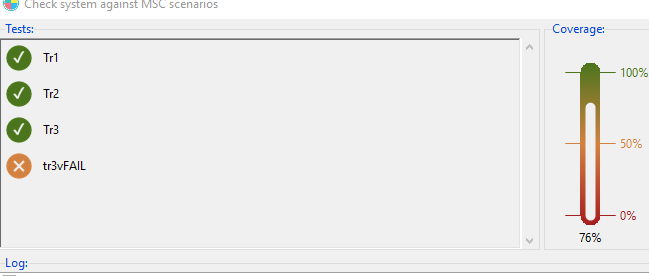
* check de la trace toute seule :FAIL et couverage de 32%



Dans cette trace on a deux signaux *bknote10* reçus juste après la saisie de nombre de billets qu’on veut débiter, ce signal existe dans la spécification, cependant le comportement n’est pas présents dans notre spécification ce qui fait que le check échoue sur cette trace.

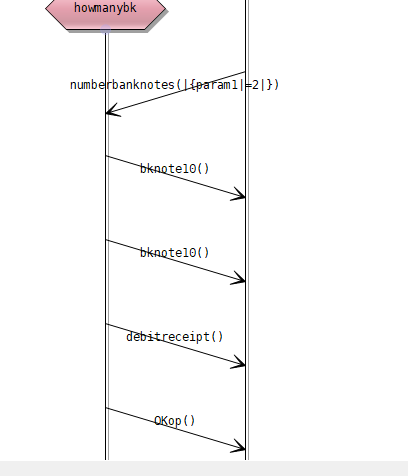
Le couverage est de 32% car cette trace couvre des comportements qui existent dans notre spécification : comme insertcard, insertcode, debit …ect

* check avec mes traces : FAIL avec le même couverage obtenu sans cette trace

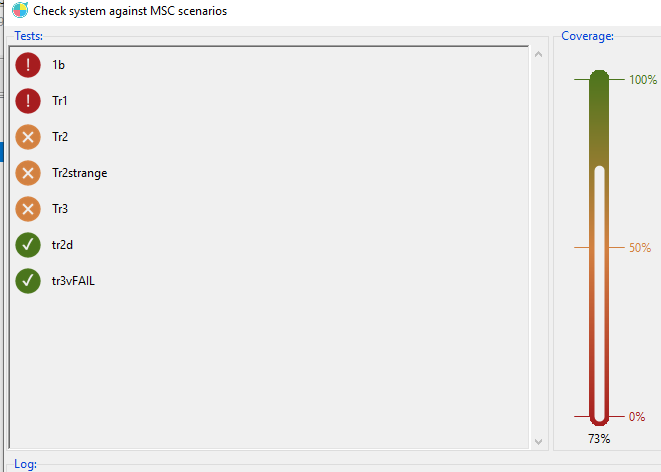


Comme expliqué dans *b* le comportement couvert par cette trace n’est pas présents dans notre spécification ce qui fait que le check échoue sur cette trace.

Le couverage ne change pas car la trace ne nous apporte aucun autre comportement qui est présent dans la spécification mais pas couvert par les trace Tr1, Tr2 et Tr3 autrement dit tous les comportements représentés par cette trace qui sont présents dans la spécification ont était déjà couverts par les traces Tr1, Tr2 et Tr3.



1. check de toutes les traces sur la nouvelle spécification :



**Tr2 et Tr2strange fail** : elles représentent le comportement où on fait *debit* avec *numberbankonote*=2 mais le signal *bknote* correspondant au type de billet (bknote50 dans ce cas) n’est pas présent, dans la spécification on a un signal *bknote* pour chaque billet débiter.

**Tr3 fail** : pour la même raison que Tr2, en plus on a débiter plusieurs fois avec un paramètre *numberbankonote* diffèrent de 0 mais le signal *bknote* et non pressent.

**Tr2d PASS :** tous les comportement qui sont représentés dans cette trace sont présents dans la spécification, et aucun comportement n’est étrange à la spécification. On remarque que contrairement aux traces précédentes dans cette trace on a le signal bknote :

* Au début on a débiter 2 billet de a10 et on a deux fois le signal bknote10.
* Puis 3 billet de a20 et on a trois fois le signal bknote20.

**Tr3Vfail :PASS :** sur cette trace on a eu un verdict pass (toujours on testant toutes les traces ensemble).en regardant cette trace on voit que la dernière action effectuée est débit avec comme paramètre *bknote = 3* (nombres de billet à débiter) ;dans notre spécification il faudra donc avoir 3 signaux *bknote20(),* or on en a eu que 2, et puisque cette trace est testé au même temps que ***Tr2d*** dans laquelle on a aussi *bknote=3* mais on a eu 3 signaux *bknote20()* donc dans notre spécification on a eu à la fin les trois signaux *bknote20()* attendus par les deux traces ***Tr2d et Tr3Vfail.***

**1b et Tr1 error :** sur ces deux traces on a eu error, et cela est dû au fait qu’à la fin de ces traces on a une requête *okcancel* (signal) et à laquelle on a une réponse *returncard* dans la première spécification, or dans la nouvelles spécification on aura pas de réponses pour cette requête , parce que *returncard* n’est pas présent dans la spécification.

* **TTCN3 :**

****

Dans le fichier TTCN3 on a pour chaque trace une cas de test .

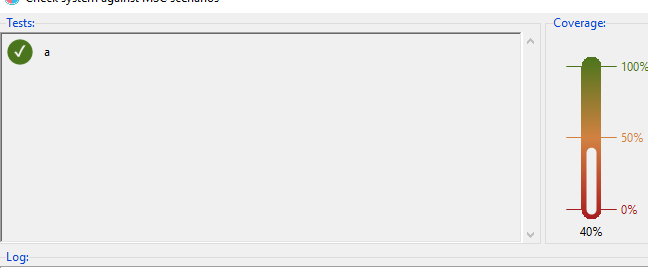
Dans le cas de test **TC\_tr2d\_tr3vFAIL** (correspandant à la capture precedente) des deux traces **tr2d et tr3vFAIL** :

* On a des *send* pour l’envoie des signaux et *receive* pour la réception des signaux.
* Tout au debut on active RTDS\_fail et cela pour qu’il puisse retourner un fail si à la fin de cette trace y’a pas de verdict PASS
* Ce qui est avant ***alt*** correspond à tous les signaux envoyés et reçus dans notre spécification dans les traces **tr2d et tr3vFAIL** jusqu’au deuxième signal bknote20() ( les deux traces sont équivalentes jusqu’au là)
* Après le deuxième signal bknote20() On a deux alternatives :
* Soit on reçoit le signal *bknote20 (*par **tr2d***)* alors dans ce cas on envoie le signal *okcancel*
* Soit on reçoit le signal *debitreceipt (par* **tr3vFAIL***)*et dans ce cas on retourne le signal *returncard*.

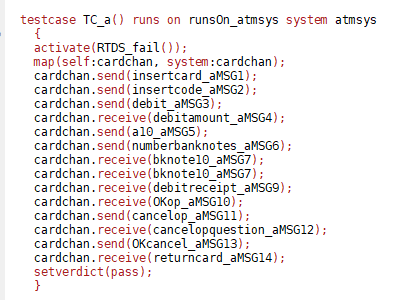
Dans le cas de test des autres test on a des send et des receive et à la fin il peut envoyer le verdict pass,Et si celui n’est pas envoyé alors grâce à RTDS\_fail on envoie un verdict Fail.

**3- SCRIPTS TTCN3 :**

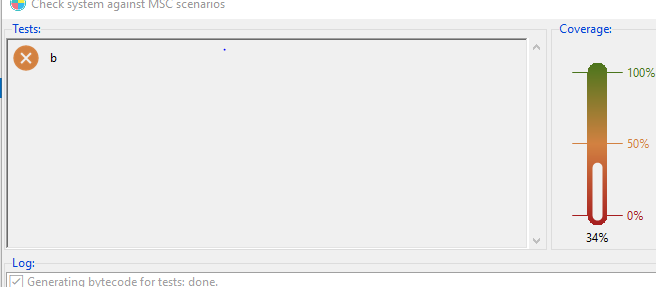
1. **Comportement B1 PASS :** Si le client demande 2 billets de 10, il doit recevoir 2 billets de 10

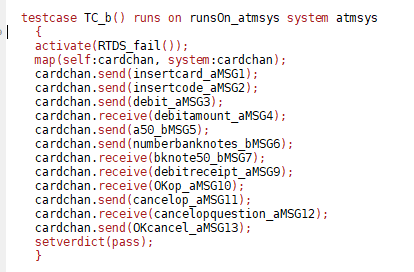


**Test case :**

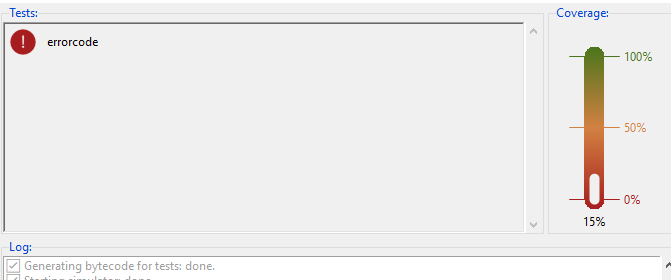
****

1. si le client demande 1 billet de 50, il ne doit pas recevoir de billet de 20 : verdict FAILl dans spec3 : c’est un comportement non attendu car dans spec3 si l’utilisateur demande un billet de 50 il aura un billet de 20, notre trace donne Pass pour le comportement B2 dans la deuxième spécification, et donc un Fail dans spec3

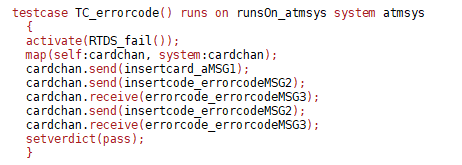
****

****

1. dans cette trace on a saisi deux fois un code erroné, or dans notre spécification on a droit à une seule erreur sur le code d’où l’erreur



Ttc3 :

****

Dans le test case : on active RTDS\_fail qui sera appelé si jamais on a pas de verdict pass.

On a un send pour le signal insertcard code , suivi d’un send insertcode correspondant à la saisie du premier code (faux) , puis on recoit un errorcode, on renvoie insertcode(saisie du deuxieme code faux) avec send et on aura un aussi un errorcode.