3DEXPERIENCE®

Réunion plénière VACSIM

Tests Progressifs Par Parties

Christophe Jaouen / Eric Mével

14 octobre 2014

DASSAULT SYSTEMES | The 3DEXPERIENCE® Company

Etat d’avancement de la tâche 1

Tests Progressifs Par Parties

Génération des scénarios
Résultats
Synchronisation entre les scénarios et le système sous test
Vérifications en boite blanche

Tests Progressifs Par Parties

Génération des scénarios

- Mise au point (correction d'anomalies et améliorations)
- Suppression de la limitation de dépendance à 32 entrées / mémoires par sortie
- Prise en compte des évolutions sur les algorithmes de sélection des vecteurs
  - Ne pas rejouer le vecteur ayant permis d'atteindre un état stable du système
  - Lorsqu'une meilleure séquence de tests est déterminée, retenir cette séquence et supprimer la séquence précédemment retenue
  - Modifier le critère de comparaison des états du système pour sélection ou non d'un vecteur de test

Tests Progressifs Par Parties

Génération des scénarios

- L'algorithme ATP peut ainsi se décrire :

Soit **S** une sortie du système à tester ;
Pour chaque état **E** déterministe associé à **S** dans la table BTP *(Si la table BTP est vide, E correspond à l'état Indéterminé)* {
    On positionne l'oracle dans l'état **E** {
        Pour chaque vecteur $V_{candidat}=(e_d, e_i)$ parmi les 2^n vecteurs (n étant le nombre d'entrées de S) {
            On simule en positionnant les entrées avec $V_{candidat}=(e_d, e_i)$
            L'état de départ du système $E_p=(M_d,T_d,S_d)$ évolue alors suivant une succession d'états $Succ(E_p) = (E_{p1}, T_{p2}...T_{pk})$ avant stabilisation en un temps $TSucc(E_p)$
            S'il existe déjà un vecteur $V_{candidat}$' retenu ayant les mêmes entrées directes e_d que $V_{candidat}$ et faisant évoluer E_p de la même manière que $V_{candidat}$, c'est-à-dire si E_p passe successivement par les mêmes états, $\{E_{p1}, T_{p2}...T_{pk}\}$ en un temps total $TSucc(E_p)'<=TSucc(E_p)$ lors de l'application de $V_{candidat}$'.
            {
                On ne retient pas $V_{candidat}$
            }
            Sinon {
                On remplace le vecteur $V_{candidat}$' par $V_{candidat}$
            }
        }
    }
}

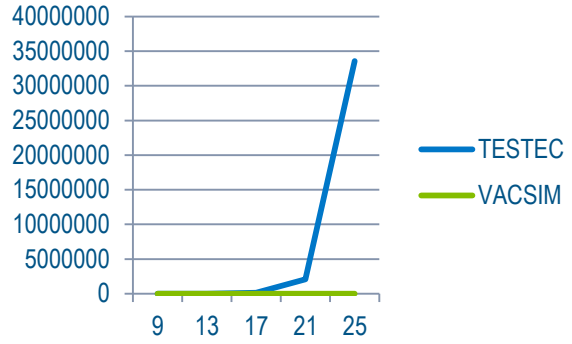
Pour tout état **E** et vecteur $V_{candidat}$ retenu :
Si $V_{candidat}$ correspond au dernier vecteur d'atteinte de l'état **E**, la séquence de test consiste à positionner **E**, et à tester la valeur de la sortie **S** après positionnement stabilisé de **E**
Sinon la séquence de test qui consiste à positionner **E** et à tester la valeur de la sortie S après positionnement stabilisé de **E** puis de façon continue lors de l'application de $V_{candidat}$

Tests Progressifs Par Parties

Résultats

- Exemple de la spécification : nombre de vecteurs retenus conforme à l'attendu
- Cas d'étude fourni par EDF
  - Une des sorties dépend de toutes les entrées
  - Comportement en combinatoire pure : pas de temporisations, pas de mémoires

Nombre d'entrées	Résultats TESTEC (Nombre de vecteurs retenus)	Résultats VACSIM (Nombre de vecteurs retenus)
9	512	89
13	8182	133
17	131072	206
21	2097152	284
25	33554432	370



Tests Progressifs Par Parties

Synchronisation entre les scénarios et le système sous test

- Modification des scénarios de test générés pour ControlBuild pour l'exécution en simulation fonctionnelle
  - Le scénario de test « pilote » la simulation fonctionnelle en exécutant des pas de simulation
  - Calcul du nombre de cycles nécessaires pour atteindre l'échéance des temporisations en fonction de la valeur du temps de cycle de la simulation

Tests Progressifs Par Parties

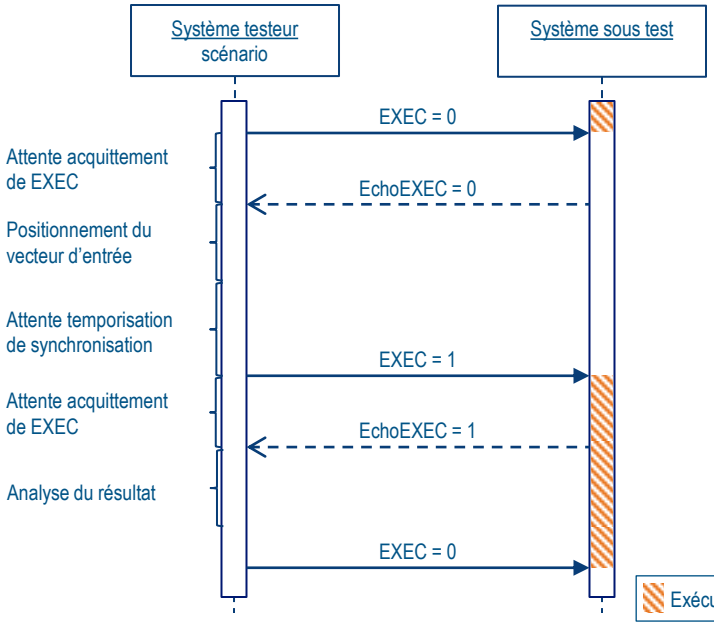
Synchronisation entre les scénarios et le système sous test

- Algorithmes définis par EDF
  - Propose deux déclinaisons d'une méthode pour assurer une synchronisation entre le scénario de test et le Système sous test : autoriser l'exécution du code du Système sous test uniquement quand le vecteur d'entrée est positionné intégralement
  - Nécessite une modification du code de ce dernier
  - Première déclinaison : attendre un temps suffisant (paramétrable) entre le positionnement du vecteur d'entrée et l'autorisation d'exécution
  - Seconde déclinaison : ajouter un principe d'acquittement

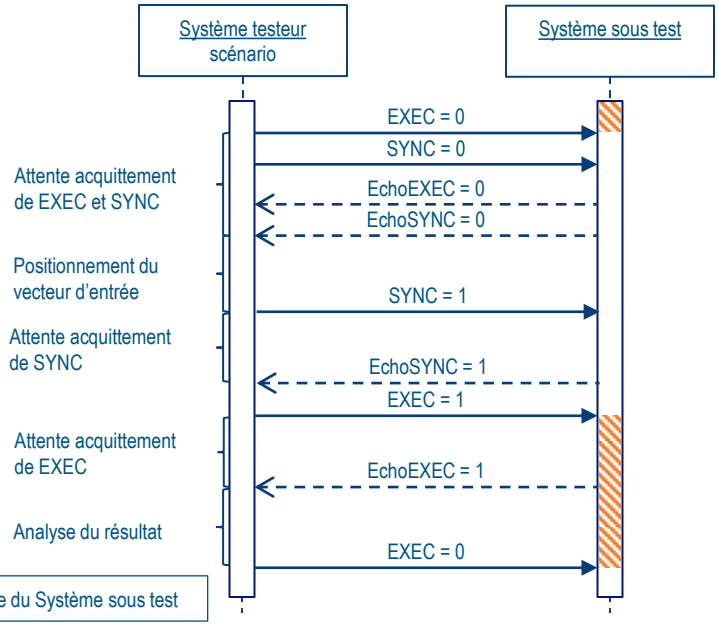
Tests Progressifs Par Parties

Synchronisation entre les scénarios et le système sous test

Algorithme 1

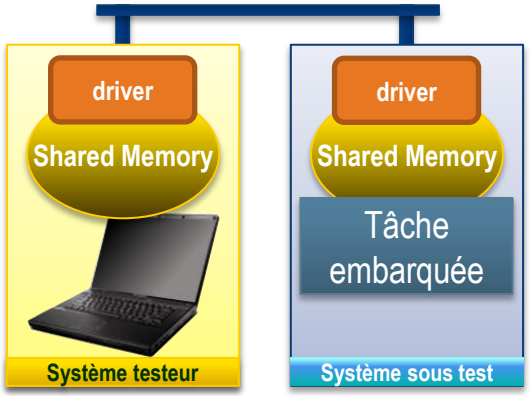


Algorithme 2



Exécution du code du Système sous test

Tests Progressifs Par Parties



Synchronisation entre les scénarios et le système sous test

- Solution étudiée
  - Pas de modification du Système sous test
  - Synchronisation entre le scénario et le driver de communication côté Système testeur par utilisation de variables système
    - Principe de « déclenchement » / « acquittement »
  - Synchronisation entre le driver de communication côté Système sous test et la tâche embarquée par utilisation d'une section critique
    - Assure la prise en compte d'un vecteur intègre pour le calcul par exclusion mutuelle

Tests Progressifs Par Parties

Vérification en boite blanche

- Objectif : Juger de la recevabilité des tests générés depuis la spécification sur une implémentation en effectuant les vérifications minimales
- Modification des tests de vérification en boite blanche
  - Conformité de la programmation
    - Absence de boucle
    - Instrumentation de toutes les mémoires
  - Comparaison des interfaces du système
  - Comparaison des dépendances pour chaque sortie entre la spécification et l'implémentation
    - Entrées / mémoires / temporisations et sorties directes

Tests Progressifs Par Parties

Vérification en boite blanche

- Vérification syntaxique à partir du format XML PLCOpen
- Pas de vérification du type des portes logiques, du type des mémoires, de la durée ou du type des temporisations
- Exemple :

Pour la sortie S : S = f(T1(E1, E2), M2, E6, T2(E7), S1, S2) ;

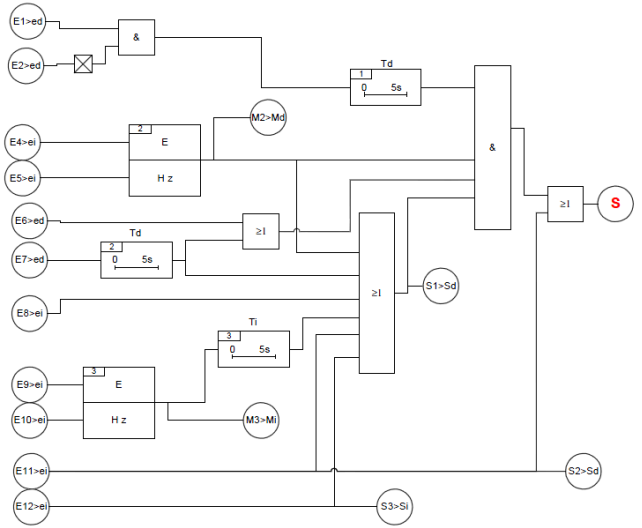
Pour la sortie S1 : S1 = f(M2, T2(E7), E8, T3(M3), S2, S3) ;

Pour la sortie S2 : S2 = f(E11) ;

Pour la sortie S3 : S3 = f(E12) ;

Pour la sortie M2 : M2 = f(M(E4, E5)) ;

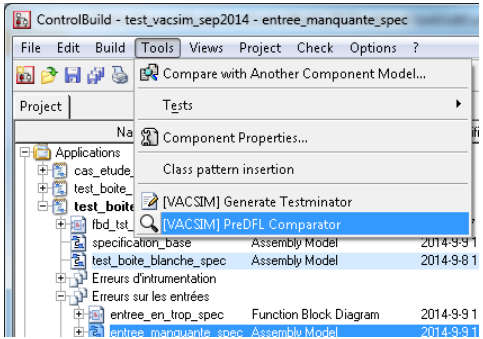
Pour la sortie M3 : M3 = f(M(E9, E10)).



Tests Progressifs Par Parties

Vérification en boite blanche

- Intégration dans ControlBuild par appel à un élément de menu
- Génération d'un rapport de vérification au format HTML



Specification (ControlBuild component)	Implementation (PLCOpen XML File)
General Information	
Project Path: e:/dev/ControlBuild/ControlBuild_2015/_Projects/test_vacsim_sep2014 Application Name: test_boite_blanche_spec ComponentName: entree_manquante_spec	File Path: E:/dev/ControlBuild/ControlBuild_2015/_Projects/test_vacsim_sep2014/Applications/test_boite_blanche_impl/functional/entree_manquante_impl/entree_manquante_impl.xml Project Path: E:/dev/ControlBuild/ControlBuild_2015/_Projects/test_vacsim_sep2014 Application Name: test_boite_blanche_impl ComponentName: entree_manquante_impl
Programmation conformance tests: OK	
Absence of loop: OK Memorization instrumentation: OK	Absence of loop: OK Memorization instrumentation: OK
Interfaces comparison: KO	
Input list: A, B, C, D, E Output list: S9, S8, S7, S6, S5, S4, S3, S1, S2, S0	Input list: A, B, C, D Output list: S9, S8, S7, S6, S5, S4, S3, S1, S2, S0
Dependencies check for each output	
Output S9: KO	
Direct inputs: E	Direct inputs: D
Output S8: KO	
Direct inputs: C, D	Direct inputs: C Direct outputs: S9
Output S7: OK	
Direct inputs: A, B	Direct inputs: A, B