Journées Systèmes et Logiciels Critiques
Institut IMAG ; 14 - 16 nombre 2000

Techniques et outils de test pour les logiciels réactifs synchrones

Farid Ouabdesselam

Méthodes de test : classification générale

- Test fonctionnel vs. Test structurel :

  information à l'origine de la sélection des données de test

  – test structurel : abstraction (modèle) du programme
    graphe de contrôle ou de données, …
  – test fonctionnel : spécification (partielle) du programme
    domaine d'entrée, fonction, propriétés
- Test déterministe vs. Test aléatoire ou statistique :

  détermination des données de test

  – par le testeur (exécution symbolique préalable éventuelle)
  – par l'outil (*aléatoire* : distribution uniforme, ou *statistique* : à partir d'un profil opérationnel)
- Test dirigé par spécifications informelles vs. formelles

Méthodes de test : classification générale

- Test statique vs. Test dynamique

  mode de génération des données de test

  – avant l'exécution du programme
  – au fur et à mesure de l'exécution du programme
- Objectif du test

  critères de pertinence des données de test

  – rechercher des défauts
  – évaluer la fiabilité
  – établir la conformité avec une spécification
- Construction de l'oracle : manuelle vs. automatique
- Définition de critères d'arrêt

Méthodes de test pour les logiciels synchrones

- Même variété que pour les méthodes générales
- Méthodes spécifiques vs. adaptation de méthodes générales
- Particularité commune : recherche d'une forte automatisation
- Méthodes spécifiques
  - le test est vu comme complémentaire à la vérification
  - l 'environnement de test doit être homogène avec ceux de spécification et de programmation
  - le test doit être bien fondé théoriquement

Méthodes spécifiques : particularités (1)

Validation des logiciels réactifs industriels

- spécifications formelles acceptées/recommandées
- model-checking souvent impraticable
- spécifications incomplètes/erronées

  $\Rightarrow$ *l'effort de spécification ne doit pas être perdu*

Apports du test : solutions pratiques à

- la validation dans des contextes imparfaits
- l'analyse de l'adéquation et la validité des spécifications

Méthodes spécifiques : particularités (2)

- Logiciels réactifs: test à base de séquences de cas de test
- Forte automatisation de la construction des cas de test
  - sélection des données de test dépend de contraintes temporelles
  - évaluation du verdict dépend de l'histoire des réponses

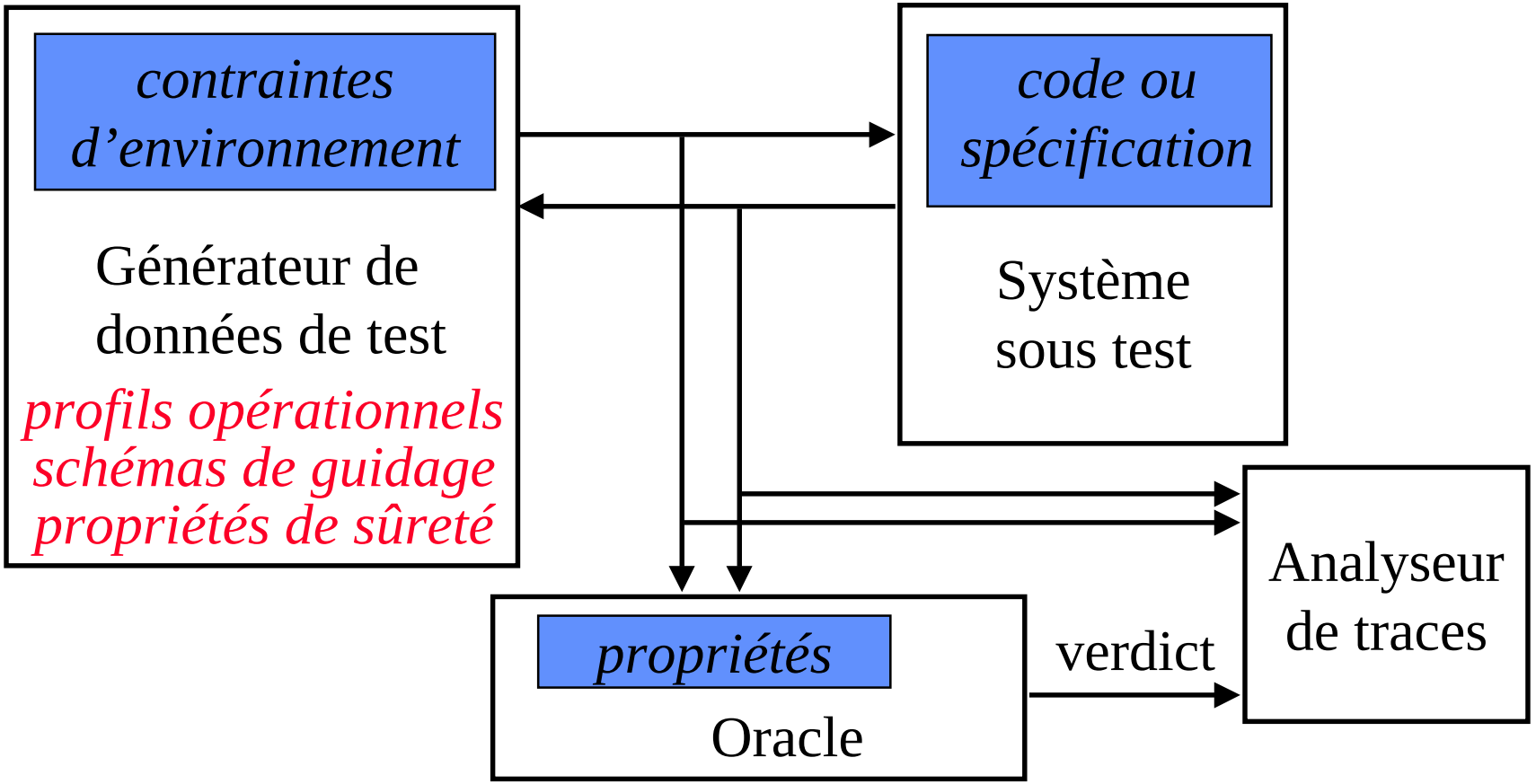
  => *meilleure utilisation des compétences du testeur humain*
- Pas d'hypothèses établies de régularité ou d'uniformité + comportements complexes à analyser => séquences de test potentiellement longues

  => *génération dynamique*

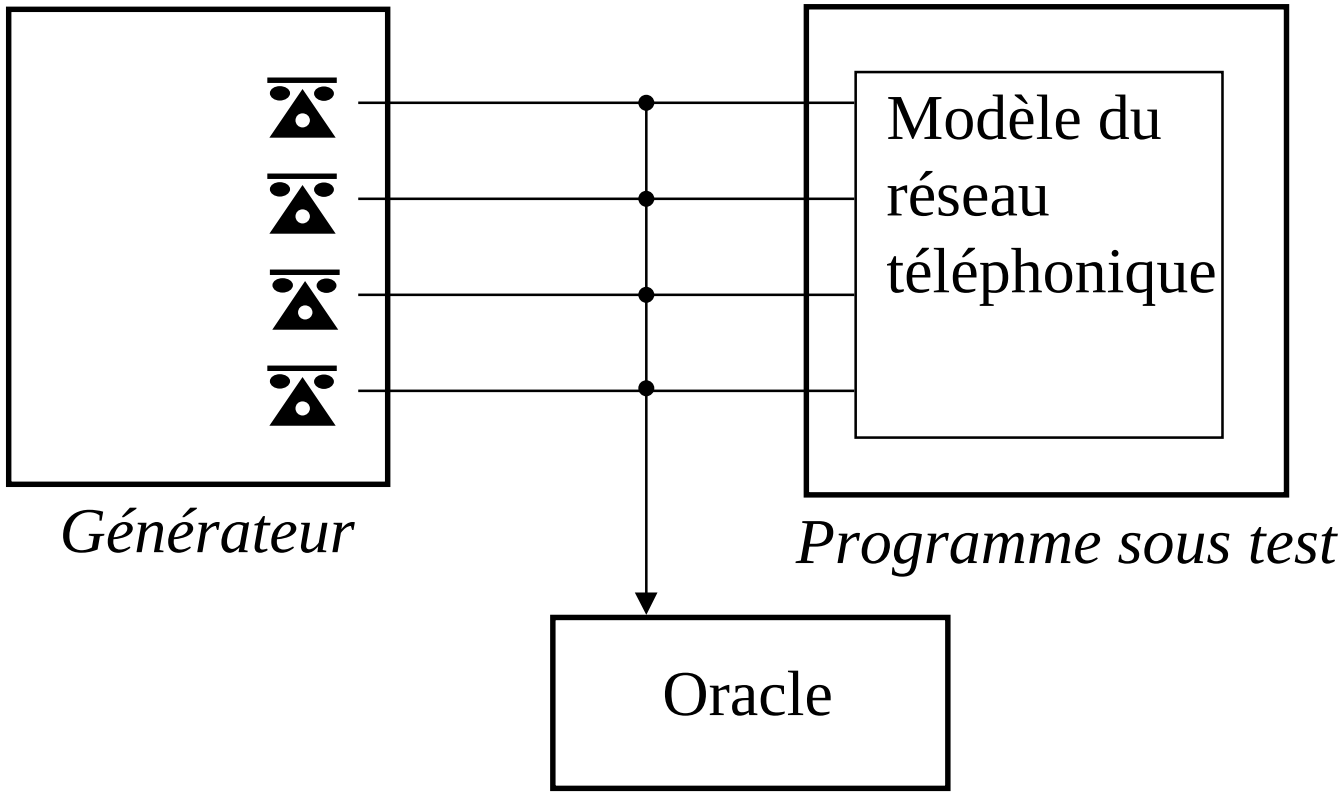
Méthodes spécifiques : particularités (3)

- Approche monoformalisme incluant
  - le langage de spécification
  - le langage de description des tests
  - le langage de programmation (éventuellement)
- Même technologie d ’implantation du test et de la vérif.
- Test dirigé par les spécifications :
  - expression complexe de l’environnement et de l’oracle
  - assurer à chaque technique de test un bon fondement théorique

Test fonctionnel
Principe de fonctionnement des outils



Application: recherche d'interactions entre services téléphoniques



Spécification de l'environnement en Lustre exemples de contraintes

- Contrainte physique :
  - *Il n'est pas possible de décrocher deux fois de suite sans raccrocher entre temps:*

    once On(A) **from pre** Off(A) **to** Off(A)
- Contraintes de modélisation :
  - *Il y a au plus une action à chaque instant:*

    #(On(A), Off(A), Dial(A),..., On(D), Off(D), Dial(D))
  - *Le nombre de numéros filtrés est limité à 10*

    (0 ≤ Black_list (A)) **and** (Black_list (A) ≤ 10)

Oracle

- Un observateur (un code synchrone exécutable)
  - soit fourni par le testeur
  - soit engendré automatiquement à partir d'un ensemble de propriétés Lustre (portant sur les entrées et sorties)

- Un verdict à chaque cycle

  => trace = séquence de triplets (entrées, sorties, verdict)

- Exemple de propriété d'oracle :

  Un appel ne peut pas être transféré par un abonné qui n'a pas souscrit au transfert d'appel

Génération automatique des données de test Algorithme de base (Lurette et Lutess)

- Problème : (q : état ; E : vecteur de valeurs pour les entrées)

  $\forall q \in Q,\ \exists E \in D_1 \times D_2 \times \ldots \times D_n$ tq Env(E , q)

  problème en général indécidable
- Solutions : Env fonction des entrées courantes et des sorties antérieures
  - $\forall i\ D_i$ = Bool (entrées = signaux)

    à chaque instant, tous les vecteurs satisfaisant Env ont la même probabilité d'être sélectionnés
  - $\forall i\ D_i$ = Bool ou Réel ou Entier, et contraintes linéaires

    à chaque instant, un vecteur satisfaisant Env est sélectionné

Génération aléatoire de données de test pour des programmes booléens (Lutess)

Génération automatique de valeurs contraintes

- Génération aléatoire équiprobable
- Génération à partir de profils opérationnels
- Génération guidée par des propriétés (de sûreté)
- Génération guidée par des schémas de comportement

en vue de

- Rechercher des erreurs de type fonctionnel
- Tester par rapport à des propriétés (de sûreté)
- Évaluer la fiabilité

Génération à base de profils opérationnels

- Bonnes estimations de fiabilité demandent un test du produit dans ses conditions d'exploitation
- Or, la génération équiprobable est loin d'être réaliste

  exemple : un abonné s'appelle aussi souvent qu'il appelle les autres
- Profil opérationnel (cf. Musa) : caractérisation quantitative de la façon dont un système est utilisé
- Distribution opérationnelle = distribution statistique des vecteurs d'entrées

  => *test fonctionnel statistique*

Génération à base de profils opérationnels

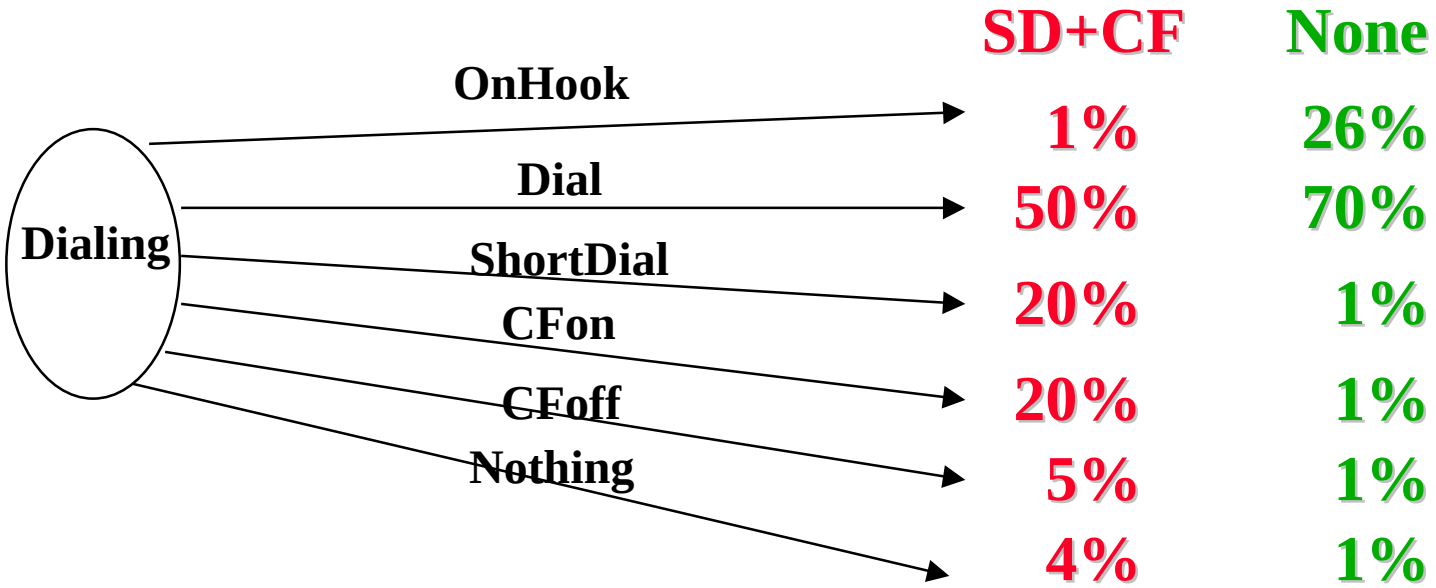
- Difficulté à construire un profil opérationnel complet
- Enrichissement des contraintes d'environnement avec des probabilités conditionnelles associées à chaque variable d'entrée
- Exemples:

  <Dialed_Number=A, 0.01, Dial(A)>

  <On(A), 0.9, pre Exception(A)>
- Lutess permet au testeur de convertir des probabilités conditionnelles en des profils opérationnels, et vice-versa

Génération à base de profils opérationnels

- Un abonné peut utiliser différents services (Short Dial, Call Forward)
- La distribution des entrées peut dépendre des services souscrits



Génération guidée par des propriétés invariantes

- Le testeur énonce des propriétés devant être vérifiées :
  - propriétés de sûreté
  - situations spécifiques

- Toute donnée de test n'est pas pertinente pour le test d'une propriété

  P : (input => output)

- Recherche d'une violation instantanée :

  A l'instant *t* une donnée teste une propriété *P* s'il existe une sortie du programme rendant *P* fausse à l'instant *t*

Génération guidée par des propriétés invariantes

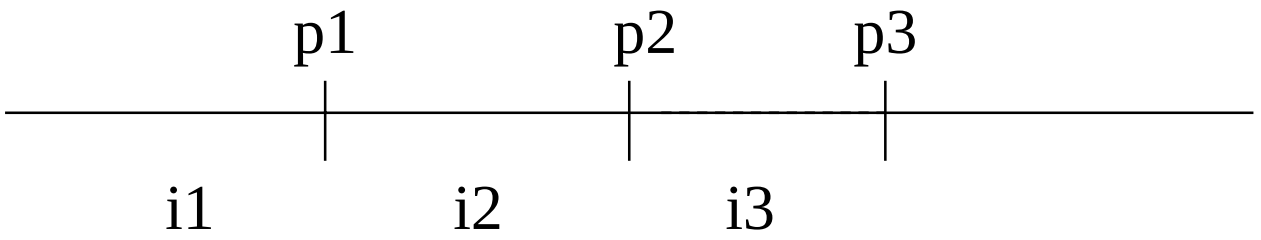
- Propriété:

  Chaque fois qu'un abonné raccroche son téléphone retourne dans l'état de repos:

  On(A) => Idle(A)

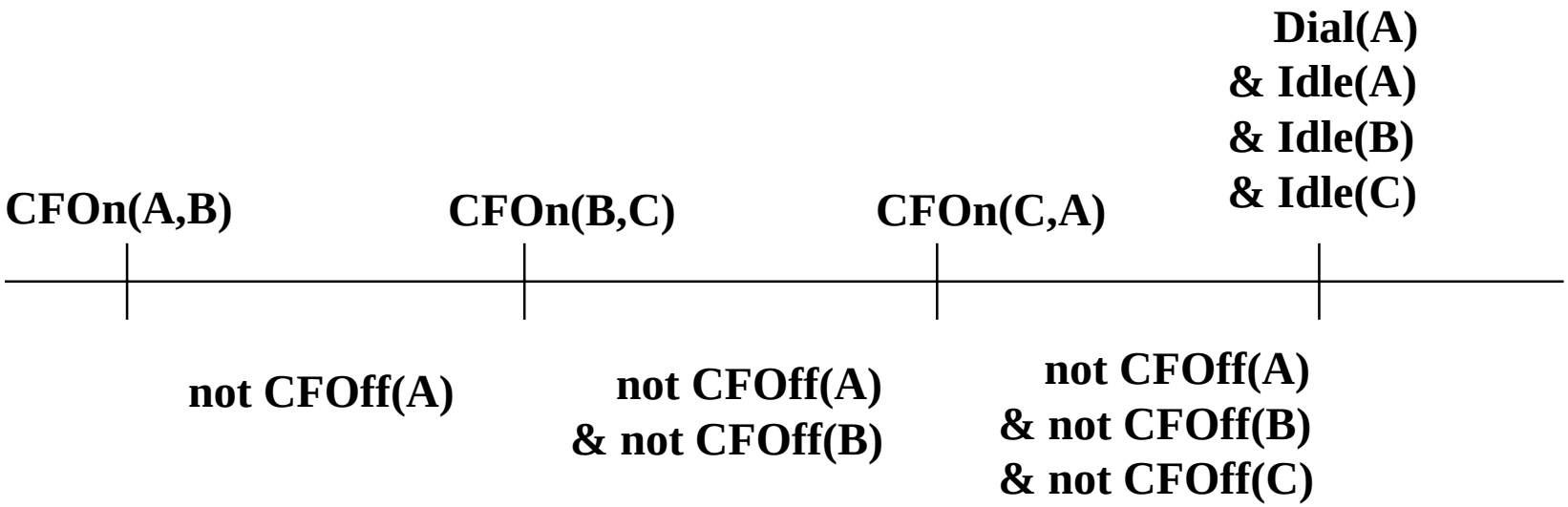
- Ce type de guidage améliore la confiance dans le logiciel testé, mais il tend à rendre impossible l'apparition d'autres comportements

Génération guidée par des schémas



- But : pouvoir observer des situations complexes d ’occurrence peu probable par les autres techniques
- Un schéma de comportement décrit une classe de séquences de test
- Langage des schémas: extension des expressions régulières
- Traduction systématique en expressions de logique temporelle (Lustre)

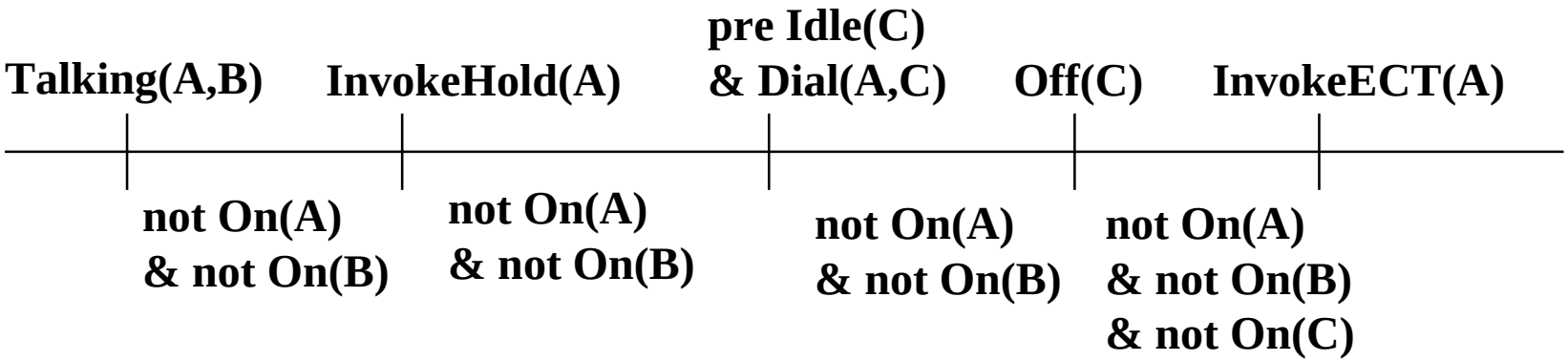
Génération guidée par des schémas



Ce schéma permet d'observer la réaction du système quand le nombre maximum de renvois est atteint

Génération guidée par des schémas

Service « Explicit Call Transfer » : permet à un abonné ayant deux communications en cours de mettre en relation ses deux correspondants



Ce schéma est observé 60 fois au lieu de 2 dans une séquence de 10 000 pas de tests

Adaptation de méthodes générales

Test structurel statistique

- modèle : automate résultant de la compilation
- critère : couverture de transitions (1 cycle)
- génération d'une distribution statistique maximisant la probabilité d'activation de chaque transition
- génération d'un critère d'arrêt en fonction d'un niveau de qualité défini et la probabilité de la transition la moins probable

complété par un choix déterministe de données

(état initial et valeurs aux bornes dans les conditions de transitions)

Adaptation de méthodes générales

Test mixte (structurel/fonctionnel) déterministe

Outil GATEL

- génération statique de séquences minimales
- objectif de test : - propriété invariante à tester
  - prédicat à vérifier au dernier cycle
- génération par « exécution symbolique » du programme (en Lustre) à partir de l'objectif de test
- testeur fixe le mode de décomposition pour l'exécution symbolique

Validation - Études de cas industrielles

- GATEL, Lurette, Prototype de test structurel statistique :

  Action Forma (96-98) : système de comptage de neutrons (Schneider-Électrique)

- Lutess :

  1er concours international de recherche d ’interactions entre services téléphoniques (98) : prix du « Meilleur outil »

Bibliographie

[1] L. du Bousquet, F. Ouabdesselam, and J.-L. Richier. Expressing and implementing operational profiles for reactive software validation. In *9th International Symposium on Software Reliability Engineering*, Paderborn, Germany, 1998.

[2] L. du Bousquet, F. Ouabdesselam, J.-L. Richier, and N. Zuanon. Lutess: a specification-driven testing environment for synchronous software. In *21st International Conference on Software Engineering*, pages 267–276. ACM Press, May 1999.

[3] B. Marre and A. Arnould. Test Sequences Generation from Lustre Descriptions : GATeL. In *15th IEEE International Conference on Automated Software Engineering*. IEEE, September 2000.

[4] F. Ouabdesselam and I. Parissis. Testing Synchronous Critical Software. In *5th International Symposium on Software Reliability Engineering*, Monterey, USA, 1994.

[5] I. Parissis and F. Ouabdesselam. Specification-based Testing of Synchronous Software. In *4th ACM SIGSOFT Symposium on the Foundation of Software Engineering*, San Francisco, USA, 1996.

[6] P. Raymond. Recognizing regular expressions by means of dataflows networks. In *23rd International Colloquium on Automata, Languages, and Programming, (ICALP'96)* Paderborn, Germany. LNCS 1099, Springer Verlag, july 1996.

[7] P. Raymond, D. Weber, X. Nicollin, and N. Halbwachs. Automatic testing of reactive systems. In *19th IEEE Real-Time Systems Symposium (RTSS'98)*. IEEE, 1998.

[8] P. Thevenod-Fosse, C. Mazuet, and Y. Crouzet. On statistical structural testing of synchronous data flow programs. In *First European Dependable Computing Conference*, Berlin, Germany, october 1994.