TOBIAS : un outil de test combinatoire pour le test de conformité

Y. Ledru, P. Bontron, L. du Bousquet, O. Maury, C. Oriat

LSR-IMAG
BP 72, 38402 St Martin d'Hères cedex, France
Tel : +33 4 76 82 72 14, Fax : +33 4 76 82 72 87
Yves.Ledru@imag.fr

Résumé

TOBIAS est un outil de génération combinatoire de cas de tests. Son principe est le dépliage combinatoire d'un « schéma de test » qui donne une vision abstraite de l'ensemble des tests à générer. Ses principales utilisations à ce jour concernent le test de conformité où il est utilisé avec une spécification exécutable en termes de contrats (JML ou VDM) ou une spécification IOLTS (TGV). Les principales forces de l'outil sont le gain de productivité dans le développement des cas de test et ses capacités à trouver des erreurs de par son caractère systématique.

1. INTRODUCTION

Le test de logiciel est aujourd'hui l'une des techniques les plus utilisées pour les activités de validation et de vérification. Nos travaux s'intéressent principalement au *test de conformité* [1], qui suppose l'existence d'une spécification du logiciel à tester.

- Elle constitue la référence à laquelle le comportement de l'implantation doit être comparé. Elle constitue un oracle et permet ainsi de décider de la réussite ou de l'échec de chaque test.
- Elle décrit l'ensemble des comportements admissibles et permet de définir des notions de couverture. Il est alors possible d'évaluer la pertinence d'un jeu de tests.
- Elle peut constituer le point de départ de la génération, manuelle ou automatique, des jeux de tests.

Au cours du projet RNTL COTE[1], notre équipe a développé l'outil TOBIAS [2], dont le principe a été élaboré à partir de constatations expérimentales [3] : la validation d'une application industrielle nécessite de nombreux tests, et nombre d'entre eux ne diffèrent que par les valeurs de leurs paramètres, les instances auxquelles ils s'appliquent, ou par le nombre d'appels consécutifs à une méthode ou un groupe de méthodes. Dans TOBIAS, l'utilisateur peut exploiter ces similarités pour produire un ensemble de tests à partir d'une description abstraite, dite « schéma de test ». Le mécanisme d'abstraction offert par ces schémas permet au testeur de gérer facilement la taille du jeu de tests. La pertinence du jeu de test reste conditionnée par la qualité des schémas de test, i.e. par l'expérience et le travail d'analyse du testeur. En résumé, TOBIAS est un outil de test combinatoire [10], qui génère l'ensemble exhaustif des cas de test correspondant à un schéma.

Cet article présente brièvement les principaux composants de l'outil (Sect. 2) et un exemple (Sect. 3) pour les illustrer. Il présente le traitement de l'exemple (Sect. 4) et fait un bref bilan des points forts et points faibles de l'outil.

2. PRINCIPAUX COMPOSANTS DE L'OUTIL

La Fig. 1 donne une vue d'ensemble de l'outil et de son environnement. Le point de départ de la génération est la description des signatures des méthodes des classes de l'application à tester. Sur cette base l'utilisateur définit, avec l'aide de l'outil des « schémas de test » qui sont une description abstraite de l'ensemble des tests à réaliser. L'outil peut ensuite déplier ces schémas en un grand nombre de données de test, cas de tests ou objectifs de test (selon le contexte). Dans la plupart des cas, l'outil fournit simplement des données de test, destinées à solliciter l'application à tester. L'oracle du test est alors fourni par une spécification exécutable de l'application. Un pilote de tests sollicite ainsi l'application et l'oracle pour produire divers résultats de test (erreurs et éventuellement couverture). Le pilote de test peut être un outil standard, comme JUnit, mais nous avons conçu des versions adaptées à TOBIAS qui exploitent les similarités des tests pour optimiser leur exécution [8].En pratique, l'outil permet la génération d'objectifs de test pour l'outil TGV [5], ou de cas de test pour VDM [6] ou Java/JML [8]. Dans tous les cas, une spécification (exécutable) de l'application est nécessaire. Nous illustrons cet article par un exemple en VDM, dont [8] présente la version JML.

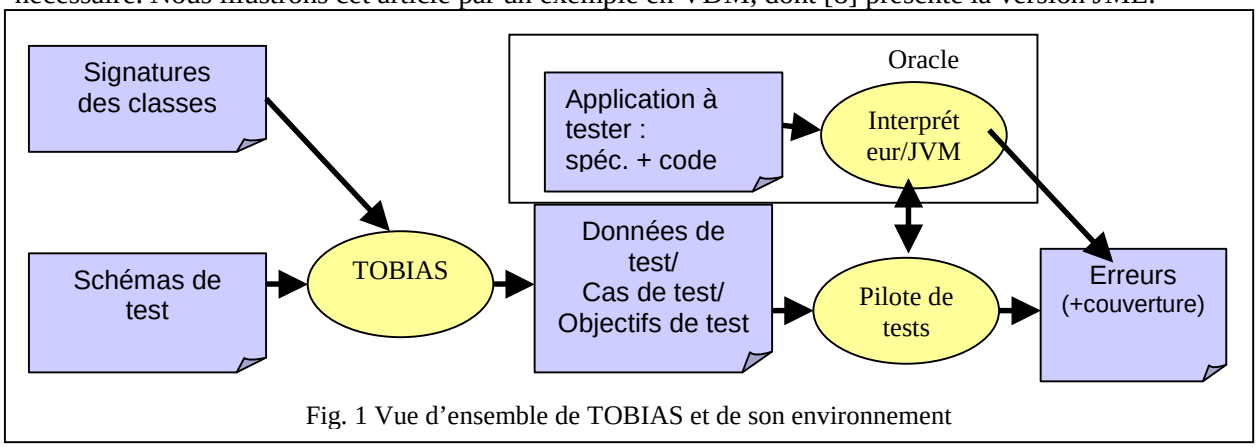

Fig. 1 Vue d'ensemble de TOBIAS et de son environnement

[1] COTE a rassemblé des équipes de Softeam, de France Télécom R&D, de Gemplus, de l'IRISA et du LSR, autour du test de conformité de composants logiciels (2000-2002).

3 UNE ETUDE DE CAS

Soit un système élémentaire de gestion de tampons, dont la spécification VDM [4] est donnée Fig. 2. Le système est composé de trois tampons (`b1`, `b2` et `b3`), représentés par des variables entières qui représentent le nombre d'éléments de chaque tampon. La taille maximale supportée par le système est de 40 éléments. Le système doit répartir les éléments pour que `b1` soit plus petit que `b2`, lui même que `b3` et qu'il y ait au plus 15 éléments d'écart entre `b1` et `b3`. Ceci est spécifié par l'invariant (Fig. 2).

L'état du système peut être modifié par trois opérations : `Initialiser` qui remet les tampons à zéro, `Ajouter` qui prend en paramètre un entier strictement positif représentant le nombre d'éléments à rajouter dans les tampons, et `Enlever` qui prend en paramètre un entier strictement positif représentant le nombre d'éléments qui sont retirés des tampons. Ces deux dernières opérations doivent répartir la charge entre les éléments de sorte que l'invariant soit respecté. La spécification des opérations est définie en termes de pré et post-conditions. Les identificateurs marqués par un ~ dans la post-condition correspondent à la valeur de ces variables dans l'état initial.

```
state buffers of
            b1 : nat
            b2 : nat
            b3 : nat
      inv mk_buffers(b1,b2,b3) ==
            b1+b2+b3<=40
            and 0<=b1
            and b1<=b2
            and b2<=b3
            and b3-b1<=15
      init B == B =
            mk_buffers(0,0,0)
      end
```

```
Initialiser: ()==>()
  Initialiser() == …
  pre true
  post b1+b2+b3=0
Ajouter: nat ==> ()
  Ajouter(nb) == …
  pre nb>0 and nb<=5
      and b1+b2+b3+nb<=40
  post b1+b2+b3 = b1~+b2~+b3~+nb
Enlever: nat ==> ()
  Enlever(nb) == …
  pre nb>0 and nb<=5
      and nb<=b1+b2+b3
  post  b1+b2+b3  =  b1~+b2~+b3~-nb
```

Fig. 2 Une spécification VDM du système des tampons.

4. TOBIAS

Le principe. Plusieurs expériences industrielles, dont [3], ont montré que les tests d'une même suite de tests présentent de nombreuses similarités. Beaucoup sont identiques aux valeurs des paramètres ou à l'ordre des opérations près. TOBIAS a été conçu pour aider l'utilisateur dans sa démarche de conception de tests similaires. L'utilisateur doit fournir une description abstraite des séquences de tests (*schéma de test*). Chaque schéma est « déplié » par TOBIAS et transformé en tests exécutables.

TOBIAS prend en entrée un diagramme de classes et en extrait les classes et signatures des méthodes. L'utilisateur sélectionne des ensembles de valeurs pertinentes pour chaque paramètre de chaque opération/méthode. Il peut aussi identifier et regrouper une ou plusieurs méthodes sous une même étiquette (appelée « groupe »). A partir de ces éléments, des schémas de test peuvent être définis. Un schéma est une forme d'expression régulière sur les opérations et les groupes.

Soit **S1** un schéma visant à tester l'ajout de nouveaux éléments dans le système après initialisation.

S1 : « `Initialiser(); AjouterGr` »

avec « `AjouterGr =` {`Ajouter(x)|x`∈{`1,2,3,4,5`}} ».

`AjouterGr` est un groupe composé de 5 instanciations différentes de la méthode `Ajouter`. A partir des valeurs des paramètres données ci-dessus et de **S1**, TOBIAS génère 5 séquences :

```
Initialiser(); Ajouter(1)
Initialiser(); Ajouter(2)
...
Initialiser(); Ajouter(5)
```

Ces séquences de test sont ensuite exécutées sur l'application à tester. Un outil comme VDMTools [4] permet de comparer les résultats calculés par le code aux propriétés exprimées par la spécification et d'ainsi détecter des divergences entre le code et sa spécification.

Les groupes permettent aussi de rassembler des méthodes différentes sous une même étiquette. Soit **S2** un schéma pour tester l'évolution du système selon les opérations `Ajouter` et `Enlever`.

S2 : « `Initialiser(); Modifier^1..2` »

avec `Modifier =`{`Ajouter(x)|x`∈{`1,2,3,4,5`}}∪{`Enlever(y)|y` ∈{`1,3,5`}}

Les séquences de **S2** initialisent le système puis exécutent un ou deux appels aux méthodes du groupe `Modifier` (8+8*8=72 séquences).

```
Initialiser(); Ajouter(1)
...
Initialiser(); Enlever(5)
Initialiser(); Ajouter(1); Ajouter(1)
...
Initialiser(); Enlever(5); Enlever(5)
```

Forces et faiblesses de TOBIAS. TOBIAS permet de décharger le testeur des tâches répétitives pour se concentrer sur les aspects créatifs de la génération de tests. En quelques lignes, l'ingénieur de test exprime un schéma qui se dépliera en centaines voire milliers de tests. L'expérience décrite dans [6] a comparé la génération de 4320 séquences de test avec TOBIAS à la production manuelle de 45 tests. Elle a montré que la conception des deux suites de test nécessitait un effort comparable, qu'elles offraient le même taux de couverture, mais que les tests TOBIAS permettaient de trouver plus d'erreurs que l'autre suite. D'autres expériences menées avec Gemplus ont confirmé les capacités de l'outil à exprimer de façon concise une suite de test, ce qui mène à des gains de productivité, ou à trouver des erreurs, notamment grâce au caractère systématique des tests et à leur grand nombre.

Côté faiblesses, il est facile d'exprimer des schémas dont le dépliage devient explosif. La maîtrise de cette explosion combinatoire passe d'abord par l'expression de schémas adaptés, qui se déplient en un nombre raisonnable de cas de test. Nous avons cependant développé des techniques de filtrage des tests, soit lors de leur génération, soit à l'exécution. Ces techniques de filtrage se basent sur l'évaluation de prédicats sur les paramètres des appels de méthodes ou lors de l'exécution. Ces prédicats peuvent se baser sur les préconditions de la spécification, ou sont fournis par l'utilisateur. Ces techniques permettent de réduire significativement le nombre de tests générés ; ainsi des exemples présentés dans [8] montrent une réduction par un facteur 6 du nombre de tests générés.

Une autre faiblesse de TOBIAS est la sélection des valeurs des paramètres. Pour l'instant, c'est l'ingénieur de test qui fournit ces valeurs à l'outil. Cependant une expérience [9] de couplage avec l'outil Ucasting [7] a montré les capacités de coupler l'outil avec des outils de synthèse de données.

5. EN RESUME

TOBIAS est un outil de test combinatoire qui génère une grande quantité de tests à partir d'un schéma de test. Il est destiné à être utilisé avec une spécification exécutable de l'application à tester. Actuellement, l'outil a été utilisé avec des spécifications JML de programmes Java, des spécifications VDM de programmes écrits dans ce langage et des spécifications en termes d'IOLTS pour l'outil TGV. Dans le cas de VDM et de JML, plusieurs techniques ont été mises en œuvre pour maîtriser l'explosion combinatoire du nombre de tests générés ou exécutés.

BIBLIOGRAPHIE

[1] B. Beizer : Software Testing Techniques. Van Nostrand Reinhold, 1990.

[2] P. Bontron, O. Maury, L. du Bousquet, Y. Ledru, C. Oriat, et M.-L. Potet : TOBIAS : un environnement pour la création d'objectifs de tests à partir de schémas de tests. In ICSSEA, déc. 2001.

[3] L. du Bousquet, H. Martin, and J.-M. Jézéquel. Conformance Testing from UML specifications, Experience Report. In Gesellschaft für Informatik (GI), p-UML workshop, Lecture Notes in Informatics. vol. P-7, 2001.

[4] The VDM Tool Group : VDM-SL Toolbox User Manual. Technical report, IFAD, Oct. 2000. ftp://ftp.ifad.dk/pub/vdmtools/doc/userman_letter.pdf.

[5] T. Jéron and P. Morel : Test Generation Derived from Model-checking. In Computer Aided Verification (CAV). LNCS 1633, Springer, 1999.

[6] O. Maury, Y. Ledru, P. Bontron, and L. du Bousquet. Using TOBIAS for the automatic generation of VDM test cases. In Third VDM Workshop (in conjunction with FME2002), Copenhagen, Danemark, 2002.

[7] L. Van Aertyck, Th. Jensen. UML-CASTING: Test synthesis from UML models. Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL), Janv. 2003

[8] Y. Ledru, L. du Bousquet, O. Maury, and P. Bontron. *Filtering TOBIAS combinatorial test suites*. In Proceedings of ETAPS/FASE'04 - Formal Aspects of Software Engineering , LNCS 2984, Springer, 2004.

[9] O. Maury, Y. Ledru and L. du Bousquet. *Intégration de TOBIAS et UCASTING pour la génération des tests.* In 16th Int. Conf. Software & Systems Engineering and their Applications-ICSSEA'2003, Paris, 2003.

[10] D. M. Cohen, S. R. Dalal, J. Parelius, and G. C. Patton. The Combinatorial Design Approach to Automatic Test Generation, *IEEE Software*, v. 13, n. 5, September 1996.