Un outil d'assistance à la construction de tests de modèles à composants et services

André, Pascal Ardourel, Gilles Mottu, Jean-Marie
Sunyé, Gerson

LS2N UMR CNRS 6004
Université de Nantes, IMT-Atlantique, Inria
Prenom.Nom@univ-nantes.fr

Dans l'article *"COSTOTest : A Tool for Building and Running Test Harness for Service-Based Component Models"* publié dans les proceedings de la conférence internationale *ISSTA 2016* [1] et présenté en session démonstration, nous décrivons comment l'outil COSTOTest nous assiste pour tester directement les modèles de composants logiciels basés sur les services. Ces travaux concernent la vérification de systèmes logiciels à base de composants et services (*Service-based Component* ou SBC) et exploitent l'ingénierie dirigée par les modèles (IDM).

Contexte Tester le plus tôt possible permet de réduire le coût du processus de vérification et de validation [2]. En IDM, c'est encore plus vrai et la correction des modèles est essentielle car ils sont le point de départ de transformations plus ou moins directes vers les modèles opérationnels et le code. Alors que les modèles servent de spécification dans les approches *Model-Based Testing* (MBT) pour générer des tests exécutés sur le code final, nous considérons la vérification de ces modèles qui peuvent être erronés [3]. Nous contribuons donc à la vérification des modèles en les testant directement (*Model Testing* MT). Ces tests sont complémentaires d'autres vérifications appliquées sur les modèles (preuves, model checking [4]). L'effort est porté sur la détection en amont des erreurs "métiers" (*platform independent*), coûteuses à corriger lorsqu'elles sont repérées tardivement et propagées dans différentes versions et implantations du système. De plus, le test de modèles permet de s'affranchir des erreurs spécifiques à l'implantation (*platform specific*), et réduit la complexité globale du test [5]. En exploitant l'IDM, les tests sur les modèles sont évolutifs car ils sont eux-mêmes des modèles. On bénéficie ainsi des mêmes outils de transformation de modèles que le système sous test.

Objectifs Nous ciblons les tests fonctionnels (unitaire, intégration, recette fonctionnelle hors IHM) pour des modèles à composants et services, ayant un niveau de description suffisamment précis et détaillé pour pouvoir exécuter les tests. Assurer la correction de ces modèles reste un défi. La vérification formelle permet de filtrer

une partie des modèles erronés mais elle ne suffit pas parce que les outils restent limités face à la combinatoire de langages expressifs. Nous considérons le test pour améliorer le niveau de correction. Notre objectif est de tester ces modèles à composants c'est-à-dire de concevoir des cas de tests, de les appliquer sur les modèles mis dans un contexte adéquat pour être exécutés et obtenir un verdict.

Processus Le processus part d'une intention de test (variables et oracles) et d'un modèle sous test. Nous considérons la construction de *harnais de test* unitaire ou d'intégration pour des systèmes à composants et services, qui permettent de passer des données de test, d'exécuter les tests, et de récupérer le verdict (succès ou échec du test). Pour réduire l'effort de construction du harnais de test, nous proposons une méthode qui guide le testeur dans le processus de conception des tests. Sachant que le testeur découvre l'application à tester et que les intentions de test restent informelles, il a besoin d'itérer jusqu'à ce que le niveau de précision soit atteint pour que le système soit testable. Il a besoin d'être assisté pour définir la forme véritable du test et déterminer la partie du système nécessaire au test. Définir concrètement l'oracle et comment trouver ou fournir les données de test est difficile : par exemple, pour atteindre une variable, il faut trouver les services qui la manipulent ou qui en dépendent. L'assistance à la construction est basée sur la détection d'incohérences et d'incomplétude entre le harnais et le modèle de test ainsi que sur des propositions générant les éléments manquants. Le programme de test est alors transformé vers une plateforme technique dédiée à l'exécution des tests.

Mise en œuvre La mise en œuvre est réalisée avec un outillage qui permet d'expérimenter l'approche. Il est mis en œuvre dans COSTO (*COmponent Study Toolbox*), la plate-forme Eclipse (`http://costo.univ-nantes.fr/`) dédiée aux modèles à composants écrits avec le DSL Kmelia. Nous avons développé un *framework* en Java qui donne un cadre d'exécution et d'animation pour les modèles de test (plateforme spécifique). L'approche est illustrée sur un cas d'étude simple, un système de *platoon* de véhicules.

Références

[1] Pascal André, Jean-Marie Mottu, and Gerson Sunyé. Costotest : a tool for building and running test harness for service-based component models (demo). In Andreas Zeller and Abhik Roychoudhury, editors, *Proceedings of the 25th International Symposium on Software Testing and Analysis, ISSTA 2016, Saarbrücken, Germany, July 18-20, 2016*, pages 437–440. ACM, 2016.

[2] Graeme Shanks, Elizabeth Tansley, and Ron Weber. Using ontology to validate conceptual models. *Commun. ACM*, 46(10) :85–89, October 2003.

[3] Martin Gogolla, Jørn Bohling, and Mark Richters. Validating uml and ocl models in use by automatic snapshot generation. *Software and Systems Modeling*, 4(4) :386–398, 2005.

[4] Pascal André, , Christian Attiogbé, and Jean-Marie Mottu. Combining techniques to verify service-based components. In *Proceedings of the International Workshop on domAin specific Model-based AppRoaches to vErificaTion and validaTiOn, AMARETTO@MODELSWARD 2017, Porto, Portugal, February 19-21, 2017.*, pages 645–656, 2017.

[5] Marc Born, Ina Schieferdecker, Hans-gerhard Gross, and Pedro Santos. Model-driven development and testing - a case study. In *First European Workshop on MDA with Emphasis on Industrial Application*, pages 97–104. Twente Univ., 2004.