BIBLIOTHEQUE DU CERIST

Doctoral School EDITE

Thesis submitted for obtaining the

PHD DEGREE IN COMPUTER SCIENCE

Doctorate jointly delivered by

Telecom & Management SudParis and Pierre et Marie Curie University- Paris 6

Speciality:

COMPUTER SCIENCE

Presented by

Fayçal Bessayah

Title

A Complementary Approach for Testing System Robustness Based on Passive Testing and Fault Injection Techniques

Committee in charge :

Nina Yevtushenko	Reviewer	Tomsk State University
Ismael Rodríguez Laguna	Reviewer	Complutense University of Madrid
Fatiha Zaidi	Examiner	Universisty of Paris-Sud XI
Sébastien Tixeuil	Examiner	Pierre et Marie Curie University
Eliane Martins	Co-advisor	State University of Campinas
Ana Cavalli	Co-advisor	IT/Telecom SudParis

BIBLIOTHEQUE DU CERIST

Acknowledgments

I would like to thank Professor Ana Cavalli for her excellent support and dedication during all the time I spent working on this PhD thesis. I am very grateful for the time she spent helping me and guiding my researches. I would like also to thank her for granting me the freedom of developing my ideas and for her suggestions and advices throughout this work.

A special thanks also to my co-adviser, Professor Eliane Martins from the State University of Campinas, for her advices and guidance in preparing this thesis. I really learned a lot of things by working with her and benefited from her experience as well as her excellent research and technical skills.

I would like also to thank Doctor Amel Mammar a lecturer at Telecom SudParis who helped me and encouraged me a lot. A particular thanks to her for having read my manuscript and for all the suggestions she made to improve the quality of this document.

More thanks go to my thesis evaluation committee consisting of Professor Nina Yevtushenko, Professor Ismael Rodríguez Laguna, Professor Sébastien Tixeuil, Doctor Fatiha Zaidi, Professor Eliane Martins and Professor Ana Cavalli.

Many thanks also to all my friends and colleagues, in no particular order, Bakr Sarakbi, Wissam Mallouli, Willy Jimenez, Mounir Lallali, Felipe lalanne, Mazen Al Maarabani, Anis laouiti, Anderson Morais, Farouk 'Aissanou, Jose Pablo Escobar, Mohamed Ahmed Mohamed Sidi and Bachar Wehbi. Thanks also to Mme. Brigitte Laurent and Mme. Jocelyne Vallet for their help and support completing the non technical part of my work.

Last and not least, I would like to thank my mother, my grandmother and all my brothers and sisters Nassim, Souad and Radia for their encouragement and support. Thanks a lot.

BIBLIOTHEQUE DU CERIST

Résumé

Que ce soit dans le domaine des transports, des énergies ou des banques, les systèmes informatiques sont immanquablement présents. Nous confions ce que nous avons de plus cher, à savoir nos vies et nos biens, à des programmes informatiques.

Parallèlement, cela va sans dire que ces systèmes sont de plus en plus complexes. Une complexité due essentiellement à une expansion sans précédent de systèmes largement distribués et hétérogènes. Sans parler de l'utilisation d'Internet comme principal réseau de transport de données, partagé par un nombre colossal de services et d'applications Web. Face à cette complexité croissante, tout dysfonctionnement, même temporaire, de ces systèmes peut avoir de lourdes conséquences économiques, voire dans certains cas, humaines. Afin de s'assurer de la fiabilité de tels systèmes, il importe donc de vérifier leurs comportements de la manière la plus rigoureuse possible.

L'utilisation des méthodes formelles pour le test de logiciels est probablement ce qu'il y a de plus sûr en matière de techniques de vérification. Ceci s'explique sans doute par les fondements mathématiques sur lesquels se basent ces méthodes, ce qui permet de développer un raisonnement plus rigoureux et de ce fait, plus fiable.

On peut requérir aux méthodes formelles pour spécifier les propriétés importantes du système testé, mais aussi pour vérifier ces propriétés sur l'implantation finale. L'utilisation de ces méthodes a permis de développer une théorie du test de conformité dont l'objectif est de réaliser un test fonctionnel qui permet de vérifier si le produit fini correspond à la spécification de référence. La recherche académique a publié de nombreux travaux sur le test de conformité. Globalement, on peut classifier l'ensemble de ces travaux en deux grandes catégories: les méthodes de test actif et les méthodes de test passif.

Le test actif consiste à appliquer au système sous test un ensemble de tests et à comparer le comportement observé avec la spécification de référence. De nombreuses méthodes de génération automatique de tests de conformité ont été proposées dans la littérature. Elles traitent généralement des systèmes protocolaires et applicatifs réactifs en faisant l'hypothèse de pouvoir interagir avec l'implantation sous test.

Le principe étant de stimuler le système testé en émettant des entrées particulières pour le faire réagir et de collecter les sorties produites pour les comparer avec celles attendues.

Ce type de test n'est malheureusement pas toujours possible à exécuter. Dans les systèmes de protocoles en couches par exemple, il est rare qu'on puisse bénéficier d'un accès direct pour interagir avec une couche particulière du système et ainsi appliquer les séquences de test. Aussi dans certains cas, la phase de test qui monopolise complètement le système, peut être très couteuse pour les industriels. Dans ce genre de situations, le test passif s'avère particulièrement intéressant.

En effet, le test passif ne requiert pas une interaction directe avec le système testé. Il consiste à observer et à collecter les entrées et les sorties produites par l'implantation sous test, et à analyser cette séquence par rapport à la spécification de référence. On vérifie alors si le comportement de l'implantation est conforme à celui prévu par la spécification.

La réalisation d'un test de conformité suppose que le système sous test s'exécute dans des conditions environnementales normales. On estime que dans de telles conditions, le comportement du système testé doit être conforme à sa spécification fonctionnelle. Cependant, lorsqu'un système informatique est susceptible d'évoluer dans un contexte hostile où les conditions environnementales sont plus ou moins stressantes, le test de conformité n'est plus suffisant. En effet dans ce genre de situations, on doit étudier le comportement du système en tenant compte de ces contraintes contextuelles. Ceci définit un autre type de test qu'on appelle : test de robustesse. L'objectif principal du test de robustesse est d'étudier le comportement d'une implantation s'exécutant dans un environnement hostile. L'implantation testée est considérée robuste si elle continue à avoir une exécution correcte en présence de fautes [1].

Les approches de test de robustesse peuvent être empiriques ou formelles. Les approches empiriques déterminent le niveau de robustesse du système étudié, tandis que les approches formelles s'intéressent à la vérification des propriétés de robustesse [2]. Les techniques d'injection de fautes sont couramment utilisées pour

BIBLIOTHEQUE DU CERIST

l'évaluation empirique de la robustesse d'une implantation. L'injection de fautes consiste à introduire de façon délibérée, des erreurs dans un système lors de son exécution et d'observer sa réaction. Cela permet, lors de la réalisation d'un test de robustesse, de simuler un environnent hostile. Par ailleurs, les approches de test de robustesse formelles ont pour but de déterminer formellement la robustesse d'une implantation en vérifiant la satisfiabilité d'un ensemble de propriétés de robustesse sur cette implantation. Ces dernières s'inspirent fortement des méthodes de test de conformité actives à la différence près que le domaine d'entrées est ici augmenté par l'introduction d'un ensemble d'aléas (fautes). Ainsi, au lieu de stimuler le système sous test par des entrées valides, le testeur de robustesse, génère et exécute des séquences d'entrées corrompues pour perturber le fonctionnement du système testé.

Contributions

Le test de robustesse est très important pour assurer la sécurité et la fiabilité des systèmes logiciels. Les techniques d'injection de fautes appliquées au test de robustesse ont montré des résultats très intéressants. Elles souffrent cependant de ne pas disposer d'oracles de tests performants leurs permettant d'évaluer la robustesse du système testé de manière plus rigoureuse. En effet, ces techniques ne vérifient pas formellement la robustesse d'un système. Une implantation est considérée robuste si elle peut continuer son exécution en présence de fautes. En d'autres termes, si le système testé ne se bloque pas, il est considéré comme robuste. On sait cependant, qu'un système peut très bien continuer son exécution sans pour autant fournir le comportement attendu. De ce fait, nous avons besoins de requérir à des approches plus rigoureuses pour évaluer la robustesse d'un système.

En outre, les techniques d'injection de fautes ne contrôlent pas efficacement le processus d'injection. Les fautes sont injectées de manière plus ou moins aléatoire et il n'y a aucun moyen de s'assurer de la bonne exécution des campagnes d'injections (est ce que toutes les fautes ont été injectées correctement ?).

D'autre part, les techniques formelles de test de robustesse définissent formellement toutes les étapes du test. Les fautes sont générées à partir d'un modèle formel et les propriétés de robustesse sont vérifiées sur la base d'un oracle de test bien défini.

BIBLIOTHEQUE DU CERIST

Toutefois, deux grandes questions peuvent être soulevées au sujet de ces méthodes. Tout d'abord, l'ensemble des fautes injectées est limité par le domaine d'entrées de l'application testée. A l'opposé des approches d'injection de fautes empiriques qui peuvent injecter n'importe quel type de fautes, les techniques formelles existantes créent le modèle de fautes en se référant au modèle fonctionnel du système testé. Ceci à l'avantage de permettre une injection mieux ciblée et plus adaptée au système testé, mais les types de fautes considérées sont limitées par le modèle fonctionnel. Si ce dernier ne prend pas en compte les aspects temporels par exemple, on ne pourra pas injecter de fautes temporelles. En plus, le modèle fonctionnel d'une implantation n'est pas toujours disponible.

Enfin, les méthodes formelles existantes appliquées au test de robustesse reprennent la même architecture que celle utilisée par les méthodes actives de test de conformité. Cette architecture impose que le testeur interagisse directement avec le système testé. Par conséquent, ces méthodes ne peuvent pas être utilisées pour tester des composants systèmes qui n'offrent pas d'interfaces d'interactions directes, ou lorsque le système testé ne peut pas être monopolisé par le testeur pour une durée importante.

Le travail que nous présentons dans ce document, consiste en un ensemble de propositions qui ont pour objectif de répondre aux défis auxquels font face les approches de test de robustesse existantes. Nous contribuons sur quatre principaux axes :

En premier lieu, nous nous intéressons aux techniques d'injection de fautes et plus particulièrement au problème de contrôle du processus d'injection. Nous proposons de formaliser les fautes injectées en utilisant une extension temporelle de la logique de Hoare [42]. Notre étude étant plus portée sur les systèmes communicants, nous proposons de spécifier chaque opération d'injection par un triplet de Hoare décrivant les pré-conditions qui doivent être satisfaites par les messages de communication interceptés avant l'exécution de l'opération d'injection, ainsi qu'un ensemble de post-conditions spécifiant comment l'exécution de cette opération devrait modifier les états de ces messages. Nous utiliserons ensuite cette formalisation comme

BIBLIOTHEQUE DU CERIST

oracle de test pour vérifier la bonne exécution du processus d'injection. Ainsi, nous proposons un algorithme de test passif qui vérifie la conformité de l'ensemble des fautes injectées (spécifiées comme un ensemble de triplets de Hoare), sur une trace d'injection. De cette manière, nous pourrons contrôler les campagnes d'injections et ainsi apporter plus de fiabilité à nos expérimentations.

Notre seconde contribution concerne la spécification et la vérification des propriétés de robustesse. Nous proposons de formaliser les propriétés de robustesse en utilisant une extension de la logique temporelle linéaire qui permet la spécification de contraintes temps réel. Il s'agit de la logique temporelle à horloge explicite, XCTL (eXplicit Clock Temporal Logic) [32], dont l'expressivité permet à la fois de spécifier des propriétés simples et complexes avec une aisance particulière.

Pour la vérification de ces propriétés, nous proposons un algorithme de test passif qui vérifie la conformité des formules XCTL sur une trace d'événements. Le choix d'une approche basée sur le test passif permet de s'affranchir des limitations du test actif, mentionnées précédemment.

Nous contribuons aussi par une nouvelle approche de test de robustesse. Nous proposons une approche hybride basée sur l'injection de fautes et le test passif. L'injection de fautes est utilisée pour créer des conditions environnementales stressantes, et le test passif permet de vérifier la satisfiabilité des propriétés de robustesse sur les traces d'exécution collectées. Les fautes injectées ainsi que les propriétés de robustesse sont formellement spécifiées. Nous utilisons la logique de Hoare pour la spécification des fautes et la logique XCTL pour la formalisation des propriétés de robustesse. Ce qui nous permet de vérifier à la fois le processus d'injection et les exigences de robustesse en appliquant les approches de test passif proposées dans nos contributions précédentes.

Finalement, nous proposons une plateforme de modélisation et de vérification de la robustesse des services Web. Les services Web sont une technologie émergente qui tend progressivement à s'imposer comme un standard du paradigme de communication programme-à-programme. Ils fournissent aussi un excellent exemple de systèmes hétérogènes fortement distribués. Les services Web peuvent être simples

BIBLIOTHEQUE DU CERIST

ou composés et ils sont largement utilisés pour la création d'applications e-commerce et de systèmes d'information distribués. Par conséquent, ils constituent un très bon exemple de systèmes critiques où le test de robustesse prend toute sa dimension.

La plateforme de test que nous proposons ici, est en réalité une instanciation de notre approche de test de robustesse, adaptée aux services Web. Cette plateforme intègre un injecteur de fautes innovant (WSInject) que nous avons conçu et développé pour pouvoir simuler un environnement d'exécution hostile. WSInject [36] est un injecteur de fautes pour services Web capable d'injecter des fautes d'interfaces et de communications, ou même de combiner les deux types de fautes en une seule injection. Il peut être utilisé pour le test de services simples ou composés.

Nous avons aussi implanté et intégré les algorithmes de test passif proposés pour la vérification du processus d'injection et des exigences de robustesse et nous avons conduit des expérimentations sur deux cas d'études pour illustrer l'utilisation de notre plateforme de test.

Organisation du manuscrit

Le présent manuscrit de thèse est organisé comme suit :

1. Dans le second chapitre, nous présentons l'état de l'art des approches de test de conformité et de robustesse. Pour le test de conformité, nous introduisons d'abord l'utilisation des méthodes formelles pour le test des systèmes logiciels. Ensuite, nous décrivons les approches les plus importantes des deux grandes familles de test : le test actif et le test passif. La deuxième partie de ce chapitre est consacrée aux méthodes de test de robustesse. Nous classons ces méthodes en deux grandes catégories. D'abord, nous exposons les techniques empiriques basées sur l'injection de fautes et ensuite nous abordons les techniques formelles.

2. Le troisième chapitre présente notre première contribution. Il s'agit de la formalisation et la vérification de l'injection de fautes. L'idée de base est de spécifier les fautes injectées par un ensemble de triplets de Hoare, puis d'utiliser

BIBLIOTHEQUE DU CERIST

cette spécification comme oracle de test pour vérifier la bonne exécution du processus d'injection. Nous définissons pour cela un algorithme de test passif qui vérifie la satisfiabilité des spécifications de fautes sur une trace d'injection. Nous présentons aussi quelques exemples de spécification pour illustrer notre approche.

3. Dans le quatrième chapitre, nous présentons notre approche de test de contraintes temps réel. Nous discutons en premier, les travaux existants qui traitent des méthodes formelles pour le test de propriétés temps réel. Ensuite, nous présentons les formalismes permettant de spécifier ce type de propriétés et justifions notre choix de XCTL [32]. Nous présentons aussi notre algorithme de test passif pour la vérification de formules XCTL sur des traces d'exécutions et discutons les résultats obtenus au terme d'une évaluation expérimentale de l'algorithme.

4. Dans le chapitre cinq, nous décrivons notre approche de test de robustesse. Il s'agit d'une approche complémentaire, basée sur l'injection de fautes et le test passif. Nous étudions d'abord les travaux existants sur le test de robustesse. Ensuite, nous présentons l'architecture générale de notre approche et détaillons chacune de ses composantes. Nous utilisons dans cette approche, la logique de Hoare pour la spécification et la validation des campagnes d'injection et la logique temporelle à horloge explicite (XCTL) pour le test des propriétés de robustesse.

5. Finalement, dans le chapitre six, nous présentons notre plateforme de test de robustesse pour les services Web. Cette plateforme est une instanciation de notre approche de test appliquée aux services Web. Nous décrivons son architecture générale et chacun de ses composants, plus particulièrement l'injecteur de fautes WSInject. Pour ce dernier, nous motivons notre choix de développer un injecteur de fautes pour les services Web et présentons son architecture et ses fonctionnalités.

   Nous présentons aussi dans ce chapitre, l'application de notre plateforme de

BIBLIOTHEQUE DU CERIST

test sur deux cas d'études et montrons comment cela a permis de détecter certains modes de défaillances que nous n'aurions pas pu déceler avec les méthodes de test traditionnelles.

6. Le dernier chapitre conclut notre travail. Nous rappelons nos principales contributions, que ce soit dans le domaine du test de conformité, de l'injection de fautes ou du test de robustesse ; et nous présentons quelques perspectives potentielles qui vont dans la continuité de notre travail.

BIBLIOTHEQUE DU CERIST

BIBLIOTHEQUE DU CERIST

Abstract

Robustness is a specialized dependability attribute, characterizing a system reaction with respect to external faults. Accordingly, robustness testing involves testing a system in the presence of faults or stressful environmental conditions to study its behavior when facing abnormal conditions.

Testing system robustness can be done either empirically or formally. Fault injection techniques are very suitable for assessing the robustness degree of the tested system. They do not rely however, on formal test oracles for validating their test. On the other hand, existing formal approaches for robustness testing formalize both the fault generation and the result analysis process. They have however some limitations regarding the type of the handled faults as well as the kind of systems on which they can be applied.

The work presented in this thesis manuscript aims at addressing some of the issues of the existing robustness testing methods. First, we propose a formal approach for the specification and the verification of the fault injection process. This approach consists in formalizing the injected faults as a set of Hoare triples and then, verifying the good execution of the injection campaigns, based on a passive testing algorithm that checks the fault specification against a collected injection trace.

Our second contribution focuses on providing a test oracle for verifying real time constraints. We propose a passive testing algorithm to check real time requirements, specified as a set of XCTL (eXplicit Clock Temporal Logic) formulas, on collected execution traces.

Then, we propose a new robustness testing approach. It is a complementary approach that combines fault injection and passive testing for testing system robustness. The injected faults are specified as a set of Hoare triples and verified against the injection trace to validate the injection process. The robustness requirements are formalized as a set of XCTL formulas and are verified on collected execution traces. This approach allows one to inject a wide range of faults and can be used to test both simple and distributed systems.

Finally, we propose an instantiation of our robustness testing approach for Web

services. We chose Web services technology because it supports widely distributed and heterogeneous systems. It is therefore, a very good application example to show the efficiency of our approach.

Keywords: Robustness Testing, Formal Specification, Fault Injection, Passive Testing, Trace Analysis.

BIBLIOTHEQUE DU CERIST

BIBLIOTHEQUE DU CERIST

Contents

BIBLIOTHEQUE DU CERIST

BIBLIOTHEQUE DU CERIST