SKRIPSI

PEMBANGKITAN KASUS UJI PROGRAM JAVA BERDASARKAN MODEL CHECKING

MUHAMMAD IRFAN

NPM: 2014730040

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2018

UNDERGRADUATE THESIS

MODEL CHECKING BASED JAVA TEST CASE GENERATION

MUHAMMAD IRFAN

NPM: 2014730040

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2018

LEMBAR PENGESAHAN

PEMBANGKITAN KASUS UJI PROGRAM JAVA BERDASARKAN MODEL CHECKING

MUHAMMAD IRFAN

NPM: 2014730040

Bandung, 17 Desember 2018

Menyetujui,

Pembimbing

Dott. Thomas Anung Basuki

Ketua Tim Penguji

Kristopher David Harjono, M.T.

Anggota Tim Penguji

Vania Natali, M.T.

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

PEMBANGKITAN KASUS UJI PROGRAM JAVA BERDASARKAN MODEL CHECKING

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 17 Desember 2018

MUHAMMAD IRFAN
NPM: 2014730040

ABSTRAK

Pengujian merupakan hal yang sangat penting dalam membangun suatu perangkat lunak. Pengujian berfungsi untuk mencari kesalahan pada program dengan cara mengidentifikasi ketepatan, kelengkapan, dan mutu dari suatu perangkat lunak. Hal ini membutuhkan biaya yang tidak sedikit. Pengotomatisan pengujian diharapkan dapat menurunkan biaya produksi dan juga meningkatkan keandalan suatu perangkat lunak. *Test case* merupakan masukan spesifik yang akan dicoba ketika sebuah perangkat lunak diuji. Pengujian adalah mencoba beberapa *test case* yang dibuat oleh penguji. Pada umumnya *test case* yang diuji hanya mewakili beberapa contoh kasus saja dan tidak mencoba seluruh kemungkinan *state* pada program. Untuk bisa menentukan *test case* yang tepat, efisien, dan menelusuri seluruh kemungkinan *state*, maka digunakan *model checking* sebagai pembangkit *test case*. Pada skripsi ini digunakan kakas Java Path Finder (JPF) untuk melakukan *model checking*. JPF adalah sebuah sistem untuk memverifikasi Java *bytecode* yang bertujuan untuk mencari cacat atau kesalahan pada program. JPF dapat melakukan *symbolic execution*. *Symbolic execution* adalah cara menganalisis program untuk menentukan masukan seperti apa yang menyebabkan setiap bagian dari program dieksekusi.

Perangkat lunak yang berhasil dibangun menerima masukan berupa sebuah *file* java lalu melakukan otomatisasi dalam konfigurasi kakas JPF untuk melakukan *symbolic execution* . Perangkat lunak akan menampilkan hasil pembangkitan *test case* dengan memproses output dari hasil eksekusi program JPF sehingga mudah dibaca oleh pengguna. Perangkat lunak yang dibangun masih terbatas dalam menangani kasus-kasus tertentu pada file Java, misalnya hanya dapat membangkitkan *test case* dengan tipe data primitif dan kondisi-kondisi tertentu saja.

Kata-kata kunci: Pemeriksa Model, Kasus Uji, Java Path Finder, keadaan

ABSTRACT

Testing is very important when it comes to building a software. Software are tested in order to find mistakes in a program to identify its precision, comprehensiveness, and quality. But, of course such a thing will require a decent amount of budget. So, in order to reduce it and also improve a software capability automated test are being ran. A test case is a specific input that will be tried when a software is tested. The test will be testing a case which is made by the tester. Usually, the test cases which are tested only represent a few sample cases, not all the possible states in the program. But in order to find the right test case efficiently model checking are being used as test case generator. On this essay, the writer use tool Java Path Finde (JPF) to do the model checking. JPF is a software to verify Java by the code in order to find imperfection or mistakes on a program. JPF could do a symbolic execution. Symbolic execution is a way of analyzing programs to determine what input causes each part of the program to be executed. A software which is succsessfully build will recieve an input in a form of Java file then it performs an automated checking in the JPF configuration to do symbolic execution. The software will show the result of the generated case by processing output from the executed JPF program so it can be read by its user. Software which has been built still has its limit in handling some cases on java file for example, it can only generate a test with primitive data types and certain conditions.

Keywords: Model checking, test case , Java Path Finder , state

Dipersembahkan untuk Mamah dan Ayah

KATA PENGANTAR

Puji syukur penulis sampaikan kepada Allah SWT karena atas berkat dan rahmatNya penulis bisa menyelesaikan penyusunan skripsi dengan judul "Model Checking Based Java Test Case Generation". Penulis menyadari bahwa di dalam skripsi ini masih terdapat banyak kekurangan. Penulis juga menyadari bahwa penyusunan skripsi ini tidak terlepas dari bantuan berbagai pihak. Pada kesempatan ini, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Kedua orang tua yang sudah memberikan semangat, motivasi, dan dukungan dalam berbagai hal.
2. Bapak Thomas Anung Basuki selaku dosen pembimbing yang telah memberikan waktu, nasihat, bimbingan, kritik, masukan, dan tambahan wawasan selama proses pembuatan skripsi ini sehingga skripsi ini dapat selesai tepat waktu.
3. Bapak Kristopher David dan Ibu Vania Natali penguji yang telah meluangkan waktu untuk memberikan kritik dan saran yang membangun dalam penulisan skripsi ini.
4. Muhammad Hilman, Reynaldo Imanuel, Vinieta Abhinandaniya yang selalu memberikan semangat kepada penulis dan sering menemani serta membantu penulis saat melakukan penyusunan skripsi ini serta selalu bisa memberikan masukan jika penulis kebingungan dalam hal apapun.
5. Keluarga Komunitas Alif Movement yang selalu memberi semangat dan selalu mengatakan bahwa penulis bisa menyelesaikan skripsi ini tepat waktu.
6. Keluarga UKM Listra UNPAR yang selalu membantu, mendukung dan memotivasi penulis dalam mengerjakan skripsi ini
7. Rekan-rekan mahasiswa dari Jurusan Teknik Informatika angkatan 2014 dan senior yang selalu mau memberikan jawaban atas pertanyaan penulis terkait skripsi ini.
8. Semua pihak yang tidak mungkin disebutkan satu-persatu yang sudah memberikan bantuan dan dukungan dalam pengerjaan skripsi ini.
   Akhir kata, penulis memohon maaf jika terdapat kesalahan dan kekurangan dalam skripsi ini. Semoga skripsi ini dapat bermanfaat bagi pihak yang membutuhkan

Bandung, Desember 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Dalam membangun sebuah perangkat lunak kerapkali ditemukan masalah berupa *error*, *bug* atau *output* yang tidak sesuai dengan harapan. Pengujian perangkat lunak menjadi hal yang penting dalam membangun perangkat lunak. Pengujian perangkat lunak digunakan untuk mencari kesalahan pada program dengan mengidentifikasi ketepatan dan kelengkapan dari suatu perangkat lunak. Pengujian perangkat lunak merupakan teknik yang paling umum digunakan untuk validasi perangkat lunak.

Pengujian perangkat lunak membutuhkan biaya yang tidak sedikit. Biaya pengujian pada umumnya berjumlah sekitar setengah dari total biaya pengembangan dan pemeliharaan perangkat lunak. Membuat pengujian secara otomatis diharapkan dapat menekan biaya produksi dan juga meningkatkan keandalan suatu perangkat lunak.

Model checking adalah teknik verifikasi yang mengeksplorasi semua kemungkinan keadaan sistem dengan cara *brute-force*. *Model checking* kerapkali digunakan sebagai teknik untuk menguji perangkat lunak. Salah satu contoh model checking yang digunakan untuk memverifikasi yaitu sebuah arsitektur sistem kontrol pesawat ruang angkasa berbasis kecerdasan buatan yang dijadwalkan akan diluncurkan pada bulan Oktober 1998 sebagai bagian dari misi *DEEP SPACE* 1 ke Mars. Pada sistem tersebut sejumlah *error* yang berdampak penting dapat diidentifikasi dengan *model checking*. [6]

Test case merupakan *input* yang spesifik yang akan dicoba ketika sebuah software diuji. *Testing* perangkat lunak merupakan serangkaian teknik empiris yang menjalankan program dengan sejumlah masukan tertentu untuk mengetahui apakah performa dari perangkat lunak tersebut sudah benar atau belum. Dapat dikatakan pada umumnya *testing* hanya mencoba beberapa *test case* secara acak oleh penguji dan hanya mewakili beberapa contoh kasus saja. Pada perangkat lunak dengan sekala kecil mungkin *test case* mudah untuk ditebak. Namun jika perangkat lunak yang diuji adalah skala besar atau kompleks maka *test case* sulit untuk ditebak.

Model checking berbeda dengan *testing*, karena *model checking* tidak bergantung pada tebakan.*Model checking* akan mengeksplorasi *state* demi *state* hingga tidak ada yang tersisa atau ditemukannya cacat program. Dari kemungkinan-kemungkinan kasus yang akan terjadi, jika ada pelanggaran pada kasus *input* tertentu maka *model checking* akan menemukannya.

Untuk bisa menentukan *test case* yang tepat dan efisien maka digunakan *model checking* sebagai pembangkit *test case*. Pembangkitan *test case* yang dilakukan oleh *model checking* mengeksplorasi *state* demi *state* pada program sehingga seluruh kemungkinan pada program dijalankan. Dengan demikian *test case* yang dibangkitkan sebagai *input* bisa menguji perangkat lunak apakah sudah memenuhi seluruh kondisi yang ada atau belum. Hal ini dapat dilakukan karena *test case* yang dibangkitkan sudah mewakili seluruh percabangan dari *state* program.

Pada skripsi ini akan dibuat sebuah perangkat lunak yang dapat membangkitkan *test case* pada perangkat lunak berbasis Java secara otomatis. *Test case* ini merupakan perwakilan dari semua kemungkinan dari *input* perangkat lunak yang akan diuji, jadi dengan *test case* ini pengguna dapat menguji perangkat lunak yang telah dibuat apakah menghasilkan *output* yang sudah sesuai atau belum.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang dapat ditentukan rumusan masalah sebagai berikut:

- Apa kakas yang tepat untuk melakukan *model checking* pada program Java?
- Bagaimana cara menggunakan kakas *model checking* untuk implementasi perangkat lunak pembangkit *test case*?
- Apa kelebihan dan kekurangan perangkat lunak yang dibuat?

1.3 Tujuan

Sesuai rumusan masalah di atas maka tujuan dari penelitian ini adalah sebagai berikut:

- Mempelajari cara kerja *model checking*.
- Mempelajari kakas yang tepat untuk melakukan *model checking*.
- Mempelajari kakas untuk melakukan *model checking* pada program Java.
- Mengimplementasikan kakas model checking untuk membangkitkan *test case*.

1.4 Batasan Masalah

Hasil dari skripsi ini adalah menghasilkan sebuah perangkat lunak yang dapat membangkitkan *test case* dari program bahasa Java secara otomatis. Perangkat lunak akhir yang akan dibuat memiliki fitur minimal sebagai berikut:

- Perangkat lunak dapat menerima *input* berupa *source code* program dalam bahasa Java.
- Perangkat lunak dapat mengeluarkan *output* berupa *test case* secara otomatis.

1.5 Metodologi

Bagian-bagian pengerjaan skripsi ini adalah sebagai berikut :

1. Mempelajari konsep *testing*.
2. Mempelajari kakas *model checking* untuk program Java.
3. Menganalisis kebutuhan perangkat lunak.
4. Mengumpulkan program Java untuk pengujian.
5. Merancang perangkat lunak.

6. Mengimplementasikan perangkat lunak.
7. Melakukan pengujian terhadap perangkat lunak.
8. Menulis dokumen skripsi.

1.6 Sistematika Pembahasan

Sistematika pembahasan dalam penelitian ini berupa:

1. Bab Pendahuluan
   Bab 1 berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.
2. Bab Landasan Teori
   Bab 2 berisi teori-teori tentang konsep *testing*, logika formal, dan *model checking* beserta kakasnya
3. Bab Analisis Perangkat Lunak
   Bab 3 berisi analisis terhadap perangkat lunak yang akan di bangun.
4. Bab Perancangan Perangkat Lunak
   Bab 4 berisi perancangan terhadap perangkat lunak yang akan dibangun.
5. Bab Implementasi dan Pengujian Perangkat Lunak
   Bab 5 berisi implementasi perangkat lunak dan Pengujian perangkat lunak.
6. Bab Kesimpulan dan Saran
   Bab 6 berisi kesimpulan serta beberapa saran untuk pengembangan lebih lanjut dari penelitian yang dilakukan dan aplikasi yang dibangun.