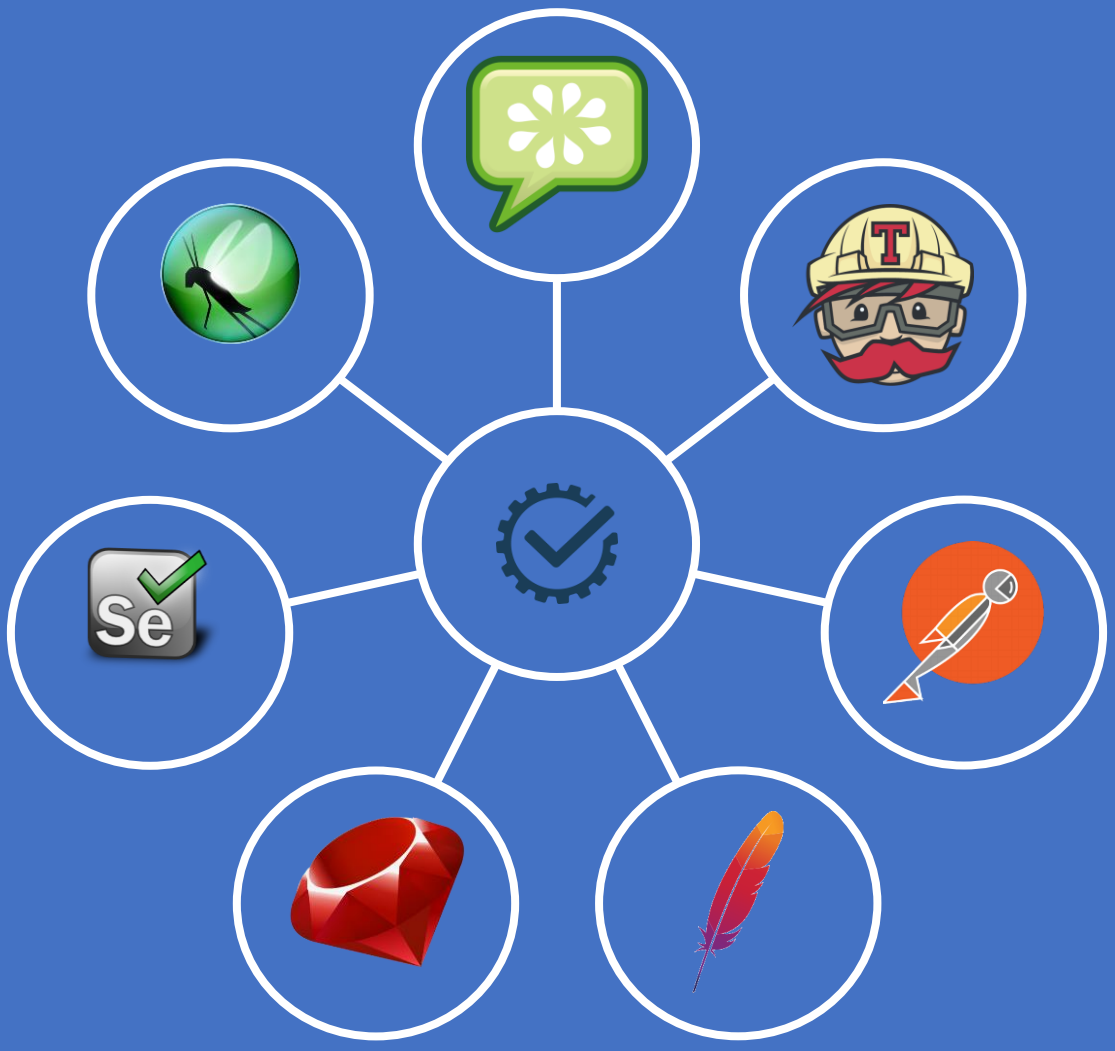
Otomatisasi Pengujian Perangkat Lunak

(Software Testing Automation)



Lamhot JM Siagian

KATA PENGANTAR

Puji dan syukur penulis sampaikan kepada Tuhan yang Maha Esa atas penyertaannya, penulis bisa menyelesaikan buku yang berjudul "Otomatisasi Pengujian Perangkat Lunak". Selain itu penulis juga ingin menyampaikan terimakasih kepada keluarga yang turut mendukung penulis. Terimakasih juga kepada teman yang memberikan dukungan (Rolastri, Mariyani, Wati, Ayu, Lestari, Frisca, Lasma, Siska, Melyana, Ulva, Trima, dan Olvi) yang turut membantu dalam proses bertukar pikiran dan memberikan saran membangun. Terimakasih juga untuk mentor saya Mas Ragapinilih untuk ilmu-ilmu baru dan sebagian saya tuliskan pada buku ini.

Automation Testing merupakan suatu teknologi yang sedang *booming* di dunia software testing. Tidak bisa dipungkiri bahwa *automation testing* sangat dibutuhkan untuk proses pengujian ulang seluruh fungsionalitas perangkat lunak. Untuk itu penulis tertarik mengambil topik mengenai *automation testing* dan terbilang jarang dibahas pada buku Bahasa Indonesia.

Buku ini akan membantu para software engineer in test atau QA engineer pemula untuk mengetahui lebih dalam mengenai automation testing. Penulis banyak membahas konsep *software testing*, *automation testing* web, API *testing*, performance testing, dan cara Konfigurasi *Continuous Integration* (CI) dengan menggunakan Travis. Penulis membuat buku dengan konsep sedikit teori, mini proyek setiap tools dan proses instalasi. Penulis mengharapkan setelah membaca buku ini, pembaca memahami dan mempraktikkan topik-topik berikut saat melakukan pengujian perangkat lunak.

1. Konsep Software Testing
2. Pemakaian Selenium IDE & Selenium Webdriver
3. API *automation testing* dengan Postman, Newman & Cucumber API
4. Web *Automation Testing* dengan Cucumber Ruby
5. *Performance Testing* dengan JMeter & Locust
6. Pengaturan *Continuous Integration* dengan Travis CI

Penulis menyadari masih terdapat kekurangan dalam buku ini untuk itu kritik dan saran terhadap penyempurnaan buku ini sangat diapresiasi. Semoga buku ini dapat memberi maanfaat bagi yang tertarik dengan *automation testing* perangkat lunak, secara khusus dan bagi semua pihak yang membutuhkan.

Jakarta, 8 Agustus 2018

Lamhot JM Siagian

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1

Konsep Testing

Pada Bab 1 akan dibahas mengenai pengertian testing, istilah dalam pengujian, metodologi testing, proses testing, pelaporan bug dan *level testing* (jenis testing).

1.1 Pengertian Testing

Menurut standar ANSI/IEEE 1059, *perangkat lunak testing* adalah proses menganalisa suatu entitas perangkat lunak untuk mendeteksi perbedaan antara kondisi yang ada dengan kondisi yang diinginkan (defects/error/bugs) dan mengevaluasi fitur-fitur dari entitas perangkat lunak.

Testing atau pengujian perangkat lunak merupakan proses inti dari jaminan kualitas perangkat lunak. Tujuan pengujian perangkat lunak[1]:

1. Pengujian adalah proses eksekusi program dengan maksud menemukan kesalahan.
2. Sebuah kasus uji yang baik adalah salah satu dengan probabilitas tinggi untuk menemukan kesalahan yang belum ditemukan.
3. Sebuah tes yang sukses adalah salah satu yang menemukan kesalahan yang belum ditemukan.

1.2 Istilah Pengujian

Test plan → adalah dokumen yang berisi definisi tujuan dan sasaran pengujian dalam lingkup iterasi (atau proyek), item-item yang menjadi target pengujian, pendekatan yang akan diambil, sumber daya yang dibutuhkan dan poin untuk diproduksi. Dengan kata lain test plan dapat disebut sebagai perencanaan atau skenario untuk melakukan testing yang akan dilakukan baik oleh expert atau user umum.

Test case → adalah sekumpulan dari input tes, kondisi yang akan dieksekusi, dan hasil yang diharapkan (IEEE). Test case adalah kondisi saat testing melakukan suatu aksi pada sistem, tester memberikan data pada sistem yang diuji, dan sistem memberikan beberapa kondisi melalui output yang dapat dibandingkan dengan hasil yang diharapkan. Tindakan, data, dan hasil yang diharapkan adalah bagian utama dari testing dan menghasilkan kondisi tes. Test

[1] Presman, Rouger S, *Software Engineering*, 6th Edition, Mc. Graw Hill,1997.

case adalah urutan langkah-langkah, sub dari langkah-langkah tersebut, dan tindakan lain yang dijalankan secara berurutan atau merupakan kombinasi dari konsekuensi yang menghasilkan kondisi tes sesuai harapan dan berguna untuk evaluasi lebih lanjut Black (2009:610).

Test plan → adalah sebuah dokumen yang berisi tentang penjelasan, strategi, metode dan juga kondisi-kondisi dalam melakukan test nantinya (Black, 2009:50).

Test tool → adalah tujuan umum dari hardware, perangkat lunak yang digunakan selama eksekusi test case untuk mengatur lingkungan, menciptakan kondisi pengujian, atau untuk mengukur hasil pengujian. Sebuah test tool terpisah dari test case itu sendiri (Black, 2009:612).

Bug report → adalah dokumen teknikal yang menjelaskan berbagai tanda atau failure mode yang terkait dengan setiap bug. *Bug report* yang baik yaitu yang menyediakan informasi yang dibutuhkan tim proyek management untuk memutuskan kapan dan bagaimana cara memperbaiki bug tersebut. *Bug report* yang baik juga menyimpan informasi yang akan dibutuhkan seorang programmer untuk memperbaiki dan debug masalah (Black, 2009:146).

Test environment → adalah pengaturan di mana pengujian terjadi, termasuk tes, platform, infrastruktur tes, uji lab, dan fasilitas lainnya (Black, 2009:611)

Test result → seorang penguji harus mendokumentasikan hasil dari pengujiannya sehingga dapat diketahui hal apa yang telah tercapai ataupun yang tidak tercapai dalam sebuah hasil pengujian (Perry, 2006:463).

Bug → adalah suatu masalah yang hadir dalam sistem yang diuji yang menyebabkan gagal dalam memenuhi harapan atau biasa disebut juga kecacatan. Gejala dari bug adalah sebuah kegagalan (Black, 2009:601-602).

Test suite → adalah kumpulan test case yang dimaksudkan untuk digunakan untuk menguji program perangkat lunak untuk menunjukkan bahwa *perangkat lunak* memiliki beberapa set perilaku tertentu. Test suite sering berisi instruksi atau sasaran terperinci untuk setiap

kumpulan test case dan informasi tentang konfigurasi sistem yang akan digunakan selama pengujian. Sekelompok test case juga dapat berisi status atau langkah prasyarat, dan uraian pengujian berikutnya.[2] *Test suite* merupakan sebuah kerangka untuk melakukan eksekusi terhadap sekumpulan *test case;* suatu cara mengatur *test case.* Dalam sebuah *test suite, test case* dapat digabungkan untuk menjadi kondisi-kondisi pengujian yang unik. (Black, 2009: 611).

Test scenario → atau skenario pengujian adalah aktivitas pengujian perangkat lunak yang menggunakan skenario: cerita hipotetis untuk membantu penguji bekerja melalui masalah kompleks atau sistem pengujian. Tes skenario yang ideal adalah cerita yang kredibel, kompleks, menarik atau memotivasi yang hasilnya mudah dievaluasi[3]

Smoke testing → juga dikenal sebagai "*Build Verification Testing*", adalah jenis pengujian perangkat lunak yang terdiri dari serangkaian tes tidak lengkap yang bertujuan untuk memastikan bahwa fungsi yang paling penting berfungsi. Hasil dari smoke testing digunakan untuk memutuskan apakah sebuah build cukup stabil untuk dilanjutkan dengan pengujian lebih lanjut.

Istilah '*smoke testing*', muncul dari dari jenis pengujian perangkat keras, di mana perangkat lulus tes jika tidak terbakar (atau berasap/*smoked*) pertama kali dinyalakan.[4]

Verifikasi → merupakan istilah yang digunakan untuk mengetahui apakah produk atau perangkat lunak yang telah dibangun sudah benar. (Apakah produk yang dibangun sudah benar?)

Validasi → merupakan istilah yang digunakan untuk mengetahui apakah produk atau perangkat lunak yang telah dibangun sudah tepat. (Apakah produk yang dibangun sudah tepat?)

---

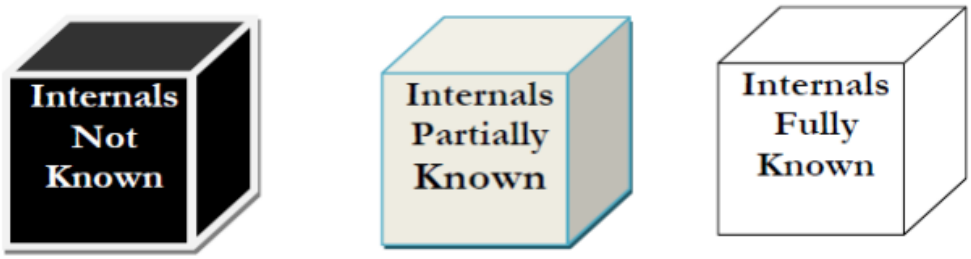
[2] https://www.softwaretestinghelp.com/software-terms-complete-glossary

[3] Cem Kaner, *Introduction to Scenario Testing*, Florida Tech, June 2003.

[4] http://softwarefundamentals.com/smoke-testing, Tanggal akses 1 Juni 2018

1.3 Metodologi Testing

Metodologi *testing* secara umum dibagi tiga berdasarkan *Box Testing* yaitu *black-box*, *white-box*, dan *grey-box tetsing*



Gambar 1.1 Perbandingan Box Testing [5]

Berdasarkan Gambar 1.1, teknik pengujian tanpa memiliki pengetahuan tentang cara kerja interior aplikasi disebut pengujian black-box. Penguji tidak menyadari arsitektur sistem dan tidak memiliki akses ke kode sumber. Biasanya, ketika melakukan tes black-box, tester akan berinteraksi dengan antarmuka pengguna sistem dengan memberikan input dan memeriksa output tanpa mengetahui bagaimana dan di mana input bekerja. Detail perbedaan metodologi testing dapat dilihat pada Tabel 1.1

Tabel 1.1 Perbedaan Metodologi

No	Black Box Testing	Gray Box Testing	White Box Testing
1	Cara Kerja internal aplikasi tidak perlu diketahui	Pengetahuan tentang cara kerja internal diketahui	Tester memiliki pengetahuan penuh tentang cara kerja internal aplikasi
2	Dikenal sebagai pengujian kotak tertutup, pengujian yang didorong data, dan pengujian fungsional	Istilah lain adalah pengujian kotak samar, Penguji memiliki pengetahuan terbatas dari bagian dalam aplikasi	Juga dikenal sebagai pengujian kotak yang jelas, pengujian struktural atau pengujian berbasis kode
3	Dilakukan oleh	Dilakukan oleh pengguna	Biasanya dilakukan oleh

[5] Acharya dkk, Bridge between Black Box and White Box – Gray Box Testing Technique, International Journal of Electronics and Computer Science Engineering, ISSN- 2277-1956.