

LAPORAN PENELITIAN

OTOMASI UJI FUNGSIONAL WEB ENTERPRISE

Oleh:

Gede Karya, S.T., M.T., CISA

Elisati Hulu, S.T., M.T.



Lembaga Penelitian dan Pengabdian Kepada Masyarakat

Universitas Katolik Parahyangan

2012

Abstrak

Salah satu rangkaian penting dalam pengembangan perangkat lunak adalah fase uji coba. Agar perangkat lunak yang dikembangkan berkualitas dan handal, maka dilakukan uji fungsional dan non fungsional. Untuk melakukan uji fungsional, diperlukan spesifikasi, skenario dan sample data uji. Pengujian dapat dilakukan berulang-ulang sampai semua kriteria fungsional dapat dipenuhi. Agar pelaksanaan pengujian berjalan konsisten sesuai dengan spesifikasi, skenario dan sample data uji, walau dilakukan berulang-ulang, maka diperlukan *tools* otomasi untuk melakukan hal tersebut. Pada penelitian ini dikembangkan model otomasi pengujian fungsional pada lingkungan sistem terdistribusi. Model ini diimplementasikan dalam bentuk perangkat lunak alat uji (*tools*) yang dapat digunakan oleh penguji secara berulang-ulang. Alat uji yang dihasilkan diimplementasikan menggunakan antarmuka berbasis web yang memiliki fitur: pengelolaan skenario, agen uji (*test agent*), pemesanan pengujian (*order*), pelaksanaan pengujian (*test*) dan pelaporan hasil (*reporting*). *Test Agent* diimplementasikan menggunakan bahasa Java yang memiliki kemampuan mengemulasikan browser yang mendukung HTML, Java Script dan Ajax. Pengujian dapat dilakukan dengan pendekatan *multi agent*, baik bersifat *standalone* maupun *service* sehingga cocok untuk aplikasi berbasis web sekala besar (*enterprise*). Untuk pembuatan skenario dikembangkan juga bahasa yang dapat digunakan untuk memanipulasi semua objek antarmuka pengguna (*user interface*) berbasis HTML. Pengujian telah dilakukan pada sistem portal akademik Unpar yang disederhanakan dengan hasil baik.

Kata kunci: otomasi uji fungsional, *test agent*, web enterprise

Kata Pengantar

Puji syukur penulis sampaikan kepada Tuhan Yang Maha Pengasih, atas berkah dan bimbingannya, penulis dapat merampungkan laporan penelitian ini. Laporan penelitian ini merupakan rekapitulasi dari sejumlah aktivitas untuk menghasilkan sebuah perangkat lunak untuk mengotomasi uji fungsional sistem aplikasi berbasis web. Perangkat lunak ini diperlukan untuk memudahkan proses *quality assurance* terutama dalam menguji kebenaran program-program aplikasi berbasis web sehingga meningkatkan produktifitas dan kualitas perangkat lunak yang dihasilkan. Penelitian ini didanai oleh Unpar melalui LPPM.

Terima kasih kepada Ketua Jurusan Teknik Informatika, Dekan Fakultas Teknologi Informasi dan Sains atas dukungan dan arahan serta persetujuan atas usulan penelitian ini. Terima kasih juga kepada Ketua LPPM atas pendanaan yang diberikan. Selain itu juga Penulis sampaikan terima kasih kepada para mahasiswa yang membantu dalam proses implementasi dan pengujian program serta dokumentasi, diantaranya: M. Aditya Nugraha, M. Iqbal, Ari Bratasena, Nurisya Hafizah dan Yurpita Zey.

Akhir kata semoga laporan ini dapat memberikan gambaran motivasi, metodologi dan hasil dari penelitian yang telah dilakukan.

Ketua Peneliti,

Gede Karya, S.T., M.T., CISA

Daftar Isi

Kata Pengantar.....	1
Daftar Isi.....	2
Daftar Gambar	4
Daftar Lampiran	6
BAB 1 PENDAHULUAN.....	7
1.1. Latar Belakang	7
1.2. Rumusan Masalah dan Batasan.....	8
1.3. Tujuan dan Hasil yang Diharapkan	8
1.4. Metodologi Penelitian	9
1.5. Sistematika Pembahasan.....	10
BAB 2 STUDI PUSTAKA	11
1.1. Konsep dan Cara Kerja Aplikasi Berbasis Web.....	11
1.2. Uji Fungsional dan Otomasi Uji.....	12
2.1. Agen Cerdas (<i>Intelligent Agent</i>).....	13
BAB 3 EKSPLORASI PERANGKAT LUNAK TOOLS OTOMASI UJI	14
3.1. Apache JMeter.....	14
3.2. HTTP Unit.....	17
3.3. HTML Unit.....	18

BAB 4 ANALISIS KEBUTUHAN DAN DISAIN SOLUSI.....	19
4.1. Analisis Kebutuhan	19
4.2. Model Solusi	20
4.3. Konstruksi Bahasa Pada Skenario	23
4.4. Disain Solusi.....	26
4.4. Disain Detail Perangkat Lunak Otomasi Uji Fungsional	28
4.4.1. Rancangan Basis Data Test Server	28
4.4.2. Rancangan Antar Muka Web App	28
4.4.3. Rancangan Algoritma Test Agent (TA).....	33
5.1. Implementasi Basis Data	36
5.2. Implementasi Web App.....	36
5.3. Implementasi Test Agent (TA)	37
5.3. Pengujian	39
BAB 6 KESIMPULAN DAN POTENSI PENGEMBANGAN	42
6.1. Kesimpulan	42
6.2. Potensi Pengembangan.....	42
DAFTAR REFERENSI	43

Daftar Gambar

Gambar 2.5. Arsitektur Perangkat Lunak Berbasis Web.....	11
Gambar 2.6. Interaksi Agent dengan Lingkungannya.....	13
Gambar 3.1. JMeter Tree.....	15
Gambar 3.2 <i>HTTP Request</i> Pada JMeter.....	15
Gambar 3.3 <i>View Result Tree</i> Pada JMeter.....	15
Gambar 3.4 <i>Aggregate Graph</i> Pada JMeter.....	16
Gambar 4.1. Model Otomasi Uji Fungsional.....	21
Gambar 4.2. Model Detail Test Server.....	22
Gambar 4.3. Disain Aplikasi Otomasi Uji Fungsional.....	26
Gambar 4.4. Disain Deployment Aplikasi Otomasi Uji Fungsional.....	27
Gambar 4.5 Rancangan Basis Data Test Server.....	28
Gambar 4.6 Struktur Menu Web App.....	29
Gambar 4.7 Daftar Object dan Action pada Language Editor.....	30
Gambar 4.8 Daftar Skenario.....	30
Gambar 4.9 Daftar Task dari Scenario Login2.....	31
Gambar 4.10 Daftar Agent.....	31
Gambar 4.11 Daftar Order.....	32
Gambar 4.12 Eksekusi Order.....	32
Gambar 4.13 Daftar Hasil Test dari Order (20 Terakhir).....	33
Gambar 4.14 Contoh Hasil Log Test No. 318.....	33

Gambar 5.1 Implementasi Basis Data.....	36
Gambar 5.2 Contoh Hasil Tampilan Web App	37
Gambar 5.3 Implementasi Test Agent (TA) pada lingkungan NetBeans 6.8.....	38
Gambar 5.4 Contoh Hasil Eksekusi TA dengan mode standalone	38
Gambar 5.5 Contoh Hasil Eksekusi TA dengan mode service.....	39
Gambar 5.6 Contoh Hasil Pengujian Skenario Login oleh Agent1	40

Daftar Lampiran

Lampiran 1 Referensi Apache JMeter

Lampiran 2 Referensi HTTPUnit

Lampiran 3 Referensi HTMLUnit

Lampiran 4 Aplikasi Portal Akademik

Lampiran 5 Konstruksi Bahasa Test Agent

BAB 1 PENDAHULUAN

Pada bab ini akan dijelaskan latar belakang, rumusan masalah, tujuan dan hasil serta sistematika pembahasan.

1.1. Latar Belakang

Salah satu hal terpenting dalam sebuah produk perangkat lunak adalah kualitas. Kualitas didefinisikan sebagai kemampuan suatu produk untuk memenuhi/ memuaskan kebutuhan penggunanya [1]. Kebutuhan perangkat lunak dapat dibagi menjadi 2, yaitu: kebutuhan fungsional dan kebutuhan non fungsional. Kebutuhan fungsional berkaitan dengan proses bisnis atau fungsi yang diotomasi oleh perangkat lunak. Fungsi ini terkait dengan aktifitas dan tata cara/ perhitungan (alur logika) yang ada di dalamnya beserta data-data yang diolah oleh perangkat lunak tersebut. Kebutuhan non fungsional mencakup kebutuhan akan reliabilitas, kapasitas, performansi dan sejenisnya.

Bagaimana cara memastikan bahwa suatu perangkat lunak berkualitas? Caranya adalah dengan melakukan uji kualitas (pengujian). Karena komponen kualitas adalah fungsional dan non fungsional, maka pengujiannya pun demikian. Pada pengujian fungsional, dapat dilakukan uji terima yaitu dengan mengecek semua fungsi yang disyaratkan apakah sudah dapat berjalan/ diakomodasi oleh perangkat lunak. Jadi fungsi perangkat lunak dibandingkan dengan fungsi-fungsi dalam proses bisnis yang diotomasi. Dengan demikian dapat dilakukan secara simulasi pada lingkungan pengembangan.

Khusus untuk uji non fungsional, ada beberapa yang tidak dapat dilakukan dengan mudah, misalnya: uji performansi pada saat *peak season*. Ini memerlukan lingkungan ekstrem sesuai dengan definisi kapasitas, berupa *sample user* dan kondisi transaksi maksimal. Demikian juga dengan uji reliabilitas, memerlukan sejumlah kapasitas dan rentang waktu yang cukup panjang untuk menjamin bahwa dalam jangka waktu yang disyaratkan perangkat lunak tidak akan bermasalah.

Untuk suatu kelayakan operasi, baik uji fungsional maupun uji non fungsional harus dilakukan, sehingga menjamin bahwa perangkat lunak sudah siap. Hal ini menjadi suatu resiko yang signifikan khususnya pada sistem enterprais, di mana akan melayani pengguna dan transaksi dalam jumlah besar dengan intensitas dan reliabilitas yang memadai. Khusus untuk uji non fungsional, diperlukan suatu alat dan lingkungan uji yang dapat mewakili kondisi yang ingin diujikan.

Pada 2 penelitian sebelumnya [2][3] sudah dihasilkan *tools* untuk melakukan uji coba secara non fungsional berupa uji performansi/ kinerja dan kapasitas situs berbasis web. Pada penelitian ini dibahas khusus bagaimana melakukan otomatisasi uji fungsional situs web enterprise. Otomatisasi uji fungsional dilakukan dengan metode black-box [1], dimana pengujian di dasarkan atas *user interface* (antar muka pengguna) berupa input dari user dan output yang dihasilkan oleh sistem perangkat lunak.

1.2. Rumusan Masalah dan Batasan

Beberapa masalah yang ingin dijawab dalam penelitian ini adalah:

1. Kriteria apa saja yang diperlukan dalam melakukan uji fungsional suatu web enterprais?
2. Bagaimana menentukan pola kriteria, skenario dan sample data uji fungsional sehingga dapat dibuat dalam bentuk model generik?
3. Bagaimana mengembangkan perangkat lunak untuk mengotomasi proses pengujian fungsional tersebut?

Adapun batasan yang diberikan adalah:

1. Mendukung operasi GET, POST, Cookies
2. Kriteria output terbatas pada halaman web dan informasi yang ada di dalamnya.
3. UI berbasis web untuk pengelolaan skenario, agent, testing dan reporting.

1.3. Tujuan dan Hasil yang Diharapkan

Berdasarkan rumusan dan latar belakang di atas, maka tujuan yang ingin dicapai pada penelitian ini untuk mengembangkan perangkat lunak untuk melakukan uji fungsional web enterprise secara otomatis. Otomatisasi di sini ditekankan pada adanya mesin pengujian yang dapat menjalankan suatu skenario uji coba yang telah ditetapkan oleh penguji, dan memberikan laporan hasil uji.

Oleh karena itu, hasil akhir yang diharapkan dalam penelitian ini adalah:

1. Perangkat Lunak Otomasi Uji Fungsional, yang memiliki fungsi:
 - a. Pemasukan skenario uji, berdasarkan elemen fungsi yang akan diuji
 - b. Pemasukan task (langkah) pengujian berikut kriteria hasil, yang mencakup task yang:
 - Sederhana
 - Kondisional

- Pengulangan
- c. Eksekusi skenario uji secara berulang menggunakan multi agen
 - d. Pelaporan hasil pengujian
2. Perangkat Lunak Target Uji berbasis Web, dalam hal ini mengemulasi proses FRS di Unpar. Perangkat lunak ini memiliki fungsi:
 - a. Halaman depan
 - b. Login dan logout
 - c. Pendaftaran mata kuliah dalam konteks FRS
 3. Hasil pengujian perangkat lunak target menggunakan perangkat lunak otomasi uji. Hasilnya mencakup berbagai skenario dari yang sederhana, mengandung kondisional dan pengulangan. Pengujian didasarkan atas skenario yang dibuat oleh penguji, kemudian eksekusi dilakukan oleh berbagai agen, secara tunggal maupun secara simultan (paralel).

1.4. Metodologi Penelitian

Penelitian ini dilakukan dengan metode rekayasa produk, khususnya produk perangkat lunak. Tahapan yang dilalui antara lain:

1. Studi konsep dan cara kerja aplikasi berbasis web. Kemudian dilanjutkan dengan studi tentang uji fungsional dan otomasi uji. Studi diakhiri dengan kajian dan eksplorasi tentang software tools otomasi uji yang ada saat ini, yaitu: JMeter, HttpUnit dan HtmlUnit.
2. Rekayasa perangkat lunak, dengan tahapan sebagai berikut:
 - a. Analisis kebutuhan perangkat lunak, baik fungsional maupun non fungsional.
 - b. Disain perangkat lunak, mulai dari disain global dan disain detail (basis data, disain protokol dan aplikasi).
 - c. Implementasi, berupa pengkodean dalam bahasa Java.
 - d. Pengujian, dilakukan terhadap perangkat lunak contoh, yaitu: sistem FRS on line Unpar dengan penyederhanaan agar lebih fokus.

1.5. Sistematika Pembahasan

Laporan penelitian ini disajikan dengan sistematika sebagai berikut:

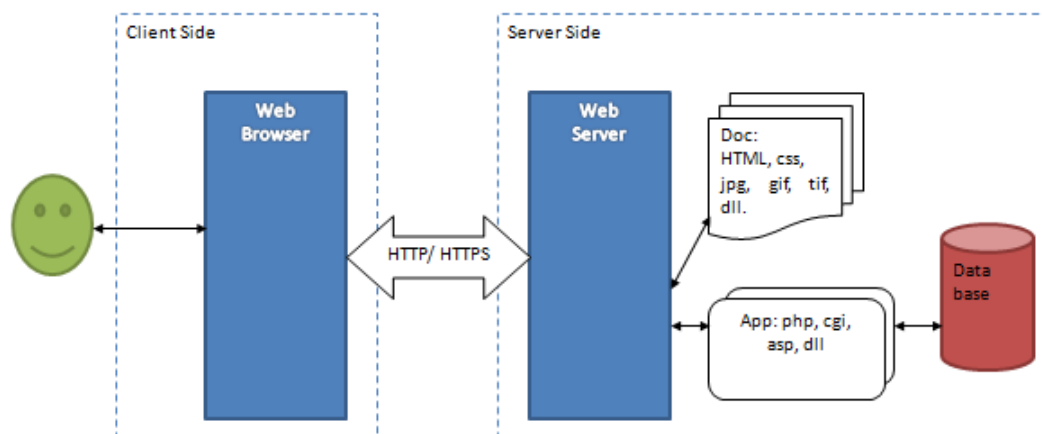
1. Pada bab 1 Pendahuluan dijelaskan tentang latar belakang, rumusan masalah, tujuan, hasil, metodologi dan sistematika pembahasan.
2. Bab 2 Kajian Pustaka, tentang konsep dan cara kerja aplikasi berbasis web, uji fungsional dan otomasi uji.
3. Bab 3 Eksplorasi software tools otomasi uji yang ada saat ini, yaitu: JMeter, HttpUnit dan HtmlUnit.
4. Bab 4 Analisis Masalah dan Disain Solusi, berisi pembahasan masalah dan gambaran solusi serta disain dari solusi yang diajukan.
5. Bab 5 Implementasi dan Pengujian, membahas hasil implementasi dan pengujian perangkat lunak yang dilakukan.
6. Bab 6 Kesimpulan dan Potensi Pengembangan, berisi kesimpulan yang diambil berdasarkan hasil penelitian dan potensi pengembangan yang mungkin untuk penelitian selanjutnya.

BAB 2 STUDI PUSTAKA

Pada bagian ini dibahas tentang hasil studi pustaka yang berhubungan dengan penelitian ini.

1.1. Konsep dan Cara Kerja Aplikasi Berbasis Web

Perangkat lunak berbasis web menggunakan protokol *Hyper Text Transfer Protocol (HTTP)* dan *Secure Hyper Text Transfer Protocol (HTTPS)*. Komponen utama dari sistem aplikasi berbasis web dapat dilihat pada Gambar 2.5.



Gambar 2.5. Arsitektur Perangkat Lunak Berbasis Web

Arsitektur komunikasi dasar yang digunakan adalah *client-server*. Di sisi client (*client side*) pengguna dapat mengakses layanan web menggunakan web browser. Di sisi server (*server side*) harus tersedia web server dan aplikasi lain yang dapat diakses melalui web server oleh web browser.

Web browser memiliki kemampuan untuk menginterpretasi dokumen dalam format Hyper Text Markup Language (HTML) berikut kelengkapannya seperti: Cascading Style Sheet (CSS), gambar dengan berbagai format seperti: JPEG, GIF, PNG dan lainnya, serta dapat mengeksekusi script tambahan seperti Java Script dan Visual Basic Script (VB Script). Web browser dapat mengirimkan permintaan dalam bentuk Uniform Resources Locator (URL) menggunakan protokol HTTP atau HTTPS (*HTTP request*) ke web server. Selanjutnya web server akan memproses URL ini dengan mengasosiasikannya ke permintaan dokumen/ file (seperti HTML atau gambar). Jika yang diminta tersebut adalah berupa file script yang didukung oleh web server, seperti: php, asp dan Common Gateway Interface (CGI), maka file

tersebut akan dieksekusi sesuai dengan mapping eksekutor yang terdaftar di konfigurasi web server. Hasil pemrosesan dalam format HTML disampaikan kembali ke web server untuk diteruskan ke web browser sebagai hasil atau *HTTP respon*.

Protokol HTTP menggunakan clear text dengan operasi GET, POST dan PUT. Pada mayoritas web server, operasi yang diimplementasikan adalah GET dan POST. Pada operasi GET semua parameter dari request ada pada URL. Dengan demikian, perintah keseluruhan terbatas pada panjang maksimal URL yang diijinkan (255 karakter). Pada operasi POST jalur antara perintah dengan data/ parameter dipisahkan. Jalur data dalam bentuk stream, sehingga panjangnya tidak terbatas. Oleh karena itu operasi POST ini cocok untuk pengiriman form dan data dalam bentuk file.

Pada konteks protokol HTTP/HTTPS, web browser disebut sebagai User Agent. Pada lingkungan Java, pengembangan User Agent dapat dilakukan dengan menggunakan library `java.net`. Class `URL` dapat digunakan untuk memproses suatu URL input dengan protokol HTTP dan HTTPS. Yang ditangani termasuk pengecekan terhadap format URL yang menghasilkan eksepsi `MalformedURLException`.

1.2. Uji Fungsional dan Otomasi Uji

Agar kualitas perangkat lunak terjaga, maka dilakukan serangkaian pengujian. Dalam pengujian perangkat lunak ada 2 aspek/ kriteria yang diuji yaitu:

1. **Uji Fungsional.** Berfokus pada kebenaran proses melalui verifikasi terhadap input yang diberikan dan output yang dihasilkan. Dalam pengujian fungsional ada 2 metode [1] yang dapat diterapkan, yaitu: metode black box dan metode white box. Pada metode black box, pengujian dilakukan per fitur berdasarkan input yang diberikan dan output yang dihasilkan oleh perangkat lunak. Sedangkan pada white box, dilakukan pelacakan terhadap proses-proses secara detail pada tingkatan algoritma yang dieksekusi oleh perangkat lunak. Uji fungsional, terutama untuk aplikasi berbasis web lebih banyak dilakukan secara black box, dimana penguji memberikan input sesuai dengan sample data dari setiap kasus fungsi, dan memverifikasi apakah output yang dihasilkan oleh perangkat lunak sudah sesuai. Semua kasus dalam pengujian fungsional didasarkan pada spesifikasi kebutuhan perangkat lunak, khususnya berkaitan dengan proses bisnis yang divalidasi dengan input dan output dari perangkat lunak.
2. **Uji Non Fungsional.** Berfokus pada kriteria non fungsional, seperti: kapasitas, kinerja/ response time, kehandalan, ketersediaan dan sejenisnya. Pengujian non fungsional memerlukan lingkungan dan data uji yang memadai sehingga dapat melihat dampak dan perilaku perangkat lunak pada kondisi-kondisi ekstrim yang diinginkan, seperti: peak season, kapasitas tertinggi sampai perangkat lunak tidak berfungsi normal.

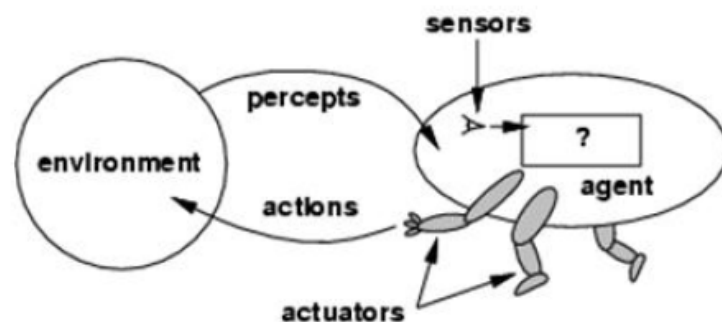
Baik dalam uji fungsional maupun non fungsional dapat menggunakan metode konvensional atau terotomasi. Dalam konteks otomasi, diperlukan sejumlah alat/ kakas yang dapat menjalankan proses uji secara otomatis melalui suatu agen uji (test agent). Dalam konteks otomasi juga diperlukan adanya mekanisme penyiapan rencana uji (test plan) berupa skenario dan task yang perlu dilakukan beserta kriteria uji yang harus dibuktikan/ dicocokkan oleh agen uji untuk menyatakan suatu hasil uji sudah valid atau tidak.

Dalam pengujian aplikasi berbasis Java sudah cukup banyak tools yang dapat digunakan untuk otomasi, seperti JMeter dan Junit. Tools ini dikhususkan untuk para programmer, sehingga harus memiliki kemampuan untuk membuat program dalam bahasa Java tersebut.

Khusus untuk otomasi uji berbasis web, ada juga beberapa library yang dapat digunakan, diantaranya HTTP Unit dan HTML Unit yang keduanya dikembangkan dalam bahasa Java. Uraian lebih lanjut tentang library ini dibahas pada bab 3.

2.1. Agen Cerdas (*Intelligent Agent*)

Agent adalah suatu sistem yang menerima informasi dari lingkungan dan memberikan aksi sebagai respon atas informasi tersebut. Agent mengamati lingkungan melalui sensor dan melakukan aksi terhadap lingkungan melalui aktuator [6]. Interaksi agent dengan lingkungan dapat digambarkan lebih lanjut seperti pada gambar 2.6.



Gambar 2.6. Interaksi Agent dengan Lingkungannya

Dalam mendisain sebuah agent diperlukan langkah awal berupa penentuan lingkungan secara lengkap berupa: (1) performance measurement, (2) environment, (3) actuators dan (4) sensors.

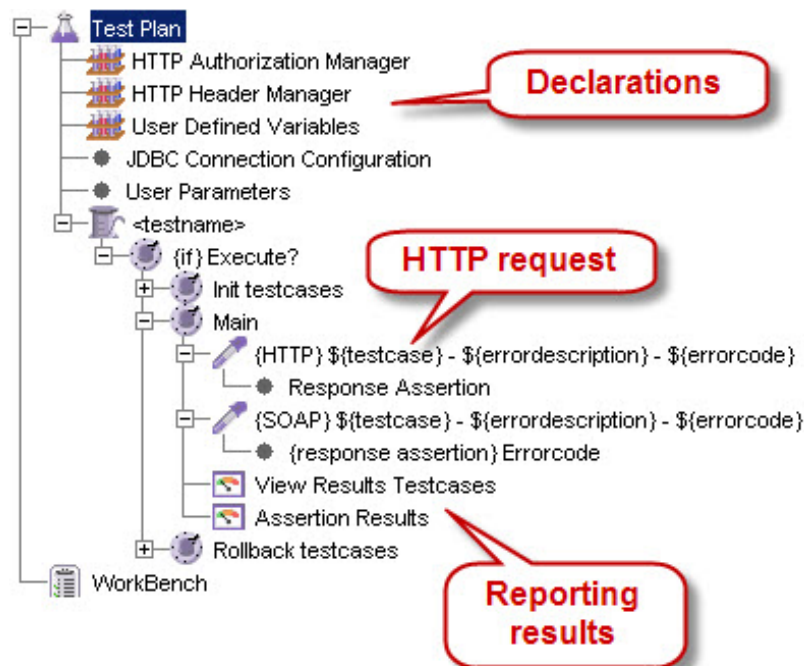
BAB 3 EKSPLORASI PERANGKAT LUNAK TOOLS OTOMASI UJI

Pada bagian ini dibahas tentang beberapa perangkat lunak yang dijadikan acuan dalam pengembangan perangkat lunak otomasi uji fungsional ini, antara lain: Jmeter, HTTPUnit dan HTMLUnit. Masing-masing perangkat lunak memiliki library yang berbasis Java, sehingga dapat diadopsi dan diadaptasi sesuai dengan kebutuhan.

3.1. Apache JMeter

Apache JMeter (selanjutnya disebut JMeter) merupakan perangkat lunak *open source* berbasis Java desktop yang digunakan untuk menguji perilaku fungsional dan mengukur performansi suatu perangkat lunak berbasis web. Pada bagian ini dijelaskan tentang prinsip kerja dan pembuatan *test plan* saja, uraian lebih detail ada pada Lampiran 1 Apache JMeter.

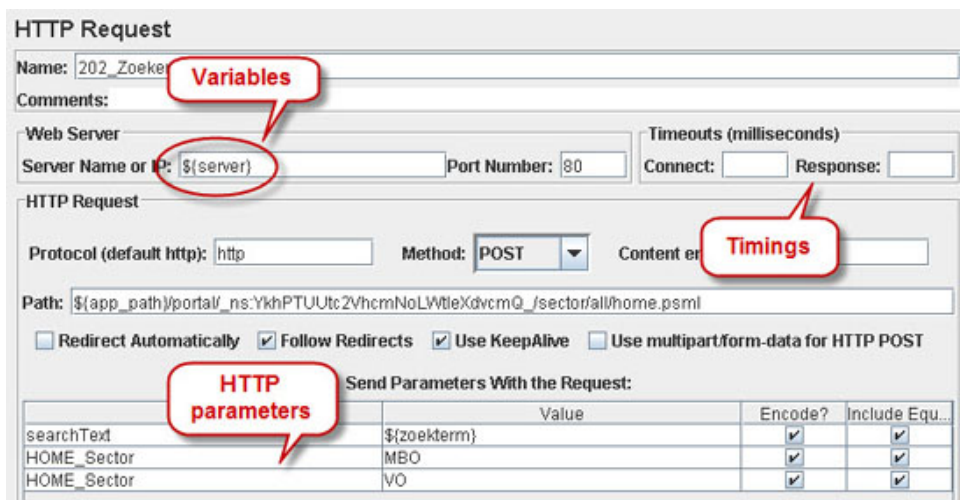
Prinsip kerja dari JMeter sangat sederhana. Jika user menginginkan test seperti pada HTTP request, yang dibutuhkan pada dasarnya adalah URL dan HTTP request. Dimulai dari situ user dapat membangun Test Plan. Pada JMeter terdapat variables, counters, paramater, file SCV, pengulangan, dll.



Gambar 3.1. JMeter Tree

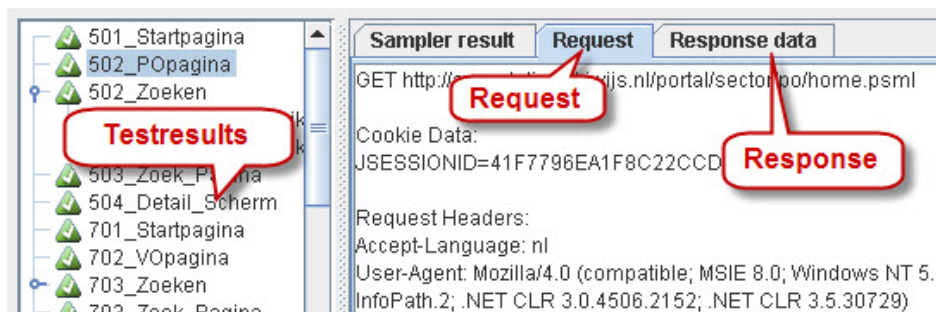
Gambar 3.1 memperlihatkan sebuah contoh *Test Plan*. *Test Plan* dimulai dengan mendeklarasikan otorisasi dan menetapkan variabel pengguna. Koneksi JDBC juga disertakan. Jika ada database, maka dapat mengatur konfigurasi database dan username/password.

HTTP request mempunyai variable pada namanya. Nama dari *HTTP request* akan diperlihatkan didalam *JMeter Tree* dan oleh karena itu berguna untuk memberikan nama dari beberapa jenis referensi dari pengujian yang dijalankan. Pada kasus ini *testcase number*, *error description* dan *error code* digunakan untuk mengidentifikasi permintaan.



Gambar 3.2 HTTP Request Pada JMeter

Gambar 3.2 menunjukkan panel detail dari *HTTP Request*. Anda dapat menambahkan semua informasi yang dibutuhkan seperti nama server, portnumber, protokol, parameter dan lain-lain.



Gambar 3.3 View Result Tree Pada JMeter

Untuk tujuan pelaporan JMeter menawarkan beberapa kemungkinan. Paling umum adalah *Results Tree*. *Results Tree* ini (lihat gambar 3.3) akan menampilkan permintaan dan memiliki kemampuan untuk menunjukkan permintaan dan respon. Tanggapan dapat ditampilkan sebagai teks, XML, HTML atau JSON. Pandangan komponen *Results Tree* ini banyak digunakan untuk pengujian fungsional.



Gambar 3.4 *Aggregate Graph* Pada JMeter

Salah satu laporan untuk melihat pengujian beban adalah *Aggregate Graph* (gambar 3.4). Selama kinerja sedang diuji pengguna dapat memonitor semua jenis data statistik seperti sampel, rata-rata, median, garis 90% dan min / max. Data dapat ditulis ke file (teks biasa, csv atau xml).

Membangun Test Plan

Untuk Melakukan Pengujian dengan menggunakan Jmeter tentu perlu membangun terlebih dahulu sebuah *Test Plan*. Berikut cara membangun sebuah *Test Plan*.

1. **Menambah dan Menghapus Elemen.** Menambahkan elemen ke *Test Plan* dapat dilakukan dengan mengklik kanan pada sebuah elemen pada pohon, lalu memilih sebuah elemen baru untuk ditambahkan dari daftar "*add*". Untuk menghapus elemen, pastikan elemen yang akan dihapus terpilih, lalu klik kanan pada elemen tersebut dan pilih pilihan "*remove*".
2. **Memuat dan menyimpan elemen.** Untuk memuat sebuah elemen dari file, klik kanan pada elemen pohon yang ingin ditambahkan sebuah elemen baru dari file, dan pilih pilihan "*merge*". Pilih file yang diinginkan. JMeter akan menggabungkan elemen yang dipilih dari file ke dalam pohon. Untuk menyimpan elemen, klik kanan pada elemen dan pilih pilihan "*Save Selection As*". JMeter akan menyimpan elemen yang dipilih, ditambah elemen semua anak di bawahnya.
3. **Konfigurasi Pohon Elemen.** Setiap elemen di dalam pohon pengujian akan menyajikan kontrol di dalam Jmeter sebelah kanan. Kontrol ini memungkinkan untuk dikonfigurasi sesuai dengan jenis elemen tersebut.
4. **Menyimpan Test Plan.** Untuk menyimpan *Test Plan*, pilih "*Save*" atau "*Save Test Plan As*" dari menu File.
5. **Menjalankan Test Plan.** Untuk menjalankan *Test Plan*, pilih "*Start*" (Kontrol + r) dari menu bar "*Run*". Ketika JMeter berjalan, hal itu menunjukkan sebuah kotak hijau kecil di ujung sebelah kanan bagian tepat di bawah menu bar. Angka-angka di sebelah kiri kotak hijau adalah jumlah *thread* aktif / jumlah total *thread*
6. **Menghentikan Test Plan.** Ada dua jenis perintah berhenti tersedia dari menu: (a) *Stop* (Kontrol + '.') untuk menghentikan *thread* dengan segera jika memungkinkan. (b) *Shutdown* (Kontrol + ',') untuk permintaan *thread* agar berhenti pada akhir setiap pekerjaan. Tidak akan mengganggu setiap sampel yang aktif.

3.2. HTTP Unit

Penguji otomatis adalah cara yang baik untuk memastikan program dipelihara secara baik. Untuk melakukan pengujian otomatis terutama untuk mengetes performa suatu web site,

penguji harus mampu melewati browser dan mengakses situs dari sebuah program. Dengan menggunakan HttpUnit hal tersebut menjadi mudah dilakukan.

Ditulis dengan bahasa java, HttpUnit mengemulasi bagian yang relevan dari perilaku browser, termasuk pengiriman form, JavaScript, otentikasi Http, cookies dan pengalihan halaman (page) otomatis, dan memungkinkan kode uji Java mampu memeriksa kembali halaman baik berupa text, DOM XML, atau bentuk tabel dan link. Ketika dikombinasikan dengan kerangka seperti JUnit, hal itu akan cukup mudah untuk menulis tes uji yang sangat cepat memverifikasi fungsi situs web.

Dari beberapa percobaan yang telah dilakukan, HTTP Unit sangat powerfull untuk mengemulasi web browser untuk mengakses situs dengan materi HTTP dan dokumen HTML. Kendala dirasakan pada saat halama web menggunakan Java Script dan Ajax. HTTP Unit telah mendukung Java Script sederhana, tapi belum mendukung secara penuh Ajax.

Informasi lebih detail tentang HTTPUnit dapat dilihat pada Lampiran 2 HTTP Unit.

3.3. HTML Unit

HtmlUnit adalah *open source* library java untuk fungsionalitas web browser. HtmlUnit merupakan model dokumen HTML dan menyediakan sebuah API yang memungkinkan untuk memanggil halaman, mengisi form, klik link, dll. seperti yang biasa dilakukan oleh "normal" browser pada umumnya.

HtmlUnit memiliki dukungan javascript yang cukup baik dan mampu bekerja bahkan dengan library Ajax yang cukup kompleks. HtmlUnit sendiri biasanya digunakan untuk tujuan pengujian atau untuk mengambil informasi dari situs web.

Informasi lebih detail tentang HTML Unit dapat dilihat pada Lampiran 3 HTML Unit.

BAB 4 ANALISIS KEBUTUHAN DAN DISAIN SOLUSI

Pada bagian ini dijelaskan tentang analisis kebutuhan perangkat lunak uji performansi dan kapasitas sistem berbasis web. Pada bagian ini juga dijelaskan bagaimana disain dan implementasi dari perangkat lunak tersebut pada lingkungan Java.

4.1. Analisis Kebutuhan

Berdasarkan latar belakang, rumusan masalah dan tujuan dapat didefinisikan bahwa yang dibutuhkan untuk melakukan otomasi uji coba aplikasi berbasis web dalam skala besar adalah:

1. Kriteria uji fungsional, khususnya dengan metode black box, yaitu berupa: input dan skenario yang dijalankan oleh user sehingga menghasilkan output yang diharapkan. Dalam hal ini perlu ada mekanisme memasukkan skenario uji (test plan) yang berisi task-task atau langkah/ instruksi yang dilakukan oleh user berikut data-data masukannya. Setelah itu diberikan kondisi-kondisi dan output yang harus dihasilkan. Jika output yang dihasilkan sesuai dengan rencana, maka uji fungsional memberikan hasil sukses. Demikian juga sebaliknya, jika output tidak sesuai rencana akan memberikan hasil gagal.
2. Untuk setiap fitur dari aplikasi harus didefinisikan skenarionya. Setiap skenario akan mengandung instruksi sederhana, kondisional maupun pengulangan. Oleh karena itu Agent harus memiliki kemampuan untuk mengeksekusi perintah sederhana, kondisional maupun pengulangan, serta memberikan status berdasarkan kriteria uji pada point (1).
3. Agar pengujian dapat dilakukan secara simultan dalam konteks enterprise, maka Agent Uji juga dapat didistribusikan pada suatu lingkungan, sehingga eksekusi dari skenario uji fungsional dapat dilakukan secara paralel. Dengan demikian pengujian dapat berjalan produktif sesuai dengan target pengembangan dalam skala besar. Oleh karena itu, lingkungan pengujian dapat memanfaatkan skema terdistribusi multi agent seperti pada gambar 4.1.
4. Sistem otomasi uji fungsional juga harus dapat mencatat/ merecord seluruh jejak pengujian, baik pendefinisian rencana uji (skenario), pemesanan kepada para agent (order) maupun hasil dari pengujian, baik hasil global yang hanya menyatakan hasil uji berhasil atau tidak, maupun jejak eksekusi setiap langkah/ task dalam pengujian yang dilengkapi dengan referensi waktu sehingga mudah dilacak. Demikian juga dengan hasil/

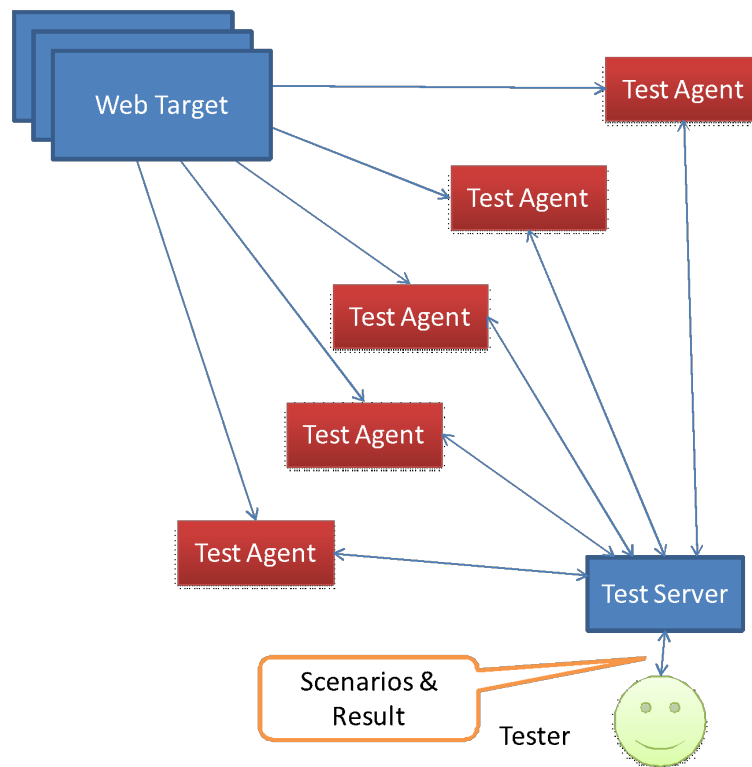
output dari pengujian perlu direkap sedemikian rupa sehingga dapat diamati pada waktunya.

5. Dari sisi teknologi dapat mengadopsi JMeter dalam perencanaan uji (test plan) sehingga dapat mengkonstruksi skenario yang detail dan kriteria hasil yang detail pula (assertion). Untuk memudahkan akses terhadap situs web dan dapat memberikan aksi yang sesuai dapat menggunakan library dari HTTPUnit dan HTMLUnit. Karena uji lebih bersifat fungsional, maka lebih tepat menggunakan HTMLUnit yang memiliki fungsi termasuk java script dan ajax yang saat ini banyak digunakan dalam pengembangan aplikasi web enterprise. Untuk lingkungan terdistribusi multi agent, dapat menggunakan teknologi java multi threading seperti pada 2 penelitian sebelumnya. Model pada 2 penelitian sebelumnya masih relevan diterapkan untuk meningkatkan produktifitas eksekusi pengujian.
6. Karena agent uji harus dapat mengeksekusi skenario, maka perlu dibuat bahasa yang dapat diberikan oleh tester kepada agent uji melalui mekanisme terpusat. Oleh karena itu perlu mengkonstruksi bahasa yang lebih manusiawi yang diberikan oleh tester untuk melakukan sejumlah aksi browsing untuk memanipulasi objek HTML dan Java Script serta memiliki kemampuan untuk menentukan apakah output dari browser sesuai dengan output yang direncanakan. Dalam hal ini diperlukan bahasa yang dapat memanipulasi semua object yang ada di browser dan semua aksi yang dapat dilakukan oleh user secara umum.

Untuk memenuhi kebutuhan di atas diajukan solusi dengan model seperti pada batian 4.2. Selanjutnya model tersebut dieksplorasi lebih jauh untuk menghasilkan disain global, detail dan diimplementasikan dengan teknologi yang sudah dieksplorasi pada bagian 3.1-3.3.

4.2. Model Solusi

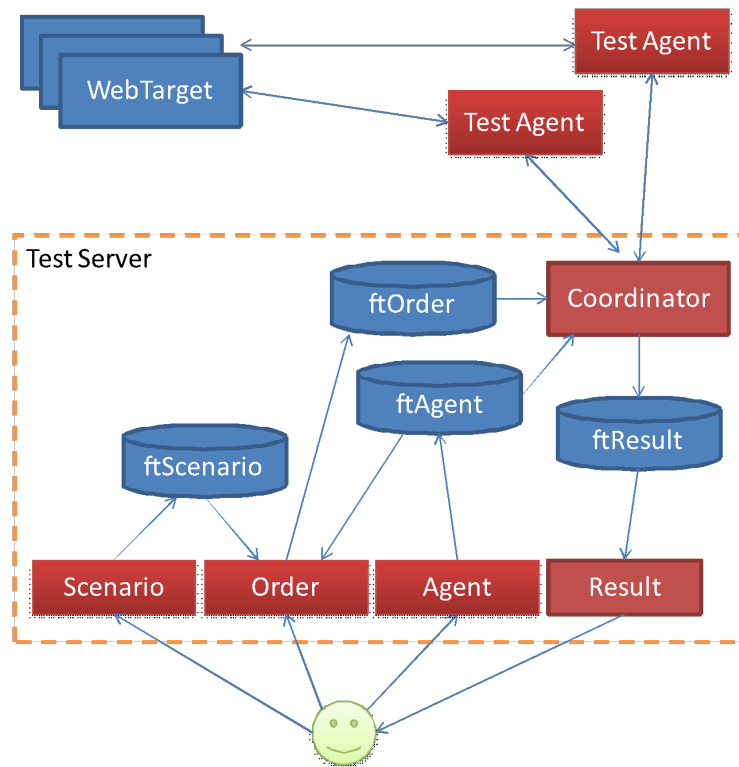
Berdasarkan definisi kebutuhan tersebut dapat diusulkan model solusi dengan arsitektur logika yang dapat dilihat pada gambar 4.1.



Gambar 4.1. Model Otomasi Uji Fungsional

Web Target (WT) merupakan web server yang menjadi host dari situs web yang akan diuji. Test Agent (TA) merupakan emulasi dari 1 User Agent (UA) berupa *browser emulator*, dengan demikian akan ada banyak TA yang akan melakukan uji fungsional. Semua TA mendapat perintah uji berupa skenario dan task dari Test Server (TS). TS menerima skenario dan task dari Tester (penguji). Skenario mewakili kasus uji fungsional, sedangkan task mewakili instruksi berupa langkah-langkah aksi yang dilakukan oleh user melalui TA.

Lebih detail, proses persiapan dan eksekusi dari otomasi uji fungsional dapat dilihat pada gambar 4.2.



Gambar 4.2. Model Detail Test Server

Pada gambar 4.2 dapat dilihat bahwa, Test Server, terdiri atas 5 elemen, yaitu: Scenario merupakan aplikasi untuk generate skenario. Skenario yang dihasilkan disimpan pada basis data ftScenario. Elemen kedua adalah Agent, merupakan aplikasi untuk mengelola Test Agent (TA) sehingga semua dapat dikordinasikan dengan baik. Informasi yang dicatat adalah AgentName yang merupakan identitas dari setiap TA. Semua informasi tentang TA disimpan pada basis data ftAgent. Elemen ketiga adalah Order, yang berfungsi untuk memesan agar suatu skenario dijalankan oleh sebuah TA. Semua pesanan disimpan pada basis data ftOrder. Elemen keempat adalah Coordinator, yang bertugas untuk mengkoordinasikan semua TA agar dapat bekerja sesuai dengan order yang ada di basis data Order. Setiap TA akan selalu berkomunikasi dengan Coordinator untuk menanyakan apakah ada order untuk dirinya, jika ada, maka akan memintanya kemudian akan menjalankan skenario yang ada di dalam pesanan. Semua hasil eksekusi order oleh TA diberikan kembali kepada Coordinator untuk disimpan di basis data ftResult. Elemen terakhir dari Test Server adalah Result yang akan menghasilkan laporan eksekusi dari semua skenario berdasarkan order yang diberikan kepada TA. Tester (penguji) dapat memasukkan

skenario melalui Scenario, kemudian mengelola semua TA melalui Agent dan mengordernya melalui Order serta dapat memantau hasilnya untuk dianalisis melalui Result.

4.3. Konstruksi Bahasa Pada Skenario

Agar skenario yang dituliskan oleh Tester dapat dieksekusi oleh TA, maka dibuat suatu bahasa yang dapat dimengerti oleh TA. Konstruksi bahasa ini didasarkan pada elemen-elemen tampilan (*user interface*) pada aplikasi berbasis web dan aksi-aksi yang dapat dilakukan oleh user kepada elemen tersebut. Setiap elemen selanjutnya disebut *Object*, dan setiap aksi selanjutnya disebut *Action*. Konstruksi ini juga diinspirasi oleh aksi yang didukung oleh library HTML Unit.

Setiap skenario terdiri atas sejumlah *task*. Setiap *task* memiliki konstruksi bahasa dengan syntax:

```
<object> <action> [<param1> [<param2>]].
```

Di mana:

Object: elemen tampilan web.

Action: aksi dari setiap elemen.

Param1: parameter pertama yang menyatakan data atau variabel jika ada.

Param2: parameter kedua jika ada.

Dengan demikian, konstruksi minimal adalah *<object> <action>*. Bahasa yang didukung dikelompokkan menjadi:

1. **Statemen Sederhana.** Menyatakan suatu task yang harus dijalankan TA berupa pernyataan sederhana seperti:

- url request <url>
- page setframe <name>
- link click <ahref>
- element get <id>
- form first // set current form to first form
- form set <name> // set current form to <name>
- form submit
- button click <name>
- submit click <name>
- text fill <name> <value>
- textarea fill <name> <value>
- radio select <name> <value>

- check select <name> <value>
- combo select <name> <value>
- combo selectmulti <name> <value> // value = (val1, val2, ...)
- text get <name> <var>
- textarea get <name> <var>
- radio get <name> <var>
- check get <name> <var>
- combo get <name> <var>
- combo getmulti <name> <var>

2. **Statemen Kondisional.** Pernyataan yang dieksekusi jika suatu kondisi terpenuhi. Seperti *if <condition> then <action> else <action>*. Dalam penelitian ini menggunakan kata kunci *assert*, dengan syntax sebagai berikut:

```
assert <operator> [<param1> [<param2>]]
  <task1>
  ...
  [<task n>]
```

Pengaruh assert berlaku sampai dengan ketemu assert berikutnya atau instruksi habis. Berikut adalah beberapa varian dari assert:

- assert title <value1>
- assert contain <value1>
- assert equal <value1> <value2>
- assert match <value1> <value2>
- assert else

3. **Statement Pengulangan.** Pernyataan yang memungkinkan eksekusi secara berulang. Konstruksi yang digunakan adalah *while <condition> do <action>*. Berikut syntax lengkapnya:

```
While [equal|contain|match] <param1> <param2> //... sama dengan assert
  <task1>
  ...
  [<taskn>]
While end
```

4. **Variabel.** Agar memiliki sistem pengendalian yang memadai, maka bahasa yang didukung juga termasuk variabel sederhana. Berikut syntaxnya:

```

var create <name> <initial value>
var setvar <name> <var> // value = [string|integer|boolean|array] as string
var setval <name> <value>
var get <name> <type> // type = [string|integer|boolean|array]
    array = (val1, val2, ...) // jason
var op <name> [name| <value>] // op = [+|-|*|/] :
    // name = name op [ name | <value>]

```

5. **Kontrol Skenario.** Untuk pengendalian terhadap eksekusi suatu skenario di dalam skenario lain, dan melaporkan status eksekusi apakah berhasil atau gagal, maka dibuat juga sintaks khusus untuk skenario, yaitu:

```

Scenario exec <name> // mengeksekusi skenario dengan nama <name>
Scenario report <value> // value = [success| fail]
    // melaporkan suatu skenario apakah memberikan hasil eksekusi

```

Berikut beberapa contoh penjelasan lebih detail tentang pernyataan di atas:

url request <url>

object url memiliki action request dengan param1 <url> yaitu alamat situs web yang diambil isinya.

page setframe <name>

object page memiliki action setframe dengan param1 <name> yaitu nama frame yang akan diset sebagai halaman yang dapat diakses elemen-elemen yang ada di dalamnya.

Penjelasan selengkapnya dapat dilihat pada lampiran 5 Konstruksi Bahasa Otomasi Web Enterprise.

Task-task di atas digunakan untuk mengkonstruksi skenario yang mewakili suatu uji fitur fungsional tertentu. Hal ini diinspirasi oleh bagaimana pembuatan Test Plan pada JMeter. Berikut adalah contoh skenario login pada situs tertentu.

Skenario Login

```

url request http://centeris.unpar.ac.id/sample/
link click "login"
form first
text fill "username" "7308001"
text fill "password" "satu"
button click "login"
assert title "Login Berhasil"
    // ..... instruksi kalau berhasil
Scenario report "success"
Scenario exec "FRS"
assert title "Login Gagal"

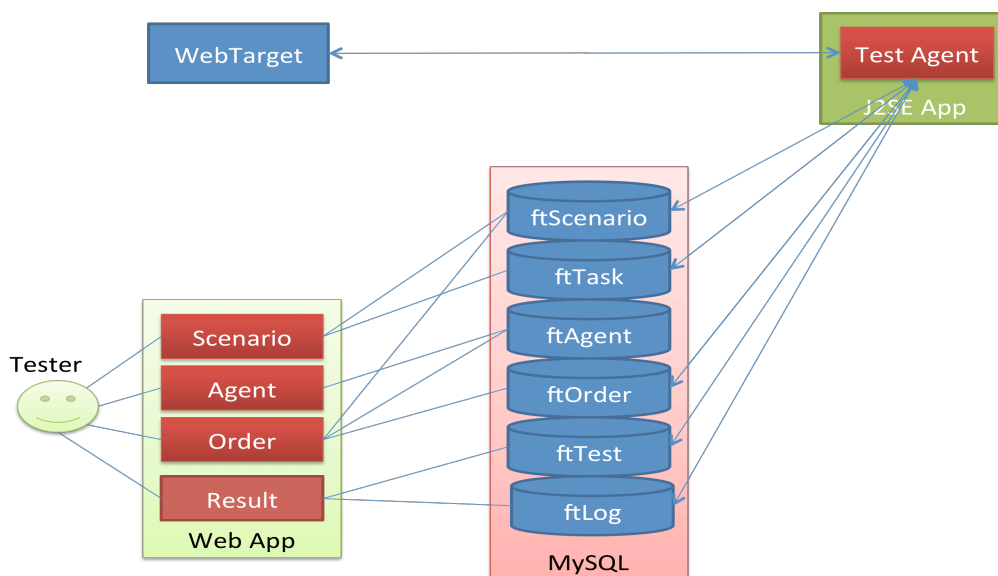
```

```
// ... instruksi kalau login gagal
Scenario report "fail"
```

Pada skenario di atas, TA diminta untuk mengakses url <http://centeris.unpar.ac.id/sample/> kemudian meklik tombol dengan label "login". Setelah itu mencari form pertama, kemudian mengisi text dengan label "username" dengan "7308001" dan mengisi text dengan label "password" dengan "satu". Setelah itu mengklik tombol dengan label "login". Jika menghasilkan halaman web dengan judul (title) "Login Berhasil" maka skenario ini dinyatakan sukses. Sementara itu, jika judul hasil adalah "Login Gagal" maka skenario ini dinyatakan gagal.

4.4. Disain Solusi

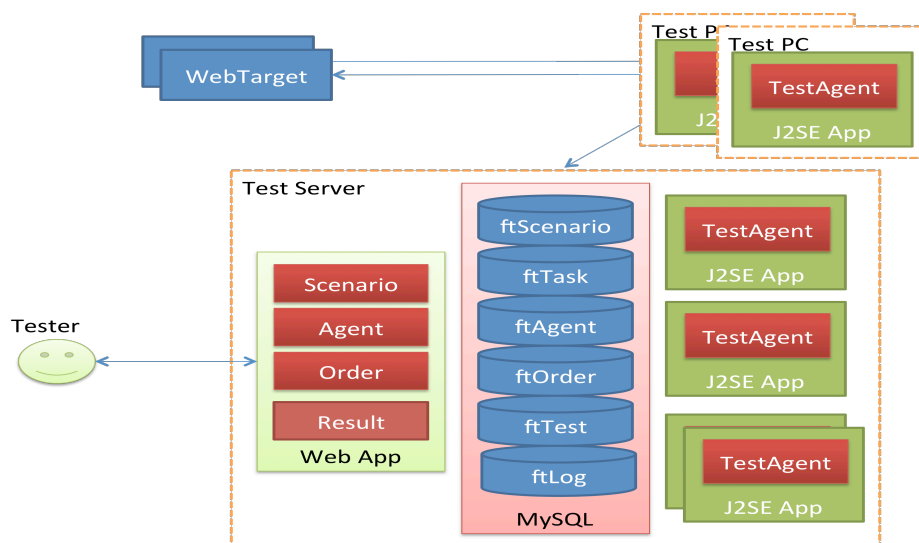
Model solusi pada bagian 4.2 diimplementasikan dengan teknologi sesuai dengan peruntukannya. Untuk Test Agent (TA) diimplementasikan menggunakan teknologi Java *multi threading* pada platform J2SE (*Java 2 Standar Edition*). Sedangkan Test Server (TS) diimplementasikan dengan teknologi web (Web App) menggunakan server Apache dan lingkungan pemrograman PHP. Untuk penyimpanan digunakan basis data MySQL. Lebih lanjut disain aplikasi dan teknologi yang digunakannya dapat dilihat pada gambar 4.3.



Gambar 4.3. Disain Aplikasi Otomasi Uji Fungsional

Pada gambar 4.3 dapat dilihat bahwa elemen Scenario, Agent, Order dan Result diimplementasikan menggunakan teknologi Web App. Untuk basis data menggunakan MySQL. Basis data ftScenario pada gambar 4.2 diimplementasikan menjadi 2 basis data, yaitu: ftScenario untuk menyimpan atribut dari suatu skenario, dan ftTask untuk menyimpan langkah-langkah (instruksi) dari suatu skenario, termasuk di dalamnya kriteria uji dari setiap instruksi. Demikian juga dengan ftResult pada gambar 4.2 juga diimplementasikan menjadi 2 basis data yaitu: ftTest untuk menyimpan status eksekusi dari setiap order oleh setiap TA, dan ftLog yang berisi log/ jejak dari hasil eksekusi setiap task/ instruksi oleh TA. Laporan dibangkitkan dari ftTest dan ftLog ini. Khusus untuk Coordinator tidak diimplementasikan tersendiri, melainkan diwakili oleh MySQL server yang langsung dapat diakses oleh masing-masing TA melalui teknologi JDBC (*Java Data Base Connectivity*).

Dari sisi deployment aplikasi, didisain bahwa aplikasi TA dapat berjalan baik pada lingkungan server maupun pada lingkungan desktop (*personal computer – PC*). Dengan demikian disain deployment menjadi seperti pada gambar 4.4.

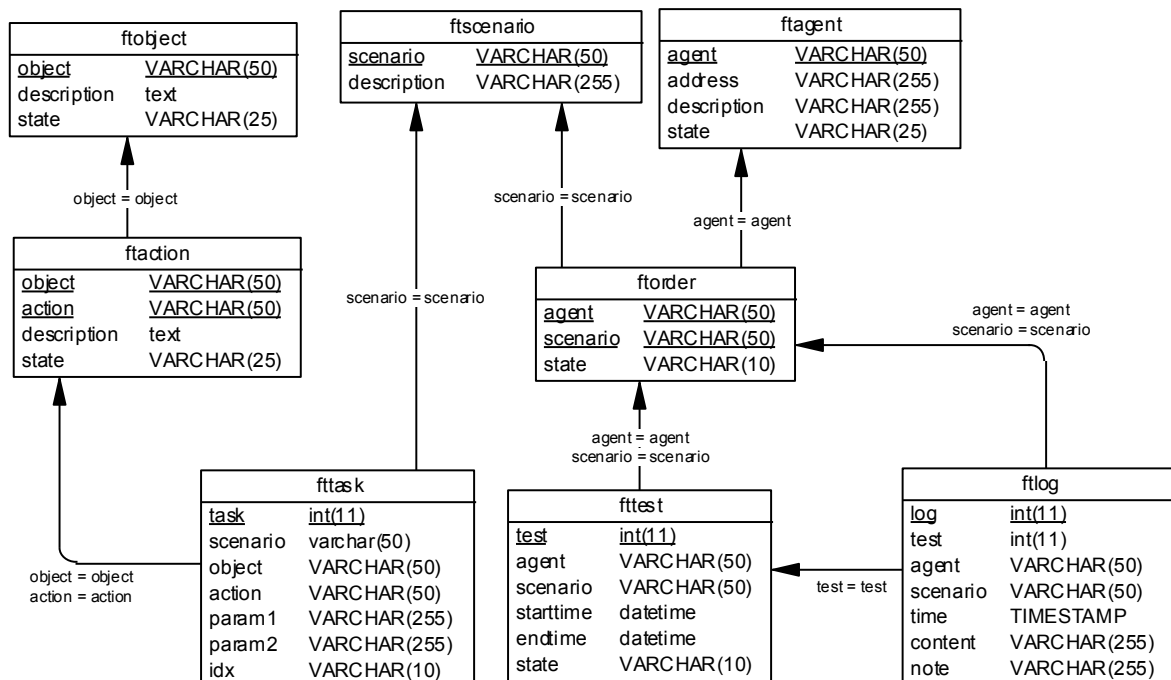


Gambar 4.4. Disain Deployment Aplikasi Otomasi Uji Fungsional

4.4. Disain Detail Perangkat Lunak Otomasi Uji Fungsional

4.4.1. Rancangan Basis Data Test Server

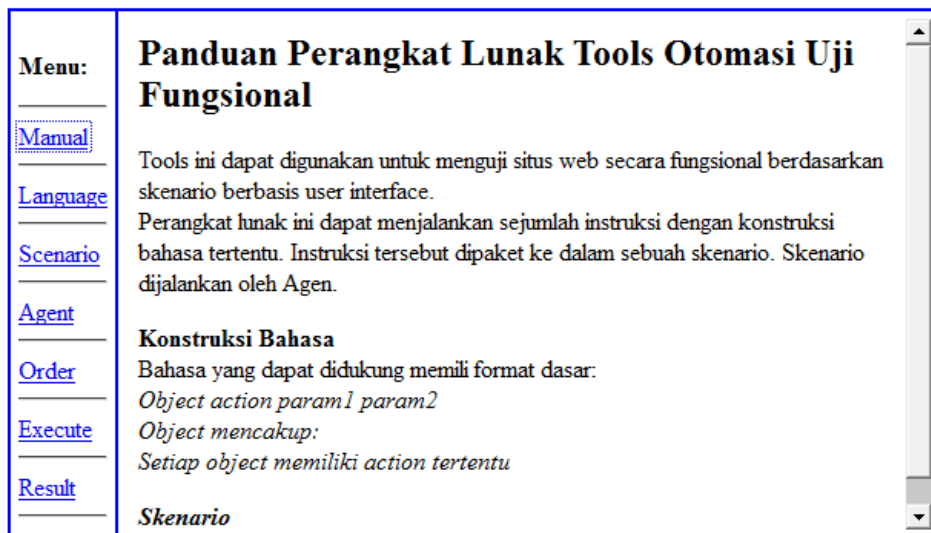
Rancangan basis data pada TS dapat dilihat pada gambar 4.5. Nama basis data adalah **function_test**. Basis data tersebut memiliki 8 tabel. Basis data pada gambar 4.4 diimplementasikan menjadi 6 tabel dengan nama yang sama. Ada 2 tabel baru, yaitu: ftObject dan ftAction yang mengimplementasikan konstruksi bahasa pada bagian 4.3. Setiap Object memiliki nama dan deskripsi, sedangkan setiap Action adalah atribut dari suatu Object dan memiliki deskripsi (description) serta status (state) apakah sedang dapat digunakan (active) atau tidak (notactive).



Gambar 4.5 Rancangan Basis Data Test Server

4.4.2. Rancangan Antar Muka Web App

Antarmuka Web App terdiri atas menu utama dan halaman-halaman web turunannya. Struktur menu dapat dilihat pada gambar 4.6.



Gambar 4.6 Struktur Menu Web App

Pada gambar 4.6 dapat dilihat bahwa ada 7 menu yang disajikan pada Web App, yaitu:

1. Manual: menampilkan petunjuk penggunaan dari aplikasi ini.
2. Language: berisi penjelasan dari konstruksi bahasa, berupa Object dan Action yang didukung oleh aplikasi ini.
3. Scenario: berisi daftar skenario dan task yang telah terdaftar, berikut manipulasi (add/edit/delete) dari skenario dan task tersebut.
4. Agent: berisi daftar agent dan pengelolaannya (add/ edit/ delete).
5. Order: berisi daftar order/ pesanan eksekusi suatu skenario oleh sebuah agent, berikut pengelolaannya (add/edit/ delete).
6. Execute: berisi daftar order dan status berjalannya. Pada saat penambahan Order dilakukan, maka status order adalah Idle. Jika ingin menjalakkannya, maka status harus diubah melalui menu ini.
7. Result: menampilkan hasil dari eksekusi suatu Order. Selain itu juga dapat dilihat log detail yang berisi hasil eksekusi dari setiap task dari skenario yang dieksekusi oleh sebuah agent.

Jika menu Language diklik, maka akan muncul tampilan seperti pada gambar 4.7.

Menu:

[Manual](#)

[Language](#)

[Scenario](#)

[Agent](#)

[Order](#)

[Execute](#)

[Result](#)

Language Editor

Instruksi: Anda dapat menambah, mengubah atau menghapus konstruksi bahasa berupa Object atau Action pada halaman ini.

List of Object

Id	Name	Description	Do
1.	url	Uniform resource locator, menyatakan alamat web site yang akan diakses	Edit Delete Action
2.	page	Halaman web	Edit Delete Action
3.	link	link web	Edit Delete Action
4.	element	Elemen yang ada di page	Edit Delete Action
5.	form	Penanganan form	Edit Delete Action
6.	button	button pada form	Edit Delete Action

Menu:

[Manual](#)

[Language](#)

[Scenario](#)

[Agent](#)

Action of Object form

Instruksi: Anda dapat menambah, mengubah atau menghapus Action pada halaman ini.

Id	Action Name	Description	Do
5.	first	Mengakses ke form pertama	Edit Delete
6.	set	Mengakses form dengan id tertentu	Edit Delete
7.	submit	Mensubmit form	Edit Delete

Gambar 4.7 Daftar Object dan Action pada Language Editor

Untuk menu Scenario dapat dilihat pada gambar 4.8.

Menu:

[Manual](#)

[Language](#)

[Scenario](#)

[Agent](#)

[Order](#)

[Execute](#)

[Result](#)

List of Scenario

Instruksi: Anda dapat menambah, mengubah atau menghapus Skenario, serta mengatur Task pada halaman ini.

Id	Name	Description	Do
1.	login	Skenario untuk menguji login	Edit Delete Clone Task
2.	frs	Skenario untuk menguji frs	Edit Delete Clone Task
3.	login2	Login yang dimodifikasi oleh Gede Karya	Edit Delete Clone Task
4.	kompas	Akses ke situs kompas	Edit Delete Clone Task
5.	webunpar	Akses ke web Unpar	Edit Delete Clone Task
6.	loginwhile	testwhile	Edit Delete Clone Task
7.	TambahAgen	Menambah agen pada sistem ini!	Edit Delete Clone Task
8.	TambahAgenImprove	perbaikan	Edit Delete Clone Task
9.	login4	login yang dimodifikasi oleh Ari dan Iqbal	Edit Delete Clone Task

Gambar 4.8 Daftar Skenario

Pada gambar 4.8, jika Task diklik, maka akan menampilkan daftar task seperti pada gambar 4.9. Selain itu juga ada fasilitas Clone untuk mengcopy semua task dalam suatu skenario untuk keperluan modifikasi.

Menu:	List of Task login2							
Manual	Instruksi: Anda dapat menambah, mengubah atau menghapus Task pada halaman ini.							
Language	No	Id	Idx	Object	Action	Param1	Param2	Do
Scenario	1.	11.	11	url	request	"http://centeris.unpar.ac.id/sample/"		Edit Delete
Agent	2.	12.	12	link	clickHref	"Login.php"		Edit Delete
Order	3.	13.	13	form	first			Edit Delete
Execute	4.	14.	14	text	fill	"username"	"7308001"	Edit Delete
Result	5.	15.	15	text	fill	"pass"	"satu"	Edit Delete
	6.	16.	16	submit	click	"login"		Edit Delete
	7.	17.	17	assert	content	"Rencana Studi"		Edit Delete
	8.	18.	18	scenario	report	success		Edit Delete

Gambar 4.9 Daftar Task dari Scenario Login2

Untuk menu Agent, hasilnya dapat dilihat pada gambar 4.10.

Menu:	List of Agent				
Manual	Instruksi: Anda dapat menambah, mengubah atau menghapus Agent pada halaman ini.				
Language	Id	Name	Description	Port	Do
Scenario	1.	agent1	Agent yang dipakai I	50	Edit Delete
Agent	2.	agent2	Agen yang dipakai Ar	51	Edit Delete
Order	3.	agent3	Agen yang dipakai Ad	50	Edit Delete
Execute	4.	agent4Otomat	Agen yang bekerja secara otomatis di Server	212	Edit Delete
Result	5.	agent5Otomat	Agen yang bekerja secara otomatis di server. Silakan ditugaskan saja!!!!	212	Edit Delete
	6.	AgenContoh1	Pada server ini	21	Edit Delete

Gambar 4.10 Daftar Agent

Sedangkan untuk menu Order, hasilnya seperti pada gambar 4.11.

Menu:	List of Order																																													
Manual	Instruksi: <i>Anda dapat menambah, mengubah atau menghapus Order pada halaman ini.</i>																																													
Language	<table border="1"> <thead> <tr> <th>Id</th> <th>Agent</th> <th>Scenario</th> <th>State</th> <th>Do</th> </tr> </thead> <tbody> <tr> <td>1.</td> <td>agent1</td> <td>login</td> <td>Idle</td> <td>Edit Delete</td> </tr> <tr> <td>3.</td> <td>agent2</td> <td>login</td> <td>Idle</td> <td>Edit Delete</td> </tr> <tr> <td>4.</td> <td>agent2</td> <td>login2</td> <td>Idle</td> <td>Edit Delete</td> </tr> <tr> <td>6.</td> <td>agent2</td> <td>kompas</td> <td>Idle</td> <td>Edit Delete</td> </tr> <tr> <td>7.</td> <td>agent2</td> <td>webunpar</td> <td>Idle</td> <td>Edit Delete</td> </tr> <tr> <td>8.</td> <td>agent4Otomat</td> <td>kompas</td> <td>Idle</td> <td>Edit Delete</td> </tr> <tr> <td>9.</td> <td>agent5Otomat</td> <td>login</td> <td>Idle</td> <td>Edit Delete</td> </tr> <tr> <td>10.</td> <td>agent4Otomat</td> <td>login2</td> <td>Idle</td> <td>Edit Delete</td> </tr> </tbody> </table>	Id	Agent	Scenario	State	Do	1.	agent1	login	Idle	Edit Delete	3.	agent2	login	Idle	Edit Delete	4.	agent2	login2	Idle	Edit Delete	6.	agent2	kompas	Idle	Edit Delete	7.	agent2	webunpar	Idle	Edit Delete	8.	agent4Otomat	kompas	Idle	Edit Delete	9.	agent5Otomat	login	Idle	Edit Delete	10.	agent4Otomat	login2	Idle	Edit Delete
Id		Agent	Scenario	State	Do																																									
1.		agent1	login	Idle	Edit Delete																																									
3.		agent2	login	Idle	Edit Delete																																									
4.		agent2	login2	Idle	Edit Delete																																									
6.		agent2	kompas	Idle	Edit Delete																																									
7.		agent2	webunpar	Idle	Edit Delete																																									
8.		agent4Otomat	kompas	Idle	Edit Delete																																									
9.		agent5Otomat	login	Idle	Edit Delete																																									
10.		agent4Otomat	login2	Idle	Edit Delete																																									
Scenario																																														
Agent																																														
Order																																														
Execute																																														
Result																																														

Gambar 4.11 Daftar Order

Kita dapat mengeksekusi Order yang telah terdaftar melalui menu Execute, seperti pada gambar 4.12. Pada gambar 4.12, setiap order sedang berstatus Idle. Untuk mengeksekusi klik link Start.

Menu:	List of Order to Execute																																				
Manual	Instruksi: <i>Anda dapat men-start atau me-reset order pada halaman ini.</i>																																				
Language	<table border="1"> <thead> <tr> <th>Id</th> <th>Agent</th> <th>Scenario</th> <th>State</th> </tr> </thead> <tbody> <tr> <td>1.</td> <td>agent1</td> <td>login</td> <td>Start</td> </tr> <tr> <td>3.</td> <td>agent2</td> <td>login</td> <td>Idle (Start)</td> </tr> <tr> <td>4.</td> <td>agent2</td> <td>login2</td> <td>Idle (Start)</td> </tr> <tr> <td>6.</td> <td>agent2</td> <td>kompas</td> <td>Idle (Start)</td> </tr> <tr> <td>7.</td> <td>agent2</td> <td>webunpar</td> <td>Idle (Start)</td> </tr> <tr> <td>8.</td> <td>agent4Otomat</td> <td>kompas</td> <td>Idle (Start)</td> </tr> <tr> <td>9.</td> <td>agent5Otomat</td> <td>login</td> <td>Idle (Start)</td> </tr> <tr> <td>10.</td> <td>agent4Otomat</td> <td>login2</td> <td>Idle (Start)</td> </tr> </tbody> </table>	Id	Agent	Scenario	State	1.	agent1	login	Start	3.	agent2	login	Idle (Start)	4.	agent2	login2	Idle (Start)	6.	agent2	kompas	Idle (Start)	7.	agent2	webunpar	Idle (Start)	8.	agent4Otomat	kompas	Idle (Start)	9.	agent5Otomat	login	Idle (Start)	10.	agent4Otomat	login2	Idle (Start)
Id		Agent	Scenario	State																																	
1.		agent1	login	Start																																	
3.		agent2	login	Idle (Start)																																	
4.		agent2	login2	Idle (Start)																																	
6.		agent2	kompas	Idle (Start)																																	
7.		agent2	webunpar	Idle (Start)																																	
8.		agent4Otomat	kompas	Idle (Start)																																	
9.		agent5Otomat	login	Idle (Start)																																	
10.		agent4Otomat	login2	Idle (Start)																																	
Scenario																																					
Agent																																					
Order																																					
Execute																																					
Result																																					

Gambar 4.12 Eksekusi Order

Hasil eksekusi dari semua Order dapat dilihat melalui menu Result, seperti pada gambar 4.13.

Menu:	List of Test State (20 terakhir)																																																															
Manual	Instruksi: klik link log pada masing-masing test untuk melihat hasil detailnya!																																																															
Language																																																																
Scenario																																																																
Agent																																																																
Order																																																																
Execute																																																																
Result																																																																
	<table border="1"> <thead> <tr> <th>ID</th> <th>Agent</th> <th>Scenario</th> <th>StartTime</th> <th>EndTime</th> <th>State</th> <th>Log</th> </tr> </thead> <tbody> <tr> <td>319.</td> <td>agent2</td> <td>login5</td> <td>2012-07-31 07:01:19</td> <td>2012-07-31 07:01:23</td> <td>fail</td> <td>Log</td> </tr> <tr> <td>318.</td> <td>agent2</td> <td>login4</td> <td>2012-07-31 06:57:57</td> <td>2012-07-31 06:58:07</td> <td>fail</td> <td>Log</td> </tr> <tr> <td>317.</td> <td>agent2</td> <td>login5</td> <td>2012-07-31 06:55:27</td> <td>2012-07-31 06:55:28</td> <td>inprogress</td> <td>Log</td> </tr> <tr> <td>316.</td> <td>agent5Otomat</td> <td>loginwhile</td> <td>2012-07-31 06:54:32</td> <td>2012-07-31 06:54:32</td> <td>success</td> <td>Log</td> </tr> <tr> <td>315.</td> <td>agent5Otomat</td> <td>loginwhile</td> <td>2012-07-31 06:52:32</td> <td>2012-07-31 06:52:32</td> <td>success</td> <td>Log</td> </tr> <tr> <td>314.</td> <td>agent2</td> <td>login4</td> <td>2012-07-31 01:26:48</td> <td>2012-07-31 06:51:34</td> <td>Idle</td> <td>Log</td> </tr> <tr> <td>313.</td> <td>agent2</td> <td>login4</td> <td>2012-07-31 01:25:21</td> <td>2012-07-31 06:51:34</td> <td>Idle</td> <td>Log</td> </tr> <tr> <td>312.</td> <td>agent2</td> <td>login4</td> <td>2012-07-31 01:23:37</td> <td>2012-07-31 06:51:34</td> <td>Idle</td> <td>Log</td> </tr> </tbody> </table>	ID	Agent	Scenario	StartTime	EndTime	State	Log	319.	agent2	login5	2012-07-31 07:01:19	2012-07-31 07:01:23	fail	Log	318.	agent2	login4	2012-07-31 06:57:57	2012-07-31 06:58:07	fail	Log	317.	agent2	login5	2012-07-31 06:55:27	2012-07-31 06:55:28	inprogress	Log	316.	agent5Otomat	loginwhile	2012-07-31 06:54:32	2012-07-31 06:54:32	success	Log	315.	agent5Otomat	loginwhile	2012-07-31 06:52:32	2012-07-31 06:52:32	success	Log	314.	agent2	login4	2012-07-31 01:26:48	2012-07-31 06:51:34	Idle	Log	313.	agent2	login4	2012-07-31 01:25:21	2012-07-31 06:51:34	Idle	Log	312.	agent2	login4	2012-07-31 01:23:37	2012-07-31 06:51:34	Idle	Log
ID	Agent	Scenario	StartTime	EndTime	State	Log																																																										
319.	agent2	login5	2012-07-31 07:01:19	2012-07-31 07:01:23	fail	Log																																																										
318.	agent2	login4	2012-07-31 06:57:57	2012-07-31 06:58:07	fail	Log																																																										
317.	agent2	login5	2012-07-31 06:55:27	2012-07-31 06:55:28	inprogress	Log																																																										
316.	agent5Otomat	loginwhile	2012-07-31 06:54:32	2012-07-31 06:54:32	success	Log																																																										
315.	agent5Otomat	loginwhile	2012-07-31 06:52:32	2012-07-31 06:52:32	success	Log																																																										
314.	agent2	login4	2012-07-31 01:26:48	2012-07-31 06:51:34	Idle	Log																																																										
313.	agent2	login4	2012-07-31 01:25:21	2012-07-31 06:51:34	Idle	Log																																																										
312.	agent2	login4	2012-07-31 01:23:37	2012-07-31 06:51:34	Idle	Log																																																										

Gambar 4.13 Daftar Hasil Test dari Order (20 Terakhir)

Untuk melihat detail eksekusi setiap task, dapat dilihat dengan mengklik link Log, dengan hasil seperti pada gambar 4.14.

Menu:	Log of Test																												
Manual	Test ID : 318																												
Language	Agent : agent2																												
Scenario	Scenario : login4																												
Agent																													
Order																													
Execute																													
Result																													
	<table border="1"> <thead> <tr> <th>ID</th> <th>Time</th> <th>Content</th> <th>Note</th> </tr> </thead> <tbody> <tr> <td>42228361.</td> <td>2012-07-31 06:57:59</td> <td>url.request:"http://centeris.unpar.ac.id/sample/"; "-"</td> <td>OK</td> </tr> <tr> <td>42228747.</td> <td>2012-07-31 06:57:59</td> <td>link.clickHref:"Login.php"; "-"</td> <td>OK</td> </tr> <tr> <td>42229394.</td> <td>2012-07-31 06:58:00</td> <td>assert.content:"ba"; "-" [true]</td> <td>OK</td> </tr> <tr> <td>42229490.</td> <td>2012-07-31 06:58:00</td> <td>scenario.exec:"login5"; ""</td> <td>OK</td> </tr> <tr> <td>42230193.</td> <td>2012-07-31 06:58:01</td> <td>var.setval:"java"; "3"</td> <td>OK</td> </tr> <tr> <td>42230319.</td> <td>2012-07-31 06:58:01</td> <td>while.notequal:"java"; "5" [true]</td> <td>OK</td> </tr> </tbody> </table>	ID	Time	Content	Note	42228361.	2012-07-31 06:57:59	url.request:"http://centeris.unpar.ac.id/sample/"; "-"	OK	42228747.	2012-07-31 06:57:59	link.clickHref:"Login.php"; "-"	OK	42229394.	2012-07-31 06:58:00	assert.content:"ba"; "-" [true]	OK	42229490.	2012-07-31 06:58:00	scenario.exec:"login5"; ""	OK	42230193.	2012-07-31 06:58:01	var.setval:"java"; "3"	OK	42230319.	2012-07-31 06:58:01	while.notequal:"java"; "5" [true]	OK
ID	Time	Content	Note																										
42228361.	2012-07-31 06:57:59	url.request:"http://centeris.unpar.ac.id/sample/"; "-"	OK																										
42228747.	2012-07-31 06:57:59	link.clickHref:"Login.php"; "-"	OK																										
42229394.	2012-07-31 06:58:00	assert.content:"ba"; "-" [true]	OK																										
42229490.	2012-07-31 06:58:00	scenario.exec:"login5"; ""	OK																										
42230193.	2012-07-31 06:58:01	var.setval:"java"; "3"	OK																										
42230319.	2012-07-31 06:58:01	while.notequal:"java"; "5" [true]	OK																										

Gambar 4.14 Contoh Hasil Log Test No. 318

4.4.3. Rancangan Algoritma Test Agent (TA)

TA bertugas untuk menerjemahkan bahasa di dalam setiap task pada suatu scenario serta mengeksekusinya sesuai dengan semantiknya. Dalam eksekusinya menggunakan aturan dan algoritma sebagai berikut:

1. Setiap Test Agent memiliki identitas AgenX, diambil dari parameter (param1) saat eksekusi. Sehingga command eksekusi menjadi: `java -jar <namajar> param1`.
2. Pada saat dieksekusi, AgenX akan memeriksa tabel Order yang Oder.agent=AgenX, dan Order.status="start".
3. Jika ada, maka AgenX akan mengupdate table Order.status="inprogress", kemudian akan menyimpan nama scenario pertama pada variabel AgenXScenario, kemudian mengambil semua task yang bersesuaian dengan AgenXScenario dan menyimpannya ke array AgenXTaskArray; jika tidak ada maka AgenX akan keluar.
4. Setelah mendapat daftar task, maka AgenX akan mengeksekusi task mulai yang pertama sampai terakhir dengan menggunakan method AgenX.execute(scenario, object, action, param1, param2) . Sebelum eksekusi, AgenX akan menginsert table Test dengan Test.status="inprogress" dan Test.starttime dan mencatat Test.id ke variabel AgenXIDTest.
5. Jika object.action = "scenario.report" maka AgenX akan mengupdate table Test.status "success" atau "fail" sesuai dengan AgenXIDTest. Kemudian akan menset AgenXScenario="".
6. Jika object.action = "scenario.exec" maka tuliskan ke log terlebih dahulu, kemudian secara otomatis akan menjalankan "scenario.report" jika AgenXScenario <> "" dengan Test.status="done", baru menjalankan scenario baru.
7. Jika object = "assert": jika kondisi terpenuhi (true) maka eksekusi semua task selanjutnya. Jika kondisinya tidak terpenuhi (false) maka lewati semua task selanjutnya sampai ketemua task assert selanjutnya. Pada saat melewati suatu task tuliskan ke table Log dengan note="skipped".
8. Jika object="while" dan action <> "end": jika kondisi terpenuhi (true) maka catat dulu nomor urut task tersebut (index) pada AgenXTaskArray ke variable AgenXIDTask. Kemudian eksekusi semua task dibawahnya.
Jika kondisi tidak terpenuhi (false) maka set variable AgenXIDTask="", kemudian skip semua task dibawahnya sampai object.action = "while.end".
Jika ketemu task dengan object.action="while.end" dan AgenXIDTask<>"" maka eksekusi kembali ke task AgenXTaskArray[AgenXIDTask], else lanjutkan eksekusi task selanjutnya.
9. Untuk setiap eksekusi task, AgenX juga menginsert ke table Log dengan Log.content=object.action:param1:param2
10. Jika semua task di suatu skenario telah selesai dieksekusi, maka secara otomatis menjalankan "scenario.report" dengan Test.status="done", kemudian AgenX akan mengupdate tabel Order dengan Order.status="idle", kemudian mengecek kembali apakah ada order yang berstatus "start" untuk dirinya. Jika tidak ada lagi, baru keluar.

11. Jika dalam mengeksekusi suatu task tidak ada eksepsi (semua log = ok) maka pada saat suatu skenario selesai dieksekusi akan mengupdate table Test.state="success". Jika ada eksepsi maka Test.state="fail".

Untuk eksekusi konstruksi bahasa, Test Agent menggunakan object dan method yang ada pada library HTMLUnit seperti pada bagian 3.3.

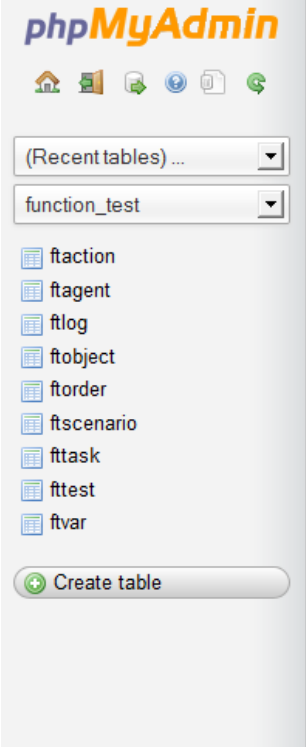
Demikianlah hasil analisis dan rancangan dari perangkat lunak uji otomatis uji fungsional web enterprise.

BAB 5 Hasil Implementasi dan Pengujian

Berikut adalah hasil implementasi basis data, program dan pengujian yang telah dilakukan.

5.1. Implementasi Basis Data

Dari rangkangan pada bab 4, telah diimplementasikan pada lingkungan MySQL versi 5.1, dan telah digunakan untuk melakukan uji coba dengan visualisasi seperti pada gambar 5.1.



centeris » [function_test](#)

Table	Rows	Type	Size	Comments
ftaction	39	MyISAM	5.4 KiB	Creation: Jul 30, 2012 at 06:17 PM
				Last update: Jul 31, 2012 at 11:29 PM
ftagent	6	InnoDB	32 KiB	Creation: Jul 30, 2012 at 04:07 AM
ftlog	206,379,354	InnoDB	30.6 GiB	Creation: Jul 30, 2012 at 04:07 AM
ftobject	16	MyISAM	3.8 KiB	Creation: Jul 30, 2012 at 03:58 PM
				Last update: Jul 31, 2012 at 06:08 AM
ftorder	14	InnoDB	48 KiB	Creation: Jul 30, 2012 at 04:07 AM
ftscenario	11	InnoDB	32 KiB	Creation: Jul 30, 2012 at 09:48 PM
fttask	80	InnoDB	32 KiB	Creation: Jul 30, 2012 at 04:07 AM
fttest	267	InnoDB	64 KiB	Creation: Jul 30, 2012 at 04:07 AM
ftvar	8	InnoDB	16 KiB	Creation: Jul 30, 2012 at 04:07 AM
9 tables	206,379,795	--	30.6 GiB	

Gambar 5.1 Implementasi Basis Data

Dari gambar 5.1. dapat dilihat bahwa telah dilakukan pengujian sebanyak 267 kali, dengan jumlah log sebanyak 206 juta log dengan ukuran 30,6 Giga Byte.

5.2. Implementasi Web App

Aplikasi Web (Web App) telah diimplementasikan, dan dapat diakses melalui situs http://centeris.unpar.ac.id/function_test/ dan contoh tampilannya dapat dilihat pada gambar 5.2. Semua disain pada bagian 4.4.2 telah diimplementasikan dengan baik.



Gambar 5.2 Contoh Hasil Tampilan Web App

5.3. Implementasi Test Agent (TA)

Aplikasi Test Agent (TA) telah diimplementasikan pada lingkungan J2SE menggunakan lingkungan pengembangan NetBeans 6.8. Visual pengembangannya dapat dilihat pada gambar 5.3 sedangkan hasil eksekusinya dapat dilihat pada gambar 5.4.

Hasil akhir kompilasi aplikasi adalah TA.jar. Eksekusinya memiliki 2 mode, yaitu:

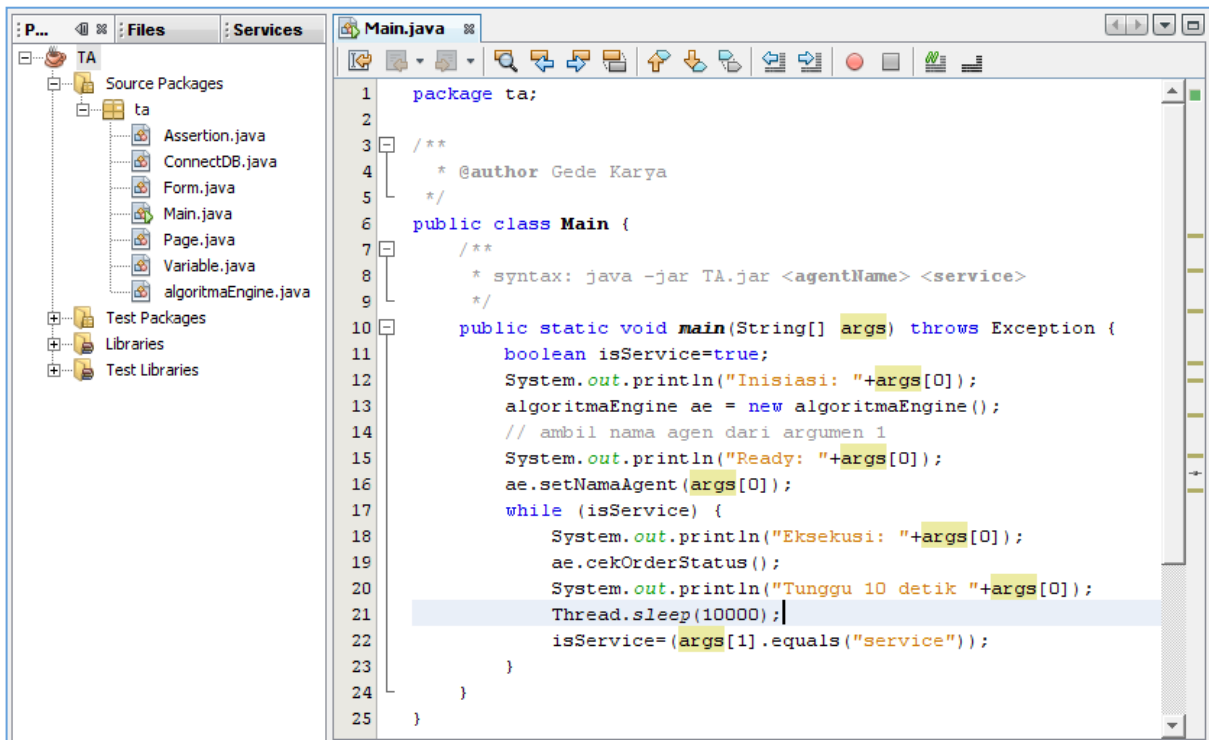
1. Mode service yang akan terus berjalan dengan sintaks:

```
java -jar TA.jar <agentName> service.
```

2. Mode standalone, dimana akan berjalan satu putaran saja, dengan sintaks:

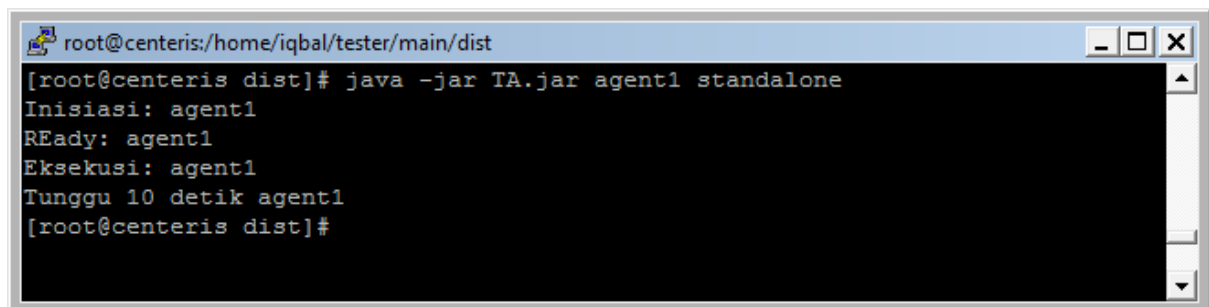
```
java -jar TA.jar <agentName> standalone
```

d

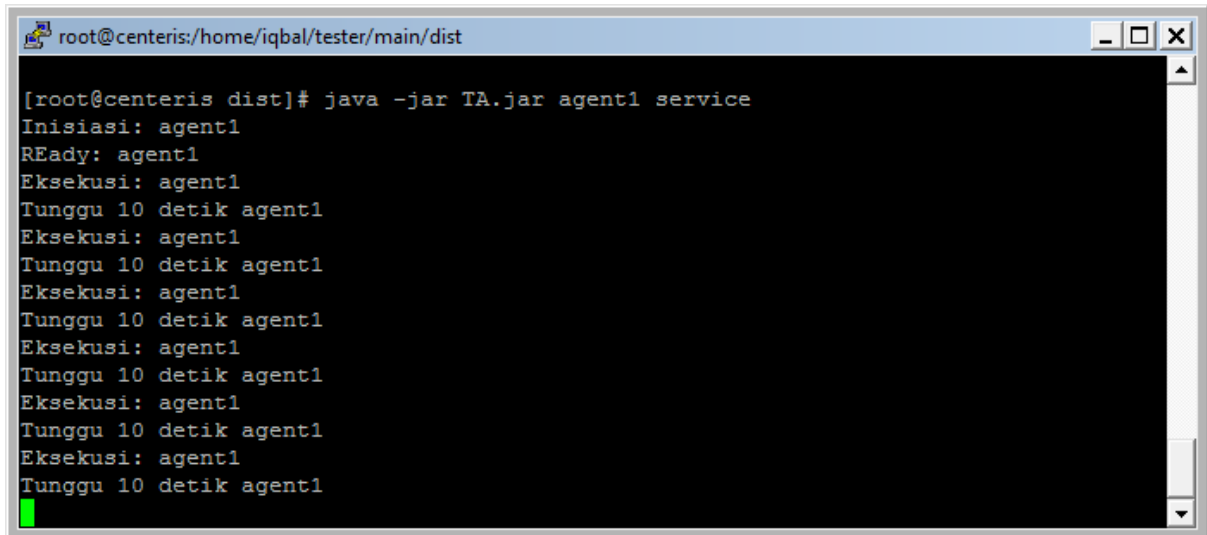


Gambar 5.3 Implementasi Test Agent (TA) pada lingkungan NetBeans 6.8

Pada gambar 5.4 TA dijalankan dengan mode standalone, sedangkan mode service dapat dilihat pada gambar 5.5. Pada mode ini, TA akan berjalan terus untuk mengeksekusi Order yang disiapkan untuknya.



Gambar 5.4 Contoh Hasil Eksekusi TA dengan mode standalone



```
root@centeris:/home/iqbal/tester/main/dist
[root@centeris dist]# java -jar TA.jar agent1 service
Inisiasi: agent1
REady: agent1
Eksekusi: agent1
Tunggu 10 detik agent1
Eksekusi: agent1
Tunggu 10 detik agent1
Eksekusi: agent1
Tunggu 10 detik agent1
Eksekusi: agent1
Tunggu 10 detik agent1
Eksekusi: agent1
Tunggu 10 detik agent1
Eksekusi: agent1
Tunggu 10 detik agent1
Eksekusi: agent1
Tunggu 10 detik agent1
```

Gambar 5.5 Contoh Hasil Eksekusi TA dengan mode service

5.3. Pengujian

Untuk melakukan pengujian, telah dikembangkan sistem aplikasi berbasis web yang merupakan emulator dari Student Portal Unpar. Aplikasi ini dapat diakses pada situs <http://centeris.unpar.ac.id/sample/> dan fitur-fiturnya dijelaskan lebih jauh pada Lampiran 5.

Pengujian telah dilakukan dengan:

1. **Multi agent.** Menggunakan 6 agent, diantaranya 2 agent menggunakan mode service, sedangkan 4 agent lagi menggunakan mode standalone. Penempatan agent juga ada 2, yaitu: pada Test Server sebanyak 2 agen otomatis, dan 4 agent pada PC Desktop.
2. **Multi scenario.** Skenario yang diujikan ada sebanyak 11 skenario, baik yang sederhana, condition, while loop, dan eksekusi skenario lain. Dari 16 skenario tersebut melibatkan 80 task (baris instruksi).
3. **Multi test.** Telah dilakukan pengujian sebanyak lebih dari 250 kali test, dengan log sebanyak lebih dari 200 juta baris.

Pada gambar 5.6 dapat dilihat contoh hasil pengujian sederhana skenario Login.

List of Task login

Instruksi: Anda dapat menambah, mengubah atau menghapus Task pada halaman ini.

No	Id	Idx	Object	Action	Param1	Param2	Do
1.	1.	1	url	request	"http://centeris.unpar.ac.id/sample/"		Edit Delete
2.	2.	2	link	clickHref	"Login.php"		Edit Delete
3.	3.	3	form	first	-	-	Edit Delete
4.	4.	4	text	fill	"username"	"7308001"	Edit Delete
5.	5.	5	text	fill	"pass"	"satu"	Edit Delete
6.	88.	88	submit	click	"login"		Edit Delete
7.	89.	89	assert	content	"Rencana Studi"		Edit Delete
8.	90.	90	scenario	report	success		Edit Delete

(a) Skenario Login

List of Test State (20 terakhir)

Instruksi: klik link log pada masing-masing test untuk melihat hasil detailnya!

Id	Agent	Scenario	StartTime	EndTime	State	Log
322.	agent1	login	2012-08-22 04:27:06	2012-08-22 04:27:07	success	Log
321.	agent1	login	2012-08-22 04:23:21	2012-08-22 04:23:22	success	Log

(b) Status Eksekusi Skenario Login oleh agent1

Log of Test

Test ID : 321
 Agent : agent1
 Scenario : login

Id	Time	Content	Note
206379296.	2012-08-22 04:23:22	url.request:"http://centeris.unpar.ac.id/sample/"	OK
206379297.	2012-08-22 04:23:22	link.clickHref:"Login.php"	OK
206379298.	2012-08-22 04:23:22	form.first:"-",""	OK
206379299.	2012-08-22 04:23:22	text.fill:"username";"7308001"	OK
206379300.	2012-08-22 04:23:22	text.fill:"pass";"satu"	OK
206379301.	2012-08-22 04:23:22	submit.click:"login"	OK
206379302.	2012-08-22 04:23:22	assert.content:"Rencana Studi";" [true]	OK
206379303.	2012-08-22 04:23:22	scenario.report:"success";"	OK
206379304.	2012-08-22 04:23:22	scenario.report:`done`	OK

(c) Log Detail Hasil Test Login oleh agent1

Gambar 5.6 Contoh Hasil Pengujian Skenario Login oleh Agent1

Dengan demikian semua implementasi sudah dilakukan dan berhasil dengan baik. Dari banyaknya pengujian juga dapat disimpulkan bahwa aplikasi sudah dapat berjalan dengan stabil.

Dari model pengujian yang dilakukan juga terdapat temuan informasi bahwa sistem ini dapat digunakan sebagai robot yang mewakili user untuk mengoperasikan aplikasi berbasis web.

BAB 6 KESIMPULAN DAN POTENSI PENGEMBANGAN

Pada bagian ini dijelaskan tentang kesimpulan yang diambil berdasarkan metodologi, kajian pustaka, dan hasil-hasil penelitian. Bagian ini diakhiri dengan potensi pengembangan yang dapat dilakukan untuk penelitian selanjutnya.

6.1. Kesimpulan

Berdasarkan hasil penelitian di atas, dapat disimpulkan:

1. Perangkat lunak uji yang dikembangkan sudah dapat berfungsi dengan baik, mencakup fitur: (a) pemasukan skenario/ task dengan variasi statement sederhana, kondisional dan pengulangan serta eksekusi skenario lain, (b) pengelolaan agent, (c) pemesanan eksekusi suatu skenario oleh sebuah agent, (c) pelaporah hasil, baik status penyelesaian maupun log detail dari eksekusi. Di dalamnya sudah termasuk otomasi dan multi agen.
2. Berdasarkan hasil pengujian, bawa perangkat lunak sudah cukup stabil. Disamping itu juga dimungkinkan untuk menggunakan perangkat lunak tersebut sebagai robot untuk melakukan sejumlah aksi browsing dengan aksi yang cukup kompleks.

6.2. Potensi Pengembangan

Dari hasil penelitian yang telah dilakukan, ada beberapa hal yang perlu dikembangkan yaitu:

1. Melengkapi konstruksi bahasa khususnya yang berkaitan dengan waktu, diantaranya: delay dari satu task ke task yang lain, order yang terjadwal, fitur pengiriman laporan/ hasil via email atau channel komunikasi lain kepada Tester. Dalam hal ini dapat dikembangkan lebih lanjut menjadi robot yang menjalankan sebagian atau seluruh operasi aplikasi berbasis web. Hal ini dapat dilakukan dengan batasan tidak ada capcha dalam situs tersebut.
2. Mengembangkan suatu lingkungan pengujian berbasis web yang terintegrasi sehingga memudahkan proses audit sistem informasi berbasis web, baik fungsional maupun non fungsional.

DAFTAR REFERENSI

- [1] Roger S. Presman, *Software Engineering A Practioner's Approach*, fitfh editon, McGraw-Hill, 2001
- [2] Gede Karya, *Perangkat Lunak Uji Performansi Dan Kapasitas Situs Web Terotomasi Multi Agen, Laporan Penelitian*, Lembaga Penelitian dan Pengabdian Kepada Masyarakat (LPPM – Unpar), Juni 2011.
- [3] Gede Karya, *Pembangkitan Skenario Dan Data Uji Performansi Dan Kapasitas Situs Web Terotomasi Multi Agen Dengan Metode Back Propagation, Laporan Penelitian*, Lembaga Penelitian dan Pengabdian Kepada Masyarakat (LPPM – Unpar), Desember 2011.
- [4] David G. Messerschmitt, *Understanding Networked Applications: A First Course*, Morgan Kaufmann, 2000
- [5] George Coulouris, Jean Dollimore, Tim Kindberg, *Distributed Systems Concepts and Design*, Addison Wesley, 2001.
- [6] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2009.
- [7] Bruce Eckel, *Thinking in Java*, Second Edtion, Printice-Hall, 2000.
- [8] Chaffee, Alexander Day, *Building a Robust Multithreaded Server in Java*, Jguru Java Training, 1998
- [9] Anonim, *Apache JMeter*,
<http://www.methodsandtools.com/tools/tools.php?jmeter>, diakses terakhir tanggal: 12 Juni 2012.
- [10] Anonim, *Extending Jmeter*, <http://www.jajakarta.org/jmeter/1.7/en/extending/index.html>, diakses terakhir tanggal 13 Juni 2012.
- [11] Anonim, *HTTPUnit Home*, <http://httpunit.sourceforge.net/index.html>, diakses terakhir tanggal 21 Juli 2012.
- [12] Anonim, *HTTPUnit Cook Book*,
<http://httpunit.sourceforge.net/doc/cookbook.html>, diakses terakhir tanggal 27 Juli 2012.
- [13] Anonim, *HTMLUnit*, <http://htmlunit.sourceforge.net> , diakses terakhir tanggal 21 Juli 2012.
- [14] Anonim, *HtmlUnit: A Quick Introduction*, <http://java.dzone.com/articles/htmlunit-%E2%80%93-quick-introduction>, diakses terakhir tanggal 22 Juli 2012.