PEMBANGKITAN SKENARIO DAN DATA UJI PERFORMANSI DAN KAPASITAS SITUS WEB TEROTOMASI MULTI AGEN DENGAN METODE BACK PROPAGATION

Gede Karya
Jurusan Teknik Informatika, Fakultas Teknologi Informasi dan Sains, Universitas Katolik Parahyangan, Jalan Ciumbuleuit No. 94, Bandung, 40141
Telp : (022) 2041964, Fax : (022) 2042131
E-mail : gkarya@unpar.ac.id

Abstrak

Untuk memastikan qualitas suatu perangkat lunak aplikasi web, diperlukan uji kualitas. Agar layak operasi, pengujian aspek fungsional saja tidak cukup, tapi juga aspek non fungsional, khususnya aspek performansi, kapasitas dan reliabilitas. Pada pengujian non fungsional, selain memerlukan alat uji, juga memerlukan skenario/data/trafik ekstrim/peak yang mewakili kondisi sesungguhnya. Untuk itu, pada penelitian ini kami mengembangkan teknik untuk membangkitkan skenario dan data uji performansi aplikasi web menggunakan metode back propagation. Metode ini dapat diimplementasikan pada Apache Web Server menggunakan mod_security dengan mode audit, yang memungkinkan mengekstrak jejak request HTTP dari user secara lengkap. Hasil pengujian menunjukkan bahwa metode ini cukup efektif diterapkan.

Kata kunci: *pembangkitan skenario/data uji performansi web, metode back propagation, mod_security*

Abstract

To ensure the quality of a web application software, required test quality. For proper operation, testing the functional aspect alone is not enough, but also non-functional aspects, in particular aspects of performance, capacity and reliability. In the non-functional testing, in addition to requiring test equipment, also require the scenario/data/extreme traffic/peak representing the actual conditions. Therefore, in this study we developed a technique to generate scenarios and test data web application performance using back propagation method. This method can be implemented using the Apache Web Server with mod_security audit mode, which allows to extract traces of user HTTP requests in full. The results show that this method is effectively applied.

1. PENDAHULUAN

Salah satu hal terpenting dalam sebuah produk perangkat lunak adalah kualitas. Kualitas didefinisikan sebagai kemampuan suatu produk untuk memenuhi/ memuaskan kebutuhan penggunanya. Kebutuhan perangkat lunak dapat dibagi menjadi 2 [1], yaitu: kebutuhan fungsional dan kebutuhan non fungsional.

Bagaimana cara memastikan bahwa suatu perangkat lunak berkualitas? Dengan melakukan uji kualitas (pengujian). Pada pengujian fungsional, dapat dilakukan uji terima yaitu dengan mengecek semua fungsi yang disyaratkan apakah sudah dapat berjalan/ diakomodasi oleh perangkat lunak. 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.

Pada pengujian non-fungsional, khususnya sisi *performance* dan *load*, Minesce [2] telah mengusulkan model uji beban web site menggunakan konsep emulasi kondisi *peak season* dengan mempertimbangkan prilaku user. Dalam model tersebut, skenario uji dimasukkan sesuai dengan kasus-kasus uji fungsional, kemudian diduplikasi menjadi sejumlah skenario untuk mewakili kondisi peak. Model yang diusulkan oleh Minesce ini banyak diadopsi oleh kalangan industri untuk membuat produk *propietary*-nya sendiri. Seng [3] menyatakan bahwa model benchmark di dunia industri yang digunakan untuk mengukur kinerja (*performance*) aplikasi seperti TREC, TPC, SPEC, SAP, Oracle, Microsoft, IBM, Wisconsin, AS^{3}AP, OO1, OO7, XOO7 terlalu spesifik untuk sistem tertentu saja (propietary), sehingga tidak mudah diterapkan pada aplikasi berbasis web lainnya. Untuk itu, dikembangkan model "*generic construct based*

workload" yang diturunkan dari spesifikasi kebutuhan, kemudian ditranslasi menjadi 3 model, yaitu: *page model, query model dan control model*. Dalam melakukan pengujian aplikasi web, Sampath [4] meneliti bagaimana meningkatkan efektifitas pengujian aplikasi web yang menggunakan *user-session-based*. Pada penelitian tersebut disimpulkan bahwa pengurutan dapat mengurangi test suite sehingga meningkatkan efektifitas pengujian aplikasi web yang memiliki karakteristik tersebut.

Pada penelitian [5], kami telah mengembangkan *tools* untuk mensimulasikan kondisi *peak* dan emulasi dari pihak-pihak yang melakukan transaksi menggunakan model multi agen terdistribusi. Tools ini menerapkan konsep dari model yang diusulkan oleh Minesce [2].

Baik pada Minesce [2], Seng[3] maupun Sampath [4] serta pada penelitian [5], skenario uji dibangkitkan dengan memasukkan skenario uji fungsional yang diduplikasi. Hal ini menyebabkan skenario sangat teratur, padahal realitanya user tidak selalu berprilaku teratur seperti skenario tersebut. Sehingga skenario hasil kurang mewakili kondisi sesungguhnya. Oleh karena itu, pada penelitian ini pembangkitan skenario didasarkan pada jejak transaksi user pada server yang dipropagasi balik (*back propagation*) ke tools uji untuk selanjutnya dimanipulasi menjadi skenario uji performansi/ load.

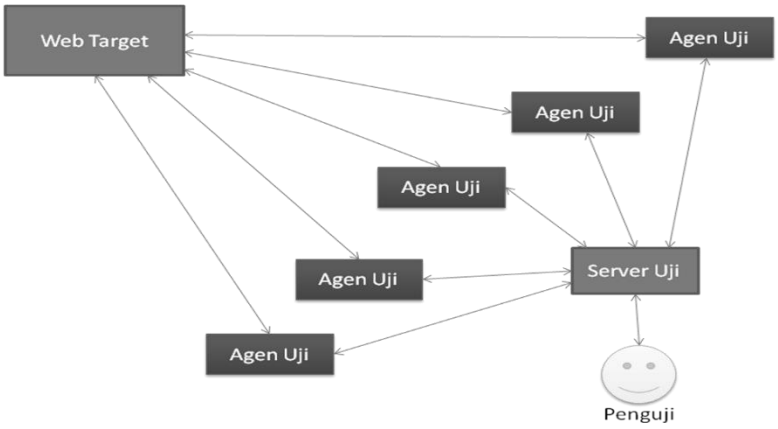
Penelitian ini dilaksanakan dengan metodologi sebagai berikut: (1) Review tentang hasil penelitian [5] untuk melihat model dan perangkat lunak uji kapasitas dan performansi aplikasi web terotomasi multi agen. Hal ini penting mengingat konstruksi selanjutnya didasarkan atas model dan perangkat lunak ini. (2) Studi pustaka dan eksplorasi tentang teknik ekstraksi log transaksi HTTP pada web server, dalam hal ini dipilih Apache Web Server, dengan pertimbangan keluasan pemakaiannya. (3) Analisis dan disain, dengan mengusulkan model pembangkitan skenario uji berdasarkan model pada penelitian [5] dan menerapkan teknik hasil langkah (2), kemudian dibuat disain aplikasi yang mengimplementasikan model tersebut. (4) Implementasi dan pengujian, dengan mengimplementasikan disain aplikasi menjadi perangkat lunak, khususnya fokus pada ekstraksi log dan modifikasi terhadap agen dan koordinator testing. Pengujian difokuskan untuk mengecek apakah proses ekstraksi log berjalan efektif untuk menghasilkan skenario sesuai model. Atas dasar langkah-langkah di atas, ditarik kesimpulan tentang penelitian ini secara keseluruhan. Selanjutnya hasil dari langkah (1) dan (2) diuraikan pada bagian 2, sedangkan langkah (3) dan (4) diuraikan pada bagian 3.

2. TINJAUAN PUSTAKA

Pada bagian ini diuraikan tinjauan hasil penelitian [5] berupa *tools* uji kapasitas web menggunakan pendekatan multi agen terdistribusi. Selain itu juga dibahas tentang bagaimana membaca jejak transaksi user dari log server web, khususnya pada Apache Web Server.

2.1 Perangkat Lunak Uji Kapasitas dan Performansi Aplikasi Web Multi Agen

Aplikasi hasil penelitian [5] menerapkan model seperti pada Gambar 1.

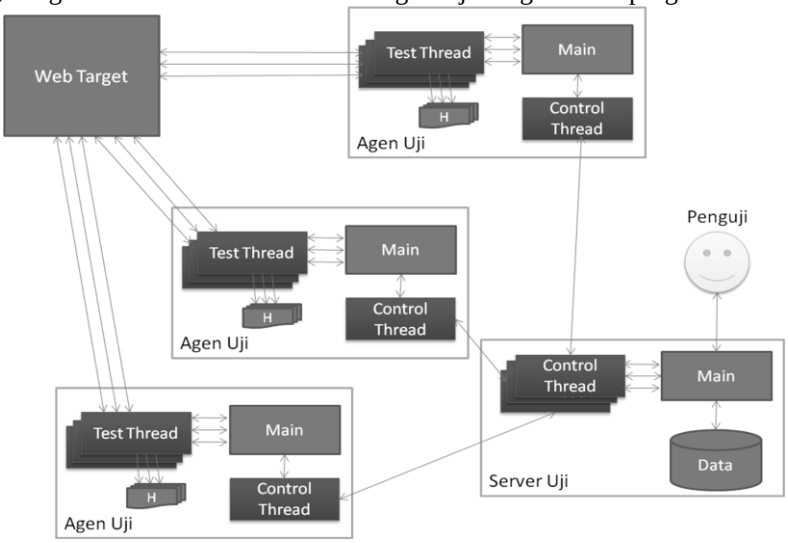


Gambar 1. Model Logika Aplikasi Uji Kapasitas dan Performansi Multi Agen

Aplikasi ini dikembangkan dengan teknologi *multi threading* [6] yang diimplementasikan pada lingkungan *Java 2 Standar Edition* (J2SE) [7]. Disain aplikasinya disajikan pada Gambar 2.

Pada Gambar 2 dapat dilihat bahwa ada 2 aplikasi *multi threading* yang dikembangkan, yaitu: (1) Agen Uji yang berperan sebagai emulator dari banyak *user agent* (pengguna) dan (2) Server Uji, yang menjadi *tools* untuk memberikan perintah penujian oleh Penguji kepada Agen Uji. Server Uji dan Agen Uji

dikembangkan dengan template pengembangan server yang diambil dari [8]. Hasilnya berupa dua program yaitu Server Uji, dengan nama WebTestServer dan Agen Uji dengan nama program WebTest.



Gambar 2. Disain Aplikasi Uji Kapasitas dan Performansi Multi Agen

2.2 Teknik Ekstraksi Log HTTP pada Apache Web Server

Format *request* dan *response* dalam HTTP didefinisian secara jelas pada RFC 2616 HTTP [9]. Untuk mendapatkan data *request* dan *response* tersebut, dapat dilakukan dengan 2 cara, yaitu: (1) dengan merekam semua paket data yang lewat dari *user agent* (browser) ke *web server* dan sebaliknya. (2) dengan mengesktrak log/ jejak di *web server*. Pada penelitian ini digunakan teknik (2), khususnya pada *Apache Web Server*.

Pengelolaan log pada *Apache Web Server* dilakukan melalui modul **mod_log** [10]. Informasi yang di-log hanya sebatas *headear* dari *request*. Itupun tidak lengkap, terutama terkait dengan kebutuhan untuk memperoleh informasi *cookies* dan *body* jika methodnya POST. Untuk mendapatkan informasi yang lengkap dapat menggunakan modul pihak ketiga yang dikembangkan oleh TrustWave Spiders Labs [11]. Modul yang dapat diterapkan adalah **mod_security** dengan mode audit. Dengan menggunakan modul ini, akan dihasilkan file log khusus dengan format [12] seperti contoh berikut:

```
--ae720000-A--
[23/Nov/2011:15:40:43 +0700] Tsyxi38AAAEAABCsfbIAAACU 127.0.0.1 49353
127.0.0.1 80
--ae720000-B--
POST /vonis/viewevent.php?i=8 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:7.0.1) Gecko/20100101
Firefox/7.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
Referer: http://localhost/vonis/viewevent.php?i=8
Cookie: first=0; HstCfa1686645=1319160505890; HstCla1686645=1321347362945;
HstCmu1686645=1319160505890; HstPn1686645=1; HstPt1686645=12; HstCnv1686645=8;
HstCns1686645=9; PHPSESSID=kundir5hu2l1dqqi1jvp2f0kt5
Content-Type: application/x-www-form-urlencoded
Content-Length: 37

--ae720000-C--
nama=dfdg&email=qweqw&komentar=dfgdfg
--ae720000-Z--
```

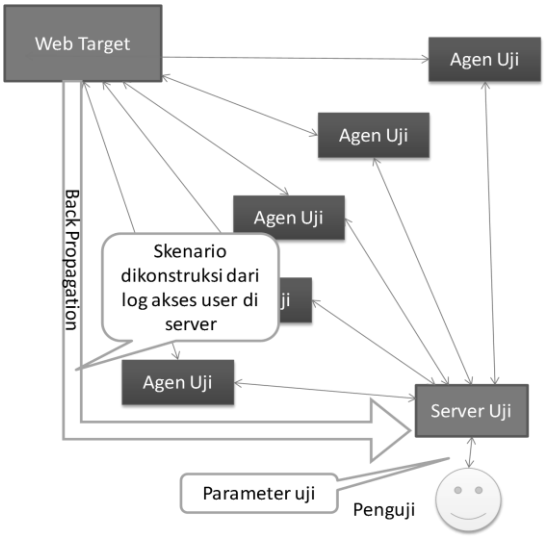
Setiap log terdiri atas sparator seksi yang diawali dan diakhiri dengan dengan "- -" . Arti dari setiap seksi pada log audit, adalah: (A) *audit log header*, (B) *request headers*, (C) *request body*, (D) *intended response headers*, (E) *intended response body*, (F) *response headers*, (G) *response body*, (H) *audit log trailer*, (I) *reduced multipart request body*, (J) *multipart files information* (K) *matched rules information*, (Z) *audit log footer*. Dengan demikian, dalam kasus *request* yang diperlukan adalah *header* dari seksi B dan *body* dari seksi C.

3. PEMBANGKITAN SKENARIO DAN DATA UJI DENGAN METODE *BACK PROPAGATION*

Pada bagian ini diuraikan model usulan dan disain aplikasi sistem pembangkitan skenario dan data uji menggunakan metode *back propagatian*, menggunakan teknik yang dibahas pada bagian 2.

3.1 Model Usulan

Model pada Gambar 1, dapat diperluas dengan mem-propagasi-kan balik (*back propagation*) data transaksi yang terekam pada Web Target sebagai bahan untuk mengkonstruksi skenario uji. Lebih detail dapat dilihat pada Gambar 3.



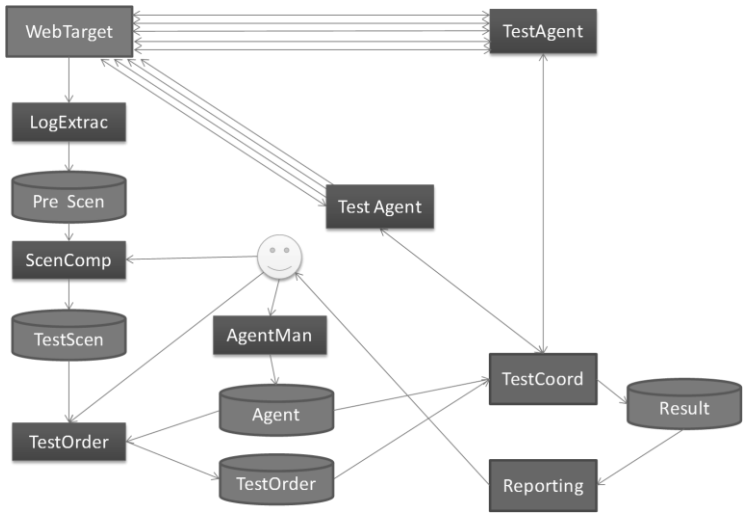
Gambar 3. Model Usulan Pembangkitan Skenario Uji

Pembacaan log pada Web Target, dapat menggunakan teknik (2) yang diuraikan pada bagian 2.2. Dengan demikian diperoleh semua *request* yang dilakukan oleh *user* pada rentang waktu tertentu. *Request* berupa *header* dan *body* tersebut, selanjutnya dimanipulasi untuk membangkitkan skenario, dengan menambahkan parameter uji yang merepresentasikan banyaknya agen uji (*multi user agent*), dan variansi data (*heterogeneous data*).

Skenario hasil manipulasi disampaikan ke Server Uji, untuk selanjutnya dikomposisi sebagai instruksi ke masing-masing Agen Uji. Setiap Agen Uji akan mengkomposisi *request* dalam bentuk pesan HTTP seperti dengan arsitektur aplikasi pada bagian 2.1. Model ini sesungguhnya mem-propagasi-kan semua transaksi yang dilakukan oleh user agent dari Web Target, untuk dimanipulasi terlebih dahulu, kemudian dijadikan transaksi kembali oleh Agen Uji.

3.2 Disain Aplikasi

Berdasarkan prinsip model usulan pada Gambar 3, maka dapat dibuat aplikasi dengan disain seperti pada Gambar 4. Pada Gambar 4 dapat dilihat bahwa ada beberapa modul aplikasi yang diperlukan, yaitu: (1) LogExtrac, berfungsi untuk membaca log audit di web terget untuk menghasilkan data PreScen. Data PreScen merupakan daftar request yang dilakukan oleh user ke WebTarget. Data ini merupakan sumber untuk membuat skenario uji. (2) ScenComp berfungsi untuk memfasilitasi penguji untuk mengkomposisi skenario uji berdasarkan data sumber PreScen. Dengan demikian konstruksi skenario benar-benar berdasarkan request user yang sesungguhnya. Dengan menggunakan aplikasi ini juga penguji dapat memasukkan varian dengan menentukan berapa varian yang diperlukan, *key* mana saja yang *value*-nya bervariasi. Semua skenario final disimpan ke data TestScen (skenario uji coba). (3) AgentMan berfungsi untuk mengelola data TestAgent dan menyimpan datanya di basis data Agent. (4) TestOrder berfungsi untuk membuat pesanan eksekusi uji coba ke TestAgent. Pesanan dibuat berdasarkan skenario di TestScen dan data Agent. (5) TestCoord berfungsi untuk berkomunikasi dengan semua TestAgent. Dalam komunikasi ini dilakukan proses otentifikasi dan pemberian perintah uji berdasarkan TestOrder dan menerima laporan hasil uji coba dari TestAgent. Semua hasil disimpan pada data Result. (6) TestAgent berfungsi untuk melakukan uji beban berdasarkan perintah dari TestCoord. (7) Reporting berfungsi untuk menampilkan laporan hasil uji berdasarkan hasil uji (Result) dan hal-hal yang perlu ditampilkan berkaitan dengan uji beban.



Gambar 4. Disain Aplikasi Pembangkitan Skenario Uji

3.3 Implementasi dan Pengujian

Pada penelitian ini, ada 3 modul aplikasi yang diimplementasikan menggunakan J2SE, yaitu: (1) LogExtractor sebagai hasil implementasi dari aplikasi LogExtrac. (2) UserAgent sebagai hasil implementasi dari TestAgent, dan (3) UserCoordinator sebagai hasil implementasi dari TestCoord. Sedangkan ScenCom, AgenMan dan TestOrder menggunakan akses queri langsung ke basis data.

Pengujian dilakukan dengan skenario sebagai berikut: (1) WebTarget menggunakan dilengkapi dengan **mod_security** (mod_security2.so) dengan mode audit. Setelah itu diakses oleh user. Semua jejak log akan tersimpan pada file audit.log pada direktori logs. (2) LogExtractor dijalankan untuk membaca audit.log menghasilkan rekam jejak dari user dan disimpan pada basis data. (3) Komposisi skenario dilakukan dengan query langsung ke basis data. (4) Test Order dilakukan dengan query langsung ke basis data. (5) UserCoordinator dijalankan dan menunggu koneksi dari UserAgent. (6) UserAgent beberapa *instance* dijalankan. UserAgent mengakses UserCoordinator untuk mendapatkan pesan uji dan mengeksekusi kemudian melaporkannya. (7) Hasil uji dapat dimonitor pada tampilan UserAgent dan basis data.

Berikut adalah potongan dari file audit.log hasil dari penggunaan mod_security pada Apache Web Server.

```
--7d760000-A--
[23/Dec/2011:15:01:57 +0700] TvQ1dcCoAmQAABQwEUYAAADt 127.0.0.1 54386
127.0.0.1 80
--7d760000-B--
GET /administrator/index.php HTTP/1.1
...
Host: localhost
Connection: Keep-Alive
Cache-Control: no-cache
...
--7d760000-Z—

--035d0000-A--
[23/Dec/2011:15:01:58 +0700] TvQ1dMCoAmQAABQwEUUAAADt 127.0.0.1 54386
127.0.0.1 80
--035d0000-B--
POST /administrator/index.php HTTP/1.1
...
Referer: http://localhost/administrator/
...
--035d0000-C--
username=admin&passwd=admin123&lang=&option=com_login&task=login
--035d0000-Z—
```

Dari audit.log di atas LogExtractor menghasilkan tabel 1 Request dan tabel 2 Params pada basis data Pre_Scen.

Tabel 1. Tabel Request

IdRequest	Methode	ReqLine
1	GET	/administrator/index.php
2	POST	/administrator/index.php

Tabel 2. Tabel Params

IdParams	IdRequest	Type	Key	Value
1	2	param	username	Admin
2	2	param	passwd	admin123
3	2	param	lang	
4	2	param	option	com_login
5	2	param	task	Login

Berdasarkan Tabel Request dan Tabel Params tersebut dikonstruksi skenario dengan cara menduplikasi dan memvariasi Key username, passwd nya sehingga membentuk skenario login sebanyak user yang diinginkan pada basis data TestScen. Dari data TestScen tersebut dibuat order untuk menjalankan request pada Tabel 1 ke masing-masing UserAgent yang berjalan banyak. Dengan demikian dapat diperintahkan oleh UserCoordinator untuk melakukan simulasi uji login sebanyak user yang dikehendaki.

4. SIMPULAN DAN SARAN

Berdasarkan hasil penelitian, dapat ditarik beberapa kesimpulan, yaitu: (1) Semua jejak akses user agent pada WebTarget dapat diekstrak dengan mod_security, khususnya mode audit. (2) Semua aksi user dapat direkonstruksi dan dijadikan kasus uji. Hal ini dapat dilakukan dengan membuat kasus uji berdasarkan log yang diekstrak dari WebTarget. Variasi dapat diberikan dengan merekonstruksi pasangan *key=value* untuk setiap user. (3) Dengan demikian model *back propagation* dapat digunakan untuk mengkonstruksi skenario/ data uji untuk uji peformansi dan kapasitas web yang mendekati situasi *peak season* sebenarnya. (4) Metode ini dapat digunakan untuk menutupi kelemahan pada tools yang dikembangkan sebelumnya, khususnya membangkitkan data uji yang heterogen yang mencerminkan kondisi sesungguhnya.

5. UCAPAN TERIMA KASIH

Dalam kesempatan ini penulis mengucapkan terima kasih kepada Lembaga Penelitian dan Pengabdian Masyarakat, Universitas Katolik Parahyangan atas fasilitasi dan pendanaan yang diberikan pada penelitian ini.

6. DAFTAR RUJUKAN

[1] ISO/IEC 25010. 2011. *SquaRE – System and Software Quality Models*",
[2] Minesce, A. D., Load Testing of Web Sites, IEEE - Internet Computing Journal, Juli 2002.
[3] Seng, Jia-Lang; Ko, I-Feng; Lin, Binshan. A Generic Construct Based Workload Model For Web Search, Information Processing & Management, Sep 2009, Vol. 45 Issue 5, p529-554, 26p; DOI: 10.1016/j.ipm.2009.04.004.
[4] Sampath, Sreedevi; Bryce, Renée C. Improving The Effectiveness of Test Suite Reduction For User-Session-Based Testing of Web Applications, Information & Software Technology, Jul2012, Vol. 54 Issue 7, p724-738, 15p; DOI: 10.1016/j.infsof.2012.01.007.
[5] Karya G., 2012. Perangkat Lunak Uji Performansi Dan Kapasitas Situs Web Terotomasi Multi Agen, Konferensi Nasional Sistem Informasi (KNSI) 2012, STMIK STIKOM Bali.
[6] George C., dkk.. 2001. *Distributed Systems Concepts and Design*. Addison Wesley.
[7] Bruce E. 2000. *Thinking in Java*, Second Edtion, Printice-Hall.
[8] Chaffee, Alexander D. 1998. *Building a Robust Multithreaded Server in Java*, Jguru Java Training.
[9] Fielding R., dkk. 1999. *RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1*. The Network Working Group of The Internet Society.
[10] Anonym.*Apache Web Server Documentation: Apache HTTP Server version 2.2 > Log Files*. http://httpd.apache.org/docs/2.2/logs.html, diakses: 6 Agustus 2011.
[11] Anonym.*Mod Security*. http://www.modsecurity.org/, diakses: 6 Agustus 2011.
[12] Anonym.*Mod Security Documentation > Audit Log > Data Format*. http://sourceforge.net/apps/mediawiki/mod-security/index.php?title=Data_Format#Audit_Log, diakses: 6 Agutus 2011.