Pembangkitan Kasus Uji untuk Pengujian Aplikasi Berbasis *Sequence Diagram*

Test Case Generator for testing Sequence Diagram Based Applications

Tafifa Redita Putri, Sri Widowati Ir.,MT , Iman Lukmanul Hakim SMB, MM

[1,2,3]Prodi S1 Teknik Informatika, Fakultas Teknik, Universitas Telkom

[1]**tafifaredita@gmail.com**, [2]**swdswd99@gmail.com**, [3]**imanlhakim@gmail.com**

Abstrak

Pada era sekarang aplikasi berbasis objek banyak dikembangkan dalam pembuatan sebuah aplikasi. Tahap *design* dan implementasi aplikasi tidak menjamin sebuah aplikasi terbebas dari kesalahan, sehingga aplikasi berbasis objek perlu dilakukan pengujian. Salah satu tahap yang penting dalam pengujian adalah pembangkitan kasus uji sebuah aplikasi.

Pembangkitan kasus uji untuk pengujian dapat dilakukan dengan beberapa cara, salah satu cara untuk membangkitkan kasus uji pada aplikasi berbasis objek adalah dengan memanfaatkan UML model. Diagram UML model selain digunakan untuk mengoreksi sebuah perancangan dapat pula digunakan untuk menghasilkan serangkaian kasus uji sehingga kasus uji dapat dihasilkan pada saat tahap *design*. Terdapat delapan UML Model yang dapat digunakan untuk membangkitkan kasus uji.

Sequence diagram merupakan diagram yang menggambarkan interaksi *behavior* sistem yang dapat digunakan untuk membangkitkan kasus uji. Dengan dibangkitkannya kasus uji dengan *sequence diagram* maka kasus uji dapat dihasilkan pada tahap *design,* sehingga dapat digunakan pengujian saat tahap *coding* selesai. Oleh karena itu pada Tugas Akhir dihasilkan sekumpulan kasus uji dengan membuat sebuah *tools* yang dapat membangkitkan kasus uji berdasarkan *sequence diagram*. Kasus uji yang telah dihasilkan oleh *tools* kemudian digunakan untuk menguji aplikasi dengan penerapan *automatic testing* dengan Selenium WebDriver.

Kata Kunci: *software testing,* kasus uji, *sequence diagram, use case diagram,*

Abstract

Nowadays Object Oriented Programming widely used for application developments. The Design and Implementation stage of an application do not ensure that application to be free from errors, so Object oriented Application needs to be tested. One of the important stage in Testing is generating the test cases.

The generation of test case for testing can be done in various ways, one of the way is to use UML Model. UML Model diagram besides used for correcting the design can also be used to generate series of test cases, so the test cases can be generated on Design Stage. There are 8 UML Model that can be used to generate test case.

Sequence diagram is a diagram that describe the System's behavior interaction that can be used to generate test cases. By generating the test case using Sequence diagram, then the test case can be generated on the Design Stage and can be used right after the Code Stage is finished. Therefore in this Final Project, there will be series of test case that generated using a tool that can generate it from sequence diagram. Test case that have been generated by the tool then will be used to test the application using Automatic testing with Selenium WebDriver.

1. Pendahuluan

Pada saat ini aplikasi berbasis objek banyak dikembangkan dalam pembuatan sebuah aplikasi. Aplikasi berbasis objek dirancang dengan menggunakan notasi UML model yang terdiri dari delapan jenis diagram yang dapat digunakan untuk menjelaskan dan mendeskripsikan sebuah aplikasi [1]. Tentunya untuk menjamin bahwa sebuah aplikasi berbasis objek telah terbebas dari *defect* perlu dilakukan pengujian. Pengujian adalah sebuah proses, atau serangkaian proses yang dirancang untuk memastikan bahwa program telah berjalan sesuai dengan *requirement* dan kebutuhan [2]. Pengujian harus dilakukan oleh seorang *developer* pada sebuah aplikasi, sebelum aplikasi tersebut diberikan kepada *user*. Salah satu tahap yang penting dalam pengujian adalah pembangkitan kasus uji sebuah aplikasi.

Pembangkitan kasus uji dapat dilakukan dengan beberapa cara, sejauh ini pembangkitan kasus uji pada sebuah aplikasi dilakukan pada saat tahap *coding* atau tahap pengujian. Namun pembangkitan kasus uji pada tahap tersebut membuat sebuah pengujian tidak dapat langsung dilakukan setelah tahap *coding* selesai. Pembangkitan kasus uji pada aplikasi berbasis objek dapat dilakukan pada saat tahap *design* dengan memanfaatkan UML Model. UML model selain dapat digunakan untuk mengoreksi aplikasi yang akan dibangun juga dapat digunakan untuk membangkitkan kasus uji sehingga dapat dihasilkan pada tahap *design*.

Terdapat delapan diagram UML model yang dapat digunakan untuk memodelkan sebuah perancangan model. Dari kedelapan diagram tersebut yang dapat digunakan untuk membangkitkan kasus uji misalnya *sequence diagram*. *Sequence diagram* dapat digunakan untuk melihat perilaku sistem. Keunggulan dari *sequence diagram* dibandingkan dengan diagram model lain untuk membangkitkan kasus uji adalah *sequence diagram* menggambarkan interaksi *behavior* sistem dan menggambarkan interaksi antar objek dalam sebuah urutan waktu. Pembangkitan kasus uji yang dilakukan dengan *sequence diagram* tentunya akan mempercepat pembuatan sebuah kasus uji. Kasus uji yang biasanya dihasilkan setelah tahap *coding* selesai, kini dapat dihasilkan pada saat tahap *design*. Sehingga kasus uji dapat langsung digunakan untuk menguji aplikasi saat tahap *coding* selesai. Oleh karena itu pada tugas akhir mengangkat judul **"Pembangkitan Kasus Uji untuk Pengujian Aplikasi Berbasis *Sequence Diagram*"** yang bertujuan untuk menghasilkan sekumpulan kasus uji berdasarkan *sequence diagram* yang dapat diperoleh pada saat tahap *design*. Kasus uji yang telah dihasilkan oleh *tools* kemudian diterapkan untuk pengujian aplikasi, pengujian dengan kasus uji yang telah dihasilkan oleh *tools* dilakukan dengan *automatic testing* dengan menggunakan Selenium WebDriver.

2. Penelitian Terkait

Terdapat beberapa penelitian terkait dengan pembangkitan kasus uji dengan *sequence diagram*. Beberapa penelitian tersebut antara lain adalah penelitian dengan judul An Approach to Generate Test Case from Sequence Diagram [2] dan Automatic Test Case Generator using Sequence Diagram [3]. Penelitian tersebut menggunakan diagram uml yaitu *sequence diagram* dan penunjangnya dengan *use case* diagram untuk menghasilkan sebuah kasus uji.

3. Landasan Teori

3.1 Aplikasi Berbasis Objek

Aplikasi berbasis objek adalah sebuah aplikasi yang dibangun dengan pemrograman berbasis objek. Pemrograman berbasis objek adalah pemrograman yang berdasarkan konsep objek, dimana struktur data berisi data. Aplikasi berbasis *sequence diagram* termasuk pada aplikasi berbasis objek.

3.2 UML Diagram [3]

Unified Modeling Language (UML) adalah sebuah metode pemodelan secara visual sebagai sarana untuk merancang atau membuat sebuah perangkat lunak berorientasi objek. UML merupakan sebuah *tool* atau model untuk merancang pengembangan perangkat lunak yang berbasis objek. UML memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam Bahasa pemrograman yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem sebuah perangkat lunak. UML menggunakan *class* dan *operation* sebagai konsep dasarnya, maka lebih cocok untuk penulisan perangkat lunak dalam bahasa berorientasi objek misalnya adalah C++, Java, C# atau VB.NET.

3.3 Sequence Diagram

Sequence diagram menggambarkan interaksi antara objek dalam urutan waktu. Umumnya sebuah *sequence diagram* menangkap *behavior* dari sebuah skenario [4]. Diagram ini menunjukan sejumlah objek dan pesan yang dilewatkan antara objek-objek di dalam sebuah skenario [4]. Terdapat beberapa simbol yang digunakan untuk membuat sebuah *sequence diagram* antara lain [4]:

a. *Object*, Merupakan *instance class* dan dituliskan tersusun secara horisontal.
b. *Found Message*, merupakan suatu pesan yang men-stimulus terjadinya skenario.
c. *Activation bar*, disebut juga dengan *focus of control* dalam setiap *lifeline* menunjukan kapan suatu *instance* aktif dalam interaksi, *activation bar* ini juga berhubungan dengan fungsi dari *instance* yang berada dalam *stack*.

d. *Lifeline*, merupakan jalur hidup suatu *instance* kelas tertentu dan berfungsi untuk mengetahui kapan suatu *instance* hidup dan dihapus, juga untuk melinierkan urutan pemanggilan pesan *instance* yang bersangkutan.
e. *Asynchrounous* & *synchronous message*, yaitu sebuah komunikasi antara peran.
f. *Interaction use*, merupakan referensi atau acuan untuk sebuah interaksi yang ada didalam definisi dari interaksi lain.
g. *Fragment*, merupakan alur dari sebuag sequence diagram. Berikut merupakan contoh dari *sequence diagram*.

3.4 Software Testing

Software testing adalah proses menganalisis item sebuah perangkat lunak untuk mendeteksi perbedaan antara kondisi yang ada dan diperlukan atau bug dan mengevaluasi fitur dari item sebuah perangkat lunak [5, 6]. Secara umum *software testing* bertujuan untuk memvalidasi, memverfikasi, dan menemukan *defect* sebuah perangkat lunak [7]. Memvalidasi adalah menguji apakah perangkat lunak yang dihasilnya sudah benar dan sesuai dengan *requirement* seorang *user* [7]. Memverifikasi adalah menguji apakah proses pembuatan sebuah perangkat lunak sudah sesuai dengan prosedur atau memenuhi spesifikasi teknis [7]. Spesifikasi adalah deskripsi dari suatu fungsi dalam hal nilai *output* terukur diberi sebuah *input* sesuai dengan persyaratan tertentu [7]. *Defect* adalah suatu variansi antara hasil yang diharapkan dan hasil yang aktual, sebuah *defect* dapat ditelusuri kesalaannya mulai dari spesifikasi, desain, atau saat *coding* [7]. Salah satu tahap penting dalam *software testing* adalah pembangkitan *test case* atau kasus uji.

3.5 Kasus Uji [2]

Kasus uji adalah sebuah masukan, kondisi, dan ekspektasi hasil yang digunakan untuk menguji sebuah perangkat lunak atau aplikasi. Dengan menggunakan kasus uji seorang penguji dapat menemukan *defect* atau *bug* pada aplikasi sebelum aplikasi digunakan user. Kasus uji harus berisikan pengujian setiap menu sebuah aplikasi untuk mencegah adanya *defect*. Kasus uji yang baik adalah kasus uji yang terdiri dari beberapa unsur misalnya adalah menemukan kesalahan yang banyak, tidak menyalin kasus uji yang lain, dibuat untuk menemukan *error* atau *defect*, tidak terlalu sederhana atau terlalu kompleks, dan jelas untuk menguji ketika terjadi kesalahan pada aplikasi.

A. XML [5]

eXtensible Markup Language (XML) dikembangkan pada tahun 1996 dan diakui oleh W3C. XML menggunakan elemen yang ditandai dengan *tag* pembuka (diawali dengan ‘<’ dan diakhiri dengan ‘>’), *tag* penutup (diawali dengan ‘</’ diakhiri dengan ‘>’) dan atribut elemen (parameter yang dinyatakan dalam *tag* pembuka misalnya adalah <form name=”isidata”>). XML hanya digunakan untuk menyimpan informasi yang dikemas dengan *tag-tag* XML. Untuk mengirim, menerima, atau menampilkan informasi dari XML dibutuhkan sebuah perangkat lunak

B. XMI [6]

XML Metadata Interchange (XMI) adalah Object Management Group (OMG) standar untuk bertukar informasi metadata melalui Extensible Markup Language (XML). XMI (XML Metadata Interchange) adalah penggunaan yang diusulkan dari Extensible Markup Language (XML) yang dimaksudkan untuk menyediakan cara standar untuk programmer dan pengguna lain untuk bertukar informasi tentang metadata (dasarnya, informasi tentang apa satu set data terdiri dari dan bagaimana itu diselenggarakan) .Secara khusus, XMI dimaksudkan untuk membantu programer menggunakan Unified Modeling Language (UML) dengan bahasa yang berbeda dan alat-alat pengembangan untuk bertukar model data mereka dengan satu sama lain.

3.6 Automation Testing Tools

Automation Testing tools adalah sebuah tools yang dapat digunakan untuk melakukan pengujian secara otomatis [14]. Automation Testing tools memiliki beberapa manfaat untuk melakukan sebuah pengujian. Manfaat tersebut antara lain adalah dapat meningkatkan kecepatan, efesien, kualitas dan menurunkan biaya pengujian [14]. Salah satu tools yang banyak digunakan untuk pengujian sebuah web dari beberapa tools tersebut adalah Selenium. Selenium adalah sebuah tools yang dapat digunakan untuk pengujian aplikasi berbasis *web* [10]. Terdapat beberapa bahasa yang dapat digunakan untuk *coding* antara lain adalah Java,

C, Groovy, Perl, PHP, Python, dan Ruby [10]. Selenium dapat dibangun di *windows*, *linux*, atau *machitosh platform* [10]. Selenium merupakan *JavaScript-based* dan berjalan pada sebuah *web* sehingga penguji dapat melihat tes yang sedang berjalan. Selenium merupakan *tools* yang banyak digunakan karena bersifat opensource sehingga mudah untuk didapatkan.

4. Perancangan Sistem

4.1 Metodelogi Penelitian

Pada tugas akhir ini dibangun sistem pembangkitan kasus uji untuk pengujian aplikasi berbasis *sequence diagram*. Kasus uji dibangkitkan dengan beberapa tahap. Pada tugas akhir ini terdapat dua tahap penting dalam pembangkitan kasus uji untuk pengujian aplikasi berbasis *sequence diagram*. Tahap tersebut adalah seagabai berikut.

a. Tahap pembangkitan Kasus uji

Tahap pembangkitan kasus uji adalah tahap yang harus dilakukan untuk dapat menghasilkan sekumpulan kasus uji. Pada tahap ini terdapat beberapa subtahap yang harus dilakukan. Subtahap tersebut antara lain adalah:

1. Memilih *sequence diagram* yang dibangkitkan kasus ujinya. *Sequence diagram* yang dipilih sudah dalam bentuk file XMI. Pengubahan *sequence diagram* menjadi bentuk XMI dilakukan dengan IBM Rasional.
2. *Sequence diagram* dalam file XMI yang telah dipilih kemudian diubah menjadi GraphXML tujuannya adalah untuk mempermudah melihat dan memvisualisasikan sebuah graph yang merepresentasikan *sequence diagram.*
3. *Sequence diagram* dengan format GraphXML kemudian diubah menjadi *Sequence Dependency Graph.* Pengubahan dilakukan bertujuan untuk mempermudah menerapkan algoritma untuk menghasilkan sekumpulan kasus uji.
4. Bentuk SDG digunakan untuk membangkitkan kasus uji. Kasus Uji yang dihasilkan merupakan sebuah skenario yang digunakan untuk menguji alur dari sebuah proses yang ada pada aplikasi. Skenario Kasus uji tersebut didapatkan dengan mengadopsi algoritma *depth first search*. Mengadopsi disini adalah menggunakan algoritma *depth first search* namun melakukan perubahan pada saat implementasi.

b. Tahap pengujian terhadap aplikasi berbasis *sequence diagram*

Pada tahap ini adalah proses pengujian untuk aplikasi berbasis *sequence diagram.* Pada tahap ini kasus uji yang telah dihasilkan pada tahap pembangkitan kasus uji digunakan untuk mengujia aplikasi berbasis *sequence diagram*. Pengujian dilakukan dengan menggunakan Selenium Web driver dengan menggunakan bahasa pemrograman *java*.

5. Pengujian dan Analisis

5.1 Tujuan Pengujian

Tujuan dari dilakukannya pengujian adalah :

1. Memastikan bahwa *tools* yang dibangun telah mampu menghasilkan kasus uji berdasarkan *sequence diagram*
2. Melakukan pengujian terhadap tools sistem pembangkit kasus uji telah menghasilkan kasus uji yang sesuai.

5.2 Skenario Pengujian 1

Skenario pengujian sistem pembangkit kasus uji dapat menghasilkan sekumpulan kasus uji adalah dengan melihat hasil kasus uji yang dihasilkan sudah sesuai dengan kebutuhan yang dilihat berdasarkan dokumen *requirement* dan *sequence diagram*. Pada skenario pengujian 1 ini yang dijadikan sebagai contoh adalah *sequence diagram* login dan *sequence diagram* tingkatkan skill dari aplikasi Delima. Berikut merupakan hasil analisis terkait kasus uji yang dihasilkan oleh *tools* yang dibandingkan dengan kebutuhan dari aplikasi delima.

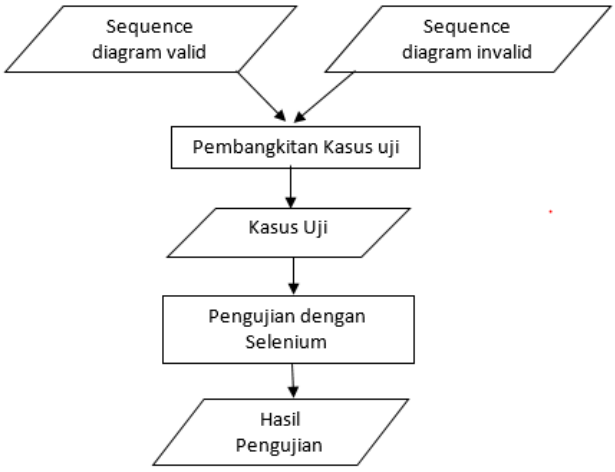
Tabel 5-1 Skenario pengujian 1

Test Scenario ID	Test Scenario Description	TC(Step)	Pre condition	Test Data	Expected result	Post condition	Importance	No. Of Test Case
L_SC1	Validasi terhadap username dan password benar	1. Menuju halaman utama Delima 2. Klik login siswa 3. Masukan username 4. Masukan password 5. Klik login 6. Login check	Masih diluar sistem, Belum dapat mengakses halaman siswa	Username : valid Password : valid	Login berhasil	Masuk halaman siswa, muncul nama siswa yang telah melakukan login	High	6
L_SC2	Validasi terhadap username dan password salah	1. Menuju halaman utama Delima 2. Klik login siswa 3. Masukan username 4. Masukan password 5. Klik 6. Login check	Masih diluar sistem, Belum dapat mengakses halaman siswa	Username : invalid Password : invalid	Login gagal	Tidak dapat masuk halaman utama siswa, diarahkan pada failure page. Mengisi username dan password ulang	High	6

Dari tabel diatas dapat dilihat bahwa hasil skenario dan kasus uji yang dihasilkan tools sesuai dengan skenario dan kasus uji hasil analisis dari requirement dan sequence diagram aplikasi Delima.

5.3 Skenario Pengujian 2

Skenario pengujian aplikasi pembangkit kasus uji berdasarkan *sequence diagram* ini dilakukan dengan metode *equivalence partitioning*. Metode tersebut adalah membagi data masukan menjadi dua bagian, yaitu data *valid* dan data *invalid*. Pada pengujian ini dimasukan *sequence diagram* yang benar (valid) sesuai dengan sistem dan *sequence diagram* yang salah (invalid) yang tidak sesuai dengan sistem. Kedua *sequence diagram* tersebut kemudian menghasilkan kasus uji yang kemudian dieksekusi dengan menggunakan selenium.



Gambar 5-1 Skenario pengujian

a. Spesifikasi sequence diagram *invalid*

Berikut adalah sebuah *sequence diagram* yang berbeda dengan sistem yang telah dibangun. Hasil dari *sequence diagram* ini nantinya akan menghasilkan kasus uji yang jika di ujikan pada aplikasi akan terdapat proses yang gagal (*fail*), atau tidak sesuai ekspektasi. Spesifikasi invalid yang digunakan pada skenario pengujian adalah *sequence diagram* login siswa dan *sequence diagram* tingkatkan skill pada aplikasi Delima.

b. Pembangkitan Kasus Uji

Kasus uji yang didapatkan untuk pengujian adalah dengan menggunakan tools yang telah dihasilkan dengan masukan *sequence diagram valid* dan *sequence diagram invalid.*

c. Pengujian Dengan Selenium

Kasus uji yang telah di dihasilkan selanjutnya di eksekusi dan diujikan pada aplikasi. Pengujian dilakukan dengan selenium web driver. Selenium merupakan tools yang dapat digunakan untuk melakukan pengujian secara otomatis dengan menggunakan bahasa pemrograman. Pada Tugas Akhir ini penulis menggunakan bahasa java untuk mengeksekusi selenium.

5.4 Hasil Pengujian Tools Pembangkit Kasus Uji

Dari hasil pengujian kasus uji yang telah dihasilkan dengan menggunkan aplikasi yang telah dibangun dan dieksekusi dengan menggunakan Selenium dapat dilihat bahwa *sequence diagram* yang *valid* jika diekseskusi dengan Selenium mendapatkan hasil yang diharapkan yaitu pass. Sedangkan sequence diagram yang invalid mendapatkan hasil *fail* sesuai dengan ekspektasi. Namun dari *sequence invalid* tersebut masih terdapat proses yang *valid*, hanya diubah beberapa proses yang *invalid*. *Sequence diagram invalid* yang prosesnya masih terdapat proses yang *valid* akan bernilai pass.

Selain itu *sequence diagram valid* dan *sequence diagram invalid* menghasilkan jumlah skenario dan kasus uji yang berbeda. Berikut adalah tabel yang dapat digunakan untuk melihat jumlah kasus uji dan skenario yang dihasilkan oleh *tools*, serta hasil pengujian dengan Selenium pada *sequence diagram valid* dan *sequence diagram invalid*.

Tabel 5-2 kesimpulan Hasil Pengujian

Spesifikasi	Modul	Jumlah		Hasil Pengujian		
		TC	SC	Pass	Fail	Jumlah Pengujian
Invalid	Login Siswa	12	2	2	10	12
	Tingkatkan Skill	13	2	11	2	13
valid	Login Siswa	12	2	12	0	42
	Tingkatkan Skill	20	3	20	0	20
	Input Bank Soal	11	2	11	0	11
	Jelajah Materi	7	1	7	0	7
	Forum Diskusi	9	2	9	0	9

Tabel 5-3 Hasil Pengujian terhadap sequence diagram Invalid

Spesifikasi	**Modul**	**Hasil Pengujian (Fail) /Jumlah eksekusi yang dibuat salah**	**Persentase**
Invalid	Login Siswa	10/10	100 %
Invalid	Tingkatkan Skill	2/2	100 %

6. Kesimpulan

Berdasarkan dari hasil penelitian dan pembuatan tugas akhir ini didapatkan beberapa kesimpulan sebagai berikut.

1. Pada tugas akhir ini telah dibuat sebuah aplikasi yang dapat membangkitkan kasus uji berdasarkan *sequence diagram* pada aplikasi berbasis objek
2. Aplikasi yang dibangun dapat menghasilkan sebuah kasus uji dengan tingkat kesesuaian sekitar 100% sesuai dengan hasil pengujian *tools* dengan skenario pengujian *tools* 2.
3. Menggunakan Selenium WebDriver untuk melakukan pengujian sesuai dengan kasus uji yang dihasilkan oleh *tools*.

7. Saran

Berikut merupakan beberapa saran yang dapat digunakan untuk menyempurnakan penelitian selanjutnya:

1. Pengujian dapat menggunakan dan mengkombinasikan diagram uml lain misalnya adalah *class diagram* atau *state chart* agar kasus uji yang dihasilkan lebih detail dan lengkap.
2. Pengujian dapat dilakukan pada aplikasi dengan spesifikasi yang lebih kompleks

Daftar Pustaka

[1] Gunadarma, "Unified Modeling Language," [Online]. Available: http://jamilah.staff.gunadarma.ac.id/Downloads/files/34878/UML_2IA.pdf. [Accessed 8 Mei 2015].

[2] J. E. Bentley, W. Bank and C. NC , "Software Testing Fundamentals—Concepts, Roles, and Terminology," *Paper 141-30,* pp. 1-12, 2004.

[3] M. Dhineshkumar and Galeebathullah, "An Approach to Generate Test Case from Sequence Diagram," *078-1-4799-3966-4*, no. ICICA, pp. 345-349, 2014.

[4] V. Panthi and D. P. Mohapatra, "Automatic Test Case Generation using Sequence Diagram," *2249-0868*, vol. 2, pp. 22-29, 2012.

[5] Informatics Lab, Modul Praktikum RPL Teknik Berorientasi Objek, Bandung: Laboratorium Informatika, Telkom Engineering School, 2013/2014.

[6] IEEE, "IEEE Standart for Software Unit Testing," *ANSI/IEEE Standart 1008-1987,* 1986.

[7] IEEE, "IEEE Standart Glossary of Software Engineering Terminology," *ANSI/IEEE Standart 6.10.12-1990,* 1990.

[8] J. E. Bentley, "Software Testing Fundamental-Concepts, Role, and Terminology," *Paper 141-30,* pp. 1-22, 2004.

[9] J. Kounitz, "Undestanding Software Test Cases".

[10] M. Junaedi, "Pengantar XML," IlmuKomputer.com, 2003.

[11] D. Kundu, D. Sumanta and R. Mall, "An Approach to Convert XMI Representation of UML 2.x Interaction Diagram into Contro Flow Graph," *International Scholarly Research Network, ID 265235,* p. 22, 2012.

[12] Ranorex, "Automated Testing," Ranorex, 2015. [Online]. Available: http://www.ranorex.com/why-test-automation.html. [Accessed 7 09 2015].

[13] Wikipedia, "Selenium (software)," Wikipedia, Rabu November 2014. [Online]. Available: http://en.wikipedia.org/wiki/Selenium_(software). [Accessed Jum'at Nove,ber 2014].