Implementasi Metode *Cause Effect Graphing* (CEG) dalam Pengujian *Requirement* Perangkat Lunak (Studi Kasus: Aplikasi G-College)

Implementation Cause Effect Graphing (CEG) Method in Requirement Testing Software (Case Study: G-College Application)

Suri Karuniawati, Sri Widowati, Ir., MT.[2], Iman Lukmanul Hakim SMB, MM[3]

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

[1]**suri_karuniawati@yahoo.co.id**, [2]**swdswd99@gmail.com**, [3]**milhcbt@gmail.com**

Abstrak

Pengembangan sebuah perangkat lunak harus melalui proses penjaminan mutu perangkat lunak. Untuk mendapatkan perangkat lunak dengan kualitas yang bagus maka pengembangan perangkat lunak harus melalui proses penjaminan mutu. Salah satu proses penjaminan mutu perangkat lunak adalah pengujian perangkat lunak.

Salah satu aspek penting dalam pengujian perangkat lunak adalah pembangkitan kasus uji. Dalam pembangkitan kasus uji terdapat beberapa teknik yang digunakan diantaranya *Whitebox testing* dan *Blackbox testing*. *Blackbox testing* merupakan pengujian yang dapat membangkitkan kasus uji dengan menggunakan spesifikasi kebutuhan perangkat lunak.

Pada Tugas Akhir ini dibangun sebuah aplikasi yang mengimplementasikan metode *Cause Effect Graphing* pada teknik *Blackbox testing* yang dapat menghasilkan kasus uji dengan menggunakan spesifikasi kebutuhan perangkat lunak dengan tingkat kebenaran sebesar 100 % dan dapat mereduksi kasus uji sekitar 90%.

Kata Kunci : *Cause Effect Graphing, Blackbox Testing*

Abstract

The development of software should pass through the process of software quality assurance. To get the software that have a good quality then the software development should pass through a process of quality assurance. One of the software Quality Assurance process is software testing.

The important aspect of testing is a generation test case. In generation of testcase there are several techniques applied include in Whitebox testing and Blackbox testing. Blackbox testing is a test that can generate test cases by using the software requirements specification.

This project tesis is to establish an application that implements the Cause Effect Graphing on Blackbox testing technique that can generate test cases by using software requirement specification with the rate of 100% truth and can reduce test cases by 90%.

Keywords : *Cause Effect Graphing, Blackbox Testing*

1. Pendahuluan

Perangkat Lunak merupakan instruksi atau program komputer yang bila dieksekusi dapat menjalankan fungsi tertentu [1]. Untuk mendapatkan perangkat lunak dengan kualitas yang bagus maka dalam tahapan pengembangan sebuah perangkat lunak harus melalui proses penjaminan mutu. Salah satu aktifitas dalam penjaminan mutu perangkat lunak adalah pengujian perangkat lunak.

Pengujian perangkat lunak adalah sebuah proses, atau serangkaian proses yang dirancang untuk memastikan bahwa program telah berjalan sesuai dengan yang diinginkan [2].Salah satu aspek penting dalam pengujian adalah pembangkitan kasus uji. Dalam pembangkitan kasus uji terdapat beberapa teknik yang digunakan diantaranya *Whitebox testing* dan *Blackbox testing*. *Whitebox testing* merupakan pengujian yang didasarkan pada kode program suatu perangkat lunak, namun pembangkitan kasus uji dengan cara ini relatif sulit diterapkan pada perangkat lunak yang kompleks. Sedangkan *Blackbox testing* merupakan pengujian yang tidak didasarkan pada kode program melainkan melainkan dapat membangkitkan kasus uji dengan menggunakan spesifikasi kebutuhan perangkat lunak.

Pada teknik *Blackbox testing* terdapat beberapa metode yang dapat digunakan untuk membangkitkan kasus uji diantaranya *Equivalence Partitioning*, *Boundary Value Analysis*, dan *Cause Effect Graphing*. Metode *Equivalence Partitioning* merupakan pengujian yang membagi data masukan dari perangkat lunak menjadi

partisi data yang kemudian diturunkan menjadi kasus uji [3]. Metode *Boundary Value Analysis* merupakan pengujian yang berfokus pada batas dimana batas nilai – nilai ekstrim dipilih [3].

Dan metode *Cause Effect Graphing* adalah metode pengujian yang membantu dalam membangkitkan kasus uji berdasarkan pada hubungan antar *cause*s (*input*) dan *effect* (*output*) yang terdapat pada spesifikasi kebutuhan perangkat lunak [4]. Namun, metode *Cause Effect Graphing* ini relatif lebih unggul dibadingkan metode lainnya dikarenakan metode ini memerhatikan integrasi antar kombinasi *input* dan *output* dan dapat mereduksi kasus uji. Oleh karena itu pada Tugas Akhir ini dibangun sebuah aplikasi pembangkit kasus uji yang mengimplementasikan metode *Cause Effect Graphing*.

2. Landasan Teori

2.1 Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan sebuah proses, atau serangkaian proses yang dirancang untuk memastikan bahwa program telah berjalan sesuai dengan yang diinginkan [2].

Dalam Pengujian sebuah *software* terdapat beberapa teknik yang dapat digunakan, diantaranya :

a. White Box Testing

White box *testing* merupakan teknik pengujian yang menggunakan struktur dan perancangan prosedural untuk memperoleh kasus uji. White box *testing* dapat mengungkap kesalahan penerapan dengan menganalisa kerja internal dan struktur sebuah software [5]. Pada pengujian ini tester perlu melihat kode suatu program untuk mengetahui bahwa unit dari kode berperilaku tidak tepat. Beberapa metode yang terdapat pada pengujian ini adalah *basis path testing, control flow testing , branch testing* dan lainnya [5].

b. Black Box Testing

Blackbox testing adalah teknik pengujian tanpa perlu mengetahui struktur internal dari suatu *software* yang akan diuji karena pengujian ini hanya berfokus kepada *input* dan *output* terhadap spesifikasi suatu *software* [6]. Beberapa teknik yang terdapat didalam *Blackbox testing* adalah,*equivalence partitioning, boundary value analysis*, dan *Cause Effect Graphing* [6].

Cause Effect Graphing merupakan teknik pengujian yang diciptakan oleh Bill Elmendorf dari IBM pada tahun 1973. Teknik ini membuat kasus uji dengan menggunakan grafik Boolean *Cause-Effect* dalam menguji fungsionalitas suatu perangkat lunak.*Cause Effect* Graph membentuk hubungan antara output dan input pada sebuah grafik Boolean dimana input sebagai *cause* dan output sebagai *effect* [3].

Tahapan yang dilakukan dalam membuat kasus uji dengan metode *Cause Effect Graphing*, diantaranya [2]:

1. Mengidentifikasi *Cause* dan *Effect* dari spesifikasi yang ada. *Cause* merupakan kondisi input dan *Effect* merupakan kondisi output atau transformasi sebuah sistem
2. Mengambarkan hubungan antar *cause* dan *effect* menggunakan grafik *cause* dan *effect*
3. Mengidentifikasi *constraint* untuk menjelaskan kombinasi antar *cause* dan antar *effect* yang tidak mungkin karena adannya batasan linkungan.
4. Mengubah grafik *cause* & *effect* menjadi Decision Table.

Pada *constraint* terdapat beberapa aturan yang dapat digunakan untuk menjelaskan kondisi hubungan antar *cause* dan antar *effect* yang tidak mungkin karena adannya batasan berdasarkan pada jenis *constraint*. Jenis – jenis *constraint* diantaranya [8] :

- Anchor(A) : kondisi dimana *cause* selalu bernilai True
- Exclusive (E): kondisi antar *cause* dimana kedua *cause* atau lebih tidak bisa sama – sama bernilai True
- One and Only One(O) : kondisi antar *cause* dimana kedua *cause* atau lebih hanya satu yang bernilai true dan tidak boleh sama – sama bernilai true atau bernilai false
- Inclusive(I) : kondisi antar *cause* dimana kedua *cause* tidak bisa sama – sama bernilai False
- Require (R): kondisi suatu *cause* mempengaruhi *cause* yang lain. Jika C1 bernilai true maka C2 bernilai true
- Mask(M) : kondisi suatu *effect* mempengaruhi *effect* yang lain. Jika E1 bennilai true maka E2 bernilai false

2.2 Selenium

Selenium merupakan *tool* yang berbasis *open source* yang digunakan untuk *automation testing*. Selenium dapat dijalankan pada *web browser* dan dapat digunakan untuk melakukan pengujian pada Aplikasi *web* [10].

Selenium dapat terintegrasi dengan berbagai *tool* dan platform dalam pengujian automation *testing*. Selenium terdiri dari beberapa bagian yaitu [10]:

- *Selenium Core*, framework Java Script yang memungkinkan untuk menulis uji kasus di HTML. Selenium Corem memerlukan instalasi Selenium IDE untuk dapat menuliskan uji kasus yang ditulis dalam HTML.
- *Selenium IDE*, berupa plugin integrasi untuk menghubungkan Selenium RC dengan web browser.

- *Selenium Client API*, mendukung beberapa Bahasa pemograman seperti java, C#, phyton dan PHP.
- *Selenium Remote Control(RC)*, server yang ditulis dengan Java yang dapat diinstall secara lokal. Server ini mengeksekusis automation kasus uji
- *Selenium WebDriver*, merupakan penerus Selenium RC, tetapi Selenium Webdriver tidak memerlukan Selenium Server
- *Selenium Grid*, server yang memungkinkan mengeksekusi sedikit kasus uji . Selenium dapat membantu meningkatkan efisiensi pengujian dengan mengeksekusi kasus uji secara paralel.

2.3 Aplikasi G-College

Aplikasi G-College merupakan sebuah aplikasi berbasis web yang menyediakan fasilitas bagi pengguna situs yang sedang mempersiapkan diri untuk menghadapi ujian masuk Universitas dengan menyajikan fitur utama aplikasi ini yaitu memberikan latihan bagi pengguna dengan mengerjakan soal – soal yang telah disediakan beserta dengan hasil koreksi dan solusi. Untuk dapat menggunakan aplikasi ini, pengguna harus terlebih dahulu terdaftar sebagai anggota. Pada aplikasi ini terdapat beberapa fitur tambahan yang dapat digunakan oleh pengguna diantaranya search, edit profile, challenge dan lainnya [11].

Fitur – fitur yang terdapat pada aplikasi G-college, diantaranya [11] :

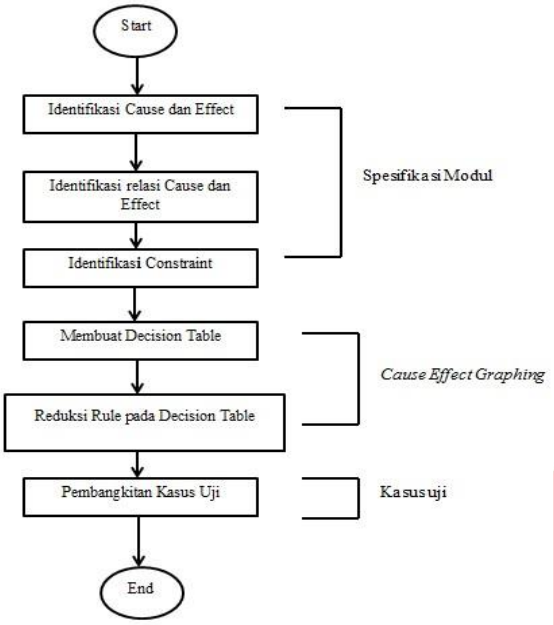
Tabel 2-3 Fitur - fitur aplikasi G-college

No	Modul	Deskripsi
1	Course	Fitur yang menyediakan soal – soal untuk dapat dikerjakan oleh pengguna beserta solusi
2	Sign In	Fitur yang digunakan oleh pengguna untuk dapat memasuki halaman utama G-College
3	Sign Up level 1 & 2	Fitur yang digunakan untuk melakukan pendaftaran sebagai anggota G-College
5	Edit Profile	Fitur yang digunakan untuk mengelola profile pada aplikasi G-college
6	Update status	Fitur yang digunakan untuk menulis status yang diinginkan oleh pengguna
7	Challange	Fitur yang digunakan pengguna untuk menantang pengguna lain dalam mengerjakan soal
8	View Profile	Fitur yang digunakan untuk melihat informasi terkait aktifitas pengguna
9	Tabel	Fitur yang digunakan untuk melakukan pencarian
10	Posting	Fitur yang digunakan pengguna untuk mengomentari halaman dinding pengguna lain

3. Pengembangan Sistem

Pada Tugas Akhir ini dilakukan pengembangan sebuah sistem yang dapat menghasilkan kasus uji yang digunakan untuk melakukan pengujian *requirement* dari spesifikasi modul yang terdapat pada dokumen SKPL (Spesifikasi Kebutuhan Perangkat Lunak) aplikasi G-College.

Gambaran umum sistem yang dibagun pada Tugas Akhir ini adalah sebagai berikut:



Gambar 3-1 Gambaran umum sistem *Cause Effect Graphing*

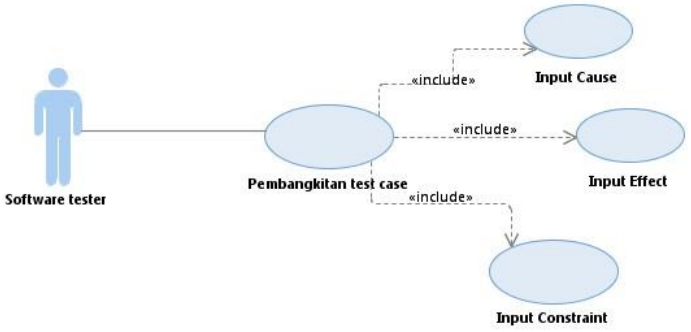
Tahapan utama yang terdapat pada Gambar 3-1 dapat dijelaskan sebagai berikut:

1. Pada dokumen SKPL (Spesifikasi Kebutuhan Perangkat Lunak) digunakan deskripsi spesifikasi modul – modul pada aplikasi G-College. Deskripsi spesifikasi modul – modul aplikasi G-College ini digunakan sebagai inputan sistem untuk menghasilkan kasus uji.
2. Spesifikasi yang digunakan untuk melakukan pengujian aplikasi dibagi menjadi dua yaitu spesifikasi *valid* dan spesifikasi *invalid*. Spesifikasi *valid* adalah spesifikasi modul yang berasal dari dokumen spesifikasi kebutuhan perangkat lunak pada aplikasi G-College. Sedangkan spesifikasi *invalid* adalah spesifikasi yang berasal dari dokumen spesifikasi kebutuhan perangkat lunak pada aplikasi G-College tetapi isinya dirubah menjadi salah.
3. Spesifikasi *valid* yang akan digunakan untuk melakukan pengujian adalah spesifikasi modul Sign In, Sign Up level 1, Course, dan Update Status sedangkan Spesifikasi *invalid* yang digunakan untuk melakukan pengujian adalah spesifikasi modul Sign Up level 1 dan Course.
4. Kemudian spesifikasi tersebut dijadikan inputan ke dalam sistem yang mengimplementasikan metode *Cause Effect Graphing* yang menghasilkan keluaran berupa kasus uji.

3.1 Pemodelan Sistem Cause Effect Graphing

Sistem yang dibangun merupakan sistem yang mengimplementasikan metode *Cause Effet Graphing*. Sistem ini dirancang dengan menggunakan pendekatan berbasis *Obeject Oriented.*

a. Use Case Diagram



Gambar 0-1 Use Case Diagram *Cause Effect Graphing*

4. Hasil Pengujian dan Kesimpulan

Pada Tugas Akhir ini, Pengujian dilakukan pada sistem yang dibangun menggunakan metode *Equivalence Partitioning* yaitu dengan membagi data input yaitu spesifikasi modul menjadi dua kelas data yaitu data *valid*

dan *invalid*. Data *valid* merupakan data yang didapat dari spesifikasi modul *valid* dan data *invalid* merupakan data yang didapat dari spesifikasi modul *invalid*. Spesifikasi modul *valid* yang digunakan adalah spesifikasi modul Sign Up level 1, Sign In, Update Status dan Course. Sedangkan spesifikasi modul *invalid* yang digunakan adalah spesifikasi modul Sign Up level 1 dan Course.

Data *valid* dan *invalid* ini kemudian dijadikan inputan ke dalam sistem untuk menghasilkan kasus uji. Kasus uji yang dihasilkan ini kemudian dieksekusi menggunakan *tool* yaitu selenium untuk memperoleh hasil pengujian.

4.3.1 Hasil Reduksi Kasus Uji

Dari hasil *rule* yang didapatkan dapat dilihat bahwa dalam melakukan reduksi kasus uji digunakan constraint yang dapat membantu dalam memilih kasus uji yang mencakup keseluruhan fungsionalitas, dan dapat mengurangi kasus uji yang ambigu.

Setiap kasus uji pada setiap modul memililki presentase tingkat reduksi yang terdapat pada Tabel 4-8, sebagai berikut:

Table 0-8 Persentase Reduksi kasus Uji

No	Modul	Jumlah rule Sebelum tereduksi	Jumlah rule sesudah tereduksi	Persentase
1	Sign Up level 1 (valid)	512	18	96,48 %
2	Course (valid)	256	5	98, 04 %
3	Sign In level 1 (valid)	1024	18	98,24 %
4	Update Status (valid)	64	3	95, 31 %
5	Sign Up level 1 (invalid)	512	18	96,48 %
6	Course (invalid)	256	5	98, 04 %

Dari hasil presentase diatas dapat dilihat bahwa, rata – rata kasus uji tereduksi sebesar 90 %.

4.3.2 Hasil Eksekusi Kasus uji dengan Selenium

Kasus uji yang telah dihasilkan oleh sistem selanjutnya diuji menggunakan selenium untuk mengetahui apakah hasil pengujian yang didapatkan sudah sesuai dengan *output* yang diharapkan.

Tabel 0-10 *Summary* Hasil Eksekusi Kasus Uji

Spesifikasi	Modul	Hasil Pengujian		
		Pass	Fail	Jumlah Eksekusi
Valid	Sign Up level 1	18	0	18
	Sign In	18	0	18
	Course	14	0	14
	Update Status	4	0	4
Invalid	Course	10	4	14
	Sign Up level 1	3	15	18

Dari hasil pengujian yang didapatkan dapat dianalisis pada Tabel 4-10 bahwa kasus uji yang dihasilkan oleh spesifikasi *valid* memiliki hasil pengujian dengan ekspektasi *output* yang sama yaitu *Pass* sedangkan pada Tabel 4-11, kasus uji yang dihasilkan oleh spesifikasi *invalid* memiliki hasil pengujian dengan ekpektasi *output* yang tidak sama atau berbeda pada bagian yang telah diubah isi spesifikasinya yaitu *Fail*, tetapi pada bagian yang spesifikasinya tidak diubah hasil pengujian sama dengan ekspektasi output(Pass).

Tabel 0-11 Persentase Hasil Pengujian Spesifikasi *Invalid*

Spesifikasi	Modul	Hasil Pengujian (Fail) /Jumlah eksekusi yang dibuat salah	Persentase
Invalid	Course	4/4	100 %
Invalid	Sign Up Level 1	15/15	100 %

Sehingga dari hasil pengujian tersebut dapat dikatakan bahwa sistem dapat menghasilkan kasus uji dengan benar.

Daftar Pustaka:

[1] R. S. Pressman, P.hd, "Software Enggineering a Practitioneer's Approach," Mc Graw hill, p. 6.

[2] G. J Myers, T. Badgett and S. C, "The Art of Software *Testing* 3 rd Edition," John Wiley & Sons, Inc., 2012.

[3] G. E. Mogyordi, "Requirements-Based *Testing* - *Cause-Effect* Graphing," *Software Testing Services*, pp. 1-12, 2005-2010.

[4] "IEEE Standart Glosary of Software Engineering Terminology," *IEEE Standart 610.12.1990*, 1990.

[5] M. Ehmer Khan, "Different Approaches to White Box *Testing* Technique for Finding Errors," *International Journal of Software Engineering and its Appliation*, vol. 5 No.3, pp. 1-14, 2011.

[6] M. Ehmer Khan, "Different Approach to *Blackbox Testing* Technique for finding Errors," *International Journal of Software Enggineering & Application*, vol. 2 No.4, pp. 1-10, 2011.

[7] L. Willian , "*Testing* Overview and *Blackbox Testing* Techniques," 2006, pp. 34-59.

[8] D. Graham, E. Van Veenendaal and I. Evans, Foundation of Software *Testing*, Cengage Learning EMEA, 2008.

[9] Software *Testing* Class, "Software *Testing* Class," 2013. [Online]. Available: http://www.software*testing*class.com/software-*testing*-life-cycle-stlc/. [Accessed Juni 2015].

[10] H. Antawan and K. Marc, "Automating Funtional Test Using Selenium," pp. 1-6, 2006.

[11] "Software Requirement Specification G-College," 2013.