Jurnal Kreativitas Mahasiswa Informatika
Volume 2 Nomor 2 Tahun 2021 Page 304 - 306
p-ISSN: 2797-6327 e-ISSN:

Implementasi Metode Cause Effect Graphing (CEG) Pengujian Requirement Perangkat Lunak (Aplikasi CAT-CPNS)

Dion pernandi[1], Muhamad Fadly[2], Sufriyah[3], Tri Rachmad Saputro[4]
[1-11]Universitas Pamulang; Jl. Raya Puspitek No. 46 buaran, serpong, Kota Tangerang Selatan. Provinsi Banten 15310. (021) 741-2566 atau 7470 9855
[1-11]Jurusan Teknik Informatika, Fakultas Teknik, Universitas Pamulang

e-mail: [1]dionpernandi19@gmail.com, [2]Maniafadly39@gmail.com, [3]sufriyahanida@gmail.com, [4]trirachmad.saputro17@gmail.com

Abstrak

Suatu pengembangan perangkat lunak harus melalui proses penjaminan mutu perangkat lunak. Agar mendapatkan perangkat lunak dengan kualitas yang bagus maka pengembangan perangkat lunak harus melalui proses penjaminan yang bermutu. Salah satu proses penjaminan mutu perangkat lunak adalah pengujian pada perangkat lunak. Salah satu aspek terpenting dalam pengujian perangkat lunak adalah pembangkitan kasus uji. Dalam pembangkitan kasus uji tersebut terdapat beberapa teknik yang digunakan diantaranya adalah *White Box Testing* dan *Black Box Testing*. Blackbox testing merupakan pengujian yang dapat membangkitkan kasus uji dengan menggunakan spesifikasi kebutuhan perangkat lunak. ini dibangun sebuah aplikasi yang mengimplementasikan metode *Cause Effect Graphing* pada teknik *Black Box 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: Implementasi, Cause Effect Graphing, Black Box Testing Pengujian

I. PENDAHULUAN

Perangkat Lunak merupakan instruksi atau program komputer yang bila dieksekusi dapat menjalankan fungsi tertentu. 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. Salah satu aspek penting dalam pengujian adalah pembangkitan kasus uji.

Aplikasi CAT-CPNS merupakan sebuah aplikasi berbasis web yang menyediakan fasilitas bagi pengguna situs yang sedang mempersiapkan diri untuk menghadapi ujian menjadi CPNS dengan menyajikan fitur utama aplikasi ini yaitu memberikan latihan bagi pengguna dengan mengerjakan soal – soal yang telah disediakan beserta dengan hasil koreksi dan standar minimal kompetensi dasar bagi pelamar CPNS. 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 materi, simulasi CAT, favorit dan lainnya. *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.

II. METODE PELAKSANAAN

Dalam pembangkitan kasus uji terdapat beberapa teknik yang digunakan diantaranya *White-box testing* dan *Black-box testing*.

Jurnal Kreativitas Mahasiswa Informatika
Volume 2 Nomor 2 Tahun 2021 Page 304 - 306
p-ISSN: 2797-6327 e-ISSN:

White-box testing merupakan pengujian yang didasar kan pada kode program suatu perangkat lunak, namun pembangkitan kasus uji dengan cara ini relatif sulit diterapkan pada perangkat lunak yang kompleks. Sedangkan *Black-box testing* merupakan pengujian yang tidak didasarkan pada kode program melainkan melainkan dapat membangkitkan kasus uji dengan menggunakan spesifikasi kebutuhan perangkat lunak. Pada teknik *Black-box 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. Metode *Boundary Value Analysis* merupakan pengujian yang berfokus pada batas dimana batas nilai – nilai ekstrim dipilih. Dan metode *Cause Effect Graphing* adalah metode pengujian yang membantu dalam membangkitkan kasus uji berdasarkan pada hubungan antar *causes* (*input*) dan *effect* (*output*) yang terdapat pada spesifikasi kebutuhan perangkat lunak. 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 dibangun sebuah aplikasi pembangkit kasus uji yang mengimplementasikan metode *Cause Effect Graphing*.

III. HASIL DAN PEMBAHASAN

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 3.1-3.3, sebagai berikut:

Tabel 3. 1 *Persentase* Reduksi kasus Uji

No	**Modul**	**Jumlah rule Sebelum te reduksi**	**Jumlah rule sesudah te reduksi**	**Per sentase**
1.	Sign Up level 1 (valid)	714	19	96,48%
2.	Course (valid)	254	5	98,04%
3.	Sign In level 1 (valid)	1054	19	98,24 %
4.	Update Status (valid)	64	4	95,31%
5.	Sign Up level 1 (invalid)	512	19	96,48%
6.	Course (invalid)	256	5	98,04%

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

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 3. 2 *Summary* Hasil Eksekusi Kasus Uji

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

Dari hasil pengujian yang didapatkan dapat dianalisis pada Tabel 3.1 bahwa kasus uji yang dihasilkan oleh spesifikasi valid memiliki hasil pengujian dengan ekspektasi *output* yang sama yaitu Pass sedangkan pada Tabel 3.2, 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).

Jurnal Kreativitas Mahasiswa Informatika
Volume 2 Nomor 2 Tahun 2021 Page 304 - 306
p-ISSN: 2797-6327 e-ISSN:

Tabel 3. 3 *Persentase* Hasil Pengujian Spesifikasi *Invalid*

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

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

IV. SIMPULAN

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.

DAFTAR PUSTAKA

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

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

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

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

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.

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.

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

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

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

H. Antawan and K. Marc, "*Automating FuntionalTest Using Selenium,*" pp. 1-6, 2006. "https://pemerintah.net/simulasi-cat-online/," 2018.