Pengujian Sistem Informasi Pengelolaan Pelatihan Kerja UPT BLK Kabupaten Kudus dengan Metode Whitebox Testing

Yudie Irawan.
Program Studi Sistem Informasi, Universitas Muria Kudus
yudie.irawan@umk.ac.id

Abstrak - Kegiatan Rekayasa perangkat lunak tidak akan lepas dari aktifitas pengujian(*testing*). Metode pengujian telah berkembang seiring dengan berkembangnya teori rekayasa software. Penelitian ini merupakan salah satu eksperimen penerapan pengujian unit pada sistem yang telah dibangun pada penelitan sebelumnya. Pengujian unit pada sistem ini menggunakan metode pengujian *white box testing*. Metode penelitian ini mengambil beberapa tahapan pada siklus pengujian sistem(*Software Testing Life Cycle*) yang meliputi *Requirement Analysis, Test Planning, Test Case Development, Environtment Setup, Test Execution dan Test Cycle Closure*. Pengujian *white box* adalah metode desain *test case* yang menggunakan struktur kendali dari desain prosedural. Hasil penelitian ini bermanfaat untuk mengetahui tingkat kesiapan sistem pada tahap implementasi. Walaupun masih ada pengujian lainnya, namun pengujian dengan metode *whitebox* testing merupakan awal dari rangkaian pengujian sistem, dimana pengujian lainnya dapat dilakukan setelah melewati metode ini. Hasil dari penelitian ini berupa dokumentasi pengujian yang akan menyajikan tingkat kelayakan sistem sesuai dengan kaidah logika algoritma dan cara kerja sistem.
Kata kunci: pengujian, unit, white box, software testing life cycle

Abstrak - Kegiatan Rekayasa perangkat lunak tidak akan lepas dari aktifitas pengujian(*testing*). Metode pengujian telah berkembang seiring dengan berkembangnya teori rekayasa software. Penelitian ini merupakan salah satu eksperimen penerapan pengujian unit pada sistem yang telah dibangun pada penelitan sebelumnya. Pengujian unit pada sistem ini menggunakan metode pengujian *white box testing*. Metode penelitian ini mengambil beberapa tahapan pada siklus pengujian sistem(*Software Testing Life Cycle*) yang meliputi *Requirement Analysis, Test Planning, Test Case Development, Environtment Setup, Test Execution dan Test Cycle Closure*. Pengujian *white box* adalah metode desain *test case* yang menggunakan struktur kendali dari desain prosedural. Hasil penelitian ini bermanfaat untuk mengetahui tingkat kesiapan sistem pada tahap implementasi. Walaupun masih ada pengujian lainnya, namun pengujian dengan metode *whitebox* testing merupakan awal dari rangkaian pengujian sistem, dimana pengujian lainnya dapat dilakukan setelah melewati metode ini. Hasil dari penelitian ini berupa dokumentasi pengujian yang akan menyajikan tingkat kelayakan sistem sesuai dengan kaidah logika algoritma dan cara kerja sistem.
Kata kunci: pengujian, unit, white box, software testing life cycle

PENDAHULUAN

Perkembangan metode pengujian sistem berkembang seiring dengan berkembangnya metode pengembangan sistem dan teknologi aplikasi yang dibangunnya. Tahapan testing atau pengujian selalu ada dalam salah satu tahapan pengembangan sistem yang ada. Pengujian sistem menjadi penting karena hanya melewati tahap yang mampu menunjukkan kesiapan sistem sebelum diterapkan di lapangan. Pengujian perangkat lunak memiliki peranan penting dalam siklus hidup sistem untuk mendeteksi tingkat kesulitan yang ada dalam tiap proses dengan baik (Khan & Khan, 2014).

Pengujian perangkat lunak dilakukan untuk mendeteksi adanya kesalahan, yang menyebabkan kegagalan perangkat lunak. Namun, pengujian perangkat lunak menjadi tugas yang memakan waktu dan mahal. Tahap pengujian mampu menghabiskan hampir 50% sumber daya pengembangan perangkat lunak. Pengujian perangkat lunak juga dapat didefinisikan sebagai proses verifikasi dan validasi pada perangkat lunak untuk memastikan bahwa aplikasi telah memenuhi persyaratan teknis dan bisnis seperti yang diharapkan. (Chayanika Sharma, Sangeeta Sabharwal, Ritu Sibal, 2013). Pengujian pada perangkat lunak dapat didefinisikan sebagai sejumlah aktifitas yang dapat direncanakan dengan baik dan terarah secara sistematik. Oleh karenanya maka sebuah template pengujian - sejumlah fase dimana didalamnya dapat ditempatkan teknik desain test case dan metode pengujian – harus ditetapkan tiap dalam proses pembuatan software. (Pressman & Maxim, 2014). Secara garis besar terdapat dua metode yang dikenal dalam pengujian software, yang pertama melakukan pengujian dari sisi fungsionalitasnya, pengujian diarahkan untuk menunjukkan sejauh mana produk dapat memenuhi fungsi sebagaimana mestinya. Metode ini disebut dengan pengujian Black-box. Pengujian cara kedua adalah dengan

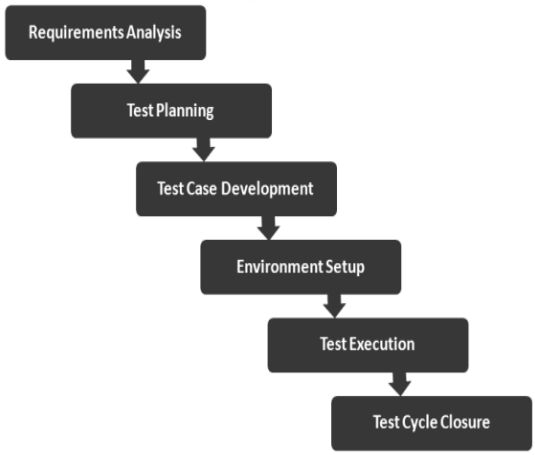
menguji cara kerja dari produk tesebut, pengujian ini diarahkan untuk menunjukkan tingkat kebenaran metode yang digunakan, cara kerja yang sesuai prosedur dan spesifikasi internal lainnya. Metode kedua disebut dengan pengujian White-box.

Hingga saat ini prioritas penggunaan metode pengujian baik *black-box* maupun *whitebox* masih dan akan terus digunakan pada pengujian sistem, bahkan dalam beberapa rilis sistem yang telah dibuat. (Christopher Henard, Mike Papadakis, Mark Harman, Yue Jia, Yves Le Traon, 2016). Metode pengujian yang digunakan dalam penelitian ini adalah *whitebox*. Pengujian *whitebox* juga dikenal sebagai pengujian struktural, pengujian *transparent box*, pengujian berbasis logika atau pengujian berbasis kode. Kata *whitebox*(kotak putih/transparan) mengacu pada sebuah metode *test case*, perangkat lunak yang sedang diuji dianggap sebagai kotak(*box*), sedangkan kata *white*/*transparent* mengacu pada bahwa kotak itu terlihat jelas isinya. Penguji dapat melihat jelas bagian dalam kotak dan cara kerjanya di dalamnya. Seringkali *logical errors* dan pemahaman *script*(kode program) yang keliru, akan berbanding terbalik dengan alur sistem yang seharusnya. Karena banyaknya yang mengatakan bahwa *logical path* adalah hal sulit untuk dieksekusi, sebenarnya dapat saja dieksekusi dalam keadaan normal. Dan karena kesalahan ketik dalam kode itu bersifat acak dan tidak dapat dihindari, dari banyak hal itulah maka kita harus melakukan pengujian *whitebox*. Tujuannya adalah untuk memeriksa *logical path* perangkat lunak dengan memeriksa struktur logis perangkat lunak. Poin - poin pengujian utama akan ditentukan pada kasus yang berbeda-beda untuk menguji program, sehingga mampu menyimpulkan apakah perangkat lunak sudah berjalan sesuai dengan kebutuhan yang diharapkan atau sebaliknya. Arah pengujian telah ada pada desain pengujian yang telah ditentukan pada tahap perancangan sistem, biasanya dituangkan dalam dokumen pengujian perancangan perangkat lunak, rincian prosedur pengujian, rancangan *test case* dengan spesifik studi kasus dan pada pengujian cakupan jalur logika. (Bo Li, Lichen Zhang, 2017).

Sistem yang akan diuji ini meliputi 13 modul aplikasi. Pengujian menggunakan metode whitebox dengan teknik *basis path* dan disajikan dalam diagram *control flograph testing*.

METODE

Metode penelitian yang dilakukan mengacu pada *Software Testing Life Cycle* yang dapat dilihat pada gambar 1 :



Gambar 1. Software Testing Life Cycle

Berikut adalah penjelasan masing – masing tahapan :

1.*Requirement analysis*; Tahap mengumpulkan data dan informasi kebutuhan aktifitas pengujian, referensi dari jurnal dan artikel ilmiah, serta buku yang membahas metode pengujian sistem.

2.*Test Planning;* Tahap penjadwalan dan rencana kegiatan pengujian menyesuaikan jumlah unit yang akan diuji dan jumlah tester.

3.*Test Case Development* ; Pada tahap ini dibuat *desain test* case pengujian *white box* dengan teknik cakupan dengan *control flowgraph*.

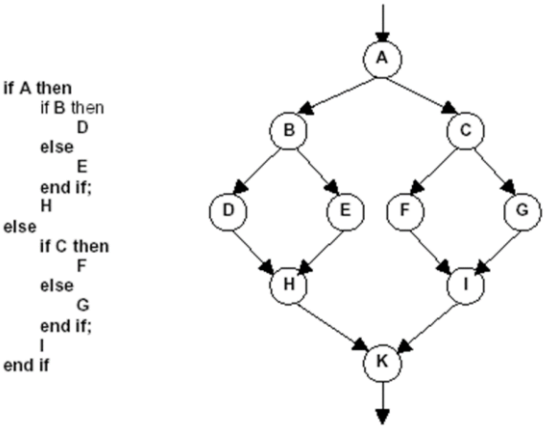
4.*Environtment Setup ;* Pengaturan *hardware* dan *software* yang akan digunakan antara lain ; sistem operasi, kapasitas memori penyimpanan dan memory pemrosesan serta kapasitas jaringan yang digunakan dalam pengujian. Dalam tahap ini dilakukan pengecekan terhadap kebersihan sistem dari gangguan eksternal termasuk bersih dari virus komputer.

5.*Test Execution ;* Pengujian sistem seperti rencana yang telah disusun sebelumnya. Tahap ini dilakukan pengujian dengan pembagian tugas pengujian menggunakan 3 komputer dengan spesifikasi yang sama.

6.*Test Cycle Closure ;* Penyusunan dokumentasi hasil pengujian yang berisi kesimpulan kelayakan sistem. (ISTQBExamCertification.com, 2013).

Cakupan pernyataan, cabang dan jalur adalah suatu teknik white box testing yang

menggunakan alur logika dari program untuk membuat test cases. Yang dimaksud dengan alur logika adalah cara dimana suatu bagian dari program tertentu dieksekusi saat menjalankan program. Alur logika biasanya direpresentasikan dalam bentuk *flowgraph* seperti pada gambar 2 .



Gambar 2. Notasi *Flowgraph*

Pada gambar 2 menunjukkan konstruksi struktural pada *flow graph*, dimana tiap siklus melambangkan satu atau lebih pernyataan kode (source code statement). Tiap lingkaran, disebut *flow graph node*, yang mewakili satu atau lebih pernyataan prosedural. (Pressman & Maxim, 2014)

HASIL DAN PEMBAHASAN

Pengujian dilakukan dalam 3 komputer dengan spesifikasi yang sama. Masing – masing komputer menjalankan test case yang telah disusun sebelumnya. Teknik basis path diawali dengan mengubah kode program ke dalam node yang kemudian digambarkan dalam diagram *flowgraph*. Untuk mentransformasi kode – kode program dalam node, penguji harus memahami algoritma dan sintak program, sehingga penguji haruslah seorang yang menguasai bahasa pemrograman juga. Hal ini sesuai dengan konsep pengujian *transparent box* dimana penguji bisa melihat jelas isi kotak. Selain baris perintah, node juga mewakili blok fungsi yang terdiri dari beberapa baris perintah. Notasi pada node diwakili dengan angka dan disusun secara urut sesuai alur logika program yang dikonversikan. Dalam penelitian ini terdapat tiga jenis cakupan yang diuji, yaitu cakupan pernyataan, cakupan cabang dan cakupan jalur.

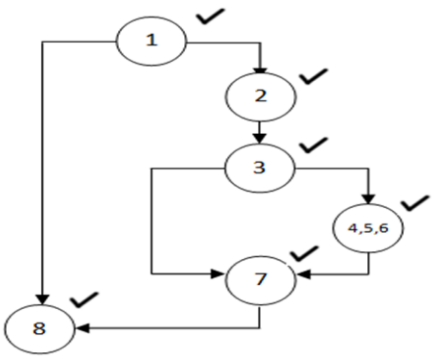
Langkah membuat *test case* dimulai dengan membuat flowchart dari sebuah kode. Setiap perintah pada program diberikan nomor urut didepannya. Pemberian nomor ini dapat berdasarkan blok fungsi perintah atau dapat juga tiap kode perintah program, dapat dilihat pada gambar 3.

```
1  if ($_POST['btnSimpan']=="Selanjutnya-->>") {
2       if (empty($_POST['txtno_ktp'])){
3            echo"<script>alert('Maaf Text KTP harus
   Diisi')</script>";
4            echo "<meta http-equiv='refresh' content='0;
   url=?page=tambah_peserta_pelatihan'>";
       }elseif(empty($_POST['txtnama'])){
5
6            echo"<script>alert('Maaf Text Nama harus
   Diisi')</script>";
7            echo "<meta http-equiv='refresh' content='0;
   url=?page=tambah_peserta_pelatihan'>";
       }elseif(empty($_POST['txttmp_lahir'])){
8            echo"<script>alert('Maaf Text Tempat Lahir harus
   Diisi')</script>";
9            echo "<meta http-equiv='refresh' content='0;
   url=?page=tambah_peserta_pelatihan'>";
10     }elseif(empty($_POST['txttgl_lahir'])){
            echo"<script>alert('Maaf Text Tanggal Lahir harus
   Diisi')</script>";
11
12           echo "<meta http-equiv='refresh' content='0;
   url=?page=tambah_peserta_pelatihan'>";
       }elseif(empty($_POST['taalamat'])){
13           echo"<script>alert('Maaf Text Alamat harus
   Diisi')</script>";
14           echo "<meta http-equiv='refresh' content='0;
   url=?page=tambah_peserta_pelatihan'>";
15     }elseif(empty($_POST['txtnope'])){
           echo"<script>alert('Maaf Text Alamat harus
16 Diisi')</script>";
           echo "<meta http-equiv='refresh' content='0;
```

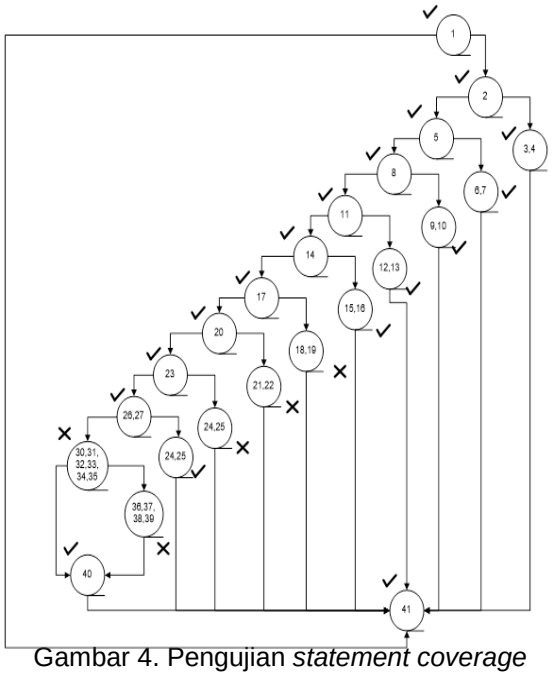
Gambar 3. Pemberian nomor pada kode program

Selanjutnya nomor tersebut digunakan sebagai node yang disusun menurut alur program untuk membuat *flowgraph.* Dengan bantuan *flowgraph* arah pengujian menjadi lebih mudah. Berikutnya melakukan pengujian pada level *statement coverage*/perintah. Pada node yang kode perintahnya telah dijalankan dengan baik maka node tersebut diberi tanda centang. Contoh pada gambar 4a, semua node berhasil dijalankan dan diberi tanda centang. Sedangkan node yang mengalami kegagalan diberi tanda silang, seperti pada gambar 4b.

a.



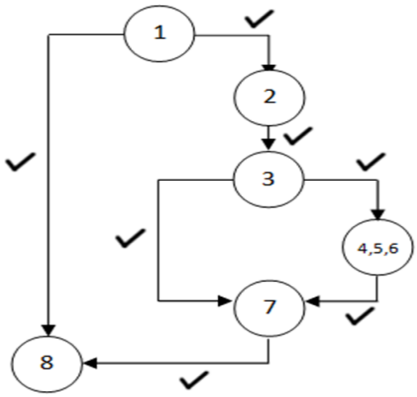
b.



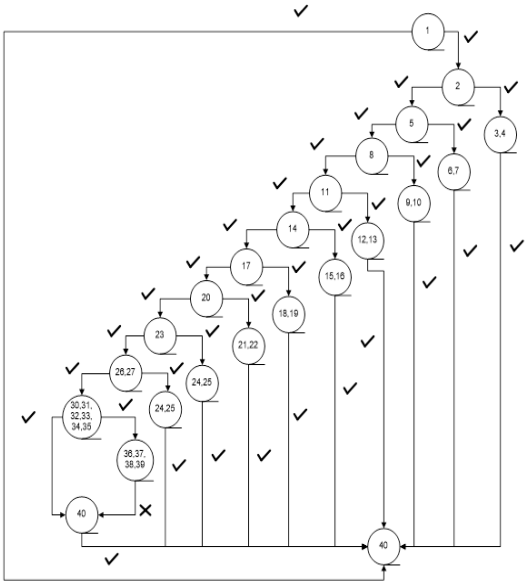
Gambar 4. Pengujian *statement coverage*

Langkah selanjutnya melakukan pengujian cabang(*branch/decision coverage*), dimana semua kondisi pilihan diuji untuk memastikan tidak ada kesalahan pada kode program untuk setiap pilihan. Pada pengujian cabang ini dilakukan juga pengujian yang berkaitan dengan pengulangan. Pengujian cabang disajikan dalam gambar 5a dan 5 b. Tampak bahwa pada flowgraph a tidak ditemukan kesalahan, sedangkan pada flowgraph 5b ditemukan kesalahan satu node dengan angka 36,37,38,39.

a.



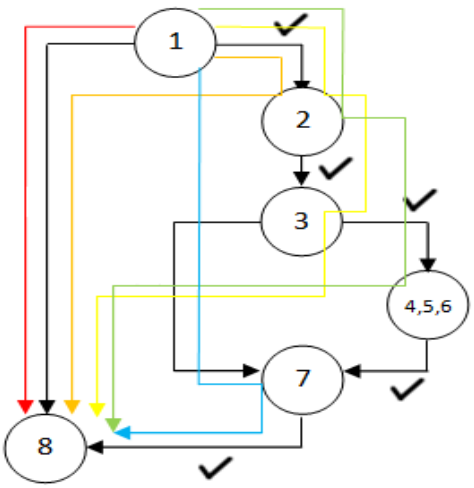
b.



Gambar 5. Pengujian *decision coverage*

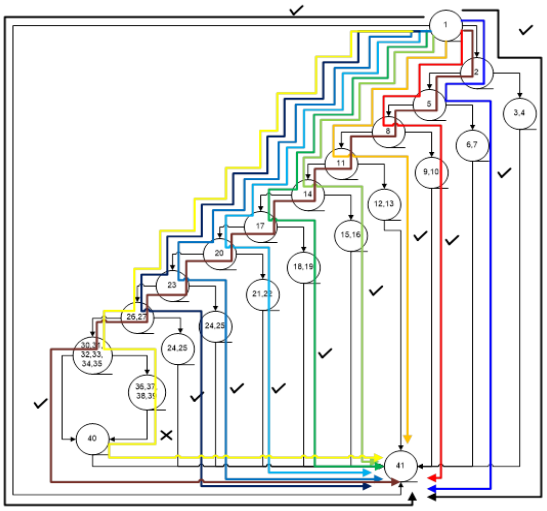
Pengujian berikutnya ada pada level jalur (*path coverage*). Pengujian jalur adalah pengujian dari awal dijalankannya sebuah modul sampai pada akhir modul. Biasanya diakhiri dengan penyimpanan atau pembatalan fungsi. Pengujian ini mengharuskan mengunjungi setiap node yang ada pada jalur yang berkesinambungan. Kegagalan pada pengujian jalur biasanya akan mengakibatkan berhentinya program ditengah eksekusi. Pada gambar 6a 5 jalur/ path dan pada gambar 6b terdapat 8 jalur/*path*. Garis jalur ditandai dengan warna yang berbeda untuk memudahkan pembacaan dan pelacakan alur program.

A



.

b.



Gambar 6. Pengujian *path coverage*

Dari 13 modul yang dilakukan pengujian ada 21 bugs yang ditemukan sesuai node pada beberapa modul. Kesalahan yang paling sering ditemui adalah kesalahan sintak, baik yang disebabkan karena salah ketik, kurang lengkap atau salah variabel. Kesalahan ditemui pada *statement coverage, branch coverage* maupun *path coverage*. Rekap kesalahan yang ditemukan dapat dilihat pada tabel 1.

Tabel 1. Rekap kesalahan yang ditemukan

Nama Modul	Posisi Node	Keterangan
Input Identitas Pendaftar	18, 19, 21, 22, 24, 25,31,36, 37, 38, 39	Salah perintah, salah ketik.
Input Pemilihan Jurusan	19, 25, 27	Salah variabel, Salah query, Salah sintak
Input Identitas Instruktur	18, 19, 21, 22, 24, 25, 39, 40	Salah sintak, salah ketik
Penerimaan Peserta	3,7	Sintak tidak lengkap

SIMPULAN DAN SARAN

Pengujian pada tingkat statemen, cabang maupun jalur telah dilakukan sampai 100 %. Hasil pengujian menunjukkan bahwa dari 13 modul terdapat 4 modul yang didalamnya ditemukan kesalahan. Kesalahan yang ditemukan termasuk pada kesalahan mayor dan berakibat sistem berhenti, oleh karena itu perlu perbaikan kode program sebelum aplikasi dapat digunakan oleh user.

Pengujian adalah tahapan yang membutuhkan banyak waktu dan tenaga, oleh karena itu sangat diperlukan kesiapan *testcase* yang sebaik – baiknya. Metode pengujian whitebox memiliki banyak teknik, sehingga perlu pengujian dengan teknik yang lain, misalnya *basispath* dan *graph matrixs*.

DAFTAR PUSTAKA

[1] Bo Li, Lichen Zhang. (2017). Analysis of White Box Test of Cyber-Physical System. *American Institute of Physics Conference Proceedings*, 020183-1, 020183-7. doi:doi: 10.1063/1.4982548

[2] Chayanika Sharma, Sangeeta Sabharwal, Ritu Sibal. (2013, January). A Survey on Software Testing Techniques using Genetic Algorithm. *IJCSI International Journal of Computer Science Issues, 10*(1, No 1), 381 - 393. Retrieved from www.IJCSI.org

[3] Christopher Henard, Mike Papadakis, Mark Harman, Yue Jia, Yves Le Traon. (2016, May). Comparing White-box and Black-box Test Prioritization. *International Conference on Software Engineering*, 14-22. doi:DOI: http://dx.doi.org/10.1145/2884781.2884791

[4] Dr. A. Nirmal Kumar, Dr. B.G. Geetha. (2016, Jan.-March). Whitebox Testing Technique for SOA. *International Journal of Advanced Engineering Technology, VII*(I), 707-709.

[5] ISTQBExamCertification.com. (2013). *What is Software Testing Life Cycle (STLC)?* Retrieved from ISTQB Exam Certification: http://istqbexamcertification.com/what-is-software-testing-life-cycle-stlc/

[6] Khan, M. E., & Khan, F. (2014, March). Importance of Software Testing in Software Development Life Cycle. *IJCSI International Journal of Computer Science Issues, 11*(2), 120-123.

[7] Pressman, R., & Maxim, B. (2014). *Software Engineering, a Practitioner's Approach* (8th ed.). New York: McGraw-Hill.

[8] Syed Roohullah Jan, Syed Tauhid Ullah Shah, Zia Ullah Johar, Yasin Shah, Fazlullah Khan. (2016, March-April). An Innovative Approach to Investigate Various Software Testing Techniques and Strategies. *International Journal of Scientific Research in Science, Engineering and Technology, 2*(2), 682-689.