PROSIDING Seminar Nasional SISFOTEK ISSN 2597-3584 (Media Online)
Sistem Informasi dan Teknologi
website : www.seminar.iaii.or.id | email : seminar@iaii.or.id

Penerapan Metode *Basis Path Analysis* dalam Pengujian *White Box* Sistem Pakar

Cindy Pamela C Munaiseche[1], Gladly C Rorimpandey[2]
[1,2]Prodi Teknik Informatika, Fakultas Teknik, Universitas Negeri Manado
cindymunaiseche@unima.ac.id

Abstract

Testing is an important thing to do for a software. Early stage testing can be done through unit testing. The unit testing uses whitebox testing with the basis path analysis method. This method using the program structure from a flowchart program and then it will be converted into a flowgraph, but actually there are so much program code that is divided into many files that do not allow for flowgraphs to be created. This research aims to apply the basis path analysis method for the white box testing in the expert system of diagnosing heart disease by using the flowchart system to overcome the problem of the number of program code files that resulted in the difficulty of making flowgraphs. The Basis Path Analysis method consists of four stages, namely: flowgraph creation, calculation of cyclomatic complexity (CC), independent path determination, and finally test case testing. The test results showed that the system procedures were more complex with the risk level being at medium levels and all the tested pathways in the form of test cases were appropriate. This research contribution is expected to help software developers in testing whitebox by using flowchart system.

Keywords: *white box testing, basis path analysis, flowgraph, independent path, expert system.*

Abstrak

Dalam membangun suatu perangkat lunak atau *software* sistem, pengujian adalah hal penting yang harus dilakukan. Pengujian tahap awal dapat dilakukan melalui pengujian unit (*unit testing*). Pengujian ini menggunakan pengujian *whitebox* (*White box testing*) dengan metode *Basis Path Analysis*. Metode ini melakukan pengujian pada struktur program dalam bentuk ***flowchart* program** yang kemudian dikonversi ke dalam bentuk *flowgraph*, namun bagaimana apabila perangkat lunak tersebut tersusun atas begitu banyak kode program yang terbagi atas banyak file yang tidak memungkinkan untuk dibuatkan *flowgraph*. Penelitian ini bertujuan untuk menerapkan metode *Basis Path Analysis* dalam pengujian *white box* pada sistem pakar diagnosa penyakit jantung dengan menggunakan ***flowchart* sistem** untuk mengatasi permasalahan di atas. Metode *Basis Path Analysis* terdiri dari empat tahapan, yaitu: pembuatan *flowgraph*, perhitungan *cyclomatic complexity* (CC), penentuan *independent path*, dan terakhir pengujian *test case*. Hasil pengujian menunjukkan bahwa prosedur sistem yang diteliti sudah lebih kompleks dengan tingkat risiko berada pada level sedang (*medium*) dan semua jalur yang diuji (*independent path*) dalam bentuk kasus uji (*test case*) sudah sesuai. Kontribusi penelitian ini diharapkan dapat membantu para *software developer* dalam melakukan pengujian *whitebox* dengan menggunakan *flowchart* sistem.

Kata kunci: *white box testing, basis path analysis, flowgraph, independent path*, sistem pakar.

1. Pendahuluan

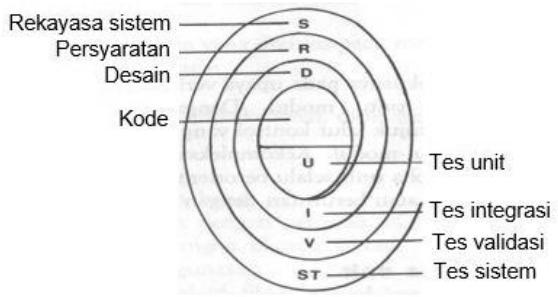
Setiap perangkat lunak atau *software* yang dibangun, perlu adanya pengujian yang dilakukan. Salah satunya pada perangkat lunak aplikasi Sistem Pakar Diagnosa Penyakit Jantung. Pengujian sangat penting dilakukan baik dalam bentuk Verifikasi dan Validasi (V&V).

Pengujian verifikasi dan validasi adalah suatu proses yang menjamin bahwa hasil dari setiap tahapan (fase) pada siklus hidup pengembangan perangkat lunak sudah memenuhi spesifikasi dan kebutuhan pengguna. Proses verifkasi dan validasi perangkat lunak sangat penting dalam siklus hidup pengembangan sistem karena terdiri dari jenis-jenis pengujian yang menjamin kualitas sebuah produk perangkat lunak [1].

Pengujian perangkat lunak dapat menggunakan *Blackbox testing*, *Whitebox testing* dan *Usability testing* [2]. ***Black box testing*** adalah menguji Perangkat Lunak (PL) dari segi spesifkasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian black-box dilakukan dengan membuat kasus uji (*test case*) yang bersifat mencoba semua fungsi dengan menggunakan PL apakah sudah sesuai dengan spesifikasi yang dibutuhkan. *Test case* yang dibuat harus dengan kasus benar dan kasus salah. ***White box testing*** adalah menguji PL dari segi desain dan kode program dengan cara memeriksa lojik dari kode program, contoh:

menguji alur pengulangan (*looping*) pada logika pemrograman.

Menurut Pressman [3], strategi pengujian perangkat lunak dapat digambarkan dalam konteks spiral seperti terlihat pada Gambar 1. **Tes unit** *(unit testing)* dimulai dari pusaran spiral yang berpusat pada masing-masing satuan perangkat lunak pada saat diimplementasikan ke dalam **kode program/**sumber. Pengujian berjalan dengan bergerak keluar sepanjang spiral dari tes unit ke **tes integrasi** (*integration testing)* dengan fokusnya adalah **desain** perangkat lunak. Setelah itu, pengujian berjalan keluar lagi, menemukan **tes validasi** (*validation testing)*. Di dalam tes validasi, persyaratan (*requirement*) yang dibangun dari analisis kebutuhan/persyaratan perangkat lunak divalidasi terhadap perangkat lunak yang telah dibuat. Selanjutnya, pengujian akan sampai pada tahap terakhir akan diuji secara keseluruhan.



Gambar 1. Stratetegi pengujian perangkat lunak dalam konteks spiral

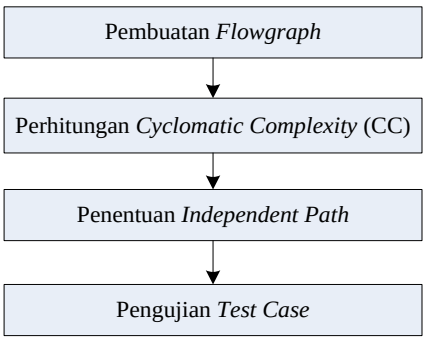
Berdasarkan strategi pengujian di atas, maka dibuatlah suatu tes unit (*Unit Testing*) terhadap perangkat lunak apliksi sistem pakar menggunakan pengujian kotak putih atau disebut dengan pengujian white box (*White Box Testing*) dengan menerapkan metode *Basis Path Analysis.* Metode ini merupakan salah satu teknik pengujian struktur kontrol yang menjamin bahwa semua statemen dalam setiap jalur independen program dieksekusi minimal 1 kali.

Metode *Basis Path* melakukan pengujian pada struktur program dalam bentuk ***flowchart* program** yang kemudian dikonversi ke dalam bentuk *flowgraph,* namun bagaimana apabila perangkat lunak tersebut tersusun atas begitu banyak kode program yang terbagi atas banyak file yang tidak memungkinkan untuk dibuatkan *flowgraph*.

Penelitian ini bertujuan untuk menerapkan metode *Basis Path Analysis* dengan menggunakan ***flowchart* sistem** untuk mengatasi permasalahan banyaknya file kode program yang mengakibatkan sulitnya pembuatan *flowgraph*. Dengan adanya penelitian ini diharapkan dapat membantu para *software developer* dalam melakukan pengujian *whitebox* dengan menggunakan *flowchart* sistem.

2. Metode Penelitian

Perangkat lunak atau software sistem yang diujikan dalam penelaitian ini adalah Sistem Pakar Diagnosa Penyakit Jantung [4]. Pengujian unit dalam penelitian ini menggunakan pengujian *white box* dengan metode *Basis Path Analysis* yang memiliki beberapa tahapan yang dapat dilihat pada Gambar 2.



Gambar 2. Tahapan *Basis Path Analysis*

2.1. Pembuatan *Flowgraph*

Flowgraph menggambarkan alur dari logika program. *Flowgraph* dapat dibuat dari kode program atau *flowchart* sistem. Notasi pada *flowgraph* digambarkan dengan lingkaran (*node*) dan anak panah (*edge*). *Node* menunjukkan pernyataan prosedural, sedangkan *Edge* menunjukkan alur perjalanan logika program. Pembuatan *flowgraph* untuk aplikasi sistem pakar diagnosa penyakit jantung ini lebih sesuai dibuat berdasarkan *flowchart* sistem karena aplikasi yang dibuat berbasis MVC (*Model*, *View*, *Controller*) sehingga ada begitu banyak kode program yang terbagi dalam banyak file yang tidak memungkinkan untuk dibuatkan *flowgraph* dari kode program yang ada. *Flowchart* sistem dapat dilihat pada Gambar 3a dan Gambar 3b.

2.2. Perhitungan CC (*Cyclomatic Complexity*)

Cyclomatic Complexity memberikan pengukuran kuantitatif terhadap kompleksitas logika sebuah program. Pada teknik pengujian *whitebox* yang dilakukan dengan metode *Basis Path*, nilai yang diperoleh dari perhitungan CC akan menentukan berapa jumlah jalur independent dalam suatu basis set program. Dari banyaknya jalur independent inilah yang menentukan jumlah kasus uji (*test case*) minimal yang harus dilakukan terhadap jalur independent tersebut untuk memastikan semua persyaratan yang ada pada jalur independent telah dieksekusi minimal satu kali. Rumus penghitungan CC adalah sebagai berikut:

$$V(G) = E - N + 2 \text{ atau } V(G) = P + 1 \qquad (1)$$

dimana:
V(G) = *Cyclomatic Complexity*
E = jumlah edge pada *flowgraph*
N = jumlah node pada *flowgraph*
P = jumlah predicate node pada *flowgraph*

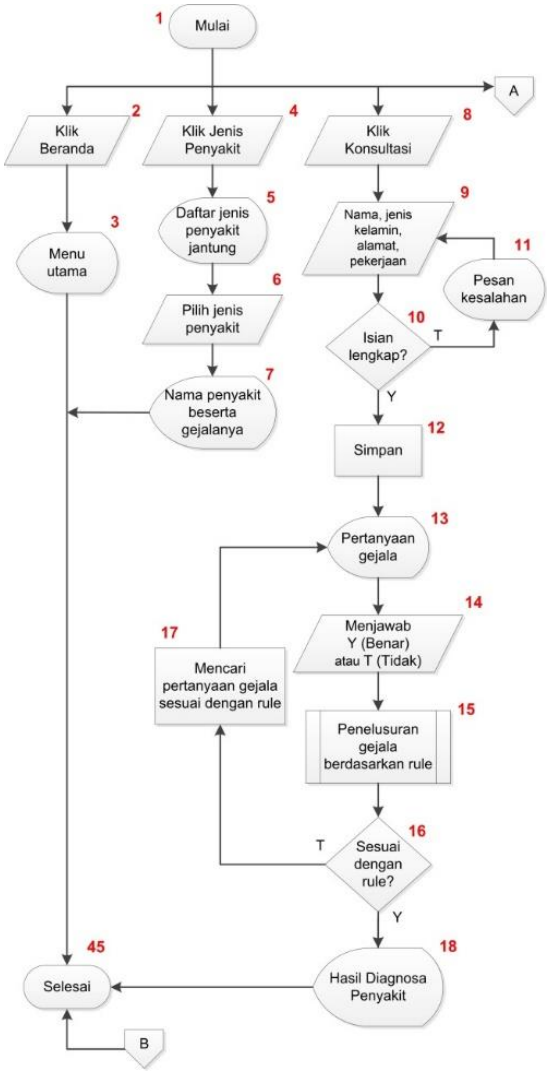
Jika jumlah baris kode programnya sedikit maka CC dapat dengan mudah dihitung secara manual. Jika programnya kompleks maka dapat menggunakan alat uji otomatis, dan jika programnya sudah banyak dan kompleks maka akan melibatkan lebih banyak *Control Flow Graph* (CFG), itu artinya pembuatan *flowgraph*-nya akan menjadi lebih rumit [5].

2.3. Penentuan *Independent Path*

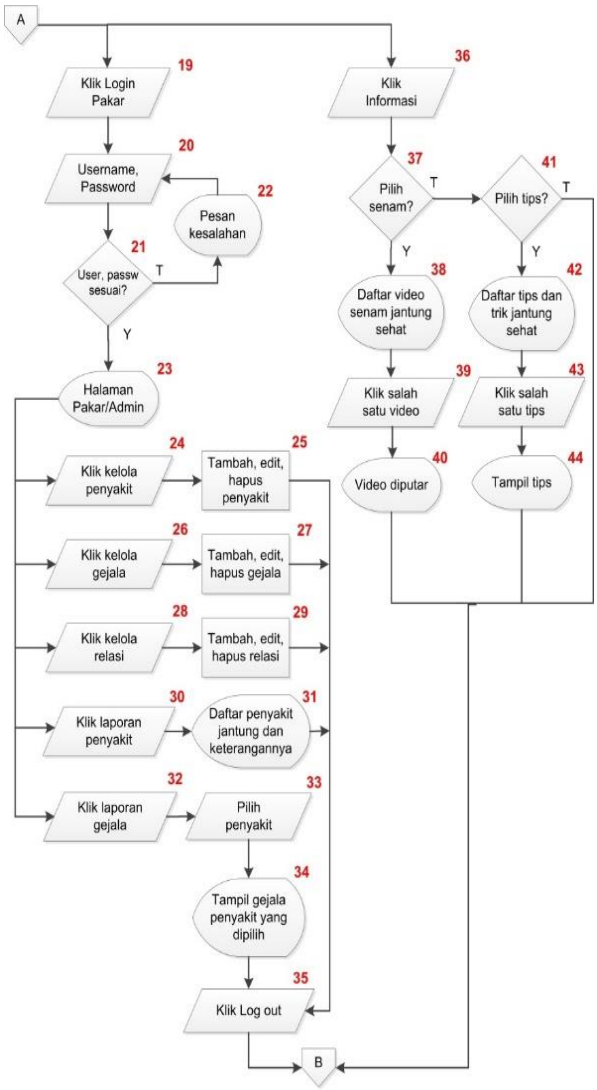
Independent path merupakan jalur pada program yang menghubungkan node awal dengan node akhir. Independent path minimal melewati sebuah Edge baru dengan alur yang belum pernah dilalui sebelumnya.

2.4. Pengujian *Test Case*

Jumlah kasus uji (*test case*) yang dibuat minimal sebanyak jumlah *independent path* yang diperoleh dari perhitungan CC. *Test case* ini dibuat untuk menguji atau mengeksekusi semua jalur independent (alur logika) yang sudah dibuat.



Gambar 3a. Flowchart Konsultasi Pakar



Gambar 3b. Flowchart Login Pakar

3. Hasil dan Pembahasan

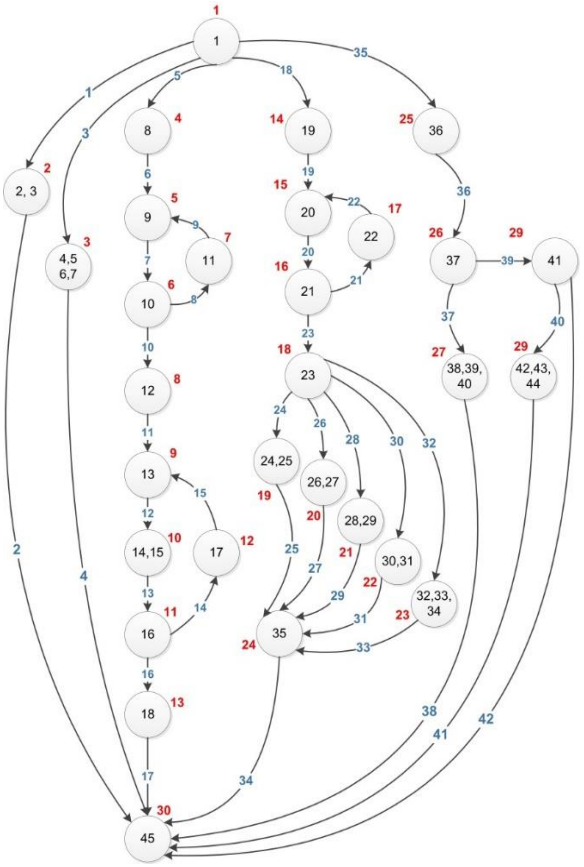
Pengujian *whitebox* dengan metode *Basis Path* membuat *flowchart* sistem terlebih dahulu sebelum membuat *flowgraph.*

3.1. Pembuatan *Flowgraph*

Berdasarkan *flowchart* pada Gambar 3a dan Gambar 3b, dibuatlah *flowgraph* seperti terlihat pada Gambar 4.

Gambar *flowgraph* diberi angka-angka berwarna merah dan biru untuk memudahkan dalam perhitungan Node dan Edge. Warna merah menunjukkan Node dan warna biru menunjukkan Edge. Dari *flowgraph* tersebut dapat dilihat jumlah Node yang ditunjukkan oleh urutan angka berwarna merah ada sebanyak 30 buah, sedangkan jumlah Edge yang ditunjukkan oleh urutan angka berwarna biru ada sebanyak 42 buah. Keterangan untuk setiap nomor yang terdapat dalam Node dapat dilihat pada Tabel 1. Total terdapat 45 nomor pada Node yang sesuai dengan jumlah angka (berwarna

merah) yang terdapat pada *flowchart* sistem dalam Gambar 3 (Gambar 3a dan Gambar 3b).



Gambar 4. *Flowchart* Aplikasi Sistem Pakar

3.2. Penghitungan CC (*Cyclomatic Complexity*)

Setelah *flowgraph* dibuat, kemudian dilakukan penghitungan CC (*Cyclomatic Complexity*) dengan menggunakan rumus: V(G) = E – N + 2. Dari *flowgraph* diperoleh jumlah Edge, E = 42 dan Node, N = 30, maka:

$$V(G) = (42 – 30) + 2 = 12 + 2 = 14$$

Jadi, nilai V(G) atau CC (*Cyclomatic Complexity*) untuk *flowgraph* pada Gambar 4 adalah **14,** ini artinya terdapat **14 jalur independent (*independent path*) yang harus diuji**.

Apabila perolehan nilai CC di atas dihubungkan dengan Tabel 1 maka nilai 14 menunjukkan bahwa **prosedur sistem sudah lebih kompleks** dengan **level kompleksitas tinggi**, **kemampuan untuk diuji** serta **biaya** dan **usaha** yang diperlukan masih berada pada **level menengah (*medium*)**, ini berarti **tingkat risikonya** masih tergolong **sedang (menengah).**

Tabel 1. Keterangan untuk setiap nomor yang terdapat dalam Node

No	Keterangan
1	Mulai
2	Pengguna mengklik menu Beranda
3	Tampil halaman menu utama
4	Pengguna mengklik menu Jenis Penyakit
5	Tampil daftar semua jenis penyakit jantung
6	Pilih salah satu jenis penyakit jantung untuk melihat gejalanya
7	Tampil nama penyakit jantung yang dipilih beserta gejalanya
8	Pengguna mengklik menu Konsultasi
9	Input nama pasien, jenis kelamin, alamat dan pekerjaan
10	Isian data pasien lengkap?
11	Jika isian tidak lengkap maka muncul pesan kesalahan
12	Jika isian lengkap maka data pasien disimpan
13	Tampil pertanyaan gejala penyakit yang dialami pasien
14	Pasien menjawab dengan memilih benar (ya) atau salah (tidak)
15	Sistem menelusuri jawaban (gejala) berdasarkan aturan (*rule*)
16	Jawaban pasien sudah sesuai dengan rule?
17	Jika tidak maka sistem akan mencari pertanyaan sesuai dengan rule
18	Jika ya maka tampil data pasien beserta hasil diagnosa penyakit
19	Pengguna mengklik menu Login Pakar
20	Input username dan password
21	Usernama dan password sudah cocok?
22	Jika tidak maka muncul pesan kesalahan
23	Jika ya maka tampil halaman utama pakar atau menu utama admin
24	Pengguna mengklik menu Kelola Penyakit
25	Tampilan halaman tambah, edit, hapus penyakit
26	Pengguna mengklik menu Kelola Gejala
27	Tampil halaman tambah, edit, hapus gejala
28	Pengguna mengklik menu Kelola Relasi
29	Tampil halaman tambah, edit, hapus relasi
30	Pengguna mengklik menu Laporan Penyakit
31	Tampil daftar penyakit jantung dan keterangannya
32	Pengguna mengklik menu Laporan Gejala
33	Pilih salah satu jenis penyakit
34	Tampil gejala dari jenis penyakit yang dipilih
35	Pengguna mengklik Menu Logout
36	Pengguna mengklik menu Informasi
37	Pengguna memilih fitur Senam?
38	Jika ya maka tampil daftar video senam jantung sehat
39	Pengguna mengklik salah satu video
40	Video senam diputar
41	Jika tidak maka dilanjutkan apakah memilih fitur Tips?
42	Jika ya maka tampil daftar tips dan trik memiliki jantung sehat
43	Pengguna mengklik salah satu tips
44	Tampil tips dan trik yang dipilih
45	Selesai

Tabel 2. Hubungan *Cyclomatic Complexity* dengan tingkat risiko

CC	*Procedure*	*Level of Comple-xity*	*Testabi-lity*	*Cost and Effort (RiskLevel)*
1-10	*A well structured and stable procedure*	*Less*	*High*	*Less*
11-20	*A more complex procedure*	*High*	*Medium*	*Medium*
21-40	*A complex procedure, alarming*	*Very High*	*Low*	*High*
>40	*An error-prone, extremely troublesome, untestable procedure*	-	*Not at all Testable*	*Very High*

3.3 Penentuan *Independent Path*

Setelah diketahui nilai CC (*Cyclomatic Complexity)*, selanjutnya **dibuatkan *independent path*-nya sebanyak 14 jalur** yang menghubungkan node awal dan node akhir seperti yang terlihat pada Tabel 3.

Dari Tabel 1 terlihat bahwa node awal dan node akhir sudah terhubung dan semua edge sudah dilalui pada setiap jalur *independent*, ini berarti sudah memenuhi persyaratan dari sebuah *independent path*.

Tabel 3. Empat Belas (14) Jalur *Independent*

No.	*Independent Path*
1	1 – 2,3 – 45
2	1 – 4,5,6,7 – 45
3	1 – 8 – 9 – 10 – 12 – 13 – 14,15 – 16 – 18 – 45
4	1 – 8 – 9 – 10 – 11 – 9 – 10 – 12 – 13 – 14,15 – 16 – 18 – 45
5	1 – 8 – 9 – 10 – 12 – 13 – 14,15 – 16 – 17 – 13 – 14,15 – 16 – 18 – 45
6	1 – 19 – 20 – 21 – 23 – 24,25 – 35 – 45
7	1 – 19 – 20 – 21 – 22 – 20 – 21 – 23 – 24,25 – 35 – 45
8	1 – 19 – 20 – 21 – 23 – 26,27 – 35 – 45
9	1 – 19 – 20 – 21 – 23 – 28,29 – 35 – 45
10	1 – 19 – 20 – 21 – 23 – 30,31 – 35 – 45
11	1 – 19 – 20 – 21 – 23 – 32,33,34 – 35 – 45
12	1 – 36 – 37 – 38,39,40 – 45
13	1 – 36 – 37 – 41 – 42,43,44 – 45
14	1 – 36 – 37 – 41 – 45

3.4 Pengujian *Test Case*

Selanjutnya *independent path* ini **diimplementasikan ke dalam bentuk kasus uji atau *test case*.** Caranya yaitu dengan mengganti angka-angka pada setiap jalur dengan keterangan yang terdapat dalam Node (Tabel 1), ini berarti terdapat **14 jumlah *test case* yang diuji.**

4. Kesimpulan

Hasil pengujian unit pada aplikasi sistem pakar diagnosa penyakit jantung menggunakan metode *Basis Path Analysis* menunjukkan bahwa prosedur sistem sudah lebih kompleks dengan tingkat risiko berada pada level sedang (*medium*) dan semua jalur yang diuji (*independent path*) dalam bentuk kasus uji (*test case*) sudah sesuai.

Daftar Rujukan

[1] Alsayed, A.O., Bilgrami, A.L. 2017. *Improving Software Quality Management: testing, Review, Inspection and Walkthrough*. International Journal of Latest Research in Science and Technology, Vol. 6, Issue 1, pn. 1-12, January-Februari 2017. ISSN (online): 2278-5299. Sumber: https://www.researchgate.net/profile/Anwar_Bilgrami2/publication/315393787_improving_software_quality_management_testing_review_inspection_and_walkthrough/links/59f19ddb458515bfd07fce1c/improving-software-quality-management-testing-review-inspection-and-walkthrough.pdf, tanggal akses: 25 Agustus 2020

[2] Munaiseche, C.P.C., Liando, O.E.S. 2016. *Evaluation of Expert System Application Based On Usability Aspects*. Jurnal IOP Conference Series: Materials Science and Engineering, Vol. 128, 24 Mei 2016. Sumber: http://iopscience.iop.org/article/10.1088/1757-899X/128/1/012001/pdf , tanggal akses: 26 Agustus 2020

[3] Pressman, R.S. 2012. Rekayasa Perangkat Lunak (Pendekatan Praktisi) Edisi 7: Buku 1. ISBN: 9789792931044, Juni 2012. Andi, Yogyakarta

[4] Munaiseche, C.P.C., Rantung, V.P., Bawiling, N.S., Manggopa, H.K. 2019. *A Knowledge Based System for Diagnosing Heart Diseases*. Journal of Physics: Conference Series, Volume 1402, Issue 2, 16 Desember 2019. Sumber: https://iopscience.iop.org/article/10.1088/1742-6596/1402/2/022090/pdf, tanggal akses: 26 Agustus 2020

[5] Madhavi, D. 2016. *A White Box Testing Technique in Softwre Testing: Basis Path Testing*. Journal for Research, Vol. 01, Issue 04, June 2016. ISSN: 2395-7549. Sumber: http://www.journal4research.org/articles/J4RV2I4007.pdf , tanggal akses: 26 Agustus 2020