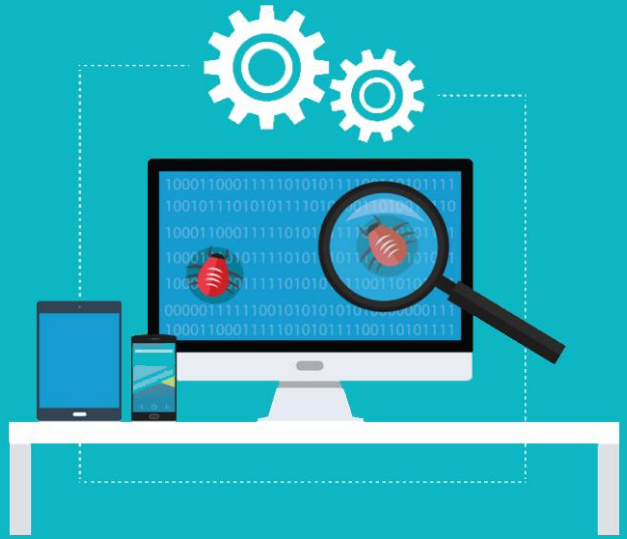


Instrumentasi *Source Code* Secara Otomatis untuk *Basis Path Testing*

Raden Asri Ramadhina Fitriani
G64154007

Pembimbing
Irman Hermadi, Skom, MS, PhD

Mengapa Pengujian Penting ?

- Untuk memastikan perangkat lunak melakukan apa yang seharusnya dilakukan

- Untuk meningkatkan kepercayaan bahwa perangkat lunak berfungsi/berperilaku dengan benar

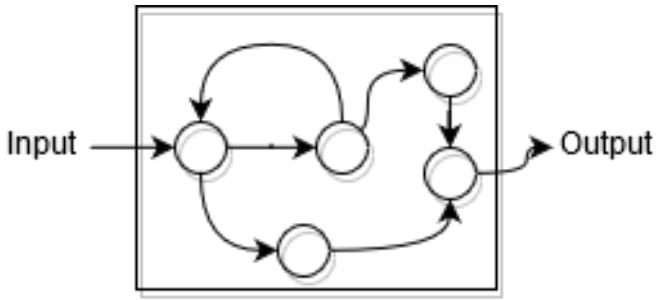
Jenis Pengujian

BLACK BOX TESTING

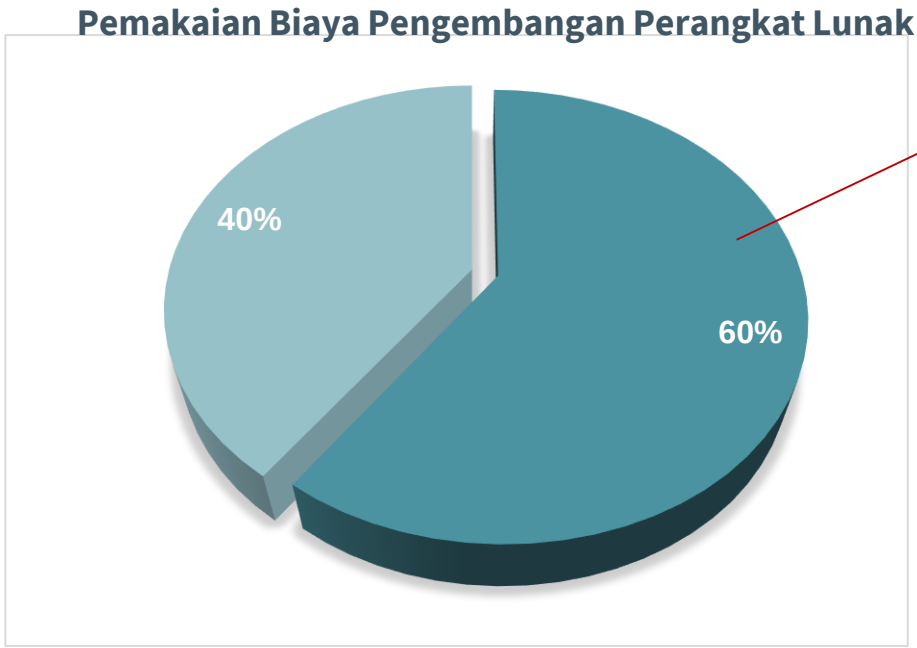
Melakukan pengujian dengan cara memeriksa fungsionalitas apakah output sudah sesuai dengan yang ditentukan

WHITE BOX TESTING

Melakukan pengujian dengan cara memeriksa struktur internal dan alur logika (proses) sebuah perangkat lunak.



Latar Belakang



Digunakan untuk pengujian (Kumar dan Mishra 2016)

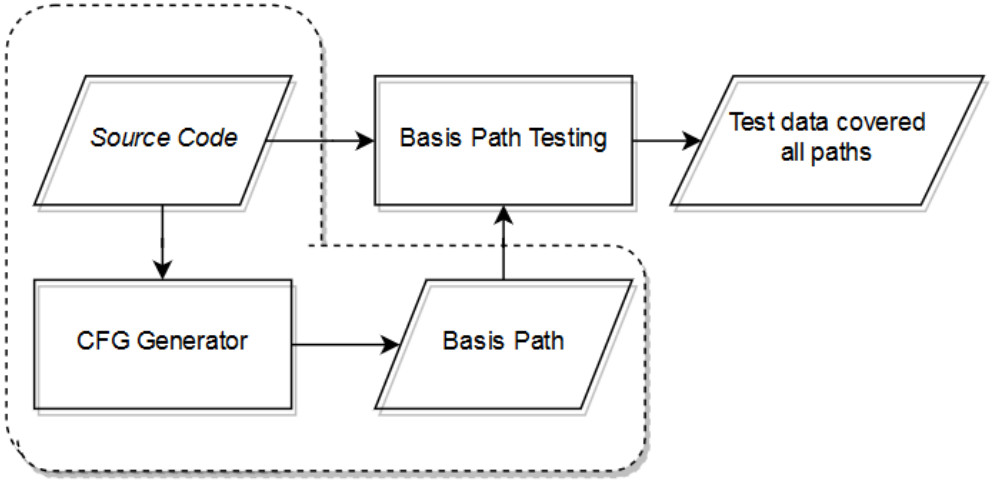
Latar Belakang

Hermadi (2015) melakukan penelitian untuk membangkitkan data uji untuk pengujian jalur menggunakan algoritma genetika.

Pembangkitan Control flow Graph (CFG) dan instrumentasi masih dilakukan secara manual

Otomasi proses tersebut dapat

- Mempercepat
- Mengurangi kerawanan akan kesalahan.



Rumusan Masalah

Bagaimana membangun sebuah aplikasi untuk melakukan pembangkitan jalur dasar dan instrumentasi secara otomatis untuk pengujian jalur

Tujuan

Membangun sebuah aplikasi yang dapat digunakan untuk membangkitkan CFG dan melakukan instrumentasi secara otomatis.

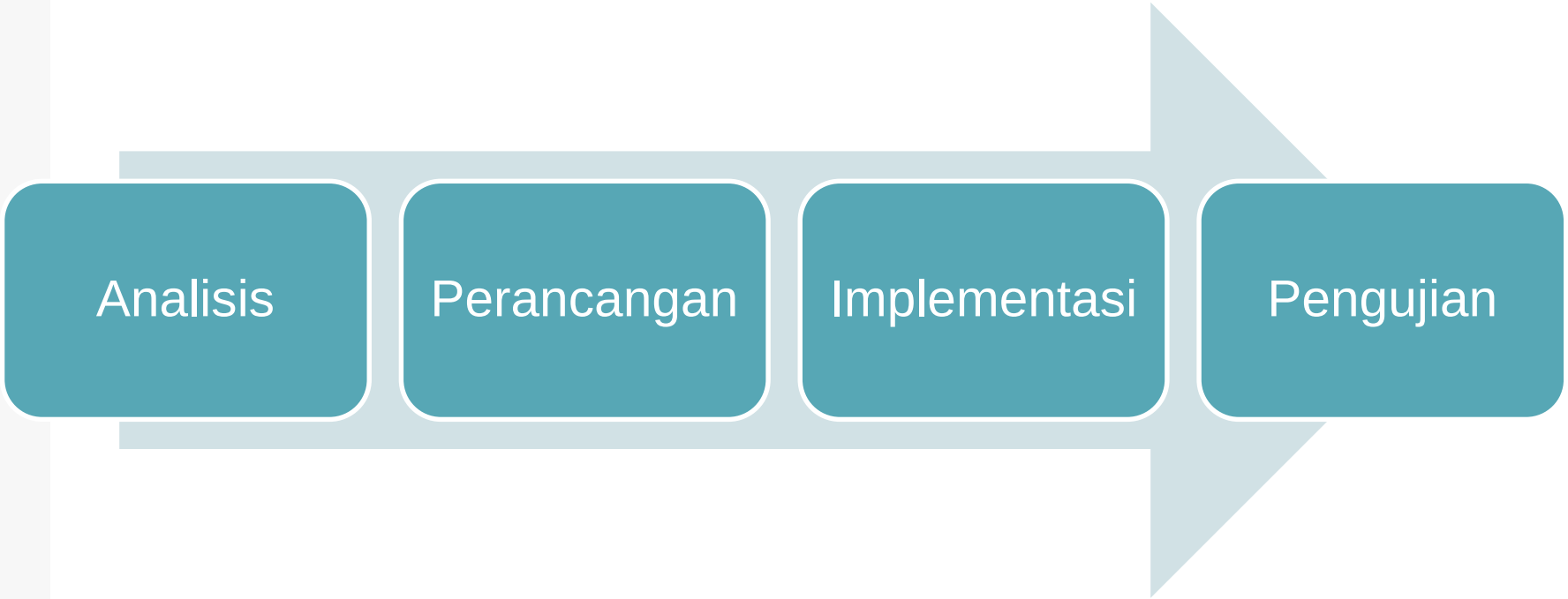
Ruang Lingkup

- Bahasa pemrograman yang diakomodasi adalah bahasa Matlab
- Model diagram yang dibangkitkan adalah CFG

Manfaat

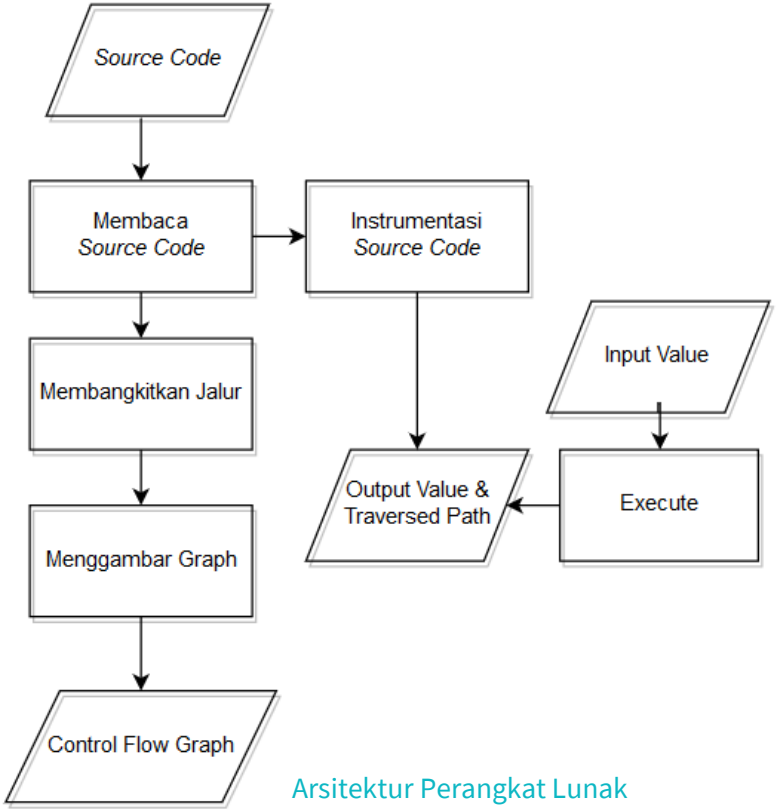
Menyisipkan tag-tag sebagai instrumentasi program ke dalam *source code* secara otomatis	• Mempercepat
Membangkitkan jalur-jalur dasar	• Pembangkitan data uji
Membangkitkan diagram CFG	• Memahami struktur dan alur dari suatu program → *re-engineering* perangkat lunak

METODE PENELITIAN



METODE PENELITIAN
Analisis

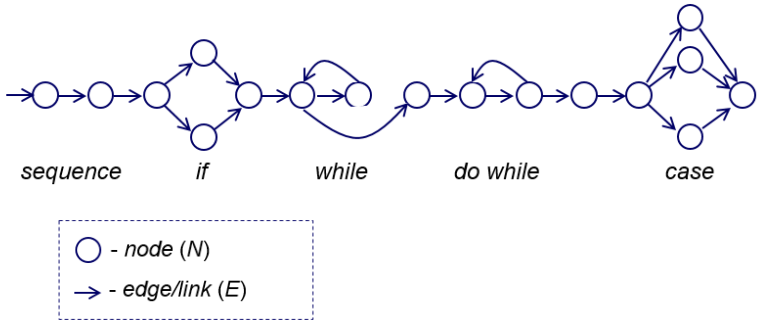
METODE PENELITIAN
Perancangan



Arsitektur Perangkat Lunak

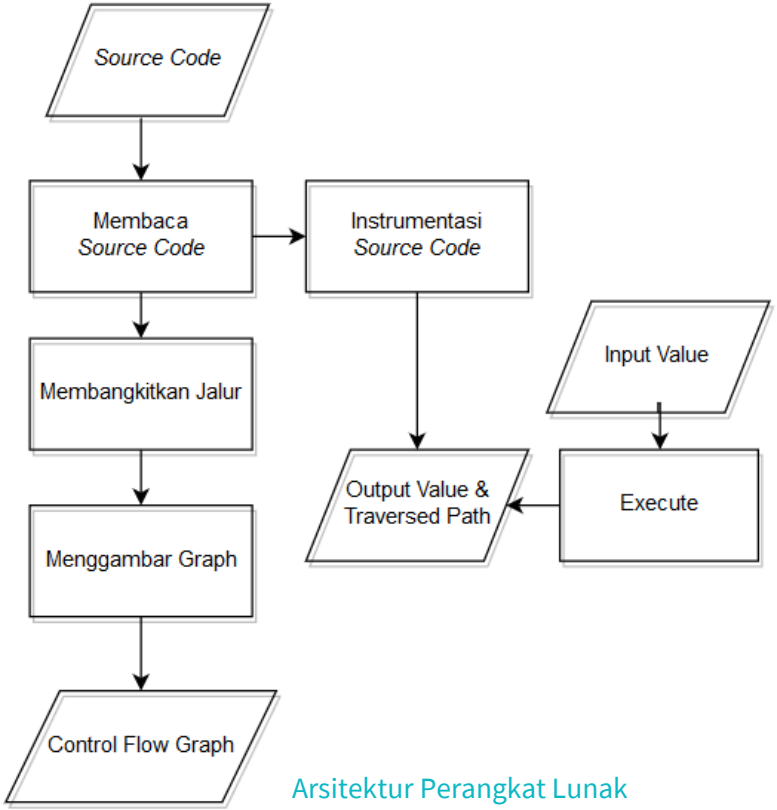
- Instrumentasi : proses menyisipkan sebuah penanda (tag) di awal atau di akhir setiap blok kode seperti awal setiap fungsi, sebelum atau sesudah kondisi terpenuhi atau tidak.

- *Control Flow Graph* (CFG) : graph berarah yang merepresentasikan aliran dari sebuah program. Terdiri dari *nodes* dan *edges*.



Notasi Control Flow Graph (CFG)

METODE PENELITIAN
Perancangan



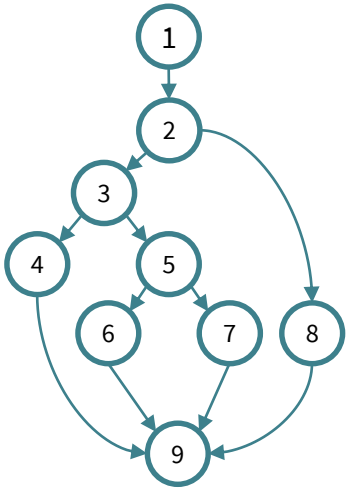
- *Cyclomatic complexity :* ukuran yang menunjukan jumlah jalur dasar dan tingkat kompleksitas dari suatu program.

- Perhitungan *cyclomatic complexity:*

$$V(G) = E - N + 2$$

Contoh *Control Flow Graph* (CFG)

```
        function [type] = triangle(sideLengths)
(1)     A = sideLengths(1); % First side
        B = sideLengths(2); % Second side
        C = sideLengths(3); % Third side
(2)     if ((A+B > C) && (B+C > A) && (C+A > B)) % Branch # 1
(3)         if ((A ~= B) && (B ~= C) && (C ~= A)) % Branch # 2
(4)             type = 'Scalene';
(5)         else
                if (((A == B) && (B ~= C)) || ((B == C) && (C ~= A)) || ...
                      ((C == A) && (A ~= B))) % Branch # 3
(6)                 type = 'Isosceles';
                else
(7)                 type = 'Equilateral';
                end
            end
        else
(8)         type = 'Not a triangle';
(9)     end
```



E=11
N=9

V(*G*) = *11* - *9* + 2
= 4

Kemungkinan jalur yang terbentuk:
P1 = 1-2-3-4-9
P2 = 1-2-3-5-6-9
P3 = 1-2-3-5-7-9
P4 = 1-2-8-9

Contoh Instrumentasi

```
function [traversedPath, type] = triangle(sideLengths)
traversedPath = [];
A = sideLengths(1); % First side
B = sideLengths(2); % Second side
C = sideLengths(3); % Third side
% instrument Branch # 1
traversedPath = [traversedPath 1 ];
if ((A+B > C) && (B+C > A) && (C+A > B)) % Branch # 1
    % instrument Branch # 2
    traversedPath = [traversedPath 2 ];
    if ((A ~= B) && (B ~= C) && (C ~= A)) % Branch # 2
        type = 'Scalene';
    else
        % instrument Branch # 3
        traversedPath = [traversedPath 3 ];
        if (((A == B) && (B ~= C)) || ((B == C) && (C ~= A)) || ...
              ((C == A) && (A ~= B))) % Branch # 3
            type = 'Isosceles';
        else
            type = 'Equilateral';
        end
    end
else
    type = 'Not a triangle';
end
```

METODE PENELITIAN

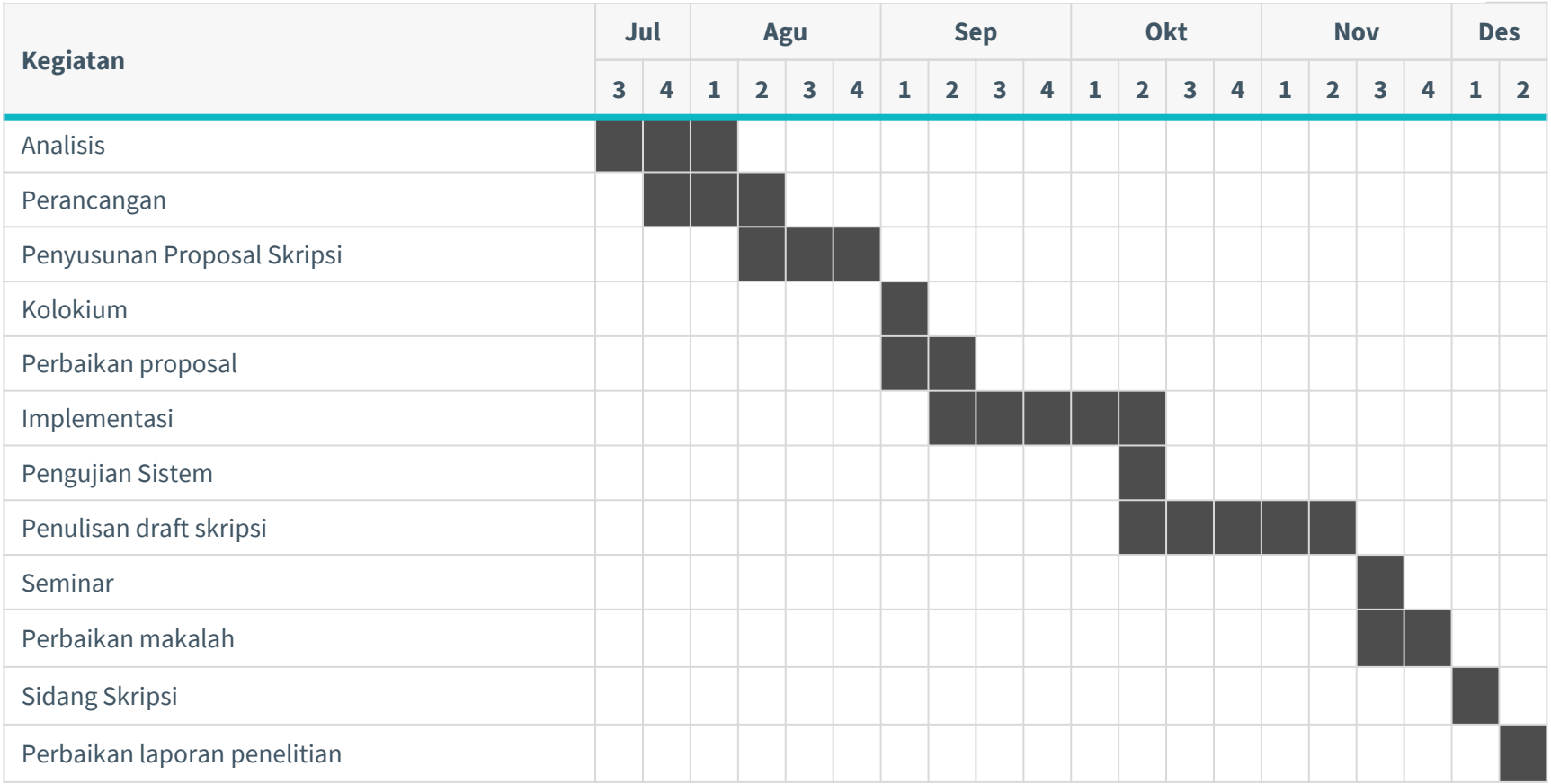
Implementasi

- Berbasis web
- Menggunakan bahasa pemrograman Java
- Editor menggunakan IDE Eclipse
- CFG akan divisualisasikan dengan menggunakan *library* Graphviz

Pengujian

Membandingkan hasil yang dikeluarkan oleh sistem dengan pembangkitan secara manual dari segi waktu eksekusi.

Rencana Jadwal Penelitian



Kegiatan	Jul		Agu				Sep				Okt				Nov				Des	
	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
Analisis	■	■	■																	
Perancangan		■	■	■																
Penyusunan Proposal Skripsi				■	■	■														
Kolokium							■													
Perbaikan proposal							■	■												
Implementasi								■	■	■	■	■								
Pengujian Sistem												■								
Penulisan draft skripsi												■	■	■	■	■				
Seminar																	■			
Perbaikan makalah																	■	■		
Sidang Skripsi																			■	
Perbaikan laporan penelitian																				■

DAFTAR PUSTAKA

Arkeman, Y, Herdiyeni, Y, Hermadi, I, dan Laxmi, GF. 2014. *Algoritma Genetika Tujuan Jamak (MultiObjective Genetic Algorithm*. IPB Press.

Basu, A. 2015. *Software Quality Assurance, Testing and Metrics*. PHI Learning Privat Limited. [Internet]. [Diunduh tanggal 14/8/2017]. Dapat diunduh dari: https://books.google.co.id/books.

Hermadi, I. 2015. "Path Testing using Genetic Algorithm". Disertasi. University of New South Wales.

Khan, M E. 2011. "Different Approaches to White Box Testing Technique for Finding Errors" dalam: *International Journal of Software Engineering and Its Applications* 5, p. 3. [Internet]. [Diunduh tanggal 21/8/2017]. Dapat diunduh dari: http://www.sersc.org/journals/IJSEIA/vol5_ no32011/1.pdf.

Kumar, D dan Mishra, K K. 2016. "The Impacts of Test Automation on Software's Cost, Quality and Time to Market" dalam: *Procedia Computer Science* 79, pp. 8–15. [Internet]. [Diunduh tanggal 20/8/2017]. Dapat diunduh dari: http://www.sciencedirect.com/science/article/pii/S1877050916001277.

Myers, G J, Sandler, C, dan Badgett, T. 2012. *The Art of Software Testing*. John Willey dan Sons, Inc, Hoboken, New York. [Internet]. [Diunduh tanggal 14/8/2017]. Dapat diunduh dari: https://books.google.co.id/books.

Tikir, M M dan Hollingsworth, J K. 2011. "Efficient Instrumentation for Code Coverage Testing" dalam: *International Journal of Software Engineering and Its Applications*. [Internet]. [Diunduh tanggal 21/8/2017]. Dapat diunduh dari: https : / / www .researchgate.net/publication/2835608_ Efficient_ Instrumentation_ for_ Code_ Coverage_ Testing

Watson, A H dan McCabe, T J. 1996. "Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric)" dalam: *NIST Special Publication*. [Internet]. [Diunduh tanggal 14/8/2017]. Dapat diunduh dari: http : / / www . mccabe . com /pdf/mccabe-nist235r.pdf.

TERIMA KASIH

Raden Asri Ramadhina Fitriani
radenasrirf@gmail.com