

**ILMU KOMPUTER**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**INSTITUT PERTANIAN BOGOR**

**BOGOR**

**2017**

**RADEN ASRI RAMADHINA FITRIANI**

**INSTRUMENTASI KODE PROGRAM SECARA OTOMATIS UNTUK *BASIS PATH TESTING***

**PERNYATAAN MENGENAI SKRIPSI DAN  
SUMBER INFORMASI SERTA PELIMPAHAN HAK CIPTA**

Dengan ini saya menyatakan bahwa skripsi berjudul Fermentasi Instrumentasi Kode Program Secara Otomatis Untuk *Basis Path Testing* adalah benar karya saya dengan arahan dari komisi pembimbing dan belum diajukan dalam bentuk apa pun kepada perguruan tinggi mana pun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka di bagian akhir disertasi ini.

Dengan ini saya melimpahkan hak cipta dari karya tulis saya kepada Institut Pertanian Bogor.

Bogor, Desember 2017

*Raden Asri Ramadhina Fitriani*

NIM G64154007

**ABSTRAK**

RADEN ASRI RAMADHINA FITRIANI. Instrumentasi Kode Program Secara Otomatis Untuk *Basis Path Testing*. Dibimbing oleh IRMAN HERMADI.

Pengujian adalah serangkaian proses yang dirancang untuk memastikan sebuah perangkat lunak melakukan apa yang seharusnya dilakukan dan bertujuan untuk menemukan kesalahan pada perangkat lunak. Path testing merupakan salah satu metode pengujian struktural yang menggunakan source code dari program untuk menemukan semua jalur yang mungkin dapat dilalui program dan memastikan semua jalur dijalankan setidaknya satu kali. Untuk menguji perangkat lunak yang kompleks secara keseluruhan akan memakan waktu yang lama dan membutuhkan sumber daya manusia yang banyak. Pada penelitian ini, akan dibangun sebuah sistem untuk membangkitkan kemungkinan jalur-jalur dari sebuah program yang dapat dijadikan dasar untuk membangkitkan data uji agar data uji yang digunakan untuk pengujian dapat mewakili semua kemungkinan. Untuk memastikan semua jalur dijalankan setidaknya 1 kali ketika diberikan masukan data uji, maka sistem ini juga akan melakukan instrumentasi kode program secara otomatis. Program yang akan diuji dalam penelitian ini adalah program yang dibangun dengan menggunakan bahasan C. Dalam pengembangannya, aplikasi ini akan dibangun dengan menggunakan bahasa pemrogramana C# dan library Graphvis untuk memvisualisasikan Control Flow Graph.

Kata kunci: Path Testing; Control Flow Graph, Instrumentasi

**ABSTRACT**

RADEN ASRI RAMADHINA FITRIANI. Automatic Source Code Instrumentation for Basis Path Testing. Supervised by IRMAN HERMADI.

Keywords:

Skripsi  
sebagai salah satu syarat untuk memperoleh gelar  
Sarjana Peternakan   
pada  
Departemen Ilmu Komputer

**RADEN ASRI RAMADHINA FITRIANI**

**DEPARTEMEN ILMU KOMPUTER**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**INSTITUT PERTANIAN BOGOR**

**BOGOR**

**2017**

**INSTRUMENTASI *SOURCE CODE* SECARA OTOMATIS UNTUK *BASIS PATH TESTING***

Judul Skripsi : Instrumentasi Kode Program Secara Otomatis untuk *Basis Path Testing*

Nama : Raden Asri Ramadhina Fitriani

NIM : G64154007

Disetujui oleh

|  |
| --- |
| Irman Hermadi, Skom, MS, PhD  Pembimbing I |

Diketahui oleh

Dr Ir Agus Buono, MSi MKom

Ketua Departemen

Tanggal Lulus: (tanggal penandatanganan skripsi oleh ketua departemen)

**PRAKATA**

Bogor, Januari 2018

*Raden Asri Ramadhina Fitriani*

**DAFTAR ISI**

DAFTAR TABEL vi

DAFTAR GAMBAR vi

DAFTAR LAMPIRAN vi

[PENDAHULUAN 1](#_Toc503105556)

[Latar Belakang 1](#_Toc503105557)

[Perumusan Masalah 1](#_Toc503105558)

[Tujuan Penelitian 2](#_Toc503105559)

[Manfaat Penelitian 2](#_Toc503105560)

[Ruang Lingkup Penelitian 2](#_Toc503105561)

[METODE 2](#_Toc503105562)

[Analisis 2](#_Toc503105563)

[Perancangan 3](#_Toc503105564)

[Implementasi 6](#_Toc503105565)

[Testing 6](#_Toc503105566)

[HASIL DAN PEMBAHASAN 7](#_Toc503105567)

[Analisis 7](#_Toc503105568)

[Perancangan 7](#_Toc503105569)

[Implementasi 9](#_Toc503105570)

[Testing 13](#_Toc503105571)

[SIMPULAN DAN SARAN 14](#_Toc503105572)

[Simpulan 14](#_Toc503105573)

[Saran 14](#_Toc503105574)

[DAFTAR PUSTAKA 15](#_Toc503105575)

LAMPIRAN 13

RIWAYAT HIDUP 15

**DAFTAR TABEL**

**No table of figures entries found.**

**DAFTAR GAMBAR**

[Tahapan Penelitian 2](#_Toc503105576)

[Contoh bahasa dot 5](#_Toc503105577)

[Contoh gambar yang memiliki lebar kurang dari 10 cm 15](file:///D:\_SKRIPSWEET\DRAFT%20SKRIPSI.docx#_Toc503105578)

**DAFTAR LAMPIRAN**

No table of figures entries found.

**No table of figures entries found.**

# PENDAHULUAN

## Latar Belakang

Pengujian adalah serangkaian proses yang dirancang untuk memastikan sebuah perangkat lunak melakukan apa yang seharusnya dilakukan. Proses ini bertujuan untuk menemukan kesalahan pada perangkat lunak. Saat pengujian, bisa saja tidak ditemukan kesalahan pada Hasil pengujian. Hal ini dapat terjadi karena perangkat lunak yang sudah berkualitas tinggi atau karena proses pengujiannya berkualitas rendah. (Myers et al. 2012).

Teknik pengujian secara umum dibagi menjadi 2 kategori diantaranya black box testing dan white box testing. Black box testing bertujuan untuk memeriksa fungsional dari perangkat lunak apakah output sudah sesuai dengan yang ditentukan. Sedangkan White box testing atau biasa disebut dengan pengujian struktural  
merupakan pemeriksaan struktur dan alur logika suatu proses. Basis Path testing merupakan salah satu metode pengujian struktural yang menggunakan source code dari program untuk menemukan semua jalur yang mungkin dapat dilalui program dan dapat digunakan untuk merancang data uji. Metode ini memastikan semua kemungkinan jalur dijalankan setidaknya satu kali (Basu 2015). Untuk melakukan monitoring jalur mana yang diambil oleh sebuah masukan pada saat eksekusi program, maka diperlukan penanda yang dapat memberikan informasi cabang mana yang dilalui. Proses menyisipkan tanda tersebut disebut instrumentasi. Biasanya tanda tersebut disisipkan tepat sebelum sebuah percabangan (Tikir dan Hollingsworth 2011).

Idealnya, pengujian dilakukan untuk semua kemungkinan dari perangkat lunak. Tetapi untuk menguji perangkat lunak yang kompleks secara keseluruhan akan memakan waktu yang lama dan membutuhkan sumber daya manusia yang banyak. Kumar dan Mishra (2016) mengatakan bahwa pengujian perangkat lunak menggunakan hampir 60% dari total biaya pengembangan perangkat lunak. Jika proses pengujian perangkat lunak dapat dilakukan secara otomatis, maka hal ini dapat mengurangi biaya pengembangan secara signifikan.

Hermadi (2015) melakukan penelitian untuk membangkitkan data uji untuk path testing menggunakan algoritma genetika. Dalam penelitian tersebut, Hermadi membangkitkan Control flow Graph (CFG) dan instrumentasi masih secara manual sehingga membutuhkan banyak waktu dan rawan akan kesalahan ketika program  
sudah semakin besar. Sehingga mengotomasi hal tersebut dapat membuat path testing menjadi lebih cepat dan mengurangi kerawanan akan kesalahan.

Pada penelitian ini, akan dibangun sebuah perangkat lunak untuk membangkitkan kemungkinan jalur dari sebuah program. Jalur-jalur ini dapat dijadikan dasar untuk membangkitkan data uji agar data uji yang digunakan untuk pengujian dapat mewakili semua kemungkinan. Untuk memonitor jalur mana yang dilalui ketika diberikan masukan data uji, maka sistem ini juga akan melakukan penyisipan tag-tag sebagai instrumentasi ke dalam kode program secara otomatis

## Perumusan Masalah

Berdasarkan latar belakang di atas dapat dirumuskan masalahnya adalah bagaimana membangun sebuah aplikasi untuk melakukan instrumentasi secara otomatis untuk pengujian jalur dan re-engineering perangkat lunak.

## Tujuan Penelitian

Penelitian ini bertujuan untuk membangun sebuah aplikasi yang dapat digunakan untuk membangkitkan CFG dan melakukan instrumentasi secara otomatis.

## Manfaat Penelitian

Hasil penelitian diharapkan dapat membantu pengembang dan penguji aplikasi untuk:

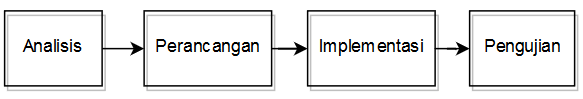
1. Menyisipkan tag-tag sebagai instrumentasi program ke dalam source code secara otomatis sehingga proses tersebut dapat dilakukan dengan lebih cepat.
2. Membangkitkan jalur-jalur dasar yang dapat digunakan sebagai dasar untuk pembangkitan data uji.
3. Membangkitkan diagram CFG yang dapat memudahkan pengembang dalam memahami struktur dan alur dari suatu program yang dapat dimanfaatkan ketika akan melakukan reengineering perangkat lunak

## Ruang Lingkup Penelitian

Bahasa pemrograman yang diakomodasi adalah Matlab dan model diagram yang dibangkitkan adalah CFG.

# METODE

Penelitian yang dilakukan terbagi menjadi beberapa tahapan proses. Gambar 1 menunjukan tahapan proses tersebut.



Gambar 1 Tahapan penelitian

## **Analisis**

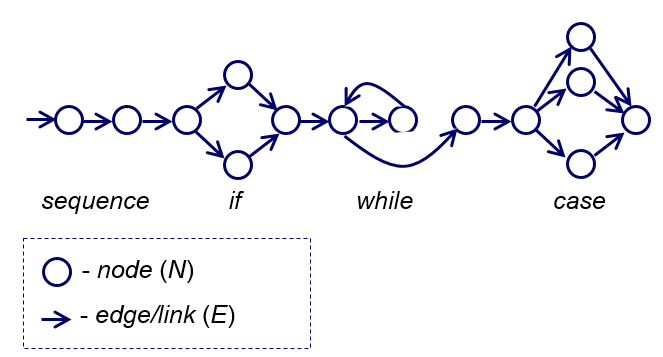
Pada tahap ini dimulai dari membaca literatur terkait dan mengumpulkan beberapa contoh program yang akan digunakan dalam penelitian. Contoh program yang digunakan dalam penelitian ini didapatkan dari penelitian yang dilakukan oleh Hermadi (2015).

*Basis path testing* merupakan salah satu metode pengujian struktural yang menggunakan kode sumber atau program (*source code*)untuk menemukan semua jalur yang mungkin dapat dilalui program dan dapat digunakan untuk merancang data uji. Metode ini memastikan semua kemungkinan jalur dijalankan setidaknya satu kali (Basu 2015).

Metode ini terbagi menjadi 4 tahapan, yaitu:

1. Menggambarkan jalur dalam bentuk *Control Flow Graph* (CFG)
2. Menghitung *cyclomatic complexity*
3. Memilih satu set jalur dasar
4. Membangkitkan data uji untuk setiap jalur dasar

*Control Flow Graph* (CFG) adalah graph berarah yang merepresentasikan aliran dari sebuah program. Setiap CFG terdiri dari *nodes* dan *edges*. *Nodes* merepresentasikan *statement* atau *expressions*. Sedangkan *edges* merepresentasikan transfer kontrol antar *nodes* (Watson dan McCabe 1996). Notasi dari CFG dapat dilihat pada Gambar 2.



Gambar 2 Notasi Control Flow Graph (CFG)

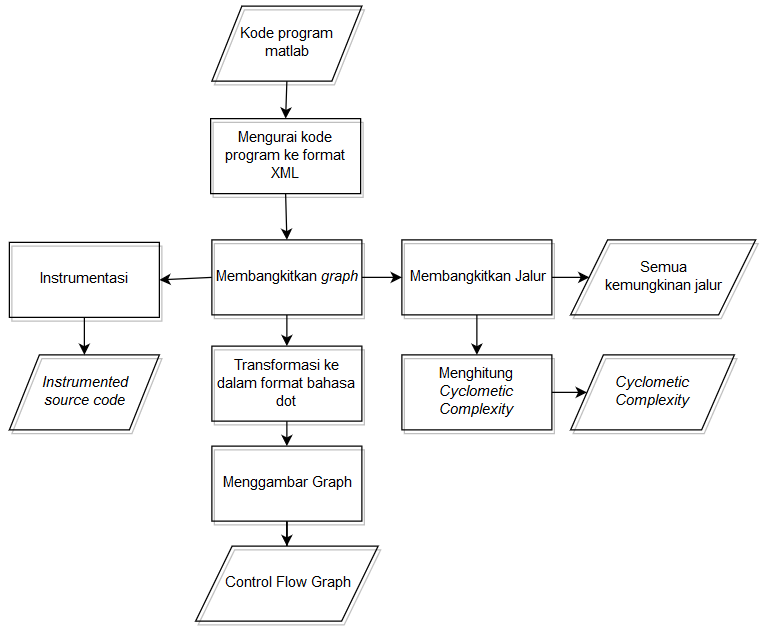
*Cyclomatic complexity* merupakan suatu sistem pengukuran yang ditemukan oleh Watson dan McCabe untuk menentukan banyaknya *independent path* dan menunjukan tingkat kompleksitas dari suatu program. *Independent path* adalah jalur yang melintas dalam program yang sekurang-kurangnya terdapat kondisi baru. Perhitungan *Cyclomatic Complexity* dapat dilihat pada persamaan berikut:

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

Dimana, E menunjukkan jumlah *edges* dan N menunjukkan jumlah *nodes*.

## **Perancangan**

Pada tahap ini ditentukan bagaimana perangkat lunak akan dibangun. Ilustrasi arsitektur sistem dapat dilihat pada Gambar 3.



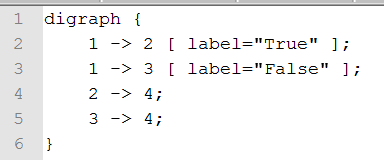
Kode program matlabakan dibaca sebagai inputan. Lalu kode program tersebut diurai menjadi file dengan format XML menggunakan *library* MATLAB-PARSER yang dibuat oleh Suffos (2015). Matlab memiliki empat struktur kontrol, yaitu IF-ELSE-END, SWITCH-CASE, FOR, dan WHILE. Contoh hasil penguraian dari struktur control perintah IF-ELSE-END dapat dilihat pada Gambar …

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | **🡪 node 1**  **🡪 node 4**  **🡪 node 3**  **🡪 node 2**  <If Line="5" Column="2" Text="if">  <If.IfPart>  <IfPart Line="5" Column="2" Text="if">  <IfPart.Expression>  </IfPart.Expression>  <IfPart.Statements>  </IfPart.Statements>  </IfPart>  </If.IfPart>  <If.ElsePart>  <ElsePart Line="30" Column="2" Text="else">  <ElsePart.Statements>  </ElsePart.Statements>  </ElsePart>  </If.ElsePart>  <If.Terminator>  </If.Terminator>  </If> |

*Extensible Markup Language* (XML) adalah bahasa yang dapat mendeskripsikan sebuah dokumen. XML memiliki banyak bagian yang tidak memiliki struktur yang pasti. XML terdiri atas dua bagian utama, yaitu elemen dan atribut. Elemen yang dapat disebut sebagai *node* merupakan bagian penting yang dapat menggambarkan struktur dari XML. Sedangkan atribut merupakan bagian yang dapat digunakan sebegai informasi tambahan dari setiap elemen (Hartwell 2017).

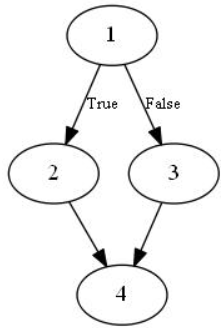
Setiap dalam tag file XML tersebut akan ditelusuri satu persatu yang termasuk struktur kontrol di dalam bahasa matlab. Sehingga terbentuklah sebuah objek graph yang terdiri dari sekumpulan node dan edge. Graph tersebut ditransformasikan ke dalam bentuk format bahasa permrograman dot. Bahasa dot adalah bahasa yang digunakan untuk mengambar *graph* berarah. Bahasa ini dapat mendeskripsikan 3 macam objek, yaitu *graph, nodes,* dan *edges* (Ganser *et all* 2015).

Setelah file dengan format bahasa dot terbentuk, CFG akan divisualisasikan dengan menggunakan library Graphviz. Graphviz merupakan perangkat lunak *open source* untuk visualisasi grafik. Gambar 2 menunjukkan contoh bahasa dot yang terdiri dari 4 *nodes* dan 4 *edges.* Gambar ….. menunjukkan hasil dari *graph*  yang dibentuk dari perinntah dot pada Gambar … yang di visualisasikan menggunakan Graphviz.



Gambar 3 Contoh bahasa dot

Contoh bahasa dot



Contoh hasil visualisasi bahasa dot menggunakan *library* graphviz

Setelah jalur terbentuk, dilakukan juga proses instrumentasi. Instrumentasi merupakan sebuah proses menyisipkan sebuah penanda (tag) di awal atau di akhir setiap blok kode seperti awal setiap perintah, sebelum atau sesudah kondisi terpenuhi atau tidak. Dalam pengujian path testing, penanda ini dapat digunakan untuk memonitor jalur yang dilalui program ketika dijalankan dengan masukan data uji tertentu (Arkeman et al. 2014).

## Implementasi

Tahapan ini adalah melakukan implementasi dari tahap sebelumnya ke dalam bentuk aplikasi web. Aplikasi ini akan dibangun dengan menggunakan bahasa pemrograman C# dan menggunakan IDE Microsoft Visual Studio Ultimate 2013. *Library* yang dibutuhkan adalah *library* MATLAB-PARSER yang dibuat oleh Suffos (2015) untuk mengurai kode program ke dalam bentuk XML dan *Graphviz* untuk menvisualisasi jalur ke dalam bentuk *graph* berarah.

## Testing

Tahapan ini adalah melakukan evaluasi dari tahapan implementasi. Evaluasi dilakukan dengan membandingkan hasil yang dikeluarkan oleh sistem dengan pembangkitan secara manual dari segi waktu eksekusi.

|  |  |  |
| --- | --- | --- |
| **No** | **Program Uji** |  |
|  |
| 1 | triangleAhmed2008 |  |
| 2 | minimaxiAhmed2008 |  |
| 3 | insertionAhmed2008 |  |
| 4 | bisectionAhmed2008 |  |
| 5 | binaryAhmed2008 |  |
| 6 | bubbleAhmed2008 |  |
| 7 | gcdAhmed2008 |  |
| 8 | remainderAhmed2008 |  |
| 9 | mmTriangleAhmed2008 |  |
| 10 | triangleMansour2004 |  |
| 11 | expintRapps1985 |  |
| 12 | quotientGallagher1997 |  |
| 13 | tritypeBueno2002 |  |
| 14 | expintBueno2002 |  |
| 15 | quotientBueno2002 |  |
| 16 | strcompBueno2002 |  |
| 17 | floatcompBueno2002 |  |
| 18 | findBueno2002 |  |
| 19 | bubbleGong2011 |  |
| 20 | flexGong2011 |  |
| 21 | spaceGong2011 |  |

# HASIL DAN PEMBAHASAN

## **Analisis**

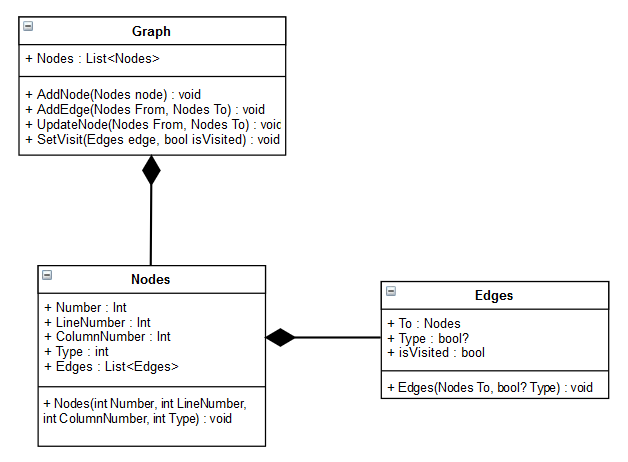
Sebelumnya sudah terdapat beberapa program yang dapat membangkitkan CFG, tetapi b

Data yang digunakan dalam penelitian ini didapatkan dari penelitian yang dilakukan oleh Hermadi (2015). Terdapat 21 contoh program

## **Perancangan**

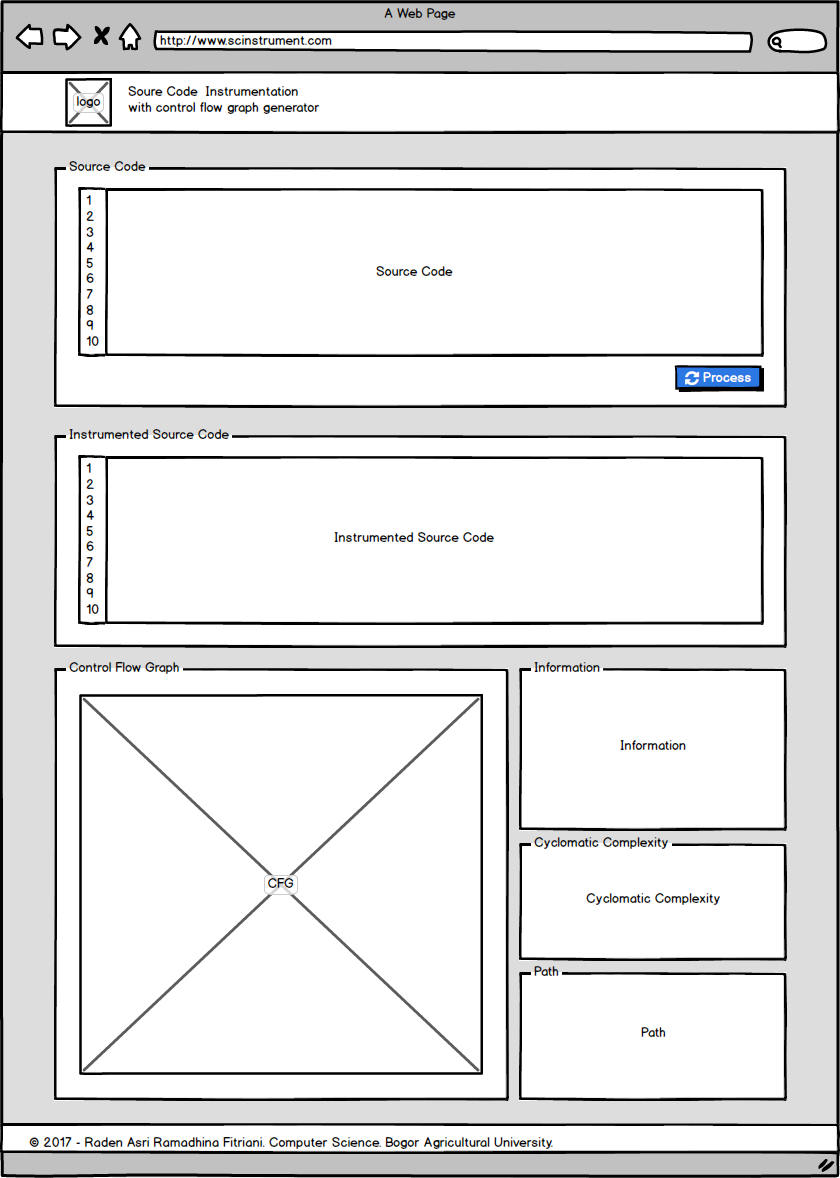
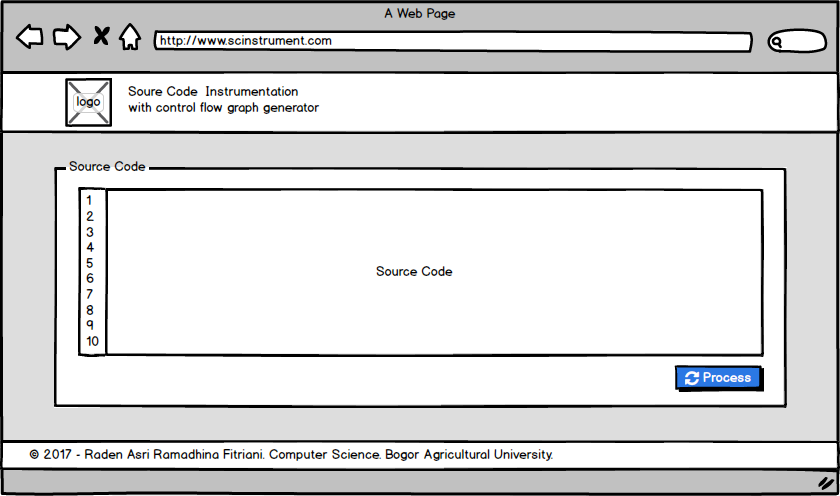
**Perancangan *Class Diagram***

*Class diagram* dibangun untuk menggambarkan struktur sistem dari segi pendefinisian *class-class*. Perancangan *Class diagram* dapat dilihat pada Gambar …….



**Perancangan Antarmuka**

Perancangan antarmuka meliputi perancangan antarmuka *form* untuk pengguna memasukkan kode program yang akan di proses dan antarmuka hasil dari proses yang telah dilakukan. Perancangan antarmuka *form* awal yang digunakan untuk memasukkan kode program yang akan di proses dapat dilihat pada Gambar …. Dan perancangan antarmuka hasil dari proses yang telah dilakukan dapat dilihat pada Gambar …



## Implementasi

Aplikasi dibangun dengan menggunakan bahasa pemrograman C# dan menggunakan IDE Microsoft Visual Studio Ultimate 2013.

Sebagai contoh, kode program yang digunakan adalah tA2008. Pada kode tA2008 terdapat perintah IF-THEN-ELSE bersarang sebanyak tiga kali.

**Kode Program**

Berikut merupakan kode program tA2008, yaitu untuk mencari jenis dari segitiga jika diketahui panjang dari setiap sisinya.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | function type = triangle(sideLengths)  A = sideLengths(1); % First side  B = sideLengths(2); % Second side  C = sideLengths(3); % Third side  if ((A+B > C) && (B+C > A) && (C+A > B))  if ((A ~= B) && (B ~= C) && (C ~= A))  type = 'Scalene';  else  if (((A == B) && (B ~= C)) || ((B == C) && (C ~= A)) || ((C == A) && (A ~= B)))  type = 'Isosceles';  else  type = 'Equilateral';  end  end  else  type = 'Not a triangle';  end |

**Mengurai Kode Progam ke Format XML**

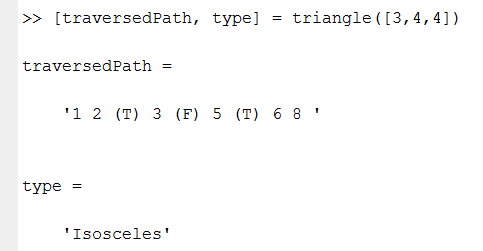
Penguraian kode program matlab dilakukan dengan menggunakan *library*  MATLAB-PARSER. Kode program yang diinputkan harus sudah dipastikan dapat dijalankan jika di *compile.* Ketika terdapat kesalahan pada kode program, *library*  ini akan mengembalikan pesan *error.* Gambar …. Menunjukkan potongan hasil penguraian kode program ke dalam format XML. Potongan kode XML yang terlihat pada Gambar menunjukkan hasil penguraian dari kode program pada Gambar …. baris ke enam dan tujuh.



**Membangkitkan Graph**

Salah satu cara untuk membaca dan menulis dokumen XML pada *framework* .NET dan C# yaitu dengan menggunakan kelas *XMLDocument* yang terdapat dalam *namespace System.XML*. Setiap elemen XML yang merupakan struktur kontrol akan menjadi *nodes* baru di dalam kelas *graph*.

**Instrumentasi**



**Membangkitkan Jalur**

**Menghitung *Cyclometic Complexity***

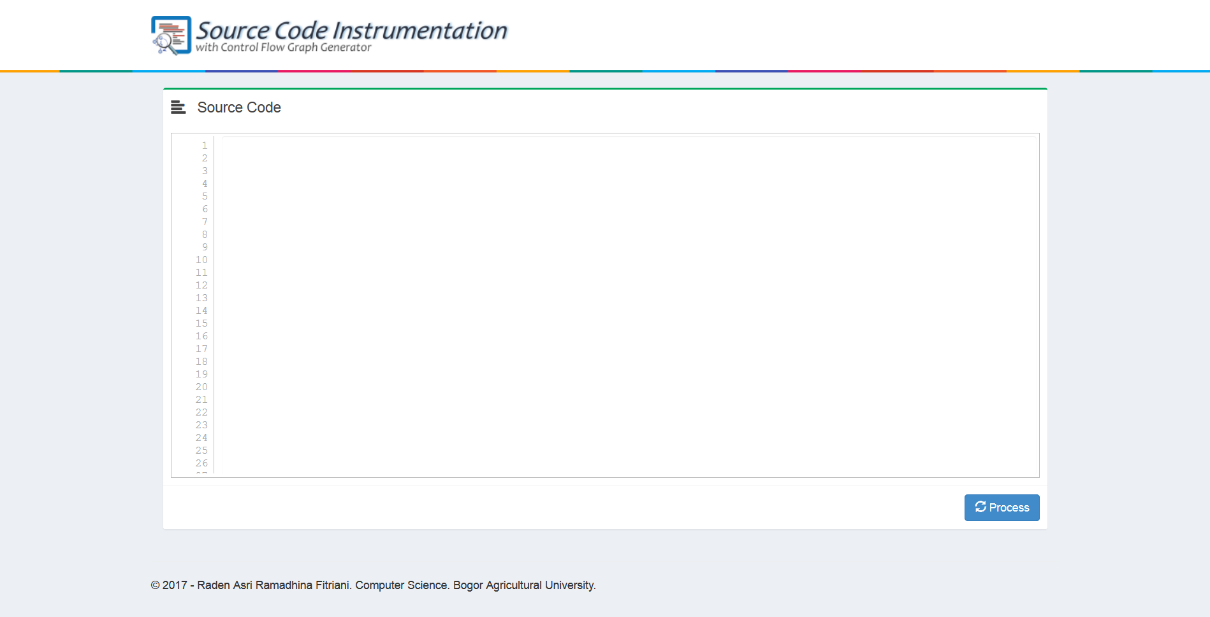
**Transformasi ke Dalam Format Bahasa Dot**

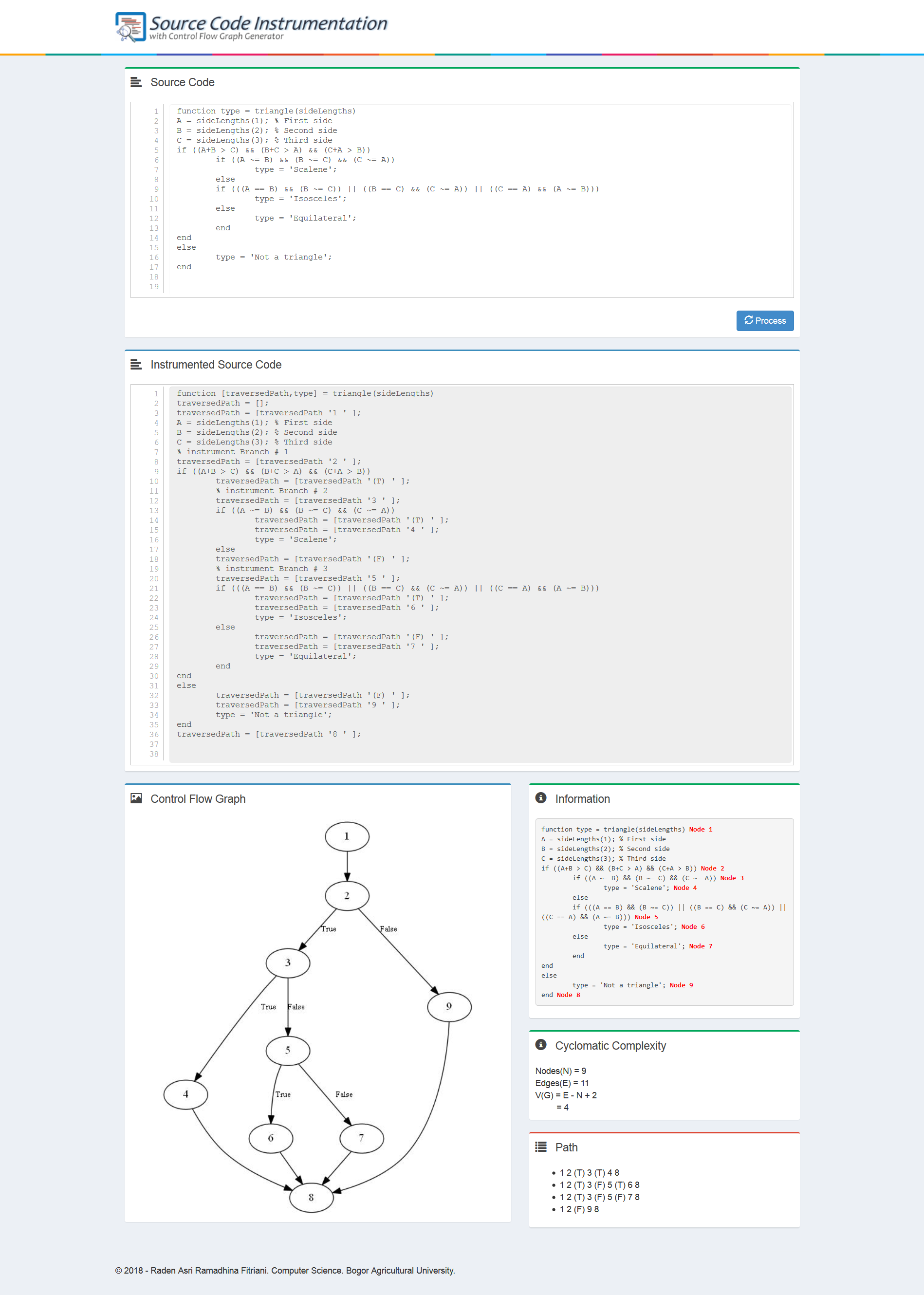
**Memvisualisasikan *Graph***

Setiap elemen akan dibaca satu persatu, an

**Hasil**

Halaman awal dari aplikasi dapat dilihat pada Gambar … dimana hanya terdapat satu text area untuk memasukkan kode program yang akan di proses. Implementasi hasil pembangkitan dapat dilihat padda Gambar …





## Testing

Tahapan ini adalah melakukan evaluasi dari tahapan implementasi. Evaluasi dilakukan dengan membandingkan hasil yang dikeluarkan oleh sistem dengan pembangkitan secara manual dari segi waktu eksekusi.

https://www.mccormick.northwestern.edu/documents/students/undergraduate/introduction-to-matlab

# SIMPULAN DAN SARAN

## Simpulan

Simpulan merupakan jawaban dari tujuan yang sudah ditentukan dan tidak dimaksudkan sebagai ringkasan hasil. Dalam Simpulan, penulis harus dan hanya menjawab masalah dan tujuan penelitian yang telah dirumuskan pada Pendahuluan. Simpulan merupakan generalisasi dari hasil penelitian dan argumentasi penulis, atau pernyataan singkat yang merupakan hakikat dari bab Hasil dan Pembahasan atau hasil pengujian berbagai hipotesis yang berkaitan.

Simpulan merupakan hasil penelitian yang boleh jadi telah dikemukakan dalam perumusan masalah dan telah diberi jawaban sementara berupa hipotesis. Dalam menulis simpulan, penulis harus membedakan dugaan, temuan, dan simpulan hasil studi. Pernyataan simpulan harus dilakukan secara cermat dan hati-hati. Penyampaian simpulan ini dapat dilakukan sebanyak 3 kali, yakni dalam Pembahasan, Simpulan, dan Abstrak sehingga diperlukan kecermatan untuk menyajikannya dengan ungkapan yang berbeda-beda.

## Saran

Saran seyogianya mengarah ke implikasi atau tindakan lanjutan yang harus dilakukan sehubungan dengan temuan atau simpulan penulis. Saran yang dikemukakan harus berkaitan dengan pelaksanaan atau hasil penelitian. Dengan demikian saran ini mengemukakan hal-hal yang perlu diteliti lebih lanjut terutama untuk memperbaiki kelemahan atau kekurangan dalam penelitian yang dilakukan atau perbaikan asumsi yang diambil sehingga didapatkan hasil yang lebih baik. Jadi, saran tersebut harus diuraikan secara spesifik. Jangan menyarankan hal-hal yang tidak dianalisis dan dibahas dalam penelitian serta terkesan menggurui atau memuaskan keinginan peneliti. Untuk penelitian yang berkaitan dengan permasalahan kebijakan, tidak perlu menyarankan kebijakan yang tidak berkaitan dengan hasil penelitian.



Gambar 4 Contoh gambar yang memiliki lebar   
 kurang dari 10 cm

# DAFTAR PUSTAKA

Bente AD, Rico-Hesse R. 2006. Model of dengue virus infection. *Drug Discov Today Dis Models*. 3(1):97-103. doi: 10.1016/j.ddmod. 2006.03.014.

Bernardo L, Izquierdo A, Prado I, Rosario D, Alvarez M, Santana E, Castro J, Martinez J, Rodriguez R, Morier L *et al*. 2008. Primary and secondary infections of *Macaca fascicularis* monkey with Asian and American genotypes of dengue virus 2. *Clin Vaccine Immunol*. 15(3): 439-446. doi: 10.1128/CVI.00208-07.

Kochel TJ, Watts DM, Gonzalo AS, Ewing DF, Porter KR, Russell KL. 2005. Cross-serotype neutralization of dengue virus in *Aotus nancyme* monkeys. *J Infect Dis*. 191(6):1000-1004. doi:10.1086/427511.

Onlamoon N, Noisakran S, Hsiao HM, Duncan A, Villinger F, Ansari AA, Perng GC. 2010. Dengue virus-induced hemorrhage in a nonhuman primate model. *Blood*. 115(9):1823-1834. doi:10.1182/blood-2009-09-241990.

[WHO] World Health Organization. 2009. Dengue and dengue haemorrhagic fever [internet]. [diacu 2009 Mei 6]. Tersedia dari: http://www.who.int /mediacentre/ factsheets/ fs117/en/ index.html.

Lampiran 1

**RIWAYAT HIDUP**

Dalam riwayat hidup dijelaskan tempat dan tanggal kelahiran mahasiswa, putra dan putri ke berapa dari orang tua, nama kedua orang tua atau wali. Untuk skripsi, tuliskan pendidikan penulis seja\k sekolah menengah hingga terdaftar sebagai mahasiswa IPB. Kegiatan penulis di luar akademik yang menunjang pendidikan juga baik dicantumkan, terutama prestasi akademik yang pernah diraih selama masa kemahasiswaan. Uraian tentang riwayat hidup tidak lebih dari satu halaman.