

# SKRIPSI

**PENGUJIAN BERBASIS *BEHAVIOUR SPECIFICATION***



**Muhammad Dipo Putra Wandara**

**NPM: 2016730091**

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS  
UNIVERSITAS KATOLIK PARAHYANGAN  
2019**



# UNDERGRADUATE THESIS

**PENGUJIAN BERBASIS *BEHAVIOUR SPECIFICATION***



**Muhammad Dipo Putra Wandara**

**NPM: 2016730091**

**DEPARTMENT OF INFORMATICS  
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES  
PARAHYANGAN CATHOLIC UNIVERSITY  
2019**



# LEMBAR PENGESAHAN

## PENGUJIAN BERBASIS *BEHAVIOUR SPECIFICATION*

Muhammad Dipo Putra Wandara

NPM: 2016730091

Bandung, 20 Agustus 2019

Menyetujui,

Pembimbing Utama

Pembimbing Pendamping

Raymond Chandra

«pembimbing pendamping/2»

Ketua Tim Penguji

Anggota Tim Penguji

«penguji 1»

«penguji 2»

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng



## PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

### **PENGUJIAN BERBASIS *BEHAVIOUR SPECIFICATION***

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,  
Tanggal 20 Agustus 2019

Meterai Rp. 6000
---------------------

Muhammad Dipo Putra Wandara  
NPM: 2016730091





## ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

**Kata-kata kunci:** «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»



## ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

**Keywords:** «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»



*«kepada siapa anda mempersembahkan skripsi ini...?»*



## KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini ...»

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Bandung, Agustus 2019

Penulis





## DAFTAR ISI

<b>KATA PENGANTAR</b>	<b>xv</b>
<b>DAFTAR ISI</b>	<b>xvii</b>
<b>DAFTAR GAMBAR</b>	<b>xix</b>
<b>DAFTAR TABEL</b>	<b>xxi</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	2
1.3 Tujuan . . . . .	2
1.4 Batasan Masalah . . . . .	2
1.5 Metodologi . . . . .	2
1.6 Sistematika Pembahasan . . . . .	3
<b>2 LANDASAN TEORI</b>	<b>5</b>
2.1 Unit Testing . . . . .	5
2.2 Bahasa Pemrograman PHP . . . . .	5
2.3 Bahasa Pemrograman HTML . . . . .	5
2.4 Codeigniter . . . . .	5
2.5 Code Coverage . . . . .	5
2.6 Xampp . . . . .	5
2.7 Travis CI . . . . .	5
<b>A KODE PROGRAM</b>	<b>7</b>
<b>B HASIL EKSPERIMEN</b>	<b>9</b>



## DAFTAR GAMBAR

B.1 Hasil 1 . . . . .	9
B.2 Hasil 2 . . . . .	9
B.3 Hasil 3 . . . . .	9
B.4 Hasil 4 . . . . .	9



## DAFTAR TABEL



# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Pengujian perangkat lunak merupakan salah satu tahapan dari *software development life cycle*. *Software Development Life Cycle*(SDLC) merupakan metodologi dari pengembangan perangkat lunak, metode ini dibagi menjadi beberapa fase seperti *analysis, design, coding, testing, installation* dan *maintenance*[7]. *Testing*/Pengujian perangkat lunak adalah proses untuk mencari kesalahan pada setiap *item* perangkat lunak, mencatat hasilnya, mengevaluasi setiap aspek pada setiap komponen (sistem) dan mengevaluasi fasilitas-fasilitas dari perangkat lunak yang akan dikembangkan[1]. Pengujian pada perangkat lunak merupakan tahapan yang wajib dilakukan sebelum perangkat lunak tersebut digunakan, agar memastikan sudah tidak ada *error/bug* pada perangkat lunak yang sedang dikembangkan. Pengujian perangkat lunak memakan *resource* yang berat, baik itu waktu maupun tenaga kerja, karena untuk melakukan pengujian perangkat lunak dibutuhkan *developer* ahli *Software Quality Assurance*(SQA). SQA yang bertanggung jawab untuk memastikan bahwa perangkat lunak yang sedang dikembangkan bebas dari *bug* agar siap untuk di-*release* dan digunakan oleh pengguna.

*Unit testing* merupakan tahapan pertama dari pengujian perangkat lunak. *Unit Testing* adalah proses pengujian bagian kode secara individu, suatu komponen, untuk menentukan apakah kode tersebut berfungsi dengan semestinya[5]. Salah satu teknik untuk membuat *unit testing* yaitu dengan *code generation*. *Code generation* adalah teknik membuat suatu program yang membuat program lain. Menggunakan *code generation* untuk *unit testing* membuatnya mudah untuk mempertahankan dan memperpanjang *unit testing*[6].

Ada beberapa metode untuk melakukan pengujian perangkat lunak, salah satunya yaitu *Test-Driven Development*(TDD). TDD adalah praktik pemrograman yang menginstruksikan pengembang perangkat lunak untuk menulis kode baru hanya ketika tes yang dilakukan secara otomatis gagal, dan menghilangkan duplikasi. Tujuan TDD adalah kode bersih yang berfungsi[2]. Tetapi, TDD memiliki masalah, bahwa TDD berfokus pada keadaan sistem daripada *behaviour* yang diinginkan oleh sistem, dan kode pada pengujian *highly coupled* dengan implementasi sistem yang ada[3]. Maka dari itu metode yang akan digunakan pada skripsi ini adalah *Behaviour-Driven Development*.

*Behaviour-Driven Development*(BDD) merupakan evolusi dari TDD untuk menyelesaikan masalah yang ada pada TDD. BDD adalah pendekatan pengembangan perangkat lunak yang *agile* untuk mendorong kolaborasi antara semua peserta dalam proses pengembangan perangkat lunak[4].

Prinsip inti dari BDD adalah "orang bisnis dan teknologi harus merujuk ke sistem yang sama dengan cara yang sama"[4]. Berbeda dengan TDD yang berpatokan pada test untuk *develope* lebih jauh program yang akan digunakan, BDD berpatokan pada keinginan *stakeholder/customer* untuk mengembangkan programnya. Untuk mencapai kesepakatan antara *developer* dan *stakeholder/customer*, maka dibutuhkan *language* untuk menspesifikasikan *behaviour* sebuah sistem agar kedua pihak paham, dan dapat mewujudkan hal berikut[4]:

- *Stakeholder/Customer* untuk menentukan persyaratan dari perspektif bisnis.
- Analisis bisnis untuk melampirkan contoh konkret (skenario atau *acceptance tests*) yang menje-

laskan *behaviour* sistem.

- *Developer* untuk mengimplementasikan *behaviour* sistem yang diperlukan secara TDD.

Pengujian dengan basis *behaviour specification* akan berfokus kepada hal yang *stakeholder/customer* harapkan pada suatu sistem dengan *behaviour* yang sudah di spesifikasikan. Pengujian akan berawal dengan membuat *Behaviour Test* yang berisi [4]:

1. *Context/Starting state* : Posisi awal sistem sebelum terjadinya suatu *event*.
2. *Event* : *Task* yang dilakukan oleh pengguna pada sistem.
3. *Outcome* : Hasil yang diharapkan dari sistem.

*Developer* akan menjadikan *Behaviour Test* sebagai patokan dasar bekerjanya suatu sistem, dan akan melakukan pengujian berbasis *Behaviour Test*. Pengujian dilakukan untuk memastikan bahwa sistem sudah bekerja sesuai apa yang *stakeholder/customer* harapkan, dan jika *Behaviour Test* berhasil dilakukan, maka sistem sudah siap untuk digunakan.

## 1.2 Rumusan Masalah

- Bagaimana cara kerja pengerjaan Pengujian Berbasis Behaviour Specification?
- Bagaimana implementasi perangkat lunak yang dapat menguji program berbasis Behaviour Specification ?

## 1.3 Tujuan

- Mempelajari cara kerja Pengujian Berbasis Behaviour Specification
- Menghasilkan perangkat penguji yang dapat menguji sebuah perangkat lunak berbasis *behaviour specification*.

## 1.4 Batasan Masalah

Mengingat banyaknya perkembangan yang bisa ditemukan dalam permasalahan ini, maka perlu adanya batasan-batasan masalah yang jelas mengenai apa yang dibuat dan diselesaikan dalam program ini. Adapun batasan-batasan masalah pada penelitian ini sebagai berikut :

## 1.5 Metodologi

Bagian-bagian pekerjaan skripsi ini adalah sebagai berikut :

1. Melakukan Studi literatur mengenai pengujian perangkat lunak, *unit testing*, dan *behaviour specification*.
2. Melakukan eksplorasi perangkat lunak.
3. Menganalisis kebutuhan perangkat lunak.
4. Membuat rancangan desain perangkat lunak, basis data, antarmuka perangkat lunak.
5. Implementasi perangkat lunak yang menerima *input behaviour spesification*, mengolah, dan mengubah ke bentuk *unit testing*.
6. Melakukan pengujian terhadap perangkat lunak.
7. Melaporkan hasil penelitian dalam bentuk dokumen skripsi.



---

## 1.6 Sistematika Pembahasan

- BAB I. PENDAHULUAN

Bab ini berisi tentang latar belakang masalah, rumusan masalah, tujuan, batasan masalah, dan metodologi.

- BAB II. LANDASAN TEORI

Bab 2 memuat uraian tentang teori yang akan digunakan pada pembuatan perangkat lunak.



## BAB 2

### LANDASAN TEORI

#### 2.1 Pengujian Perangkat Lunak

Pandangan umum pengujian perangkat lunak adalah bahwa kegiatan ini adalah untuk menemukan *bug*. Tujuan pengujian perangkat lunak adalah untuk memenuhi syarat kualitas program perangkat lunak dengan mengukur atribut dan kemampuannya terhadap ekspektasi dan standar yang berlaku. Pengujian perangkat lunak juga menyediakan informasi berharga untuk upaya pengembangan perangkat lunak.[9]

Kualitas perangkat lunak adalah sesuatu yang diinginkan semua orang. Manajer tahu bahwa mereka menginginkan kualitas tinggi, pengembang perangkat lunak tahu mereka ingin menghasilkan produk yang berkualitas, dan pengguna bersikeras bahwa perangkat lunak bekerja secara konsisten dan dapat diandalkan.[9]

Banyak kelompok kualitas perangkat lunak mengembangkan *software quality assurance plan*, dimana hal itu sama dengan *test plans*. Rencana jaminan kualitas perangkat lunak dapat mencakup berbagai kegiatan di luar yang termasuk dalam *test plan*. *Quality assurance plan* mencakup keseluruhan kualitas, rencana pengujian adalah salah satu alat kontrol kualitas dari rencana jaminan kualitas.[9]

Pada pembahasan pengujian perangkat lunak, ada *term* yang biasa digunakan, yaitu[8]:

- **Error** — Orang membuat *error*. Sinonim yang baik adalah *mistake*. Ketika orang membuat *error* saat melakukan *coding*, kami menyebut *error* ini *bug*. *Error* cenderung menyebar; *requirements error* dapat diperbesar selama proses desain dan lebih diperkuat lagi selama pengkodean.
- **Fault** — *Fault* adalah hasil dari *error*. Lebih tepat untuk mengatakan bahwa *fault* adalah representasi dari *error*, di mana representasi adalah mode ekspresi, seperti teks naratif, diagram Bahasa Pemodelan Bersatu, diagram hierarki, dan kode sumber. *Defect* adalah sinonim yang baik untuk *fault*, sama juga seperti *bug*. *Fault* bisa sulit dipahami. *Error* yang disebabkan oleh kelalaian menghasilkan *fault* di mana ada sesuatu yang hilang yang seharusnya ada di dalam representasi.
- **Failure** — *Failure* terjadi ketika kode yang sesuai dengan *fault* dijalankan. Dua kehalusan muncul di sini: satu adalah bahwa *failure* hanya terjadi dalam representasi yang dapat dieksekusi, yang biasanya dianggap sebagai kode sumber, atau lebih tepatnya, kode objek yang dimuat; kehalusan kedua adalah bahwa definisi ini hanya mengaitkan *failure* dengan *fault* komisi.
- **Incident** — Ketika *failure* terjadi, itu mungkin atau mungkin tidak mudah terlihat oleh pengguna (atau pelanggan atau penguji). Suatu *incident* adalah gejala yang terkait dengan *failure* yang memberi tahu pengguna tentang terjadinya *failure*.
- **Test** — *Testing* jelas berkaitan dengan *errors*, *faults*, *failures*, and *incidents*. *Test* adalah tindakan melatih perangkat lunak dengan *test case*. *Test* memiliki dua tujuan berbeda: untuk menemukan *failures* atau untuk menunjukkan eksekusi yang benar.

- **Test case** — *Test case* memiliki identitas dan dikaitkan dengan perilaku program. Ini juga memiliki serangkaian input dan output yang diharapkan.

## 2.2 Bahasa Pemrograman PHP

## 2.3 Bahasa Pemrograman HTML

## 2.4 Codeigniter

## 2.5 Code Coverage

## 2.6 Xampp

## 2.7 Travis CI

# LAMPIRAN A

## KODE PROGRAM

Listing A.1: MyCode.c

```
1 // This does not make algorithmic sense,
2 // but it shows off significant programming characters.
3
4 #include<stdio.h>
5
6 void myFunction( int input, float* output ) {
7     switch ( array[i] ) {
8         case 1: // This is silly code
9             if ( a >= 0 || b <= 3 && c != x )
10                 *output += 0.005 + 20050;
11             char = 'g';
12             b = 2^n + ~right_size - leftSize * MAX_SIZE;
13             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
14             strcpy(a,"hello_$@?");
15         }
16         count = ~mask | 0x00FF00AA;
17     }
18 }
19
20 // Fonts for Displaying Program Code in LATEX
21 // Adrian P. Robson, nepsweb.co.uk
22 // 8 October 2012
23 // http://nepsweb.co.uk/docs/progfonts.pdf
```

Listing A.2: MyCode.java

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID; //store the ID of all vertices
12    protected ArrayList<Double> closeDist; //store the distance of all vertices
13    protected int totaltrj; //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35
36 }
```



## LAMPIRAN B

### HASIL EKSPERIMEN

Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4