Pendeteksian kloning kode secara semantik dengan metode *IOE-BEHAVIOR* pada kode sumber Java

I Ketut Dala Cahyoga[1], Jati Hiliamsyah Husen[2], Dana Sulistyo Kusumo[3]

[1,2,3] Universitas Telkom, Bandung
dalacahyoga@student.telkomuniversity.ac.id[1], jatihusen@telkomuniversity.ac.id[2], danakusumo@telkomuniversity.ac.id[3]

Abstrak

Pada saat ini proses pemeliharaan perangkat lunak telah menjadi kegiatan utama dari pihak industri pengembang perangkat lunak. Kloning kode merupakan salah satu penyebab pemeliharaan suatu perangkat lunak menjadi susah. Cacat (*defect*) juga dapat terjadi jika pada proses pemeliharan kloning kode tidak dilakukan dengan baik. Dalam pendeteksian sebuah kloning kode yang dilakukan manual, pendeteksian dilakukan dengan membandingkan baris teks kode satu persatu. Sedangkan, pada proses pendeteksian kloning kode secara semantik, pendeteksian kloning kode dilakukan dengan mendeteksi makna atau tujuan dari suatu *method* pada kode sumber. Metode *IOE-Behavior* adalah sebuah metode yang menggunakan *input, output* dan *effect* dari sebuah *method* untuk melakukan pendeteksian jika adanya pengkloningan kode yang dilakukan secara semantik. Pada penelitian ini akan dilakukan pengukuran perbandingan tingkat kebenaran dan tingkat kesamaan dari hasil pendeteksian sistem dan manual, untuk mengetahui hasil pendeteksian yang lebih baik antara sistem dan manual. Hasil dari pendeteksian sistem mendapatkan hasil 110 *method* kloning dan 81 *method* tidak kloning dari 191 kandidat kloning. Hasil pendeteksian manual didapatkan dengan melakukan survey kepada 5 partisipan, mendapatkan hasil masing-masing 104, 102, 104, 109, 107 *method* kloning dan 87, 89, 87, 82, 84 *method* tidak kloning. Setelah dilakukan pengukuran perbandingan tingkat kebenaran dan tingkat kesamaan antara hasil pendeteksian sistem dan manual, didapatkan rata-rata presentase tingkat kebenaran hasil pendeteksian sistem sebesar 61,54% dan presentase tingkat kebenaran hasil pendeteksian manual sebesar 38,44%. Tingkat kesamaan diukur dengan metode *Cohen's Kappa* mendapatkan rata-rata nilai *Kappa* sebesar 0.715, nilai tersebut menyatakan bahwa tingkat kesamaan antara hasil pendeteksian sistem dan hasil pendeteksian partisipan sebagai kesamaan kuat (*Substantial Agreement*) berdasarkan interpretasi indeks nilai *Cohen's Kappa*.

Kata kunci : Kloning Kode, *IOE-Behavior, Java, Input, Output, Effect, Method, Cohen's Kappa.*

Abstract

Currently the software maintenance process has become the main activity of the software developer industry. Code cloning is one of the reasons why maintaining software is difficult. Defects can also occur if the code cloning maintenance process is not done properly. In detecting a code clone that is done manually, the detection is done by comparing lines of code text one by one. Meanwhile, in the process of detecting code cloning semantically, code cloning detection is done by detecting the meaning or purpose of a method in the source code. IOE-Behavior is a method that uses the input, output and effects of a method to detect if there is a semantic cloning of code. In this study, a comparison measurement of the level of truth and the level of similarity of the system and manual detection results will be carried out, to determine the better detection results between the system and manual. The results of the detection of the system get 110 cloning methods and 81 non-cloning methods from 191 cloning candidates. The results of manual detection were obtained by conducting a survey to 5 participants, getting results of 104, 102, 104, 109, 107 cloning methods and 87, 89, 87, 82, 84 non-cloning methods. After measuring the comparison of the level of truth and the level of similarity between the results of system and manual detection, the average percentage of the truth level of system detection results is 61.54% and manual is 38.44%. The level of similarity measured by the Cohen's Kappa method got an average Kappa value of 0.715, the value states that the level of similarity between the system detection results and the participant detection results is a strong similarity (Substantial Agreement) based on the interpretation of the Cohen's Kappa index value.

Keywords: Code Clone, IOE-Behavior, Java, Input, Output, Effect, Method, Cohens's Kappa.

1. Pendahuluan

1.1. Latar Belakang

Pada saat ini proses pemeliharaan perangkat lunak telah menjadi kegiatan utama dari pihak industri pengembang perangkat lunak [5]. Kloning kode merupakan salah satu penyebab pemeliharaan suatu perangkat lunak menjadi susah [13]. Tindakan kloning kode ini juga dianggap mempermudah proses pengkodean suatu kode sumber pada proses pembuatan perangkat lunak, karena pengembang perangkat lunak dengan menyalin suatu algoritma dapat menyelesaikan permasalahan yang seharusnya dibangun [4]. Akan tetapi kegiatan kloning kode ini dapat menimbulkan masalah jika pengembang perangkat lunak melakukan penyalinan suatu kode sumber pada sebuah sistem yang mengandung *defect*, hal ini menyebabkan pengembang perangkat lunak harus memeriksa seluruh kode sumber yang terdapat pada sistem tersebut [14]. Cacat (*defect*) juga dapat terjadi jika pada proses pemeliharan kloning kode tidak dilakukan dengan baik [15]. Pada penelitian Rattan (2013) menyatakan pada sebuah kode sumber terdapat 20-30% kemungkinan baris kode terkloning [2].

Dalam pendeteksian sebuah kloning kode yang dilakukan manual, pendeteksian dilakukan dengan membandingkan baris teks kode satu persatu [16]. Sedangkan, pada proses pendeteksian kloning kode secara semantik, proses pendeteksian dilakukan dengan mendeteksi makna atau tujuan dari suatu *method* pada kode sumber [4]. Pada penelitian Keivanloo dan Rilling (2013) menyatakan bahwa jika proses pendeteksian berdasarkan perbandingan teks akan sangat sulit untuk melakukan pendeteksian kloning jika dilakukan secara semantik [3]. Hal tersebut mungkin menyebabkan sulitnya proses pendeteksian kloning kode semantik jika dilakukan dengan cara manual. Sebuah kloning kode semantik juga mempunyai pengaruh terhadap peningkatan biaya pada proses pemeliharaan perangkat lunak [4].

Pada penelitian Elva dan Leavens (2012) menyatakan bahwa tujuan atau fungsi dari suatu *method* pada kode sumber java dapat dideteksi dari *input* dan *outputnya*. Elva dan Leavens menambahkan parameter yang dapat digunakan juga untuk mendeteksi kesamaan suatu *method*, parameter yang ditambahkan tersebut adalah *effect*, parameter *effect* ditambahkan dengan alasan untuk mengetahui tujuan dari sebuah *method* dalam kode sumber yang dibangun pada bahasa pemrograman java. Elva dan Leavens memberi nama metodenya dengan nama *IOE-BEHAVIOR* (*Input, Output, Effect - Behavior*) [1]. Metode *IOE-Behavior* merupakan metode pembaruan dari metode sebelumnya yang sudah pernah diusulkan, metode sebelum *IOE-Behavior* merupakan metode *input, output Behavior*. Metode sebelumnya tersebut masih mempunyai kekurangan dalam proses pemotongan fragmen kode karena masih berdasarkan pada kemiripan urutan kode [4].

Dari beberapa hal yang dijelaskan di atas penulis mencoba untuk melakukan penelitian terhadap perbandingan tingkat kebenaran dari sistem pendeteksi kloning kode dengan metode *IOE-Behavior* yang dibandingkan dengan pendeteksian kloning kode yang dilakukan manual. Pengukuran tersebut bertujuan untuk mengetahui hasil pendeteksian yang lebih baik antara sistem pendeteksian yang dibangun dengan metode *IOE-Behavior* dan pendeteksian manual. Dengan dilakukan penelitian ini diharapkan dapat memberikan informasi mengenai presentase perbandingan tingkat kebenaran antara hasil dari sistem pendeteksian kloning kode yang dibangun dengan metode *IOE-Behavior* dan hasil dari pendeteksian kloning kode secara semantik yang dilakukan secara manual, sehingga dapat diketahui hasil pendeteksian yang lebih baik antara sistem pendeteksian yang dibangun dengan menggunakan metode *IOE-Behavior* dan pendeteksian kloning kode secara semantik yang dilakukan manual.

1.2. Topik dan Batasan

Sistem yang dibangun pada penelitian ini yaitu sistem pendeteksi kloning kode menggunakan metode *IOE-Behavior*. Sistem pendeteksi kloning kode ini dilakukan secara semantik. Pada penilitian ini terdapat beberapa batasan sebagai berikut.

1. Pada sistem pendeteksi kloning kode ini hanya dapat melakukan pendeteksian pada kode sumber Java.
2. Sistem pendeteksi kloning kode ini hanya mendeteksi *method* yang berparameter *input,output, effect* dan *non-void*.

1.3. Tujuan

Penelitian ini bertujuan untuk membangun sebuah sistem pendeteksi kloning kode secara semantik yang dibangun dengan menggunakan metode *IOE-Behavior* untuk mengukur perbandingan tingkat kebenaran dan tingkat kesamaan dari hasil pendeteksian sistem pendeteksian kloning kode tersebut yang dibandingkan dengan beberapa hasil pendeteksian kloning kode yang dilakukan secara manual.

2. Studi Terkait

2.1 Kloning Kode

Kloning kode adalah sebuah kegiatan *copy* dan *paste* sebuah kode yang terdapat pada suatu kode sumber ke dalam kode sumber yang lainnya[2]. Kode dianggap sama secara semantik atau secara fungsional jika suatu kode tersebut mempunyai kesamaan secara fungsi atau kegunaannya. Tindakan kloning kode ini juga dianggap mempermudah proses pembangunan suatu kode sumber pada proses pembuatan perangkat lunak, karena pengembang perangkat lunak dengan menyalin suatu algoritma dapat menyelesaikan permasalahan yang seharusnya dibangun [4].

2.1.1 Penyebab Kloning

Para pengembang perangkat lunak melakukan kloning kode disebabkan beberapa hal diantaranya disebabkan keterbatasan waktu yang memaksa seorang pengembang melakukan penyalinan atau pengkloningan sebuah kode selain itu pengembang biasanya menggunakan fragmen kode sebagai sebuah template lalu di *edit* sesuai dengan kebutuhan dari sang pengembang perangkat lunak.

Berikut pola-pola kloning (Patterns of Cloning) yang dijelaskan oleh Kapser dan Godfrey [6].

1. *Forking*
   Pola *Forking* merupakan suatu proses pengkloningan pada sebuah kode yang dilakukan dengan cara menyalinkan sebuah kode yang diletakkan pada file yang baru. Namun, file baru itu akan di *edit* secara mandiri.
2. *Templating*
   Pola *templating* dijadikan sebagai cara untuk menyalin/mengkloning sebuah kode ketika pengembang perangkat lunak tidak mempunyai sebuah rencana untuk pembuatan sebuah alur kode. Templating dilakukan ketika ada hal yang harus dibagikan seperti *library*.
3. *Customization*
   Pola *customization* dilakukan ketika kebutuhan pada sebuah kode tidak dapat dilakukan dengan pengkloningan kode secara langsung, Namun kode tersebut terpenuhi jika dilakukan penyesuaian pada fragmen kode.

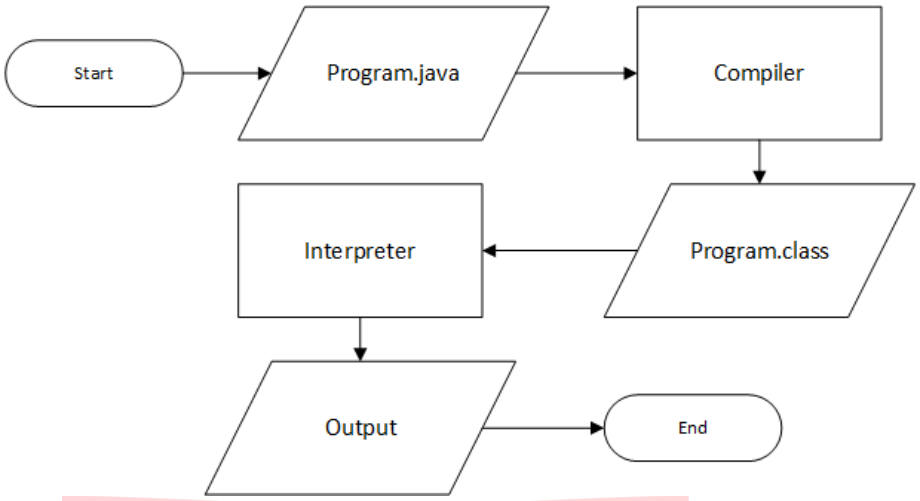
2.1.2 Jenis Kloning

Pada hasil survey penelitian yang dilaksanakan Rattan et al [2], didapatkan beberapa jenis kloning. Beberapa jenis kloning yang sudah didapatkan dari penelitiannya antara lain, sebagai berikut.

1. *Exact clones*
   Jenis *Exact clones* atau kloning yang sama persis, pada jenis ini sepasang kode yang mempunyai script yang sama persis kecuali terdapat penambahan spasi.
2. *Renamed/parameterized clones*
   Jenis *Renamed/parameterized clones* Merupakan kloning struktural, yang mempunyai struktur yang mirip dengan kode yang dituju kecuali ada perubahan pada penandaan (*identifier*), tipe data dan komentar.
3. *Near miss clones*
   Jenis *Near miss clones* mirip dengan jenis *Renamed/parameterized clones* namun disertai dengan penambahan atau penghapusan *statement* atau baris kode.
4. *Semantic clones*
   Pada kloning jenis *Semantic clones* menyatakan kemiripan secara fungsional dengan tidak dinyatakan kemiripan secara sintak ataupun tulisan.
5. *Function Clone*
   Pada jenis *Function Clone*, kloning yang terbatas pada level granularitas *function/method* atau *procedure*.
6. *Model Based Clones*
   Pada jenis *Model Based Clones*, kloning dilakukan pada fase desain saat proses pengembangan perangkat lunak

2.2 Kode Sumber Java

Java merupakan bahasa pemrograman yang digunakan pada sistem berorientasi objek. Java merupakan bahasa permorgraman berorientasi objek terbaik dibandingkan dengan bahasa pemrograman berorientasi objek yang dibuat sebelumnya (Ada, Simula, C++). Java diciptakan oleh James Gosling pada tahun 1991 seorang pengembang yang bekerja pada Sun Microsystems [8].



Gambar 2.1 Cara kerja kode sumber java

Pada kode sumber java ada isitilah *identifier*, yang digunakan sebagai nama *method*, *class* dan variabel. Sebuah variabel bisa dibuat dengan urutan huruf kecil atau besar, angka dan tanda simbol. *Identifier* Tidak boleh mempunyai awalan angka dan sesuatu *case sensitive*. Java menggunakan penamaan *identifier* pada *method public* dan sebuah variabel diawali dengan penulisan huruf kecil dan selanjutnya diawali dengan penulisan huruf besar, contohnya checkHasil, checkNilai, getName.

2.3 Semantik

Pada pembahasan tentang semantik, akan lebih banyak dikutip pada sumber yang sudah melakukan penelitian mengenai semantik sebelumnya.

2.3.1 Pengertian Semantik

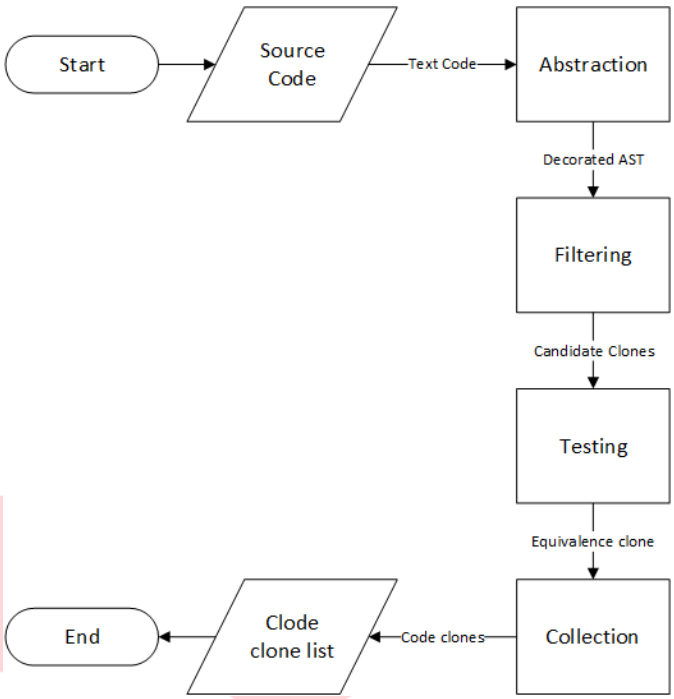
Beberapa ahli menyatakan istilah semantik sebagai berikut, menurut Verharr [7] terdapat dua istilah semantik yakni semantik secara gramatikal dan secara leslikal. Semantik digunakan oleh para ahli sebagai salah satu cabang ilmu bahasa yang mempelajari makna. Menurut kemposn [12] dalam cabang ilmu bahasa, sebuah bahasa terdiri dari kata dan kalimat yang mempunyai makna. Keterkaitan makna antar kata pada sebuah kalimat akan memperkuat arti dari kalimat tersebut. Jadi, semantik dapat dinyatakan sebagai makna.

2.3.2 Semantik Pada Kode

Pendeteksian kloning pada sebuah kode yang dilakukan secara semantik merupakan pendeteksian pada potongan kode yang mempunyai kemiripan fungsi. Fungsi atau tujuan sebuah *method* atau *function* adalah sebuah kemampuan yang bisa dilakukan oleh *method* atau *function* pada sebuah algoritma ketika dijalankan [3]. Maka, *method* atau *function* dapat dianggap sebagai kloningan secara semantik jika *input*, *output* dan *effect* nya sama [1]. Kloning kode secara semantik bisa dianggap pendeteksian yang mempertimbangkan fungsionalitas dari sebuah *method* atau *function*.

2.4 Metode *IOE-Behavior*

Metode *IOE-Behavior* adalah sebuah metode yang menggunakan *input*, *output* dan *effect* dari sebuah *method* untuk melakukan pendeteksian jika adanya pengkloningan kode yang dilakukan secara semantik [1]. Hasil *output* yang dikeluarkan dari *method* suatu *method* merupakan representasi *effect* pada metode *IOE-Behavior* [1]. Metode *IOE-Behavior* dapat digunakan untuk melakukan pendeteksian pada *method* atau *function* dalam sebuah kode sumber yang dianggap sebagai kloning kode. Pendeteksian secara semantik yang dilakukan dengan metode *IOE-Behavior* dilakukan dengan membandingkan *input, output* dan *effect* dari sebuah *method*. Terdapat empat proses dalam melakukan pendeteksian kloning kode menggunakan metode *IOE-Behavior*, yakni *abstraction, filtering, testing dan collection*. Alur dari metode *IOE-Behavior* didefinisikan pada gambar 2.2.



Gambar 2.2 Alur kerja metode IOE-Behavior

2.4.1 Proses *Abstraction*

Proses *abstraction* menghasilkan AST (*Abstract syntax tree*) yang didalamnya terdapat properti dari *method*. AST (*Abstract syntax tree*) yang didapatkan akan dijadikan informasi pada proses selanjutnya. Properti yang terdapat pada AST berisikan nama *method*, tipe *method*, efek *method* dan nama *class* dari *method* tersebut. Terdapat dua informasi utama pada proses *abstraction*. Pertama tipe *method* yang menggambarkan tipe dari parameter *input* dan *output*, informasi kedua adalah efek dari *method* [1]. Tabel 2.1 menggambarkan properti dari sebuah *class* yang terdapat tiga *method*.

Tabel 2.1 Contoh properti dari sebuah class

Code	Tipe	Efek Method
int cekNilai () { return nilai; }	void, int	{acct.nilai}
int hitungRata () { return nilai * rata }	void, int	{acct.nilai, acct.rata}
string PrintState (date d) { return “Nilainya adalah : “ + nilai; }	string, date	{acct.nilai}

2.4.2 Proses *Filtering*

Proses *filtering* menghasilkan kumpulan *method* yang dianggap sebagai kandidat kloning. Pada proses ini mempertimbangkan tipe dan efek dari *method* yang akan dideteksi secara semantik. Hasil dari proses *filtering* ini merupakan kumpulan *method* dengan tipe dan efek yang mirip, hasil ini disebut sebagai kandidat kloning. Terdapat dua tahapan yang dilakukan dalam proses *filtering* [1].

1. *Filtering* menggunakan tipe *method*. Mengelompokkan *method-method* dengan mempertimbangkan tipe (*input* dan *output*) *method* yang mirip.
2. *Filtering* menggunakan efek *method*. Mengelompokkan *method-method* dengan mempertimbangkan *effect* dari *method* yang mirip.

2.4.3 Proses *Testing*

Proses *testing* melakukan pengujian pada kumpulan *method* yang dianggap mirip yang didapatkan dari proses *filtering* [1]. Pengujian pada proses ini dilakukan dengan mengujikan file percobaan. File percobaan tersebut berisikan *random data* untuk menguji sebuah *method* dengan tujuan untuk mengetahui efek dari *method* yang diujikan tersebut.

2.4.4 Proses *Collection*

Proses *collection* merupakan akhir dari metode *IOE-Behavior* yang menyimpan hasil dari proses-proses sebelumnya berupa kumpulan *method* yang dianggap sebagai kloning kode. *Method* yang dideteksi secara semantik dianggap sebagai kloning jika *method* tersebut mempunyai *input*,*output* dan *effect* yang mirip [1]. Proses *collection* menyimpan hasil akhir dari pendeteksian yang berisi kumpulan *method* yang dianggap sebagai kloning.

2.5 *Cohen's Kappa*

Cohen's Kappa merupakan sebuah metode yang mengukur evaluasi kesepakatan pengukuran yang dilaksanakan oleh dua orang penilai [9]. Cohen's Kappa hanya digunakan untuk mengevaluasi hasil dari pengukuran data kualitatif [10]. Metode ini didefinisikan sebagai berikut [9].

$$K = \frac{Po - Pc}{1 - Pc}$$

Tabel 2.2. Relevansi Cohen's Kappa

		Penilai B		Total
		Relevan	Tidak Relevan	
Penilai A	Relevan	*r1*	*r2*	*g1*
	Tidak Relevan	*r3*	*r4*	*g2*
Total		*f1*	*f2*	*n*

Po merupakan hasil kemiripan penilaian dan *Pc* merupakan hasil yang didapatkan secara kebetulan. Relevansi data yang didapatkan dari hasil penilaian dua penilai didefinisikan pada tabel 2.2

Pada gambaran tabel 2.2, *Po* dapat didefinisikan sebagai berikut [9].

$$Po = \frac{r1 + r4}{n}$$

Sedangkan *Pc* dapat didefinisikan sebagai berikut [9].

$$Pc = \frac{\left(\frac{f1 + g1}{n}\right) + \left(\frac{f2 + g2}{n}\right)}{n}$$

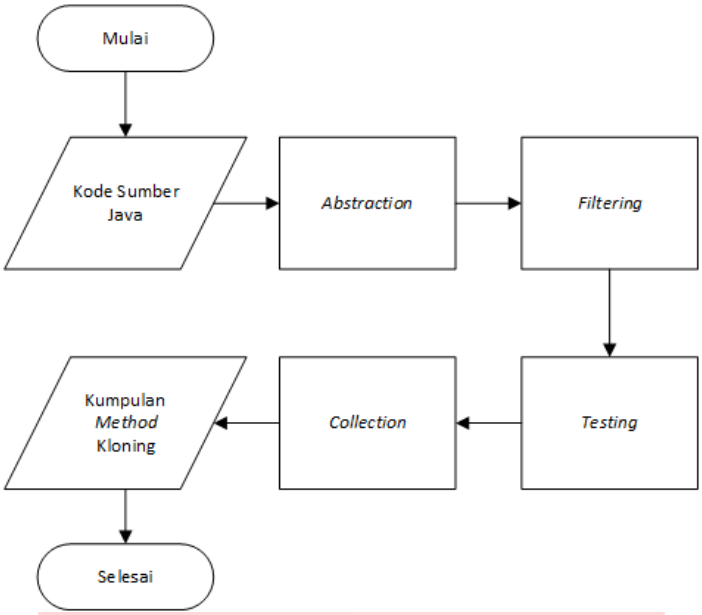
Nilai K bisa dihitung setelah mendapatkan nilai *Po* dan *Pc*. Nilai *Cohen's Kappa* dapat digunakan untuk mengukur nilai kesamaan dari hasil yang didapatkan dari sistem yang dibangun dengan hasil yang didapatkan dari partisipan. Tabel 2.3 menjelaskan interpretasi indeks nilai *Cohen's Kappa* [11].

Tabel 2.3. Interpretasi nilai indeks Cohen's Kappa [11]

Indeks Kappa	Presentase Kesepakatan
0.00 – 0.02	Rendah
0.21 – 0.40	Lumayan
0.41 – 0.60	Cukup
0.61 – 0.80	Kuat
0.80 - 1	Sangat Kuat

3. Sistem yang Dibangun

Pada bagian ini menjelaskan tentang sistem yang dibangun pada penelitian ini, yang terdiri dari empat tahap yakni *abstraction*, *filtering*, *testing* dan *collection*. Desain dari sistem yang dibangun pada penelitian ini digambarkan pada gambar 3.1.



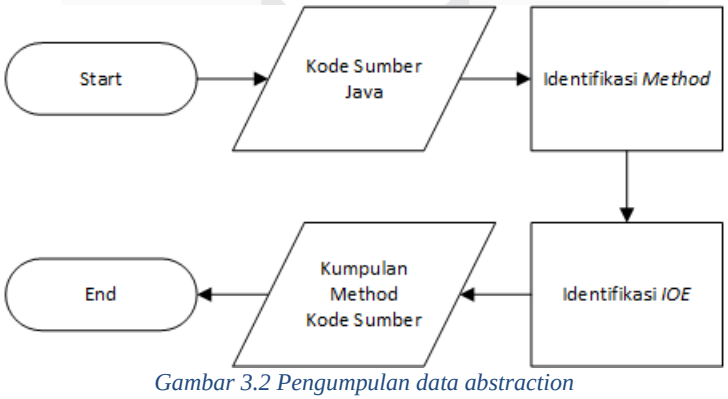
Gambar 3.1 Desain Sistem

3.1 *Abstraction*

Tahap pertama pada penelitian ini adalah *abstraction* yang bertujuan untuk melakukan pengumpulan *method* yang terdapat pada kode sumber. Pada proses pengumpulan *method* pada suatu kode sumber tersebut, terdapat beberapa data yang diambil. Berikut daftar data yang diambil pada suatu kode sumber.

1. Nama *method* merupakan sebuah nama yang dibuat pada suatu *method* yang terdapat pada suatu kode sumber.
2. Tipe *input* merupakan sebuah tipe data yang dimasukan pada sebuah *method*.
3. Tipe *output* merupakan sebuah tipe data yang dikeluarkan pada sebuah *method*.
4. Efek merupakan suatu objek yang dihasilkan oleh fungsi atau tujuan dari sebuah *method*.

Data yang diambil dari *method* tersebut selanjutnya akan disimpan menjadi sebuah kumpulan data. Kumpulan data yang didapatkan tersebut akan disimpan dalam bentuk *array*. Tahap pengumpulan data pada suatu kode sumber didefinisikan pada gambar 3.2.



Gambar 3.2 Pengumpulan data abstraction

3.1.1 Identifikasi *Method*

Pada identifikasi *method* bertujuan untuk mengidentifikasi objek yang dianggap sebuah *method* pada suatu kode sumber. Proses identifikasi ini dilakukan dengan membandingkan objek-objek yang terdapat pada kode sumber dengan melihat penulisan sebuah *method*. Pada proses identifikasi pada metode *IOE-Behavior*, tipe *method* yang diidentifikasi hanya method yang mempunyai parameter input dan method *non-void*. Berikut adalah contoh *method* yang dibedakan dengan *method* berparameter input dan tidak berparameter input.

```
int jmlTotal (int a, int b)
{
    int total;
    total = a + b;
    return total;
}
```

Gambar 3.3.Method berparameter input

```
String total()
{
    int a;
    int b;
    a = 2;
    b = 3;
    String total;
    total = "Total Penjumlahan: " + a * b;
    return total;
}
```

Gambar 3.4 Method tidak berparameter input

3.1.2 Identifikasi *IOE*

Pada identifikasi *IOE* bertujuan untuk melakukan identifikasi pada tipe *input*, *output* dan *effect* pada *method* dalam suatu kode sumber. Identifikasi *input* dilakukan dengan melihat tipe dari variabel masukan pada *method*, sedangkan pada identifikasi *output* dilakukan dengan melihat tipe dari variabel keluaran pada sebuah *method*. *Effect* diidentifikasi dari makna atau tujuan pada sebuah *method*.

3.2 *Filtering*

Pada proses filtering bertujuan untuk melakukan pendeteksian pada *method* yang dianggap kloning. Proses ini melakukan pendeteksian pada *method* yang dianggap kloning berdasarkan dua tahap. Berikut tahap-tahap *filtering* yang dilakukan pada proses ini.

1. Tahap pertama yang dilakukan yaitu *filtering* berdasarkan tipe dari sebuah *method*.
2. Tahap selanjutnya dilakukan *filtering* pada *method* berdasarkan *input* dan *output* dari *method* tersebut.

Filtering pada *method* yang dilakukan pada proses ini menggunakan daftar *method* yang diambil dari kumpulan *method* yang didapatkan dari hasil proses *abstraction*. Pada proses ini akan menghasilkan kelompok-kelompok yang berisikan *method* yang mempunyai *input*, *output* dan *effect* yang sama. *Method* yang terdapat pada kelompok-kelompok tersebut selanjutnya disimpan dalam *array* dan basis data. Kumpulan *method* yang terdapat pada kelompok-kelompok tersebut dinyatakan sebagai kandidat kloning.

3.3 *Testing*

Pada proses sebelumnya didapatkan hasil berupa kelompok-kelompok yang berisikan kumpulan *method* yang dianggap sebagai kloning. Kumpulan *method* yang terdapat pada setiap kelompok dari hasil *filtering* tersebut akan dijalankan untuk menampilkan nilai *output* dari *method* yang diuji coba tersebut. Setiap *method* yang berada pada kelompok yang sama akan diberikan nilai *inputan* yang sama. Hasil *output* dari *method* yang dijalankan tersebut akan dicatat dan disimpan dalam sebuah basis data.

Pada proses ini penentuan nilai *inputan* pada setiap kelompok akan ditentukan dengan data acak sesuai dengan tipe *inputan* dari *method* yang terdapat pada kelompok tersebut. Data acak yang digunakan tersebut berisikan data dari nilai setiap jenis tipe data yang terdapat pada kode sumber java. Hasil *output* yang dikeluarkan oleh *method* tersebut akan disimpan dalam sebuah basis data yang akan digunakan pada proses selanjutnya.

3.4 *Collection*

Proses *collection* merupakan tahap terakhir dari proses pendeteksian ini. Proses *collection* mengumpulkan hasil yang dilakukan oleh proses *testing*. Pada proses *testing* telah didapatkan hasil berupa data nilai keluaran dari setiap *method* yang terdapat pada kelompok kandidat yang dianggap kloning tersebut. Selanjutnya setiap *method* yang mempunyai nilai *output* yang sama akan di kelompokan dan akan disimpan dalam sebuah *array* dan basis data baru.

Kumpulan *method* pada *array* dan basis data tersebut merupakan hasil akhir dari proses pendeteksian ini. Kumpulan *method* yang terdapat pada *array* dan basis data baru tersebut dinyatakan sebagai kloning berdasarkan metode *IOE-Behavior*.

4. Evaluasi

Pada evaluasi dilakukan analisa terhadap pengukuran perbandingan tingkat kebenaran dan pengukuran tingkat kesamaan hasil pendeteksian antara sistem yang dibangun dan pendeteksian manual dari partisipan. Pengukuran perbandingan tingkat kebenaran dilakukan dengan melakukan survey pengecekan ulang kepada partisipan mengenai kebenaran hasil yang berbeda antara sistem dan partisipan. Pengukuran tingkat kesmaan dilakukan menggunakan metode *Cohen's Kappa*. Pengukuran tingkat kesamaan dilakukan dengan cara membandingkan hasil pendeteksian dari sistem dan hasil pendeteksian manual dari semua partisipan. .

4.1 Dataset

Penelitian ini menggunakan perangkat lunak *DNSJava v3.4.1* sebagai dataset yang diujikan. *DNSJava* adalah sebuah perangkat lunak yang mengimplementasikan *DNS* pada bahasa pemrograman java. Kode sumber dari perangkat lunak *DNSJava* didapatkan dari sistus repositori kode *github*. Terdapat 356 *method* yang mempunyai parameter *input, output* dan *effect non-void* pada perangkat lunak *DNSJava* yang akan diujikan pada penelitian ini.

4.2 Hasil Pendeteksian Sistem

Hasil pendeteksian dari sistem yang dibangun pada penelitian ini mendapatkan 191 *method* yang dinyatakan sebagai kandidat kloning yang tersebar pada 49 kelompok kandidat kloning dan 165 *method* dinyatakan bukan merupakan kandidat kloning dari total 356 *method* yang mempunyai parameter *input, output, effect* dan *non-void* pada perangkat lunak *DNSJava* yang diujikan pada penelitian ini. Berikut tabel 4.1 menjelaskan hasil pengujian dari sistem yang dibangun.

Tabel 4.1.Hasil Pendeteksian Sistem

Proses Abstraction	
Daftar Method	356
Proses Filtering	
Kandidat Kloning	191
Kelompok Kandidat Kloning	49
Proses Testing & Collection	
Method Kloning	110
Method Tidak Kloning	81

Pada tabel 4.1 dijelaskan bahwa pada proses *Abstraction* dan *Filtering* mendapatkan 356 daftar *method* dan 191 kandidat kloning yang tersebar pada 49 kelompok kandidat kloning. Pada proses *Testing* dan *Collection* mendapatkan hasil 110 *method* klon dan 81 *method* tidak kloning dari 191 kandidat kloning.

4.3 Hasil Pendeteksian Manual

Hasil pendeksian manual didapatkan dengan melakukan survey dengan menggunakan perangkat lunak *Video Conference Zoom* pada lima partisipan. Partisipan merupakan mahasiswa Informatika Telkom University yang sudah mempunyai pengetahuan tentang pemrograman berorientasi objek dengan bahasa pemrograman java. Partisipan melakukan pendeteksian kloning kode secara manual pada 191 *method* yang dianggap sebagai kandidat kloning yang tersebar dalam 49 kelompok kandidat kloning. Hasil pendeteksian kloning kode secara manual dari 5 partisipan pada perangkat lunak *DNSJava* dijelaskan pada tabel 4.2.

Tabel 4.2 Hasil Pendeteksian Manual

Partisipan	**Method Klon**	**Method Tidak Klon**
Partisipan 1	104	87
Partisipan 2	102	89
Partisipan 3	104	87
Partisipan 4	109	82
Partisipan 5	107	84

4.4 Analisa Hasil Pendeteksian

4.4.1 Analisa Perbandingan Tingkat Kebenaran

Analisa pada proses ini bertujuan untuk mengukur rata-rata perbandingan tingkat kebenaran dari sistem yang dibangun berdasarkan kesepakatan kebenaran dari hasil yang berbeda dari sistem dan partisipan. Pada proses ini dilakukan survey pengecekan ulang kepada partisipan yang bertujuan untuk memastikan kebenaran hasil yang berbeda dari pendeteksian yang dilakukan partisipan. Pada tabel 4.3. menampilkan jumlah hasil pendeteksian yang berbeda dari sistem dan partisipan.

Tabel 4.3 Jumlah Hasil Pendeteksian Berbeda

Partisipan	Jumlah Hasil Pendeteksian Berbeda
Partisipan 1	26
Partisipan 2	28
Partisipan 3	30
Partisipan 4	23
Partisipan 5	27

Terdapat beberapa perbedaan hasil dari sistem dan partisipan dikarenakan cara pendeteksian yang berbeda antara sistem dengan metode *IOE-Behvaior* dan pendeteksian manual. Metode *IOE-Behavior* melakukan pendeteksian dengan membandingkan parameter *input,output* dan *effect*, sedangkan pendeteksian manual dilakukan dengan membandingkan baris teks kode satu persatu. Setelah dilakukan survey pengecekan ulang kepada partisipan mengenai kesepakatan kebenaran dari hasil pendeteksian kloning kode yang dilakukan, didapatkan hasil perbandingan jumlah kebenaran hasil dari sistem dan partisipan seperti yang tampilkan pada tabel 4.4.

Tabel 4.4 Perbedaan jumlah kebenaran hasil pendeteksian

Partisipan	Jumlah Kebenaran Sistem	Jumlah Kebenaran Partisipan	Total
Partisipan 1	17	9	26
Partisipan 2	18	10	28
Partisipan 3	20	10	30
Partisipan 4	12	11	23
Partisipan 5	16	11	27

Selanjutnya dilakukan perhitungan presentase perbandingan tingkat kebenaran anatara sistem dan partisipan. Presentase tingkat kebenaran dapat didefinisikan sebagai berikut.

$$Presentase\ (\%) = \frac{Jumlah\ pendeteksian\ benar}{Jumlah\ total\ hasil\ pendeteksian\ berbeda}\ x\ 100\%$$

Tabel 4.5 Perbandingan presentase tingkat kebenaran sistem dan partisipan

Partisipan	Presentase Kebenaran Sistem	Presesntase Kebenaran Partisipan
Partisipan 1	65,38%	34,61%
Partisipan 2	64,28%	35,71%
Partisipan 3	66,66%	33,33%
Partisipan 4	52,17%	47,82%
Partisipan 5	59,25%	40,74%
Rata-rata	**61,54%**	**38,44%**

Pada tabel 4.5 ditampilkan hasil perhitungan presentase perbandingan tingkat kebenaran anatara sistem dan partisipan. Dari total perbandingan kebenaran hasil pendeteksian sistem dengan lima partisipan didapatkan rata-rata presentase kebenaran sistem sebanyak 61,54% dan presentase kebenaran partisipan sebanyak 38,44%.

4.4.2 Analisa Tingkat Kesmaaan

Analisa pada proses ini bertujuan untuk mengukur tingkat kesamaan hasil pendeteksian antara sistem dan manual dengan menggunakan *Cohen's Kappa*. Nilai *Cohen's Kappa* dapat dihitung dari tabel hasil pendeteksian sistem dan tabel hasil pendeteksian manual. Nilai *Cohen Kappa* didapatkan dari perhitungan *Po* dan *Pc*, *Po* merupakan presentase hasil pendeteksian yang konsisten dari sistem dengan partisipan. *Pc* merupakan presentase perubahan pengukuran antara sistem dengan partisipan. Pada tabel 4.6, 4.7, 4.8 menampilkan tabel perhitungan *Cohen's Kappa* antara hasil pendeteksian sistem dengan lima hasil pendeteksian manual dari partisipan.

Tabel 4.6 Hasil Perhitungan Cohen Kappa Partisipan 1 dan 2

	Partisipan 1			Partisipan 2		
Sistem	**Kloning**	**Tidak**	**Total**	**Kloning**	**Tidak**	**Total**
Kloning	94	16	110	92	18	110
Tidak	10	71	81	10	71	81
Total	104	87	191	102	89	191
Po	0.863			0.853		
Pc	0.010			0.010		
Kappa	0.724			0.704		

Pada pengukuran nilai *Cohen Kappa* untuk partisipan 1 didapatkan nilai *Po* 0.863 dan *Pc* 0.010. Perbandingan hasil sistem dengan partisipan 1 mendapatkan nilai *Cohen Kappa* sebesar 0.724. Hasil dari tingkat kesepakatan *Kappa* pada partisipan 1 dapat diinterpretasikan kuat. Pada pengukuran nilai *Kappa* untuk partisipan 2 didapatkan nilai *Po* 0.853 dan *Pc* 0.010. Pada perbandingan hasil sistem dengan partisipan 2 mendapatkan nilai *Cohen Kappa* sebesar 0.704. Hasil dari tingkat kesepakatan *Kappa* pada partisipan 2 dapat diinterpretasikan kuat.

Tabel 4.7 Hasil Perhitungan Cohen Kappa Partisipan 3 dan 4

	Partisipan 3			**Partisipan 4**		
Sistem	**Kloning**	**Tidak**	**Total**	**Kloning**	**Tidak**	**Total**
Kloning	92	18	110	98	12	110
Tidak	12	69	81	11	70	81
Total	110	81	191	109	82	191
Po	0.842			0.879		
Pc	0.010			0.010		
Kappa	0.682			0.754		

Pada pengukuran nilai *Cohen Kappa* untuk partisipan 3 didapatkan nilai *Po* 0.842 dan *Pc* 0.010. Perbandingan hasil sistem dengan partisipan 3 mendapatkan nilai *Cohen Kappa* sebesar 0.682. Hasil dari tingkat kesepakatan *Kappa* pada partisipan 3 dapat diinterpretasikan kuat. Pada pengukuran nilai *Kappa* untuk partisipan 4 didapatkan nilai *Po* 0.879 dan *Pc* 0.010. Pada perbandingan hasil sistem dengan partisipan 4 mendapatkan nilai *Cohen Kappa* sebesar 0.754. Hasil dari tingkat kesepakatan *Kappa* pada partisipan 4 dapat diinterpretasikan kuat.

Tabel 4.8 Hasil Perhitungan Cohen Kappa Partisipan 5

	Partisipan 5		
Sistem	**Kloning**	**Tidak**	**Total**
Kloning	95	15	110
Tidak	12	69	81
Total	107	84	191
Po	0.858		
Pc	0.010		
Kappa	0.712		

Pada pengukuran nilai *Cohen Kappa* untuk partisipan 5 didapatkan nilai *Po* 0.858 dan *Pc* 0.010. Perbandingan hasil sistem dengan partisipan 5 mendapatkan nilai *Cohen Kappa* sebesar 0.712. Hasil dari tingkat kesepakatan *Kappa* pada partisipan 5 dapat diinterpretasikan kuat. Dari semua hasil pengukuran nilai *Kappa* didapatkan rata-rata nilai *Kappa* sebesar 0.715, rata-rata nilai *Kappa* tersebut dapat diinterpretasikan sebagai tingkat kesepakatan kuat (*Subtantial Agreement*).

5. Kesimpulan

Berdasarkan penelitian yang telah dilakukan untuk mengukur perbandingan tingkat kebenaran dan tingkat kesamaan antara hasil pendeteksian sistem pendeteksi kloning kode secara semantik dengan metode *IOE-Behavior* dan hasil pendeteksian kloning kode manual yang dilakukan oleh lima partisipan pada kode sumber java yang dilakukan uji coba pada dataset *DNSJava v3.4.1*, pada tingkat kesamaan anatara hasil pendeteksian sistem dan manual yang diukur dengan metode *Cohen's Kappa* didapatkan rata-rata nilai *Kappa* sebesar 0.715, berdasarkan hasil pengukuran tingkat kesamaan tersebut dapat diketahui tingkat kesamaan antara hasil pendeteksian sistem dan partisipan dapat diinterpretasikan sebagai kesamaan kuat (*Substantial Agreement*) berdasarkan interpretasi indeks nilai *Cohen's Kappa*. Pada tingkat kebenaran didapatkan rata-rata presentase tingkat kebenaran hasil pendeteksian sistem sebesar 61,54% dan presentase tingkat kebenaran hasil pendeteksian manual sebesar 38,44%, berdasrkan perbandingan presentase tingkat kebenaran tersebut dapat diketahui bahwa hasil pendeteksian sistem pendeteksi kloning kode dengan metode *IOE-Behavior* mempunyai tingkat kebenaran yang lebih besar dibandingkan pendeteksian kloning kode yang dilakukan manual.

REFERENSI

[1] Elva, R., Leavens, G., (2012) "Jsctracker: A Semantic Clone Detection Tool for Java Code", University of Central Florida.

[2] Rattan, D., Bhatia, R., & Singh, M., (2013), "Software Clone Detection : A Systematic Review", Information and Software Technology.

[3] Keivanloo, I., & Rilling, J., (2013), "Semantic-Enabled Clone Detection", 2013 IEEE 37th Annual Computer Software and Applications Conference.

[4] Bayu Priyambadha., (2015), "Pendeteksian klon Secara Semantik Berdasarkan Perilaku Kode", Institut Teknologi Sepuluh November.

[5] Shofi Nastiti., Fajar Pradana., Tri Astoto Kurniawan., (2016), "Pendeteksian Kloning Kode Secara Semantik Dengan Metode IOE-BEHAVIOR Pada Kode Sumber PHP.

[6] Kapser, C. J., & Godfrey, M. W., (2008), ""Cloning Considered Harmful" Considered Harmful: Patterns of Cloning in Software", Empirical Software Engineering.

[7] Dosen Pendidikan. "Semantik adalah",[Online]. Tersedia: https://www.dosenpendidikan.co.id/. [diakses 11 November 2020].

[8] Noviyanto, S.T., "Pengenalan Bahasa Pemrograman Java", Universitas Gunadarma.

[9] Sim, J., & Wright, C. C., (2005), "The Kappa Statistic in Reliability Studies: Use, Interpretation, and Sample Size Requirements", Physical Therapy.

[10] ."Uji Konsistensi Cohen's Kappa", [Online] Tersedia: https://pelatihan-ui.com/uji-konsistensi-cohens-kappa/. [diakses 28 Juni 2021].

[11] Landis, J., & Koch, G. (1977). The measurement of observer agreement for categorical data. Biometrics, 159-174.

[12] Kempson, R.M., (1977), "Semantic Theory". Cambridge: Cambridge University Pres.

[13] Md. Monzur Morshed. A Literature Review of Code "Clone Analysis to Improve Software Maintenance Process", Department of Computer Science, American International University-Bangladesh.

[14] Sjoberg, D., & Yamashita, A., (2013), "Quantifying the effect of code smells on maintenance effort". IEEE Transactions On Software Engineering.

[15] Bettenburg, Shang, W. S. W., Ibrahim, W., Adams, B., Zou, Y. Z. Y., & Hassan, a. E., (2009), "An Empirical Study on Inconsistent Changes to Code Clones at Release Level", 2009 16th Working Conference on Reverse Engineering.

[16] Ashish N. Runwal, D. Waghmare., (2017), "Code Clone Detection based on Logical Similarity". International Journal of Advanced Research in Computer and Communication Engineering.