Pembuatan Kakas Pendeteksi *Unused Method* pada Kode Program PHP dengan *Framework* CodeIgniter Menggunakan *Call Graph*

Divi Galih Prasetyo Putri, Daniel Oranova Siahaan, Rizky Januar Akbar
Jurusan Teknik Informatika, Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
e-mail : daniel@if.its.ac.id

***Abstrak*—Proses evolusi dan perawatan dari sebuah sistem merupakan proses yang sangat penting dalam rekayasa perangkat lunak tidak terkecuali pada aplikasi web. Pada proses ini kebanyakan pengembang tidak lagi berpatokan pada rancangan sistem. Hal ini menyebabkan munculnya *unused method*. Bagian-bagian program ini tidak lagi terpakai namun masih berada dalam sistem. Keadaan ini meningkatkan kompleksitas dan mengurangi tingkat *understandability* sistem. Guna mendeteksi adanya *unused method* pada progam diperlukan teknik untuk melakukan *code analysis*. Teknik *static analysis* yang digunakan memanfaatkan *call graph* yang dibangun dari kode program untuk mengetahui adanya *unused method*. *Call graph* dibangun berdasarkan pemanggilan antar method. Aplikasi ini mendeteksi *unused method* pada kode program PHP yang dibangun menggunakan *framework* CodeIgniter. Kode program sebagai inputan diurai kedalam bentuk *Abstract Syntax Tree* (AST) yang kemudian dimanfaatkan untuk melakukan analisis terhadap kode program. Proses analisis tersebut kemudian menghasilkan sebuah *call graph*. Dari *call graph* yang dihasilkan dapat dideteksi method-method mana saja yang tidak berhasil ditelusuri dan tergolong kedalam unused method. Kakas telah diuji coba pada 5 aplikasi PHP dengan hasil rata-rata nilai presisi sistem sebesar 0.749 dan *recall* sebesar 1.**

Kata Kunci—abstract syntax tree, call graph, code analysis, CodeIgniter, PHP, unused method.

I. PENDAHULUAN

APLIKASI berbasis web telah menjadi salah satu jenis aplikasi yang penting dan banyak digunakan oleh masyarakat. Aplikasi ini memungkinkan pengguna untuk dapat menggunakan aplikasi tanpa perlu melakukan instalasi. Dengan adanya kemudahan ini, perkembangan aplikasi web menjadi sangat pesat. Semakin banyak pihak yang mengambangkan aplikasi berbasis web sebagai media untuk memberikan informasi maupun jasa. Bahasa pemrograman PHP menjadi salah satu bahasa yang banyak digunakan untuk mengembangkan aplikasi berbasis web. PHP merupakan bahasa *open source* yang mudah dikembangkan karena konfigurasinya yang cukup mudah dan memiliki banyak referensi. Telah banyak terdapat *framework* yang dapat digunakan untuk membantu pengembang dalam membangun aplikasi ini antara lain Yii [1], CakePHP [2], Symfony [3], dan CodeIgniter [4]. *Framework* ini memanfaatkan model MVC dalam pembangunan situs yang dinamis.

Banyaknya pihak yang memanfaatkan aplikasi berbasis web mengakibatkan permintaan yang terus bertambah dan berubah terhadap sistem. Selain itu, usaha pengembang untuk meningkatan performa, menambah fungsionalitas, atau mengadaptasi teknologi baru juga memicu adanya evolusi sistem. Pada proses ini banyak pengembang tidak lagi berpatokan pada rancangan sistem.

Banyak pengembang hanya menambahkan fungsi baru untuk menggantikan sebuah fungsi tanpa menghapuskan fungsi tersebut. Fungsi-fungsi yang tidak terpakai disebut sebagai *unused method.* Adanya *unused method* pada program akan sangat merugikan pengembang sistem. Dari sisi *maintenance*, akan sulit dilakukan pembenahan terhadap sistem karena banyak kode program yang tidak jelas kegunaannya namun masih berada dalam sistem. Identifikasi dan penghapusan *unused method* juga akan mengurangi kompleksitas dan ukuran sistem, menghambat *maturity* sistem, meningkatkan *understandability* sistem, dan memudahkan proses *maintenance* [5].

Telah terdapat berbagai macam aplikasi untuk menganalisis kode program terutama PHP seperti PHPMD (PHP Mess Detector) [6], PHPDCD (PHP Dead Code Detector) [7], PHP CodeSniffer [8], dan lain-lain. PHP Code Sniffer dapat digunakan untuk melakukan analisis terhadap kode program PHP, Javascript, dan CSS dan membantu pengembang menjaga kode tetap konsisten dari bersih dari kesalahan sesuai dengan standar kode yang telah ditentukan. Aplikasi ini juga dapat digunakan pada kode program PHP dengan *framework* CodeIgniter. Akan tetapi, untuk dapat menggunakan aplikasi ini pada kode program PHP dengan *framework* PHP pengguna harus menentukan sendiri aturan-aturan dalam pengkodean yang menjadi dasar untuk melakukan analisis. Pengguna harus membuat sebuah kelas yang memanfaatkan fungsi bawaan

dari CodeSniffer dan membangun fungsi-fungsi untuk menetapkan aturan pengkodean yang harus dipenuhi oleh aplikasi yang akan dianalisis. Oleh karena itu, dibutuhkan sebuah sistem yang dapat menganalisis kode program PHP dengan *framework* CodeIgniter dengan praktis dan mudah.

Untuk mengetahui fungsi-fungsi mana saja yang tergolong *unused method* perlu dilakukan analisis pada program. Metode yang dapat digunakan untuk menganalisis sebuah program antara lain metode *dynamic analysis* dan *static analysis*.

Metode *static analysis* lebih cocok digunakan karena proses identifikasi bagian dari program yang menunjukkan *behavior* yang tidak diinginkan dapat dilakukan tanpa perlu melakukan eksekusi terhadap program. Salah satu model yang dapat digunakan untuk membantu melakukan analisis statis pada kode program adalah *call graph*. *Call Graph* cocok digunakan dalam kasus ini karena *call graph* merupakan sebuah teknik untuk merepresentasikan pemanggilan antar prosedur dalam sebuah sistem. Diharapkan dengan menggunakan teknik ini dapat mengidentifikasi adanya *unused method* pada sebuah sistem secara efektif.

II. TINJAUAN PUSTAKA

A. Analisis Kode Sumber

Source code analysis atau analisis kode sumber adalah sebuah proses untuk mendapatkan informasi mengenai sebuah program dari kode sumbernya atau dari artefak yang dihasilkan dari kode sumbernya menggunakan kakas bantu. Proses ini sangat penting dalam mendukung proses pemeliharaan sistem. Informasi yang dihasilkan dapat berupa data-data seperti *Identifiers Table*, *Abstract Syntax Tree* (AST), *Control Flow Graph* (CFG), *Call Graph*, *Value Dependence Graph* (VDG), dan lain-lain. Informasi tersebut harus koheren dengan semantik dari bahasa pemrogramannya sehingga dapat membantu pengembang dalam memahami kode program [4]. Terdapat dua metode untuk melakukan *source code analysis* yaitu *static analysis* dan *dynamic analysis*.

1. *Static Analysis*
   Static analysis adalah sebuah teknik analisis untuk mendapatkan informasi tentang jalur yang mungkin dilalui dalam sebuah eksekusi program tanpa menjalankan program tersebut. Teknik ini menelusuri perilaku dari sebuah program dengan menggunakan semua *input* dan *output* yang mungkin didapatkan [9].
2. *Call Graph*
   Call graph merupakan sebuah graf berarah yang merepresentasikan transfer kontrol antar prosedur. Setiap *call graph* memiliki *node* sebagai prosedur dan *edge (f, g)* untuk pemanggilan prosedur *f* dari prosedur *g*. Sehingga sebuah *cycle* pada *call graph* menggambarkan pemanggilan prosedur yang rekursif. Setiap bahasa pemrograman memiliki tantangan tersendiri dalam pembangunan *call graph*. *Call graph* digunakan untuk membantu memahami jalannya program dan kegiatan pemeliharaan program lainnya. Terdapat dua tipe *call graph* yaitu *context-sensiti*ve *call graph* dan *context-insensitive call graph*. Pada *context-sensitive call graph*, setiap pemanggilan prosedur harus dianalisis secara terpisah karena memiliki konteks yang berbeda. Sedangkan pada *context-insensitive call graph*, setiap node menggambarkan satu prosedur [10].

III. ANALISIS DAN PERANCANGAN

Proses pendeteksian kode program dilakukan pada kode program PHP dengan *framework* CodeIgniter. *Framework* ini memanfaatkan model MVC (*Model View Controller*) untuk membangun sebuah aplikasi web.

MVC merupakan sebuah konsep yang cukup populer digunakan dalam pengembangan aplikasi. Konsep ini memisahkan pengembangan aplikasi menjadi tiga komponen. Ketiga komponen tersebut antara lain:

1. View
   Komponen ini menangani presentasi logic dari sebuah aplikasi. Pada aplikasi web, *view* biasanya berbentuk halaman html. *View* berfungsi untuk merepresentasikan data kepada pengguna. Komponen view tidak memiliki akses terhadap komponen *model*.
2. Controller
   Controller merupakan komponen yang mengatur hubungan antara *model* dan *view*.
3. Model
   Komponen *model* menangani manipulasi data. Biasanya berhubungan langsung dengan database. Pemanggilan oleh *model* hanya dapat dilakukan oleh komponen *controller* dan komponen model tidak dapat langsung mengakses komponen *view*.

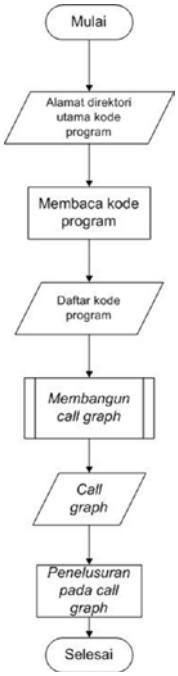
CodeIgniter menyediakan banyak *library* penunjang sehingga pengembang dapat lebih mudah dalam membangun aplikasi. Pada CodeIgniter, aplikasi web mulai dijalankan dari sebuah *default_controller* yang terletak pada file *config* yang merupakan bawaan dari CodeIgniter.

Pada direktori utama terdapat tiga folder yaitu *application*, *system*, dan *user_guide*. Seluruh file aplikasi web yang dikembangkan diletakkan pada folder *application*. Seluruh file *controller* aplikasi dimasukkan kedalam folder *controllers*, file-file antarmuka aplikasi pada folder *views*, dan kelas model untuk aplikasi pada folder *models*.

A. Deskripsi Umum Sistem

Sistem yang akan dibuat yaitu berupa kakas bantu. Kakas bantu ini menampilkan hasil deteksi adanya *unused method* dari sebuah aplikasi PHP yang dipilih oleh user. Kakas menampilkan hasil deteksi berupa daftar *method* yang terdeteksi beserta keterangan dari kelasdan komponen mana asal *method* tersebut. Pengguna dapat memilih untuk menampilkan kode program dari *method* tersebut. Kakas akan menampilkan kode program dari kelas asal *method* tersebut. *Method* yang terdeteksi akan diberi tanda berupa pewarnaan teks yang berbeda.

Diharapkan dengan adanya kakas bantu ini, pengguna dapat dengan lebih mudah mengetahui adanya *unused method* pada



Gambar. 1. Diagram Alir Proses Mendeteksi *Unused Methods*

aplikasi yang sedang dikembangkan sehingga dapat dilakukan penanganan dengan lebih cepat.

B. Spesifikasi Kebutuhan

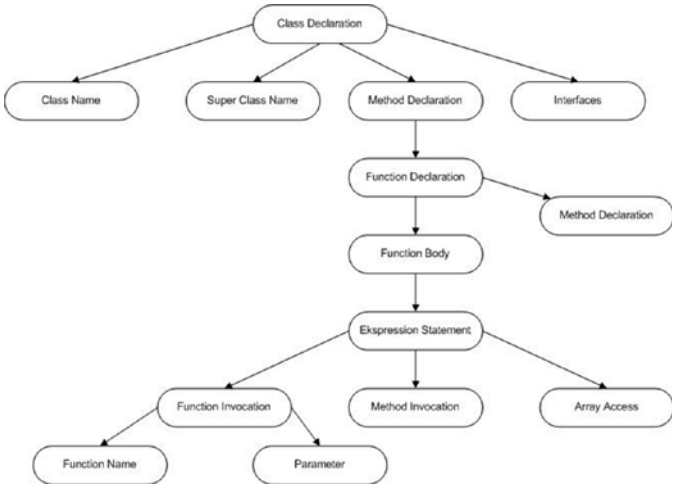
Pada sistem yang akan dibuat ini, dibutuhkan beberapa fungsi yang dapat membantu proses bisnis dalam sistem. Fungsi-fungsi tersebut antara lain:

1. Melihat hasil deteksi *unused* method dari kode program yang dimasukkan
   Pada kasus penggunaan ini, sistem menerima input berupa perintah untuk memproses kode program. Setelah itu, sistem akan menampilkan hasil deteksi berupa daftar *method*.
2. Menampilkan kode program dari hasil deteksi
   Sistem dapat menampilkan kode program dari *method* yang terdeteksi. Sistem menerima masukan berupa nama *method* dan kelas yang dipilih untuk ditampilkan.

C. Perancangan Proses Deteksi Unused Methods

Proses deteksi adanya *unused methods* dapat dilihat pada Gambar 1. Proses dimulai dengan mendaftar semua file PHP yang menyusun aplikasi. File-file tersebut kemudian diuraikan ke dalam bentuk AST untuk dapat dianalisis lebih lanjut. AST menangkap struktur penting dari sebuah kode program dan merepresentasikannya dalam bentuk *tree*. Struktur kode program tersebut dibentuk menjadi sebuah *node* dengan tipe *node* yang spesifik. AST berbeda dari *concrete syntax tree* karena AST tidak menangkap detail sintaks seperti koma untuk pemisahan argumen. Hasil AST dari kode program PHP dapat dilihat pada Gambar 2.

Dari hasil analisis akan didapatkan sebuah *call graph*. *Call graph* yang terbentuk merepresentasikan pemanggilan antar *method*. Terdapat empat jenis pemanggilan antar *method*



Gambar 2. Struktur Abstrak Syntax Tree

antara lain:

1. Pemanggilan dari kelas *controller* ke kelas *model*
   - *$this->load->model('nama_model')*
   - *$this->load->nama_model->nama_method()*
2. Pemanggilan antar *method* pada *controller* yang sama
   - *$this->nama_method()*
3. Pemanggilan dari kelas *controller* ke *view*
   - *$this->load->view->('nama_view',data)*
4. Pemanggilan dari *view* ke kelas *controller*

- Pemanggilan kelas *controller* dari *view* dapat dilakukan dengan 3 cara, yaitu:

a. Pemanggilan menggunakan kode program PHP dapat dilakukan menggunakan fungsi *form_open*.
   - *form_open(nama_controll/nama_method)*

b. Pemanggilan menggunakan kode program HTML dapat dilakukan dengan *tag href* dan *form*.
   - *<a href="nama_controller/nama_method">*
   - *<form action="nama_controller/nama_method">*

c. Sedangkan untuk pemanggilan *controller* menggunakan kode program JavaScript dapat dilakukan dengan menggunakan *ajax*, *jquery*, maupun dengan fungsi *windows.location.href*.
   - *#ajax{url="pemanggilan_controller"}*
   - *window.Location.href="pemanggilan_kontroller "*
   - *$('#...').load('pemanggilan_kontroller')*

Untuk dapat menganalisis jenis pemanggilan pertama sampai ketiga, bentuk AST yang didapatkan dari kode program PHP ditelusuri. Penelusuran dikhususkan pada *node FunctionInvocation* yang merepresentasikan ekspresi pemanggilan antar *method*. Khusus untuk jenis pemanggilan keempat, perlu adanya proses tambahan. Selain dilakukan penguraian menggunakan PHP *parser*, file *view* diuraikan kode HTML dan JavaScript-nya untuk dapat mengetahui adanya pemanggilan antar *method*.

Setiap *method* digambarkan sebagai sebuah *node* dalam *call graph*. Apabila *node* tersebut melakukan pemanggilan

Tabel 1.
Tabel Nilai Presisi dan *Recall*

Data Uji	Retrieve	Relevant	Relevant retrieve	Precision	Recall
keymanagement	7	6	6	0,857143	1
kp2	2	2	2	1	1
aps	25	17	17	0,68	1
SIMSchool	199	42	42	0,2111	1
blank_app	30	30	30	1	1
Average				0,7496486	1

terhadap *node lain*, maka kedua *node* tersebut dihubungkan satu sama lain. Proses pembangunan *call graph* dimulai dari *root_controller* aplikasi. Kemudian *call graph* yang terbentuk ditelusuri. Setiap *node* yang terhubung ke *node* lain dapat digolongkan sebagai *unused method*.

IV. IMPLEMENTASI

Kakas bantu ini dibangun dengan menggunakan perangkat pengembang Eclipse Kepler. Bahasa pemrograman yang digunakan adalah bahasa pemrograman java. Kakas yang dibangun berupa *library*. *Library* ini berisi fungsi-fungsi yang diperlukan untuk melakukan proses deteksi *unused methods*. Selain itu, data-data *input* dan *output* proses juga disimpan dalam sebuah kelas *singletone* pada *library*. *Library* tersebut kemudian disertakan dalam pembangunan Eclipse plugin. Eclipse yang digunakan adalah Eclipse Kepler versi 4.3 yang sudah terinstal RCP. Antar muka dibangun dengan menggunakan SWT.

V. PENGUJIAN DAN EVALUASI

Untuk mengukur performa dari sistem perlu dilakukan perhitungan tingkat presisi dan *recall*. Presisi dihitung dari jumlah informasi relevan yang berhasil dideteksi dibandingkan dengan seluruh hasil deteksi [15]. Secara matematis, rumus untuk menghitung presisi dan recall dapat dilihat pada (1) dan (2).

$$precision = \frac{relevant\ items\ retrieved}{retrieved\ items} \quad (1)$$

$$recall = \frac{relevant\ items\ retrieved}{relevant\ items} \quad (2)$$

Recall didapatkan dari perbandingan jumlah informasi relevan yang berhasil dideteksi dengan jumlah *method* yang tergolong *unused* berdasarkan informasi dari pengembang aplikasi. Pengembang aplikasi mendaftar semua *method* pada aplikasinya dan menelusuri pemanggilan antar *method* secara manual dimulai dari *default_controller*-nya. Kemudian pengembang aplikasi menentukan *method-method* mana yang tidak digunakan dan tergolong sebagai *unused methods.* Hasil yang didapatkan secara manual tersebut yang kemudian dibandingkan dengan hasil sistem untuk mendapatkan nilai presisi dan *recall*. Hasil pengujian untuk setiap aplikasi dapat dilihat pada Tabel 1.

Hasil uji coba pada aplikasi **keymanagement** menunjukkan nilai presisi sebesar 0,85 dan *recall* sebesar 1. Sistem dapat mendeteksi seluruh *method* yang tergolong *unused* dapat dilihat dari nilai *recall* sistem. Namun berdasarkan konfirmasi dengan pihak pengembang aplikasi, tidak semua *method* yang terdeteksi merupakan *method* yang *unused*. Terdapat satu *method* yaitu *check_database* pada kelas *verifylogin* yang bukan merupakan *unused method* namun terdeteksi oleh sistem.

Hal ini disebabkan oleh*method* yang dibangun untuk melakukan validasi terhadap inputan pengguna. *Method* tersebut didefinisikan dan dipanggil menggunakan library dari CodeIgniter sehingga sistem tidak dapat menangkap pemanggilan *method* tersebut. Kode program untuk permasalahan tersebut dapat dilihat pada Kode Sumber 1.

Hasil uji coba pada aplikasi **kp2** menunjukkan bahwa presisi dan *recall* sistem sebesar 1. Semua *method* yang merupakan *unused method* dapat dideteksi dan semua hasil deteksi relevan.

```
$this->form_validation->set_rules('username',
'username', 'trim|required|xss_clean');
   $this->form_validation-
>set_rules('password','password',
'trim|required|xss_clean|callback_check_database');
```

Kode Sumber 1.Contoh Kode Program yang Tidak Terdeteksi

Pada hasil uji coba pada aplikasi **aps** dapat dilihat bahwa nilai presisi dari sistem sebesar 0,68 dan *recall* sebesar 1. Terdapat 8 *method* yang bukan merupakan *unused method* namun terdeteksi oleh sistem. Hal ini dikarenakan tidak ada pemanggilan terhadap *method-method* atau halaman tampilan tersebut apabila penelusuran dilakukan dari *default controller*. Sehingga, *method-method* tersebut digolongkan ke dalam *unused method*.

Dari hasil dari uji coba pada aplikasi **SIMSchool** dapat dilihat tingkat presisi yang rendah yaitu 0,21. Hal ini disebabkan oleh adanya pemanggilan kelas *view* dari *controller* yang menggunakan *passing parameter* antar fungsi dalam *controller* untuk mendefinisikan nama kelas *view* yang dipanggil. Hal ini tidak dapat ditangkap oleh sistem, karena sistem menggunakan metode *static analysis*. Sedangkan untuk mendeteksi jalannya nilai dalam sistem harus digunakan metode *dynamic analysis*.

```
$this->display('content/administrator', $data);
```

Kode Sumber 2.Contoh Kode Program yang Tidak Terdeteksi (2)

```
public function display($alamat,$data)
    {
        //header
        $this->load-
>view('templates/header_chart',$data);
        $this->load->view('templates/layout_top');
        $this->load->view('templates/menu_top');
```

```
        $this->load->view('templates/content_top');

        //content
        $this->load->view($alamat,$data);
        $this->load-
>view('templates/content_bottom');
        $this->load-
>view('templates/layout_bottom');
    }
```

Kode Sumber 3.Contoh Kode Program yang Tidak Terdeteksi (3)

Kode Sumber 2 merupakan potongan kode dari kelas kontoler yang digunakan untuk mengakses *method* lain pada *controller* yang sama dengan parameter pertama berupa nama halaman yang ingin diakses dan parameter kedua berupa data yang akan ditampilkan pada halaman tersebut. *Method* yang dipanggil oleh proses tersebut adalah *method display* dan dapat dilihat pada Kode Sumber 3.

Hasil pengujian pada aplikasi **blank_app** menunjukkan bahwa nilai presisi dan *recall* sistem untuk aplikasi tersebut adalah 1. Seluruh *method* pada aplikasi tersebut tergolong dalam *unused method* kecuali 6 buah *method construct* pada kelas *controller* dan model. Hal ini dikarenakan *method-method* pada aplikasi tersebut adalah *method* kosong.

VI. KESIMPULAN DAN SARAN

Dari hasil selama proses perancangan, implementasi, serta pengujian dapat diambil kesimpulan sebagai berikut:

1. Sistem dapat mendeteksi *method* yang tergolong dalam *unused method* dengan memanfaatkan *call graph* yang dibangun dari kode program. Hal ini ditunjukkan dengan nilai *recall* dari sistem sebesar 1. Semua *method* yang tergolong *unused method* dapat ditangkap oleh sistem.
2. Sistem dapat melakukan deteksi dengan rata-rata tingkat ketelitian dan ketepatan sebesar 0.75. Terdapat beberapa *method* yang terdeteksi oleh sistem bukan merupakan *unused method*.
3. Sistem dapat menampilkan blok *method* yang dipilih pengguna untuk ditampilkan

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Diantaranya adalah sebagai berikut:

1. Meningkatkan presisi sistem untuk mendeteksi adanya *unused* method dengan menambahkan aturan-aturan pemanggilan antar *method* beredasarkan standar pengkodean bahasa pemrograman PHP.
2. Menambahkan fitur untuk melakukan proses penghapusan terhadap *method* yang terdeteksi sebagai *unused method* dari sistem.

DAFTAR PUSTAKA

[1] "yiiframework," [Online]. Available: http://www.yiiframework.com/. [Diakses Januari 2014].

[2] "CakePHP," [Online]. Available: http://cakephp.org/. [Diakses Desember 2013].

[3] "Symfony," [Online]. Available: symfony.com. [Diakses Januari 2014].

[4] "CodeIgniter," [Online]. Available: http://ellislab.com/codeigniter. [Diakses Desember 2013].

[5] K. D. Cooper dan L. Torczon, Engineering a Compiler, Houston: Morgan Kaufmann, 2009.

[6] "PHPMD," [Online]. Available: http://phpmd.org/. [Diakses Januari 2014].

[7] "PHPDCD," [Online]. Available: https://github.com/sebastianbergmann/phpdcd. [Diakses Januari 2014].

[8] "PHP CodeSniffer," [Online]. Available: http://pear.php.net/package/PHP_CodeSniffer. [Diakses Januari 2013].

[9] D. d. Cruz, P. R. Henriques dan J. S. Pinto, "Code Analysis: Past and Present," dalam *FOSE '07 2007 Future of Software Engineering*, Braga, 2009.

[10] T. Reps, M. Sagiv dan R. Wilhelm, "Static Program Analysis via 3-Valued Logic□," dalam *9th Internatinal Symposium SAS*, Madrid, 2002.

[11] D. Grove , G. DeFouw, J. Dean dan C. Chambers, "Call Graph Construction in Object-Oriented Languages," dalam *Proceedings of the 12th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, New York, 1997.

[12] P. Raghavan, C. D. Manning dan H. Schutze, "Evaluation in Information Retrieval," dalam *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[13] Eclipse. [Online]. Available: http://projects.eclipse.org/projects/tools.pdt. [Diakses 9 Januari 2014].

[14] "JSDT," [Online]. Available: https://eclipse.org/webtools/jsdt/.

[15] C. D. Wang, W. A. Appel, L. J. Kom dan S. C. Serra, "The Zephy abstract syntax description languange," dalam *Proceedings of the COnference on Domain-Sepific Language*, Berkeley, 1997.