ASPEK PEDAGOGIK IMPLEMENTASI TRANSLATOR NOTASI ALGORITMIK BERBASIS PARSING LL(*) DAN STRING TEMPLATE

Wijanarto[1], Ajib Susanto[2]
[1,2]Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro
Jl. Nakula 5 - 11, Semarang, 50131, 024-3517261
E-mail : wijanarto.udinus@gmail.com[1], ajibsusanto@gmail.com[2]

Abstrak

Pengajaran pemrograman dasar pada tahun pertama, merupakan matakuliah dasar wajib bagi mahasiswa ilmu komputer. Algoritma merupakan model untuk memecahkan masalah di bidang pemrograman yang di implementasikan dalam bahasa pemrograman. Tidak mudah bagi seseorang dalam membuat solusi dalam bentuk bahasa formal, selain pemilihan alat atau aplikasi yang tepat untuk membantunya. Paper ini hasil penelitian yang melihat aspek pedagogik dari pengajaran pemrograman dengan model notasi algoritmik yang akan menghasilkan Domain Specific Language (DSL) untuk pengajaran pemrograman dasar. Implementasi model pengajaran di wujudkan dengan metode parsing LL(*) dan string template, yang otomatis akan mentranslasikan notasi algoritmik menjadi bahasa c standar. Model notasi algoritmik yang di pilih sudah pernah diterapkan dan diajarkan di perguruan tinggi. Grammar dihasilkan dengan bantuan ANTLR dan string template, yang di sesuaikan dengan model yang di pilih. Metode eksperimen di pakai untuk mengukur apakah terdapat perbedaan mahasiswa yang menggunakan translator dengan yang tidak menggunakan translator. Hasil dari penelitian ini berupa translator yang dapat di jalankan dalam command prompt serta editor gui yang di sebut ETNA (Editor Translator Notasi Algoritmik) serta rekomendasi penggunaan alat yang tepat dalam pengajaran pemrograman dasar. Alat ini diharapkan membantu seseorang atau mahasiswa mendisain solusi dalam bentuk notasi algoritmik, tanpa memikirkan kerumitan dalam bahasa yang pakai.

Kata Kunci: *parsing LL(*), aspek pedagogik, translator,algoritmik, experimental method*

Abstract

Teaching basic programming in the first year , a compulsory basic course for computer science students. Algorithm is a model for solving problems in the field of programming is implemented in a programming language . Not easy for someone to create a solution in the formal language form , in addition to the selection of the right tool or application for help. This paper studies looking at the results of the pedagogical aspects of teaching programming model algorithmic notation which will result in Domain Specific Language (DSL) for teaching basic programming. Implementation of the teaching model embodied by the method of parsing LL (*) and string template, which automatically translates an algorithmic notation became a standard C language. Notation algorithmic models have been implemented in select and taught in college . Generated with the help of ANTLR grammar and string templates, which are customized to the selected models . Experimental methods in use to gauge whether there are differences in the use of students who did not use a translator to translator. Results

from this research is a translator that can be run in a command prompt and gui editor that called ETNA (Algorithmic Notation Editor Translator) as well as recommendations appropriate use of tools in the teaching of basic programming . This tool is expected to help a person or a student design algorithmic solutions in the form of notation , without thinking about the complexity of the language used.

.***Keywords:*** *LL (*), aspects pedagogy, translator, Algorithmics, experimental method*

1. PENDAHULUAN

Studi yang pernah dilakukan di Afrika Selatan menunjukan bahwa keberhasilan suatu pembelajaran pemrograman dasar di pengaruhi oleh 3 aspek, *pertama*, lingkungan belajar (alat atau aplikasi) yang mendukung notasi yang sederhana, yang dapat mengkonstruksi notasi umum untuk bahasa pemrograman.

K*edua*, penampilan visual dari struktur program harus memungkinkan mahasiswa pemrograman dasar dapat memahami semantik konstruksi program dan *ketiga*, lingkungan kerja aplikasi harus melindungi mahasiswa untuk tidak melakukan interpretasi dan pemahaman yang salah. Sesederhana apapun, suatu masalah pemrograman yang harus di pecahkan tetap dilakukan secara terstruktur dan ilmiah. Di bidang ilmu komputer atau teknik informatika, langkah atau urutan langkah pemecahan masalah atau metode yang logis, terstruktur dan berhingga di sebut sebagai algoritma [1][2]. Algoritma merupakan metode penyelesaian masalah yang umum dan banyak di lakukan hampir di seluruh bidang ilmu[3].

Di lain pihak pemahaman mahasiswa atau orang yang tertarik mempelajari pemrograman sering terkendala oleh bagaimana menggunakan bahasa itu sendiri. Di Indonesia studi mengenai pembelajaran pemrograman dasar sangat sedikit, apalagi yang menyangkut alat penunjang atau ketepatan penggunaan aplikasinya. Dalam penelitian yang di lakukan Hidayanti [4], lebih menyoroti metode pembelajaran dari aspek pedagogik, di mana capaian mahasiswa dalam belajar pemrograman dasar sangat rendah di karenakan rendahnya partisipasi, keaktifan dalam berdiskusi dan bertanya serta menjawab pertanyaan dalam kuliah. Sedangkan peneliti lain [5], dalam matakuliah sejenis yaitu komputer dasar, menyimpulkan (masih dari aspek pedagogik) bahwa metode belajar berbasis pada masalah dapat meningkatkan pemahaman materi dan prestasi mahasiswa, namun hanya efektif di lakukan dalam satu siklus saja.

Dengan demikian menurut hemat kami, dalam rangka mempermudah proses pembelajaran siswa dalam pemrograman dasar diperlukan model yang dapat menyederhanakan struktur dan semantik instruksi [6], sehingga dapat mempermudah pemahanan serta mengurangi interpretasi yang salah dalam rangka menyelesaikan masalah dalam bidang pemrograman.

Model sederhana yang diimplementasikan merupakan suatu translator notasi algoritmik yang secara otomatis dapat menghasilkan suatu bahasa pemrograman tingkat tinggi yang umum [7]. Sementara notasi algoritmik yang standar yang diberikan merupakan notasi yang sudah di ajarkan di perguruan tinggi [8]. Paper ini akan

mengimplementasikan model yang di pilih untuk menghasilkan suatu translator notasi algoritmik ke dalam bahasa C standard dan mengukur tingkat perbedaan yang terjadi dalam pengajaran pemrograman dasar dari aspek pedagogik.

2. METODE PENELITIAN

2.1 Parsing LL(*)

Parsing LL(*) merupakan perbaikan dar LL(k) untuk k>1, lookahead pada LL(k) terbatas pada k saja, sedangkan dalam LL(*) dapat mengestimasikan berapa kedalaman lookahead. Gambar 1 berikut selengkapnya mengenai notasi predikat grammar pada LL (*).

$A \in N$	Nonterminal
$a \in T$	Terminal
$X \in (N \cup T)$	Grammar symbol
$\alpha, \beta, \delta \in X^*$	Sequence of grammar symbols
$u, x, y, w \in T^*$	Sequence of terminals
$w_r \in T^*$	Remaining input terminals
ϵ	Empty string
$\pi \in \Pi$	Predicate in host language
$\mu \in \mathcal{M}$	Action in host language
$\lambda \in (N \cup \Pi \cup \mathcal{M})$	Reduction label
$\vec{\lambda} = \lambda_1..\lambda_n$	Sequence of reduction labels
Production Rules:	
$A \rightarrow \alpha_i$	i^{th} context-free production of A
$A \rightarrow (A'_i) \Rightarrow \alpha_i$	i^{th} production predicated on syntax A'_i
$A \rightarrow \{\pi_i\}? \ \alpha_i$	i^{th} production predicated on semantics
$A \rightarrow \{\mu_i\}$	i^{th} production with mutator

Gambar 1. Notasi Predikat Grammar LL(*)

Definisi formal dari LL(*) [9], sebagai berikut grammar *G=(N,T,P,S,Π,M),* dimana *N* adalah himpunan non terminal simbol atau rule, T adalah himpunan terminal simbol atau token, *P* adalah himpunan produksi, $S \in N$ merupakan start simbol, *Π* himpunan side effect free predikat semantik, dan *M* adalah himpunan aksi (mutator).

2.2 String Template

String Template (ST) [10][11] merupakan engine template dan file template yang di pakai bersama-sama sebagai controller untuk melakukan translasi. ST merupakan DSL untuk mengenerate teks terstruktur dari internal data struktur untuk output suatu grammar. ST program dapat di tulis dalam java yang merupakan controller dalam finite state automata. Struktur ST dapat terdiri sebagai berikut pada gambar 5,

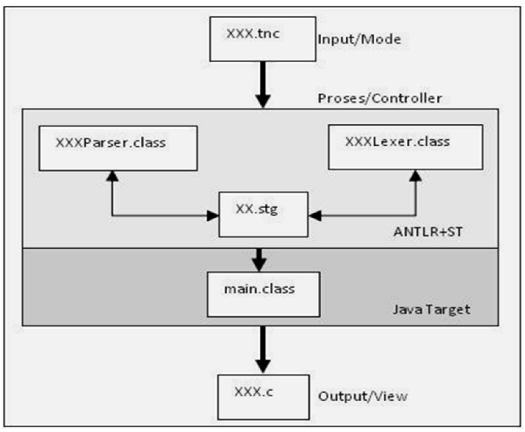
```
group groupname;
template1(a1, a2, ..., an) ::= "..."
...
```

Gambar 2. Struktur Group Template

Template berisi kumpulan referensial mutual pada output yang menyediakan pustaka untuk mengkonstruksi output bagi kontroler. Template di kompilasi menjadi instance bertipe string template yang bertindak sebagai prototype instance selanjutnya.

2.3 Arsitektur Aplikasi

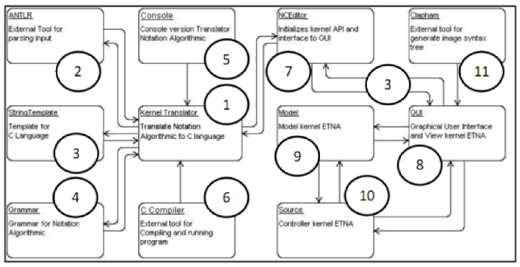
Suatu sistem aplikasi di kembangkan dengan suatu metode atau cara yang beragam, paper ini akan menggunakan pendekatan Model View Contrller (MVC) [12][13][14]. Arsitektur Aplikasi adalah seperti gambar 3 sebagai berikut,



Gambar 3. Arsitektur Translator Notasi Algoritmik berbasis MVC

Input yang berupa file text dalam bentuk notasi standar algoritma akan di baca oleh scanner yang sesuai dengan grammar yang di generate oleh

ANTLR. String Template merupakan translator (*hand coded*) notasi ke bahasa yang di spesifikasikan secara simultan saat membuat grammar. Generator notasi, yang menjadi test rig dalam bentuk class akan menghasilkan output bahasa yang valid. Sementara implementasi arsitektur dapat di lihat pada gambar 4 berikut,

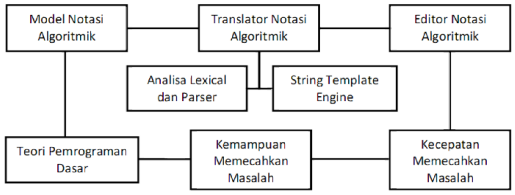


Gambar 4. Diagram Block Aplikasi

Translator (kernel ETNA) ditandai dengan lingkaran 1, saling berkomunikasi dengan ANTLR, lingkaran 2, String Template dan grammar di tandai lingkaran 3 dan 4, sebagai paket dan kelas yang di pakai kernel. Parser dan Lexer dari grammar yang di hasilkan ANTLR, serta string template yang di tulis khusus untuk bahasa c, dipakai oleh kernel selama ETNA berjalan. GUI sebagai interface ETNA dan user memakai kernel saat diperlukan. Console, lingkaran 5 merupakan translator dalam versi command line yang memakai kernel serta kompiler c, lingkaran 6, sebagai tool luar untuk menghasilkan file eksekusi juga di pakai oleh kernel. Interaksi kernel dan GUI (ETNA) melalui NCEditor, ditandai lingkaran 7. Saat aplikasi dimulai kernel akan di inisialisasikan oleh NCEditor bersama-sama GUI sekaligus sebagai viewer ditandai lingkaran 8, model Translator (kernel ETNA) ditandai dengan lingkaran 9 serta source controller di tandai lingkaran 10, sebagai implementasi model MVC. Sementara tool dari luar Clapham lingkaran 11, menggenerate image syntax tree yang saat ini di pakai untuk membantu user memahami notasi agoritmik (ke depan akan di manfaatkan untuk error trace secara visual).

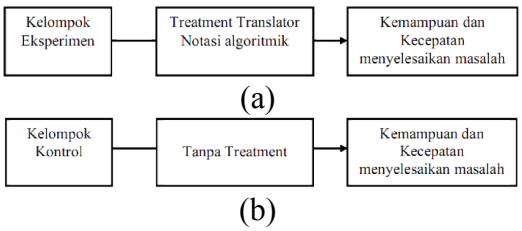
2.4. Kerangka Pikir Dan Paradigma Eksperimen [15]

Belajar pemrograman mahasiswa di harapkan tidak terjebak menggunakan bahasa pemrograman (program) yang cenderung rumit dan sukar di pahami, dan alat untuk membantu mahasiswa dalam belajar pemrograman adalah suatu bahasa natural (Notasi Algoritmik) yang mudah di mengerti dan mudah di terjemahkan oleh komputer dalam rangka menyelesaikan masalah di bidang pemrograman. Translator notasi algoritmik dapat membantu mahasiswa memecahkan masalah di bidang pemrograman dasar tanpa belajar bahasa program yang rumit, seperti kerangka pikir yang di jelaskan pada gambar 5 di bawah ini.



Gambar 5. Kerangka Pikir

Paradigma penelitian yang di pakai dalam metode eksperimen adalah seperti pada gambar 6 a dan b berikut,



Gambar 6. Paradigma Penelitian

Dari gambar paradigma penelitian di atas, variabel penelitian yang telah ditetapkan dikenal posttest dengan

pengukuran penggunaan tanpa prates. Pembelajaran tanpa menggunakan translator notasi pada kelompok eksperimen dan pembelajaran dengan menggunakan translator notasi algoritmik untuk kelompok kontrol. Setelah itu, kedua kelompok tersebut dikenai pengukuran dengan menggunakan pascates. Sedang hipotesa yang di ujian adalah sebagai berikut :

a. Hipotesa Nol (Ho)

1) Tidak ada perbedaan penggunaan translator notasi algoritmik yang signifikan antara kelompok yang menyelesaikan masalah pemrograman dengan menggunakan translator notasi algoritmik dan kelompok yang tanpa menggunakan translator notasi algoritmik.

2) Penggunaan translator notasi algoritmik tidak lebih cepat memecahkan masalah pemrograman dibandingkan dengan tanpa menggunakan translator notasi.

b. Hipotesa Alternatif (Ha)

1) Terdapat perbedaan kemampuan menyelesaiakan masalah pemrograman antara kelompok yang menggunakan translator notasi algoritmik dan kelompok yang tanpa menggunakan translator notasi algoritmik.

2) Penggunaan translator notasi algoritmik lebih cepat memecahkan masalah pemrograman dibandingkan tanpa menggunakan translator notasi algoritmik dalam menyelesaikan masalah pemrograman dasar.

3. HASIL DAN PEMBAHASAN

3.1 ETNA (Editor Translator Notasi Algoritmik)

ETNA atau editor translator notasi algoritmik merupakan hasil implementasi denga menggunakan metode parsing LL(*) dan string template dalam bentuk GUI. Potongan grammar yang di hasilkan dengan metode parsing LL(*) dan string template di sajikan dalam gambar 6 dan 7 berikut.

```
grammar Algoritmik;
......
Program
...
: declaration+
-> program(
   libs={$program::libs},
   globals={$program::global),
  functions={$program::functions},
mainfunctions={$program::mainfunctio
ns}
);
....
LINE_COMMENT
:'//'~('\n'|'\r')*'\r'?'\n'
{$channel=HIDDEN;}
;
```

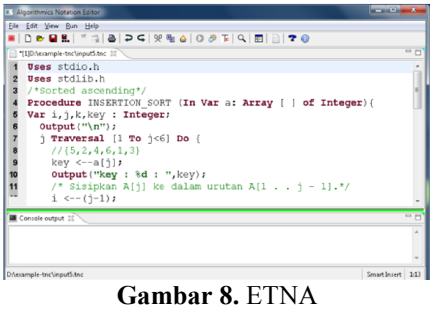
Gambar 6. Grammar notasi algoritmik

Sementara potongan string tempplate yang di hasilkan adalah sebagai berikut.

```
group Algoritmik;
program
(libs,globals,functions,mainfunctions
)
::=<<
<libs; separator="\n">
<globals; separator="\n">
<functions; separator="\n">
<mainfunctions; separator="\n">
>>
...
```

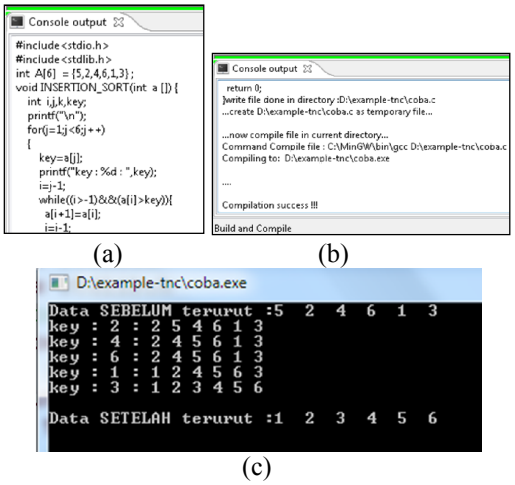
Gambar 7. String Template Algoritmik

Berikut tampilan ETNA saat sedang mengolah file source notasi algoritmik pada gambar 8.



Gambar 8. ETNA

ETNA yang di kembangkan telah berhasil mentranslasikan notasi algoritmik, melakukan kompilai dan menjalankan file hasil kompilasi seperti terlihat pada gambar 9 berikut.



Gambar 9. (a) Hasil translasi, (b) Hasil kompilasi, (c) Hasil eksekusi

3.2 Aspek Pedagogik

Hail uji normalitas sebaran data yang diperoleh dari pascates kemampuan dan kecepatan memecahkan masalah pemrograman dasar baik dari kelompok eksperimen maupun kontrol seperti pada tabel 1 dan 2 di bawah ini.

Tabel 1. Hasil Uji Normalitas Sebaran Data Pascates Kemampuan Menyelesaikan Masalah Pemrograman Dasar [Sumber : data primer]

Data	Sig.	Keterangan
Pascates Kelompok Kontrol	0.140	Sig. (2-tailed) > 0, 050 = normal
Pascates Kelompok Eksperimen	0.690	Sig. (2-tailed) > 0, 050 = normal

Kecepatan juga memperoleh hasil yang signifikan sebagai berikut,

Tabel 2. Hasil Uji Normalitas Sebaran Data Pascates Kecepatan Menyelesaikan Masalah Pemrograman Dasar [Sumber : data primer]

Data	Sig.	Keterangan
Pascates KK	0. 398	Sig. (2-tailed) > 0, 050 = normal
Pascates KE	0. 293	Sig. (2-tailed) > 0, 050 = normal

3.2.1 Perbandingan Data Kelompok Kontrol dan Kelompok Eksperimen

Berikut ini disajikan tabel perbandingan data pascatest skor tertinggi, skor terendah, *mean*, *median*, dan *mode* dari kelompok kontrol dan kelompok eksperimen. Tabel ini di buat untuk memberi kejelasan gambaran hasil penelitian yang di peroleh sedemikian rupa sehingga kita dapat memperbandingan dengan mudah.

Tabel 3. Perbandingan Data Statistik Pascates Kemampuan dan Kecepatan Menyelesaikan Masaah Pemrograman [Sumber : data primer].

	N	Min		Max		Mean		Median		Mode	
		A	B	A	B	A	B	A	B	A	B
X	38	29	60	85	21	64.68	44.58	64.00	46.33	61	56
Y	38	60	57	93	71	76.89	32.03	79.50	29.00	61	77

Keterangan :
A: Kemampuan ; B:Kecepatan
X : Pascatest Kelompok Kontrol
Y : Pascatest Kelompok Eksperimen

Sementara itu, hasil uji beda kelompok eksperimen dan control pada kemampuan dan kecepatan menyelesaikan masalah pemrograman dasar di sajikan dalam table 3 dan 4 berikut

Tabel 4. Hasil Uji-t Data Pascates Kemampuan Memecahkan Masalah Pemrograman Dasar Kelompok Kontrol dan Kelompok Eksperimen [Sumber : data primer]

Data	T Hitung	df	P	Keterangan
A	4.638	74	.000	p < 0,05 = signifikan

Keterangan
A : Pascates kelompok kontrol dan kelompok eksperimen

Dari tabel di atas maka uji beda signifikan terdapat perbedaan kemampuan kelompok yang di uji. Pada hasil uji beda kecepatan menyelesaikan masalah pemrograman dasar juga member hasil yang signifian seperti di sajikan sebagai berikut

Tabel 5. Hasil Uji-t Data Skor Kecepatan Memecahkan Masalah Pemrograman Dasar Kelompok Kontrol dan Kelompok Eksperimen. [Sumber : data primer]

Data	T Hitung	df	P	Keterangan
A	-4.718	74	.000	p > 0,05 = signifikan

Keterangan
A : Pascates kelompok kontrol dan kelompok eksperimen

4. KESIMPULAN DAN SARAN

Dari paparan di atas maka penulis menyimpulkan sementara bahwa implementasi model notasi algoritmik dengan LL(*) parsing dan string template berhasil di kembangkan menjadi suatu alat pembelajaran pemrograman dasar ETNA. Hasil uji beda terhadap ETNA dalam pemakaian pengajaran juga memberi hasil yang signifikan, sehingga dapat menerima hipotesa, bahwa terdapat perbedaan kemampuan dan kecepatan pemecahan masalah pemrograman dasar dengan ETNA pada kelompok eksperimen dan control. Kedepan perlu studi lebih lanjut mengenai pelacakan kesalahan pada ETNA dengan memanfaatkan kemampuan syntax tree, juga perlu di teliti lebih lanjut reaksi ETNA terhadap user dari aspek dimensi kognitif [2] maupun framework Nelson.

DAFTAR PUSTAKA

[1] Blass, Andreas; Gurevich, Yuri., 2003, *Algorithms: A Quest for Absolute Definitions*, Bulletin of European Association for Theoretical Computer Science.

[2] David Harel, Yishai A. Feldman, 2004 , *Algorithmics: the spirit of computing, Edition 3*, Pearson Education, ISBN 0-321784-0.

[3] Chen Shyi-Ming, Lin Chung-Hui, Chen Shi-Jay, 2005, *Multiple DNA Sequence Alignment Based on Genetic Algorithms and Divide-and-Conquer Techniques*, International Journal of Applied Science and Engineering. 3, 2: 89-100.

[4] Hindayati Mustafidah, 2007, *Prestasi Belajar Mahasiswa dalam Mata Kuliah Pemrograman Dasar Melalui Pembelajaran Kooperatif Model Jigsaw*, Paedagogia, Agustus jilid 10 No 2, hal. 126 – 131.

[5] Yuwono Indro Hatmojo, Sigit Yatmono, 2009, *Peningkatan Prestasi Mata Kuliah Komputer Dasar Mahasiswa D3 Teknik Elektro FT UNY Menggunakan Metode Belajar Berbasis Masalah*, Jurnal edukasi@Elektro Vol. 5, No.1, Maret, hal. 67 – 78.

[6] Chairmain Cilliers, Andre Calitz, Jean Greyling, 2005, *The Application of The Cognitive Dimension Framework for Notations as an Instrument for the Usability analysis of an Introductory Programming tool*, Alternation Journal, 12.1b, p 543-576 ISSN 1023-1757.

[7] Wijanarto, Achmad Wahid Kurniawan, 2012, *Model Translator Algoritmik ke Bahasa C*, Prosiding Kommit, Komputer dan Sistem Intelijen, Vol 7, 464-472 ISSN 2302-3740.

[8] Liem, Inggriani, 2007, *Draft Diktat Dasar Pemrograman (Bagian Prosedural)*, ITB , Bandung, unpublished.

[9] Parr, Terrence, Fischer, Kathleen S, 2011, LL(*) : The Foundation of the ANTLR Parser Generator, PLDI '11, Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation, ACM New York, NY USA, ISBN: 978-1-4503-0663-8

[10] Parr, Terrence, 2006, A Functional Language For Generating Structured Text, di akses 10-10-2013, 2006, http://www. cs. usfca. edu/parrt/papers/ST. pdf

[11] Parr, Terrance, Fischer, Kathleen S, 2004, Enforcing Strict Model-View Separation in Template Engines, New York, New York, USA. ACM 1-58113-844-X/04/0005.

[12] Reenskaug, Trygve M.H., 1979, *MODELS - VIEWS - CONTROLLERS. ,* XEROX PARC.

[13] Reenskaug, Trygve M.H., 1979, *THING-MODEL-VIEW-EDITOR an Example from a planningsystem.* , Xerox PARC technical note May 1979.

[14] Stanchfield, Scott. *Applying MVC in VisualAge for Java. JavaDude*. [Online] 1996 - 2009. diakses: 10-10-2012. http://javadude.com/articles/vaddmvc2/mvc2.html.

[15] Sugiyono, 2010, Metode Penelitian Kuantitatif, Kualitatif dan R &F, Bandung, Alfabeta.