LAPORAN KEMAJUAN
PENELITIAN DOSEN PEMULA

TRANSLATOR NOTASI ALGORITMIK UNTUK PENGAJARAN PEMROGRAMAN DASAR

Tahun ke 1 dari rencana 1 tahun

WIJANARTO, S.Sos., M.Kom. 0628027003
AJIB SUSANTO, S.Kom., M.Kom 0615127404

UNIVERSITAS DIAN NUSWANTORO
Oktober 2013

HALAMAN PENGESAHAN

Judul Kegiatan	:	TRANSLATOR NOTASI ALGORITMIK UNTUK PENGAJARAN PEMROGRAMAN DASAR
Peneliti / Pelaksana		
Nama Lengkap	:	WIJANARTO
NIDN	:	0628027003
Jabatan Fungsional	:	
Program Studi	:	Teknik Informatika
Nomor HP	:	
Surel (e-mail)	:	wijanarto.udinus@gmail.com
Anggota Peneliti (1)		
Nama Lengkap	:	AJIB SUSANTO M.Kom.
NIDN	:	0615127404
Perguruan Tinggi	:	UNIVERSITAS DIAN NUSWANTORO
Institusi Mitra (jika ada)		
Nama Institusi Mitra	:	
Alamat	:	
Penanggung Jawab	:	
Tahun Pelaksanaan	:	Tahun ke 1 dari rencana 1 tahun
Biaya Tahun Berjalan	:	Rp. 14.500.000,00
Biaya Keseluruhan	:	Rp. 14.852.000,00

Mengetahui
Ketua Lembaga Penelitian

(Y. Tyas Catur P. S.Si.,M.Kom)
NIP/NIK 0686.11.1994.046

Semarang, 9 - 10 - 2013,
Ketua Peneliti,

(WIJANARTO)
NIP/NIK0686.11.2009.354

RINGKASAN

Pemrograman dasar merupakan pondasi utama sesorang atau mahasiswa yang ingin belajar membuat program untuk menyelesaikan suatu masalah tertentu. Dalam disiplin ilmu komputer menyelesaikan masalah merupakan basis dari perkembangan keilmuan. Algoritma merupakan salah satu teknik untuk memecahkan masalah di bidang pemrograman yang di ekspresikan dalam bahasa pemrograman. Kesulitan utama seseorang dalam mengekspresikan solusi dalam bentuk bahasa formal merupakan masalah tersendiri yang harus di pecahkan, selain pemilihan alat atau aplikasi yang tepat untuk membantunya, bahkan untuk orang dengan latar belakang ilmu komputer.

Dengan demikian yang tujuan jangka panjang yang ingin di capai dalam peneitian ini adalah membuat suatu *Domain Specific Language* (DSL) untuk pengajaran pemrograman dasar. Untuk saat ini, penelitian ini menyajikan suatu alat atau aplikasi untuk mempermudah penyelesaian masalah berbasis notasi algoritmik dalam bidang pengajaran pemrograman dasar bagi mahasiswa di tahun pertama, dalam bentuk suatu editor teks yang dapat mentranslasikan notasi algoritmik ke bahasa c. Alat ini akan membantu seseorang atau mahasiswa untuk dapat mendisain solusi dalam bentuk notasi algoritmik, tanpa memikirkan kerumitan dalam bahasa yang di pilihnya. Model notasi algoritmik yang di pilih merupakan model yang sudah pernah diterapkan dan diajarkan di perguruan tinggi. Penelitian ini akan menggunakan metode *Rapid Application Development* (*RAD*) dan *Model View Cotroller* (*MVC*) dalam rangka membangun aplikasi translator notasi algoritmik dalam bentuk editor teks. Untuk mengetahui apakah aplikasi yang di hasilkan mempunyai manfaat nyata, metode eksperimen murni (*pure experimental*) tanpa pretest akan di terapkan untuk mengevaluasi subyek penelitian yang di bagi menjadi dua kelompok, yaitu subyek eksperimen dan subyek kontrol. Sedangkan analisa data di lakukan dengan Uji T, untuk mendapatkan signifikansi perbedaan subyek penelitian terhadap penggunaan translator notasi algoritmik yang di ukur berdasarkan kemampuan (nilai) dan efektifitas (waktu) penyelesaian masalah yang di berikan secara random terhadap subyek penelitian.

PRAKATA

Dengan selesainya pembuatan laporan penelitian dosen pemula ini, penulis bersyukur kehadirat Alloh SWT. Laporan penelitian ini merupakan kegiatan lanjutan dari hasil penelitan yang telah di lakukan penulis dalam rangka mencari jawaban atas kegelisahan intelektual di bidang pengajaran pemrograman, khususnya pemrograman dasar.

Pemlihan model pembelajaran pemrograman merupakan inti dari penelitian ini, yang dapat di aplikasikan dalam suatu notasi sederhana untuk pengajaran pemrograman dasar dan pembuatan media atau alat untuk memformulasikan model abstrak menjadi nyata dalam bentuk editor dan translator notasi algoritma ke dalam bahasa formal, dalam hal ini bahasa C.

Tentu saja, capaian ini bukan merupakan akhir dari kegiatan, pengujian dan penyempurnaan model akan secara berkelanjutan akan terus dilakukan oleh penulis guna mencapai suatu titik ideal. Dengan menyeberkan informasi penelitian melalui jurnal atau prosiding, penulis berharap akan mendapatkan masukan yang lebih baik kedepannya, selain itu baik model maupun alat hasil berupa editor akan di lakukan penyemurnaan hingga mendapatkan bentuk yang mudah di pakai dan di pahami oleh pemakai awal.

Akhirnya, penulis tidak lupa juga mengucapkan terima kasih yang sebesar-besarnya kepada para pihak yang telah membantu terselesaikannya laporan ini, terutama, mahasiswa dan asisten peneliti yang terlibat dalam proyek ini. Tak lupa rekan dosen dan segenap jajaran struktural di Fakultas Ilmu Komputer Universitas Dian Nuswantoro yang sangat mendukung kegiatan ini, penulis ingin menyampaikan apresiasi setinggi-tingginya.

Semarang, Oktober 2013

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1. PENDAHULUAN

1.1.Latar Belakang Masalah

Pemrograman dasar merupakan pondasi utama seseorang atau mahasiswa yang ingin belajar membuat program untuk menyelesaikan suatu masalah tertentu. Sesederhana apapun, masalah yang harus di pecahkan harus dilakukan secara terstruktur dan ilmiah. Dalam dunia ilmu komputer atau teknik informatika langkah-langkah pemecahan masalah atau metode yang logis, terstruktur dan berhingga di sebut sebagai algoritma (Blass et.al 2003, Harel et.al 2004). Seperti diketahui algoritma merupakan metode penyelesaian masalah yang umum dan banyak di lakukan hampir di seluruh bidang ilmu, seperti penentuan DNA (Ming, 2005), Teori graph dalam menentukan lintasan terpendek (Kruskal, 1956) dan masih banyak lagi.

Dalam studi yang pernah dilakukan di Afrika Selatan (Cilliers et.al., 2005), keberhasilan pembelajaran pemrograman dasar di pengaruhi oleh, (1) lingkungan belajar (alat atau aplikasi) yang mendukung notasi yang sederhana, yang dapat mengkonstruksi notasi umum untuk bahasa pemrograman, (2) penampilan visual dari struktur program harus memungkinkan mahasiswa pemrograman dasar dapat memahami semantik konstruksi program dan (3) lingkungan kerja aplikasi harus melindungi siswa untuk tidak melakukan interpretasi dan pemahaman yang salah. Di lain pihak pemahaman mahasiswa atau orang yang tertarik mempelajari pemrograman sering terkendala oleh bagaimana menggunakan bahasa itu sendiri. Artinya kesulitan utama mempelajari pemrograman di karenakan kesulitan bagaimana menggunakan memahami semantik dari suatu bahasa pemrograman, seperti di jelaskan dalam (Cilliers et.al., 2005).

Di Indonesia studi mengenai pembelajaran pemrograman dasar sangat sedikit, apalagi yang menyangkut alat penunjang atau ketepatan penggunaan aplikasinya. Dalam penelitian yang di lakukan Hidayanti (Hidayanti, 2007), 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 (Yuwono, 2009), 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, sehingga dapat mempermudah pemahanan serta mengurangi interpretasi yang salah dalam rangka menyelesaikan masalah dalam bidang pemrograman. Model sederhana yang dipakai merupakan suatu translator notasi algoritmik yang secara otomatis dapat menghasilkan suatu bahasa pemrograman tingkat tinggi yang umum (Wijanarto, 2012). Sementara notasi algoritmik yang standar yang diberikan merupakan notasi yang sudah di ajarkan di perguruan tinggi (Liem, 2007).

1.2. Rumusan Masalah

Berdasarkan paparan latar belakang sebelumnya, maka masalah yang di hadapi dan akan di angkat dalam penelitian terdiri dari :

1. Bagaimana menentukan dan membuat model grammar untuk menghasilkan notasi algoritmik ?
2. Bagaimana implementasi model grammar dalam bentuk aplikasi translator notasi algoritmik ke dalam suatu bahasa formal ?
3. Bagaimana menerapkan aplikasi translator notasi algoritmik untuk menyelesaikan masalah di bidang pemrograman dasar?.

1.3. Batasan Masalah

Tentu saja terdapat beberapa batasan dalam rangka menyelesaiakan masalah yang di kemukakan di atas, diantaranya adalah :

1. Model notasi algoritmik yang di pakai berupa notasi yang sudah di ajarkan dalam perkuliahan di perguruan tinggi (Sekolah Teknik Elektro dan Informatika ITB dan Universitas Dian Nuswantoro.
2. Aplikasi hanya mengenerate bahasa pemrograman (formal) prosedural yaitu bahasa C standar.

3. Penanganan terhadap fungsi input dan output sebatas yang umum di pelajari dalam pemrograman dasar.
4. Aplikasi hanya dapat di jalankan dengan Java SE 1.7 ke atas terinstall di komputer.

BAB 2. TINJAUAN PUSTAKA

2.1. Notasi Algoritmik

Dalam penelitian ini, notasi algoritmik yang dimaksud adalah suatu bahasa alami (*psuedocode*) yang mudah di pahami manusia dalam mengeskpresikan suatu solusi atau disain dalam suatu masalah. Walaupun demikian notasi yang dipakai kecenderungannya lebih mendekati ke suatu bahasa formal tertentu untuk mempermudah translasi. Penelitian ini menggunakan notasi algoritma dalam (Liem, 2007) dengan sedikit modifikasi oleh penulis, secara ringkas notasi algoritmik yang di maksud dapat di lihat pada tabel 1 di bawah ini. Selain alasan kemudahan notasi ini sudah di ajarkan pada STEI ITB dan Fakultas Ilmu Komputer, Universitas Dian Nuswantoro.

Tabel 1. Notasi Algoritmik Standar

Notasi Modifikasi	Notasi Standar
Program Berisi setidaknya satu deklarasi: pustaka, makro, type, variabel, konstanta, fungsi dan program utama	**Program <nama-program>** { Spesifikasi teks algoritmik secara umum } **KAMUS** { Definisi konstanta, type, deklarasi variabel, spesifikasi prosedur, fungsi } **ALGORITMA** { Teks algoritma - tidak berada di antara tanda kurung kurawal }
Pustaka **Uses** FILE	Sama
Makro **Def** [type][As harga] **IfNotDef** type **EndDef** **ElseDef**	Tidak Ada
Type (struct, enum, union, array) **Type** [nama = type] [nama : <fields>] [type : <fields>]	**{ Definisi Type Bentukan}** **type** namatype : < elemen1 : type1, elemen2 : type2, ... > **{ Deklarasi Variable }** nmvar1 : namatype nmvar2 : type1 {misal} **{ Akses Elemen }** nmvar2 ← nmvar1.elemen1 nmvar1.elemen2 ← <ekspresi>

	{ Definisi type enumerasi } **type** hari : (senin, selasa, rabu, kamis, jumat, sabtu) **{ Deklarasi variable }** H : Hari **{ Assignment }** H ← senin **{ Definisi type Array}** **type** hari : (senin, selasa, rabu, kamis, jumat, sabtu) **{ Deklarasi variable }** H : Hari **ALGORITMA { Assignment }** H ← senin
Variabel Var Nama:type[<--harga]	**Deklarasi Variable** <nama> : <type> **Inisialisasi/Assignment** <nama> ←<harga> **{ Deklarasi Variabel Array}** nm_array : array [0..nmax-1] of type-array **{ Cara Mengacu Elemen Array }** nm_array_{indeks} **Contoh:** TabInt : array [0..99] of integer TabInt ← 1 X ← $TabInt_{10}$
Konstanta Constant Nama:type [<--harga]	**Deklarasi Constant** constant <nama>:<type>=<harga>
Statements [forStat] [ifstat] [switchstat] [exprs;] [block] [assignStat ;] [proc_callStat] [assignFunction] [statIO ;] [;]	**Assignment Statement** <nama1>←<nama2> <nama> ←<konstanta> <nama> ← <ekspresi> **Expression Statement** nama1 ← nama1 <opr> nama2
Block {* Variabel statements Multi Komentar Komentar	Tidak ada

Whitespace *}	
Input/Output **Input**(text,[exprs]) **Output**(text,[exprs])	input(<list-nama>) **Contoh:** input (X) {x integer} input (X, Y) input (F) {F real} input (s) {s string} output(<list-nama>) **Contoh:** output (X) {x integer} output (X, Y) output ("Contoh output") output ("Namaku: ", nama) output (F) {F real} output (CC) {c character}
Kondisi(ifstat, switchstat) **if** <kondisi>**then** <statements>[**else** statments] **Depend on** <nama> <* harga : statements else : harga : statements *>	**Satu Kasus:** if kondisi then aksi **Dua Kasus Komplementer:** if kondisi-1 then aksi-1 else { not kondisi-1 } aksi-2 **Analisa > 2 kasus** depend on nama kondisi-1 : aksi-1 kondisi-2 : aksi-2 ... kondisi-n : aksi-n else : aksi-else
Pengulangan (forStat) nama **Traversal** range [step harga] **Do** [statements] **While** <exprs> **Do** [statements] **Repeat** [statments] **Until** <exprs>	**Pengulangan berdasarkan pencacah:** i traversal [Awal..Akhir] Aksi **Pengulangan berdasarkan kondisi berhenti:** repeat Aksi until kondisi-stop **Pengulangan berdasarkan kondisi ulang:** while (kondisi-ulang) do Aksi {not kondisi-ulang} **Pengulangan berdasarkan dua aksi:** iterate Aksi-1 stop kondisi-stop Aksi-2
Subprogram **Procedure**<nama>(In/Out args) **Function**<nama>(args):type	**function** NAMAF (param1 : type1, param2 : type2, …) → type-hasil { Spesifikasi fungsi } **KAMUS LOKAL** { Semua nama yang dipakai dalam algoritma dari fungsi } **ALGORITMA** { Deretan fungsi algoritmik:

	pemberian harga, input, output, analisis kasus, pengulangan } { Pengiriman harga di akhir fungsi, harus sesuai dengan typehasil } → **hasil** **Pemanggilan fungsi** nama ← **NAMAF ([list parameter aktual])** output (**NAMAF ([list parameter aktual]))** **procedure** NAMAP (input nama1 : type1, input/output nama2 : type2, output nama3 : type3) { Spesifikasi, Initial State, Final State } **KAMUS LOKAL** { Semua nama yang dipakai dalam BADAN PROSEDUR } **ALGORITMA** { BADAN PROSEDUR } { Deretan instruksi pemberian harga, input, output, analisis kasus, pengulangan atau prosedur } **Pemanggilan prosedur** NAMAP(paramaktual1,paramaktual2,paramaktual3)

2.2. Translator

Translator merupakan proses translasi input kode sumber menjadi output program yang dapat di eksekusi (Aho et.al, 2007., Aho and Ullman, 1973), melalui analisa lexical dan pemaknaan semantik, yang terdiri dari parser dan lexer yang di hasilkan oleh grammar dengan string template (ST). Baik lexer dan parser di hasilkan oleh ANTLR dan String Template (ST) (Parr et.al, 2011., Parr, 2010), serta Java sebagai target bahasa generator. Dengan demikian yang bertindak sebagai translator dan generator adalah, parser dan lexer yang di generate dari ANTLR dan ST dalam bentuk class dalam bahasa java.

2.3. Grammar

Grammar merupakan aturan kontekstual suatu sintak dengan terdapat semantik didalamnya dari suatu bahasa formal. Sintak yang di gunakan dalam penelitian ini menggunakan BNF (*Backus-Naur Form*) atau Extended BNF, karena kemudahan notasinya (Aho et.all, 2007., Appel, 1998., Watt et.all 2000), yang terdiri dari *himpunan berhingga simbol terminal, simbol non terminal, simbol awal dan*

aturan produksi $N ::= \alpha | \beta$, dimana **N** adalah simbol *non terminal* , ::= berarti *terdiri dari* serta α adalah *string terminal* atau *non terminal* yang mungkin kosong serta simbol | yang berarti *alternatif*, himpunan tadi di sebut sebagai *context-free grammar*, singkatnya *grammar*.

Suatu grammar menentukan abstraksi sintak dalam suatu himpunan *Abstract Syntax Tree* (AST), tiap simpul non terminal dari AST mempunyai label aturan produk yang berlaku dan grammar tidak menghasilkan suatu kalimat untuk simbol terminal yang tidak berperan dalam abstraksi sintak. Dalam implementasi penelitian ini grammar yang di terapkan adalah seperti dalam ANTLR (Par et.al, 2011).

2.4. Pemrograman Dasar

Pemrograman merupakan proses belajar memprogram untuk memecahkan masalah dengan metode dan sistematika tertentu lalu mengeksprsikannya dalam suatu bahasa formal, sehingga belajar pemrograman tidak sama dengan belajar bahasa pemrograman (Liem, 2007). Dari sudut pandang aspek inilah konsep pemrograman yang di pakai dalam penelitian ini menjadi sangat penting. Kerangka pikir mahasiswa di arahkan kepada bagaimana menyelesaikan suatu masalah tanpa harus di repotkan dengan menguasai bahasa tertentu. Notasi algoritmik merupakan salah satu solusi yang mencoba membuat standar bebas bahasa namun tetap menyentuh pada metode dan sistematika penyelesaian masalah di bidang ilmu komputer secara lebih sederhana (natural).

2.5. Text Editor

Hampir seluruh bahasa pemrograman dapat di pastikan menggunakan editor berbasis teks (untuk pemodelan, biasanya menggunakan bahasa pemodelan yang cenderung visual). Sulit untuk menemukan referensi standar mengenai editor text, namum secara umum editor teks adalah suatu lingkungan kerja yang di gunakan untuk mengolah data jenis text dan bukan lainnya (numerik, data terstruktur, data multimedia) (Wikipedia, 2013). Terdapat perbedaan pengolah text biasa, seperti aplikasi pengolah kata (MSWord (Microsoft (TM)), LibreOffice, openOffice (open source)), sementara editor text yang di maksud dalam penelitian ini adalah ditujukan untuk mengolah teks notasi algoritmik yang akan di translasikan menjadi kode

sumber bahasa c (seperti, notepad, wordpad), dan hanya dapat menerima input berupa data teks sederhana, tanpa memproses data visual.

BAB 3. TUJUAN DAN MANFAAT PENELITIAN

3.1 Tujuan Penelitian

Tujuan dalam penelitian yang akan di capai adalah menghasilkan aplikasi atau tool (editor teks) yang mampu mentranslasikan notasi algoritmik standar dalam bahasa C, yang di bagi menjadi :

1. Menentukan dan membuat model grammar untuk menghasilkan notasi algoritmik.
2. Mengembangkan aplikasi translator dari model grammar dari notasi algoritmik yang di pilih untuk menghasilkan bahasa c.
3. Mempermudah penyelesaikan masalah di bidang pemrograman dasar melalui translator notasi algoritmik ke bahasa c.

3.2 Manfaat Penelitian

Manfaat yang di harapkan dari penelitian ini adalah :

1. Mengasilkan model grammar untuk notasi algoritmik
2. Menghasilkan aplikasi translator editor teks yang di gunakan untuk mentranslasikan notasi algoritmik ke bahasa c.
3. Membantu mahasiswa atau orang yang tertarik di bidang pemrograman untuk fokus belajar bagaimana menyelesaikan masalah dengan notasi algoritmik standar yang sederhana tanpa memikirkan bahasa pemrograman yang terkesan rumit dan membingungkan.

3.3 Luaran Penelitian

Luaran dari penelitian ini yaitu menghasilkan suatu aplikasi editor teks yang dapat mentranslasikan notasi algoritmik ke bahasa c yang di beri nama Translator Notasi Algoritmik, dengan kemampuan seperti editor pada umumnya, code completion, Syntax Highlight, penanganan kesalahan sintak, undo dan redo, dan kemampuan lainnya.

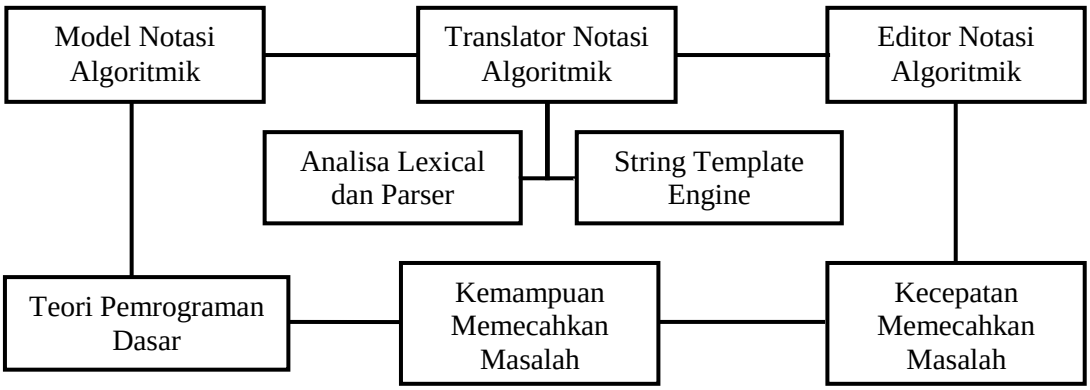
3.4 Kontibusi Penelitian

Kontribusi dari alat ini akan mendukung proses belajar pemrograman dasar dengan model notasi algoritmik yang secara berkelanjutan akan di sempurnakan untuk mencapai hasil optimal. Sehingga juga membawa dampak pada pengembangan

model grammar yang harus di sesuaikan dan dalam ilmu bahasa pemrograman, model ini dapat menjadi studi yang di kenal dengan domain specific language (DSL), yaitu studi bahasa pemrograman (textual atau visual) yang di kembangkan dengan domain yang khusus dan untuk keperluan khusus pula dalam disiplin ilmu komputer.

3.5 Kerangka Pikir

Belajar pemrograman tidak sama dengan belajar program, pemrograman merupakan teknik yang di gunakan untuk menyelesaikan masalah dengan komputer yang di jembatani dengan suatu bahasa natural dalam penyampaiannya. Bahasa natural di sini bukanlah seperti bahasa pada umumnya, misal bahasa Jawa, Sunda atau Inggris, namun merupakan bahasa yang di mengerti manusia dan mudah untuk di terjemahkan menjadi bahasa formal (bahasa formal adalah bahasa yang hanya di mengerti oleh komputer). Dengan demikian dalam 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 di jelaskan pada gambar 1 di bawah ini,



Gambar. 1 Kerangka Pikir

Penggunaan editor notasi algoritmik di harapkan mampu mempermudah mahasiswa dalam memecahkan masalah dengan efektif dalam pembelajaran pemrograman dasar, tanpa belajar bahasa program. Dengan demikian pengujian terhadap editor notasi algoritmik sangat perlu di lakukan. Hasil uji penggunaan editor di harapkan akan memberikan kepastian perbedaana kecepatan pemecahan masalah pemrograman tanpa tahu bahasa program yang di pakai.

3.6 Hipotesa Penelitian

Berdasakan kajian teoritis dan kerangka pikir di atas, maka dapat di ajukan hipotesa sebagai berikut :

1. Hipotesa Nol (Ho)
    a. 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.
    b. Penggunaan translator notasi algoritmik tidak lebih cepat memecahkan masalah pemrograman dibandingkan dengan tanpa menggunakan translator notasi .
2. Hipotesa Alternatif (Ha)
    a. Terdapat perbedaan kecepatan menyelesaiakan masalah pemrograman yang signifikan antar kelompok yang menggunakan translator notasi algoritmik dan kelompok yang tanpa menggunakan translator notasi algoritmik.
    b. Penggunaan translator notasi algoritmik lebih cepat memecahkan masalah pemrorgaman dibandingkan tanpa menggunakan translator notasi algoritmik dalam.

BAB 4. METODE PENELITIAN

4.1 Teknik Penelitian

Dalam penelitian ini, kami akan menggunakan beberapa teknik umum guna mencapai tujuan penelitian. Penulis akan membagi dua bagian besar yaitu :

1. Metode rekayasa perangkat lunak, yang di gunakan untuk membangun translator notasi algoritmik menjadi suatu aplikasi editor text. Metode rekayasa perangkat lunak yang akan dilakukan terdiri dari penentuan model notasi standar agoritmik, dilanjutkan dengan membuat disain arsitektur sistem, pembangunan sistem dengan metode RAD (Rapid Application Development) dan MVC (Model View Controller).
2. Metode eksperimen, yang di gunakan untuk mengevaluasi apakah hasil implementasi aplikasi bermanfaat untuk mahasiswa yang menggunakannya. Metode evaluasi dalam implementasi yang di pilih adalah metode eksperimen dengan disain kelompok kontrol tanpa pretest (*Posttest Only with Control Group*).

4.2 Model Penelitian

4.2.1 Model Notasi Algoritmik

Menentukan model standar standar notasi algoritmik merupakan jantung dari penelitian ini, di karenakan model ini merupakan kerangka utama dari aplikasi yang akan di hasilkan. Model notasi yang di pilih merupakan model notasi dalam (Wijanarto, 2012). Secara umum arsitektur model grammar yang di pakai adalah seperti dalam gambar 2 sebagai berikut :

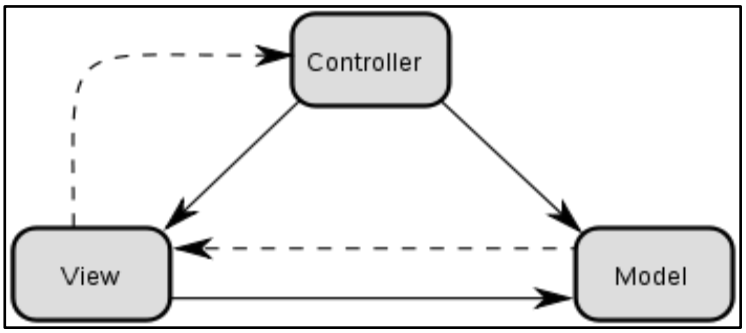
Gambar 2. Model Translator Notasi ke Bahasa C

Seperti dalam (Wijanarto, 2012), model terdiri dari 3 buah langkah yaitu Notasi Algoritmik seperti pada tabel 1, yang berupa bahasa yang mudah di pahami manusia (natural) untuk mengekspresikan disain solusi suatu masalah pemrograman yang

merupakan input yang akan di proses oleh translator dan akan menghasilkan (menggenerate) bahasa formal (bahasa C).

4.2.2 Model View Controller

Metode MVC (Model View Controller) berbasis pada paradigma object oriented. Model atau pendekatan ini pertama kali di sajikan dalam suatu laporan teknis yang di keluarkan oleh Xerox (Reenskaug, 1979) dan dalam perkembangannya pendekatan ini banyak di pakai dalam pengembangan sistem khususnya yang berbasis pada paradigma obyek oriented (Stanchfield, 2009). Metode MVC terdiri dari urutan langkah seperti pada gambar 3 di bawah ini

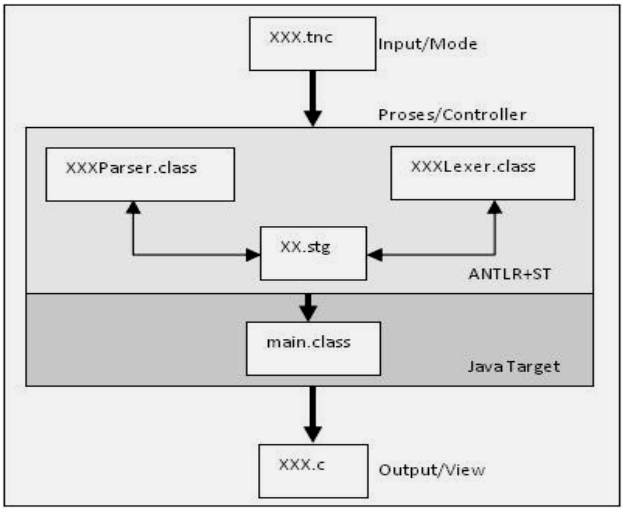


Gambar 3. Model MVC

Model mewakili bagian aplikasi yang menyimpan data dan menyediakan method untuk aksi, ***View*** merupakan bagian aplikasi yang menghasilkan data bagi user dan ***Controller*** adalah bagian yang menerima input dari user dan keperluan modifikasi pada model. Dengan menggunakan metode MVC (Stanchfield, 2009) di samping teknik ini sangat cepat, namun di perlukan keahlian dalam proses pengembangan aplikasi berbasis Java yang multiplatform.. Untuk membuat model grammar, kami memilih model EBNF (Extended Backus Naur Form) yang di generate dengan tool ANTLR. Sedang dalam rangka mentranslasikan grammar ke bahasa kami menggunakan String Template, di samping karena kemudahannya, tool dapat di integrasikan dengan ANTLR. Terakhir untuk menintegrasikan grammar dan template akan di bangun di atas Java yang menghasilkan editor text.

4.2.3 Perancangan dan Pembangunan Arsitektur Sistem

Suatu sistem aplikasi di kembangkan dengan suatu metode atau cara yang beragam, penelitian ini akan menggunakan dua pendekatan dalam mengembangkan aplikasi yaitu Rapid Application Development (RAD) dan Model View Contrller (MVC). Adapun rancangan arsitektur secara umum sebagai kerangka pikirnya adalah seperti gambar 4 sebagai berikut :



Gambar 4. Rancangan Arsitektur MVC Translator Notasi Algoritmik

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.

4.2.4 Rapid Application Development

Teknik pembangunan sistem yang di gunakan dengan pendekatan object orinted programming, dengan teknik Rapid application development (RAD). Disamping karena kemudahannya, teknik ini juga sangat cepat dalam membangun sistem skala menengah ke atas. Fase pengembangan sistem dengan metode RAD di bagi menjadi: (1) Fase Planning, untuk menentukan tujuan, fungsionalitas dan scope

yang akan di kerjakan, (2) Fase User Design, yaitu menentukan interface dan bagaimana system akan bekerja dalam bentuk prototype, (3) Fase Construction, Prototype di konversi menjadi aplikasi yang sudah berfungsi, dengan pengkodean dan pengembangan fungsionalitas aplikasi, (4) Fase Cutover, merupakan fase terakhir dimana kegiatan utamnya adalah mencoba pada pemakai dan mendidik para pemakai (Sommerville, 2011).

4.3 Implementasi Aplikasi

Dalam rangka mencapai tujuan penelitian yaitu menghasilkan aplikasi translator notasi agoritmik, kami akan mengimplementasikan penelitian sebagai berikut :

1. Menentukan dan membuat grammar untuk notasi algoritmik untuk menghasilkan suatu kerangka kerja translator notasi algoritmik ke suatu bahasa.
2. Merancang desain *input/output* translator notasi algritmik yang mudah digunakan untuk mahasiswa dengan metode RAD dalam kerangka MVC.
3. Mengimplementasikan rancangan translator notasi algoritmik dalam bentuk editor teks dengan menggunakan bahasa pemrograman *Java SE 1.7 dengan Integrated Deveopment Environtment Eclipse Juno.*

4.4 Pendekatan Penelitian

Pendekatan penelitian yang di pilih dalam penelitian ini adalah penelitian kuantitatif, artinya penelitian yang di lakukan untuk mencari data kuantitatif melalui hasil uji coba eksperimen. Data yang digunakan untuk menganalisis pendekatan kuantitatif ini adalah data berupa angka. Data dalam penelitian kuantitatif adalah berupa angka-angka (Nurgiyantoro, 2001), pendekatan kuantitatif dengan alasan semua gejala yang diamati dapat diukur dan diubah dalam bentuk angka serta dapat dianalisis dengan analisis statistik. Penelitian ini bertujuan untuk menguji suatu teori yang menjelaskan hubungan teori yang ada dengan kenyataan.

Proses pendekatan mengikuti proses berpikir deduktif. Berpikir deduktif, yaitu diawali dengan penentuan konsep yang abstrak berupa teori yang sifat-sifatnya masih umum kemudian dilanjutkan dengan pengumpulan bukti-bukti atau kenyataan untuk pengujian.

4.5 Desain dan Paradigma Penelitian

Dalam metode experimental terdapat tiga jenis rancangan penelitian yaitu (1) Pra-experimen adalah suatu rancangan yang bertujuan mengungkap hubungan sebab-akibat yang melibatkan satu kelompok subyek tanpa kontrol terhadap variabel lainnya(Sugiyono, 2010), (2) eksperimen semu (quasi eksperimental), merupakan rancangan yang mengungkap hubungan sebab akibat dengan melibatkan satu kelompok kontrol dan satu kelompok eksperimen (Sukmadinata et.al., 2007) dan (3) eksperimen murni, rancangan yang melibatkan satu variabel experimen yang berkaitan di beri suatu *treatement* khusus dan satu kelompok kontrol dengan perlakuan yang berbeda setelah menguji hasil (Nasution, 2007).

Penelitan ini menggunakan rancangan eksperimen tanpa pretest (*posttest only with control group*), karena kemudahannya dalam memberi perlakuan khusus terhadap kelompok eksperimen yaitu dengan menguji translator notasi algoritmik untuk mengerjakan masalah yang di berikan dan melakukan kontrol terhadap kelompok lainnya. Penggunaan desain ini hanya melakukan postes baik terhadap kelompok experimen maupun terhadap kelompok kontrol. Penempatan subjek dalam experimen maupun terhadap kelompok kontrol kelompok masing-masing dilakukan dengan penugasan acak, tabel 2 di bawah ini menjelaskan disain kelompok tanpa pretest.

Tabel 2. Disain Subyek Acak Kelompok Eksperimen dan Kontrol

	Kelompok	**Variabel terikat**	**Posttest**
A	Eksperimen	X	Ye
A	Kontrol	-	Yk

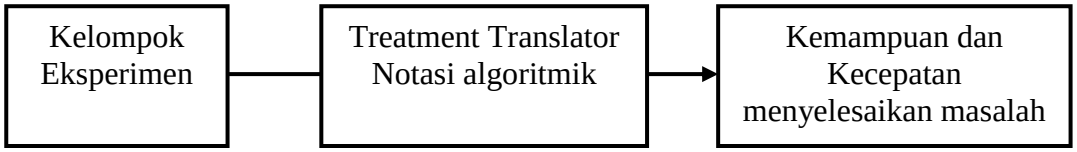
Tabel 2 akan menugaskan setiap subyek pada kelompok eksperimen dan kelompok kontrol secara acak (A). Kemudian hanya melaksanakan perlakuan pada kelompok eksperimen (tanda X) sedangkan pada kelompok kontrol tidak di perlakukan (tanda -), dan kemudian akan melaksanakan posttest pada kelompok eksperimen (Ye) dan kelompok kontrol (Yk), untuk menentukan perbedaan rata-rata nilai yang di peroleh pada Ye dan Yk dengan metode statistik (Uji T), sehingga dapat menentukan signifikansi perbedaan kedua kelompok tersebut.

4.5.1 Paradigma Penelitian

Paradigma penelitian adalah pola pikir yang menunjukkan hubungan antara variabel yang akan diteliti yang sekaligus mencerminkan jenis dan jumlah rumusan masalah yang perlu dijawab melalui penelitian, teori yang digunakan untuk merumuskan hipotesis, jenis dan jumlah hipotesis, dan teknik analisis statistik yang akan digunakan (Sugiyono, 2010).

Paradigma yang digunakan dalam penelitian ini adalah paradigma sederhana. Paradigma sederhana terdiri atas satu variabel independen dan dependen (Sugiyono, 2010). Paradigma dalam penelitian ini dapat digambarkan pada gambar 5 dan 6, sebagai berikut.

a. Paradigma Kelompok Eksperimen



Gambar 5 Paradigma Kelompok Eksperimen

b. Paradigma Kelompok Kontrol

Gambar 6 Paradigma Kelompok Kontrol

Dari gambar paradigma penelitian di atas, variabel penelitian yang telah ditetapkan dikenal paska-uji 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.

4.6 Variabel Penelitian

Dalam penelitian eksperimen kuasi, (Arikunto, 2006) mengatakan bahwa objek penelitian atau apa saja yang menjadi titik perhatian suatu penelitian disebut

sebagai variabel. Variabel dalam penelitian ini diklasifikasikan menjadi dua, yaitu variabel bebas (X) dan variabel terikat (Y). Variabel dalam penelitian ini dapat dilihat sebagai berikut.

1. Variabel bebas

Variabel bebas merupakan variabel yang mempengaruhi atau yang menjadi sebab perubahannya atau timbulnya variabel terikat (Sugiyono, 2010). Variabel bebas dalam penelitian ini adalah penggunaan translator notasi algoritmik untuk menyelesaikan masalah pembelajaran pemrograman dasar.

2. Variabel terikat

Variabel terikat adalah variabel yang dipengaruhi atau yang menjadi akibat, karena adanya variabel bebas. Variabel terikat dalam penelitian ini adalah kemampuan dan kecepatan mahasiswa dalam menyelesaikan masalah pembelajaran pemrograman dasar.

4.7 Tempat dan Waktu Penelitian

1. Tempat Penelitian

Penelitian ini mengambil lokasi di Laboratorium Dasar Fakultas Ilmu Komputer Universitas Dian Nuswantoro Semarang , dengan studi banding di Laboratosium Dasar STEI ITB Bandung. Kelas yang di ambil sebagai obyek penelitian adalah mahasiswa tahun pertama, Program studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro, Semarang.

2. Waktu Penelitian

Pelaksanaan penelitian dilakukan pada jam praktikum mata kuliah pemrograman dasar supaya siswa mengalami suasana pembelajaran seperti biasanya. Penelitian ini dilakukan pada tanggal 16 September 2013 sampai dengan tanggal 27 September 2013. Penelitian ini dilakukan dalam beberapa tahap, yaitu: 1) tahap perlakuan kelompok kontrol dan kelompok eksperimen tanpa menggunakan translator notasi algoritmik, 2) tahap perlakuan kelompok eksperimen dengan menggunakan translator algoritmik. Proses pengumpulan data dapat diamati melalui table 3 di bawah ini.

Tabel 3. Jadwal Pengambilan Data Penelitian

No	Hari / Tanggal	Kegiatan	Kelas	Sesi ke-
1	Senin, 16 September 2013	Perlakuan 2 Kelompok Eksperimen	A11.42	5, 6, 7
2	Senin, 16 September 2013	Perlakuan 1 Kelompok Kontrol	A11.41	2, 3, 4
3	Rabu, 18 September 2013	Perlakuan 2 Kelompok Eksperimen	A11.41	3, 4, 5
4	Kamis, 19 September 2013	Perlakuan 1 Kelompok Kontrol	A11.42	6, 7, 8
5	Senin, 23 September 2013	Pascates Kelompok Eksperimen	A11.41	2, 3, 4
6	Senin, 23 September 2013	Pascates Kelompok Kontrol	A11.42	5, 6, 7

4.8 Populasi dan Sampel Penelitian

Dalam Arikunto (Arikunto,2006) menyatakan bahwa populasi adalah kesuluruhan subjek penelitian. Begitu juga dalam Sugiyono (Sugiyono, 2011), populasi adalah wilayah generalisasi yang terdiri atas subyek atau obyek yang mempunyai kualitas dan karakteristik tertentu yang ditetapkan oleh peneliti untuk dipelajari dan kemudian ditarik kesimpulannya. Populasi dalam penelitian ini adalah mahasiswa tahun pertama kelompok A11.41 dan A11.42 program studi Teknik Informatika FIK UDINUS. Terdapat kelas pararel pada mahasiswa pemrograman dasar tahun pertama di program studi teknik infrmatika FIK UDINUS, yaitu sebanyak 15, masing-masing kelas terdiri dari 30 sampai 40 siswa.

Sampel adalah bagian dari jumlah dan karakteristik yang dimiliki oleh populasi. Bila populasi besar dan peneliti tidak mungkin mempelajari semua yang ada pada populasi, misalnya karena keterbatasan dana, tenaga, dan waktu, maka peneliti dapat menggunakan sampel yang diambil dari populasi itu (Sugiyono, 2010). Pada penelitian ini dibutuhkan sampel sebesar dua kelas, sampel yang nantinya akan diambil adalah dua kelas yang harus benar-benar representatif. satu kelas sebagai kelompok eksperimen yaitu pada kelas A11.41, dan satu kelas sebagai kelompok kontrol, yaitu kelas A11.42.

4.9 Alat dan Teknik Pengumpulan Data

1. Instrumen Pengumpulan Data

Teknik pengumpulan data dilakukan dengan tes. Tes dilakukan pada saat praktikum dasar pemrograman. Selanjutnya, pascates digunakan untuk mengetahui prestasi kemampuan akhir mahasiswa. Pascates dilakukan untuk mengetahui

kemampuan dan kecepatan mahasiswa setelah mendapat perlakuan. Pascates ini dilakukan pada kelompok kontrol dan kelompok eksperimen. Pembelajaran dilaksanakan di dalam kelas dan materi yang diambil adalah Fungsi dan Prosedur dalam mata pemrograman dasar.

2. Pengembangan Instrumen Penelitian

Instrumen adalah suatu alat yang digunakan untuk mengukur fenomena alam maupun sosial yang diamati. Secara spesifik fenomena tersebut adalah variabel yang diamati. Instrumen yang digunakan dalam penelitian ini berupa tes menyelesaiakan masalah pemrograman dasar yang berfungsi mengukur kemampuan dan kecepatan mahasiswa dalam menyelesaikan masalah emrograman dasar pada mahasiswa tahun pertama di program studi teknik informatika FIK UDINUS. Tes ini berupa menyelesaiakan masalah sederhana yang di kerjakan kelompok kontrol dan kelompok eksperimen. Berikut kisi-kisi instrumen tes Fungsi dan Prosedur.

Tabel 4. Kisi-kisi Instrumen Tes Masalah Fungsi dan Prosedur

No	Pokok Bahasan	Indikator	Jenis
1	Deklarasi Prototype Fungsi dan prosedur	Mahasiswa mampu mengidentifikasi dan mendefinisikan spesifikasi kebutuhan fungsi dan prosedur (Nama, argumen input/output, hasil balik dan tipe fungsi)	Definisi dan Spesifikasi Fungsi atau prosedur
2	Algoritma penyelesaian masalah dalam badan fungsi dan prosedur	Mahasiswa mampu membuat penyelesaian masalah dengan notasi algoritmik untuk menghasilkan penyelesaian yang efisien dan cepat.	Algoritma penyelesaian yang tepat
3	Hasil balik fungsi dan output prosedur	Mahasiswa mampu menghasilkan solusi output yang valid untuk input yang valid pula.	Output penyelesaian yang valid

4	Kecepatan waktu pengerjaan	Mahasiswa dapat menyelesaikan permasalahan pemrograman dasar dalam waktu yang di batasi	Waktu penyelesaian masalah dalam 60 menit

Adapun pedoman penilaian yang dipakai untuk instrumen penelitian ini berupa spesifikasi dan defnisi permintaan masalah yang harus di buat, output yang di inginkan dari masalah yang ada, penggunaan standar dasar algoritma, tipe data dan efisiensi penyelesaian masalah (algoritma) yang berkaitan dengan model matematika umum. Perancangan pedoman penilaian ini juga telah melalui proses *expert judgement. Expert judgement* dalam penelitian ini adalah team laboratorium pemrograman yang di pimpin oleh Wijanarto, M.Kom dan tim laboratrium dasar FIK selaku dosen pemrograman dasar. Pedoman penilaian masalah pemrograman dasar dapat dilihat sebagai berikut.

Tabel 5. Rubrik Penilaian Masalah Pemrograman Dasar

Keterangan Penilaian		**Nilai**
Prototype	• Menulis Fungsi atau Prosedur dengan spesifikasi dan definisi sangat jelas	40-50
	• Menulis Fungsi atau Prosedur dengan spesifikasi dan definisi cukup jelas	30-39
	• Menulis Fungsi atau Prosedur dengan spesifikasi dan definisi kurang jelas	20-29
	• Menulis Fungsi atau Prosedur dengan spesifikasi dan definisi tidak jelas	10-19
	• Tidak menulis Fungsi atau Prosedur	0 - 9
Algoritma	• Menggunakan struktur algoritma, tipe data yang tepat, variabel yang sangat	40-50
	• Menggunakan struktur algoritma, tipe data yang tepat, variabel yang cukup	30-39
	• Menggunakan struktur algoritma, tipe data yang tepat, variabel yang kurang	20-29
	• Menggunakan struktur algoritma, tipe data yang tepat, variabel yang tidak	10-19
	• Tidak menggunakan struktur dasar algoritma	0 - 9
Validitas Ouput	• Menghasilkan validitas output sangat tepat dan presisi	40-50
	• Menghasilkan validitas output cukup tepat dan presisi	30-39
	• Menghasilkan validitas output kurang tepat dan presisi	20-29
	• Menghasilkan validitas output tidak tepat dan presisi	10-19
	• Tidak menghasilkan output	0 - 9
Waktu	• Menyelesaikan dalam waktu sangat cepat, kurang dari 30	40-50
	• Menyelesaikan dalam waktu cukup cepat, antara 30 hingga 40 menit	30-39
	• Menyelesaikan dalam waktu kurang cepat, antara 40 hingga 50 menit	20-29
	• Menyelesaikan dalam waktu tidak cepat, antara 50 hingga 60 menit	10-19
	• Tidak selesai dalam waktu yang di tentukan, lebih dari 60 menit	0 - 9

4.10 Uji Validitas Instrumen

Validitas yaitu suatu ukuran yang menunjukkan tingkat kevalidan suatu instrumen. Suatu instrumen yang valid mempunyai tingkat validitas tinggi dan begitu juga sebaliknya apabila instrumen tidak valid maka validitasnya rendah (Arikunto, 2006).

Dalam penelitian ini, instrumen yang digunakan adalah tes menulis, maka validitas yang digunakan adalah validitas isi (*content validity*). Validitas ini digunakan untuk mengetahui seberapa instrumen tersebut telah mencerminkan isi yang dikehendaki. Soal tes masalah pemrograman sesuai dengan materi yang digunakan dalam Kurikulum Program Studi Teknik Informatika FIK UDINUS dan STEI ITB, khususnya matakuliah dasar pemrograman tahun pertama. Selain itu, instrumen yang digunakan dalam pembelajaran pemrograman dasar di lakukan konsultasikan terlebih dahulu pada ahlinya (*expert judgement*).

4.11 Uji Reabilitas Instrumen

Reliabilitas menunjuk pada suatu pengertian bahwa suatu instrumen dapat dipercaya untuk digunakan sebagai alat pengumpul data (Arikunto, 2006:154). Kriteria keterpercayaan tes menunjuk pada pengertian tes mampu mengukur secara konsisten sesuatu yang akan diukur dari dari waktu ke waktu.

Koefisien reliabilitas dalam penelitian ini menggunakan penghitungan rumus *Alpha Crombach*. Penghitungan rumus tersebut menggunakan bantuan komputer program SPSS 20. Pengujian reliabilitas dilaksanakan sebelum tes awal menulis dongeng kelas eksperimen dan kontrol dimulai. *Alpha Cronbach* dapat dipergunakan dengan baik untuk instrumen yang jawabannya berskala, maupun jika dikehendaki yang bersifat dikhotomis (Nurgiyantoro, 2009), Oleh karena itu, *Alpha Cronbach* juga dipergunakan untuk menguji reliabilitas pertanyaan-pertanyaan atau soal-soal esai. Pada penelitian ini ada dua instrumen yang harus diukur reliabilitasnya, yaitu solusi masalah dan waktu pengerjaan solusi. Hasil perhitungan koefisiensi reliabilitas dengan *Alpha Cronbach* tersebut diinterpretasikan dengan tingkat keandalan koefisiensi korelasi sebagai berikut.

Tabel 6. Rubrik Penilaian Masalah Pemrograman Dasar

Antara 0,800 sampai 1000 adalah sangat tinggi
0,600 sampai 0,799 adalah tinggi
0,400 sampai 0,599 adalah cukup
0,200 sampai 0,399 adalah rendah

0,000 sampai 0,179 adalah sangat rendah (Arikunto, 2006). Berdasarkan perhitungan yang telah dilakukan dengan bantuan program SPSS 15.0 di dapatkan koefisien reliabilitas isian masalah pemrograman dasar sebesar 0,883. Berdasarkan hasil tersebut, maka dapat disimpulkan bahwa instrumen tersebut memiliki indeks reliabilitas yang sangat tinggi. Hasil perhitungan dapat dilihat pada lampiran.

4.12 Prosedur Penelitian

Prosedur penelitian yang digunakan dalam penelitian ini adalah sebagai berikut.

4.12.1. Pelaksanaan (*Treatment*)

Kedua kelompok dianggap memiliki kondisi yang sama dan tanpa diberikan prates, dan selanjutnya akan diadakan *treatment* (perlakuan). Tindakan ini dengan menggunakan translator notasi algoritmik, mahasiswa dan peneliti. Peneliti berperan sebagai pengamat yang mengamati secara langsung proses perlakuan. Pada tahap ini, ada perbedaan perlakuan antara kelompok eksperimen dan kelompok kontrol. Dalam pembelajaran penyelesaian masalah pemrograma dasar, kelompok eksperimen diberi perlakuan dengan menggunakan translator notasi algoritmik, sedangkan kelompok kontrol tidak mendapatkan perlakuan tersebut. Adapun tahap-tahap eksperimen adalah sebagai berikut.

a) Kelompok Eksperimen

Kelompok eksperimen akan diberi perlakuan dengan menggunakan translator notasi algoritmik sebanyak tiga perlakuan. Mahasiswa menyelesaiakan masalah dengan menggunakan translator notasi algoritmik. Berikut langkah-langkah pembelajaran penyelesaian masalah dalam pemrograman dasar pada kelompok eksperimen.

1) Perlakuan Pertama

Baik kelompok eksperimen maupun kontrol di asumsikan dalam keadaan yang sama (tanpa pretest) dan mendapat perlakuan yaitu tanpa menggunakan translator notasi algoritmik. Proses *treatment* dengan tidak menggunakan translator notasi algoritmik melalui langkah- langkah sebagai berikut.

(a) Mahasiswa mendapat soal dan memahami spesifikasi dan definisi soal.
(b) Mahasiswa membuat disain kasar penyelesaian soal mlealui kertas.
(c) Mahasiswa mulai mengerjakan dengan kompiler standar
(d) Mahasiswa melakukan debugging dan pengujian.
(e) Peneliti mencatat hasil dan waktu penyelesaian pekerjaan mahasiswa.

2) Perlakuan kedua

Dalam pertemuan kedua kelompok eksperimen mendapatan perlakuan dengan menggunakan translator notasi algoritmik. Proses *treatment* untuk kelompok eksperimen dengan menggunakan translator notasi algoritmik adalah sebagai berikut.

(a) Mahasiswa mendapat soal dan memahami spesifikasi dan definisi soal.
(b) Mahasiswa membuat disain kasar penyelesaian soal mlealui kertas.
(c) Mahasiswa mulai mengerjakan dengan translator notasi algoritmik
(d) Mahasiswa melakukan debugging dan pengujian.
(e) Peneliti mencatat hasil dan waktu penyelesaian pekerjaan mahasiswa.

b. Kelompok Kontrol

Kelompok kontrol mendapatkan pembelajaran masalah pemrograman dasar yang dilaksanakan tanpa menggunakan translator notasi algoritmik (perlakuan 1) tetapi menggunakan apa yang biasanya digunakan sebelumnya yaitu kompiler standar dalam menyelesaikan masalah di bidang pemrograman dasar.

4.12.2. Pengukuran Sesudah Eksperimen *(Post-Experiment Measurement)*

Langkah mahasiswa setelah mendapat perlakuan, kelompok eksperimen dan kelompok kontrol diberi pascates. Tes ini bertujuan untuk melihat pencapaian peningkatan kemampuan memecakan masalah pemrograman dasar setelah diberi perlakuan dengan menggunakan translator notasi algoritmik dan yang tidak diberi perlakuan dengan menggunakan translator notasi algoritmik. Pascates juga

digunakan untuk membandingkan nilai yang dicapai mahasiswa sama, meningkat, atau menurun.

4.13 Teknik Analisis Data

Dalam penelitian ini, analisis data menggunakan rumus Uji-t dan gain skor. Uji-t dimaksudkan untuk menguji rata- rata hitung di antara kelompok-kelompok tertentu (Nurgiyantoro, 2009). Uji-t dalam penelitian ini digunakan untuk menguji perbedaan rata-rata hitung, apakah ada perbedaan signifikan atau tidak antara kelompok eksperimen dengan kelompok kontrol. Syarat data bersifat signifikan apabila nilai p lebih kecil daripada taraf signifikansi 5%.

Gain skor adalah selisih *mean* treatmen dan pascates masing-masing kelompok kontrol dan eksperimen. Gain skor digunakan untuk mengetahui adanya peningkatan atau penurunan skor, untuk mengetahui kecepatan dari penggunaan translator notasi algoritmik dala menyelesaikan masalah pemrograman. Namun, sebelum dilakukan pengujian terhadap hipotesis maka akan dilakukan uji persyaratan analisis terlebih dahulu, yaitu uji normalitas sebaran dan uji homogenitas.

4.13.1. Uji Persyaratan Analisis Data

a. Uji Normalitas Sebaran Data

Uji normalitas bertujuan untuk mengetahui apakah segala yang diselidiki memiliki distribusi normal atau tidak. Uji normalitas ini menggunakan teknik statistik Kolmogorov-Smirnov (Uji K-S). Interpretasi hasil uji normalitas dengan melihat nilai *Asymp. Sig. (2tailed).* Adapun interpretasi dari uji normalitas adalah sebagai berikut.

1. Jika nilai *Asymp. Sig. (2tailed)* lebih besar dati tingkat *Alpha 5%* (*Asymp. Sig. (2tailed)* > 0,05) dapat disimpulkan bahwa data berasal dari populasi yang berdistribusi normal.
2. Jika nilai *Asymp. Sig. (2tailed)* lebih kecil dari tingkat *Alpha 5%* (*Asymp. Sig. (2tailed)* < 0,05) dapat disimpulkan bahwa data berasal dari populasi yang berdistribusi tidak normal.

b. Uji Homogenitas Varian

Uji homogenitas bertujuan untuk mengetahui apakah sampel yang diambil dari populasi memiliki varian yang sama atau tidak menunjukkan perbedaan yang signifikan satu sama lain. Interpretasi hasil uji homogenitas dengan melihat nilai *Sig*. Adapun interpretasinya sebagai berikut.

a. Jika signifikan lebih kecil dari 0,05 (*Sig*. < *alpha*), maka varian berbeda secara signifikan (tidak homogen).
b. Jika signifikan lebih besar dari 0,05 (*Sig*. > *alpha*), maka varian berbeda secara signifikan (homogen).

4.13.2. Analisis Data

Teknik analisis data yang digunakan untuk menguji hipotesis dalam penelitian ini adalah Uji-t (*t-test*). Uji-t untuk menguji apakah nilai rata-rata dari kedua kelompok tersebut memiliki perbedaan yang signifikan teknik analisis data dilakukan dengan menggunakan komputer program SPSS 20. Interpretasi hasil Uji-t dengan melihat nilai *Sig. (2tailed),* kemudian dibandingkan dengan tingkat signifikasi 0,050. Adapun interpretasi dari Uji-t adalah sebagai berikut.

a. Jika nilai *Sig. (2-tailed)* lebih besar dari tingkat signifikasi 0,05 (*Sig. (2-tailed)* > 0,05), maka dapat disimpulkan bahwa tidak terdapat perbedaan yang positif dan signifikan antara mahasiswa yang menggunakan translator notasi algoritmik dibanding dengan mahasiswa yang tidak menggunakan translator notasi algoritmik dalam pemecahan masalah pemrograman dasar.
b. Jika nilai *Sig. (2-tailed)* lebih kecil dari tingkat signifikasi 0,05 (*Sig. (2-tailed)* < 0,05), maka dapat disimpulkan bahwa terdapat perbedaan yang positif dan signifikan antara mahasiswa yang menggunakan translator notasi algoritmik dibanding dengan mahasiswa yang tidak menggunakan translator notasi algoritmik dalam pemecahan masalah pemrograman dasar. Setelah dilakukan Uji-t, dapat diambil kesimpulan bahwa;
    1. Jika nilai *Sig. (2-tailed)* lebih besar dari tingkat signifikasi 0,05 (*Sig. (2- tailed)* > 0,05), maka dapat disimpulkan bahwa penggunaan translator notasi algoritmik tidak lebih cepat

dibandingkan dengan yang tanpa menggunakan translator notasi algoritmik dalam pembelajaran pemrograman dasar.

2. Jika nilai *Sig. (2-tailed)* lebih kecil dari tingkat signifikasi 0,05 (*Sig. (2-tailed)* < 0,05), maka dapat disimpulkan bahwa penggunaan translator notasi algoritmik lebih cepat dibandingkan dengan yang tanpa menggunakan translator notasi algoritmik dalam pembelajaran pemrograman dasar.

4.14 Hipotesis Statistik

Hipotesis statistik disebut juga hipotesis nihil (Ho). Hipotesis ini menyatakan tidak terdapat perbedaan yang signifikan kecepatan menyelesaikan masalah pemrograman dasar kelas eksperimen yang diajar dengan menggunakan translator notasi algoritmik dengan kelas kontrol yang diajar tanpa menggunakan translator notasi algoritmik.

Ho = μ1 :μ2
Ha = μ1≠μ2

Ho = Tidak terdapat perbedaan yang signifikan kecepatan menyelesaiakan masalah pemrograman dasar antara kelas eksperimen yang diajar dengan menggunakan translator notasi algoritmik dengan kelas kontrol yang diajar tanpa menggunakan translator notasi algoritmik.

Ha = Terdapat perbedaan yang signifikan kecepatan menyelesaikan masalah perograman dasar antara kelas eksperimen yang diajar menggunakan translator notasi algoritmik dengan kelas kontrol yang diajar tanpa menggunakan translator notasi algoritmik.

Ho = μ1:μ2
Ha = μ1>μ2

Ho = translator notasi algoritmik tidak efektif digunakan sebagai media pemecahan masalah pmrograman dasar pada mahasiswa tahun pertama progrm studi teknik informatika FIK UDINUS.

Ha = translator notasi algoritmik dapat cepat menyelesaikan masalah yang digunakan sebagai media pembelajaran pemrograman dasar pada mahasiswa tahun pertama program studi teknik informatika FIK.

BAB 5. HASIL YANG DICAPAI

5.1 Hasil Penelitian

Dalam penelitian ini, terdapat dua hasil yang akan di paparkan, yang terdiri dari hasil penelitian pengembangan sistem (perekayasaan) editor translator notasi algoritmik dan hasil penelitian eksperimen yang bertujuan untuk mendiskripsikan perbedaan penyelesaian masalah pemrograman dasar dengan translator notasi algoritmik. Perbedaan yang ingin di lihat adalah pada mahasiswa yang diberi perlakuan atau kelompok eksperimen dengan menggunakan translator notasi algoritmik dalam menyelesaikan masalah pemrograman dasar dan mahasiswa yang tidak menggunakan translator notasi algoritmik dalam menyelesaiakan masalah pemrograman dasar yang di sebut kelompok kontrol. Dengan demikin penelitian ini juga dapat menguji kecepataan penyelesaian masalah pemrograman dasar dengan translator notasi algoritmik. Penggunaan data di peroleh dari posttest baik pada kelompok eksperimen dan kontrol.

5.1.1. Hasil Penelitian Perekayasaan

Terdapat tiga tahap sebagai hasil dari pengembangan sistem yang di pilih, yaitu *pertama*, hasil pembuatan grammar untuk notasi algoritmik untuk menghasilkan suatu kerangka kerja translator notasi algoritmik ke suatu bahasa. *Kedua*, implementasi rancangan translator notasi algoritmik dalam bentuk editor teks dengan pendekatan RAD dan MVC, yang di implementasikan dengan menggunakan bahasa pemrograman *Java SE 1.7 dengan Integrated Deveopment Environtment Eclipse Juno.*

5.1.1.a. Grammar dan String Template Translator Notasi Algoritmik

Grammar di bangun dengan menggunakan library ANTLR dengan editor ANTLRWorks, dan di tulis langsung (hand coded). Format grammar (Parr, 2007) yang di pakai dalam peelitian ini adalah sebagai berikut :

```
grammarType grammar name;
«optionsSpec»
«tokensSpec»
«attributeScopes»
«actions»
```

```
/** doc comment */
rule1 : ... | ... | ... ;
rule2 : ... | ... | ... ;
   ...
```

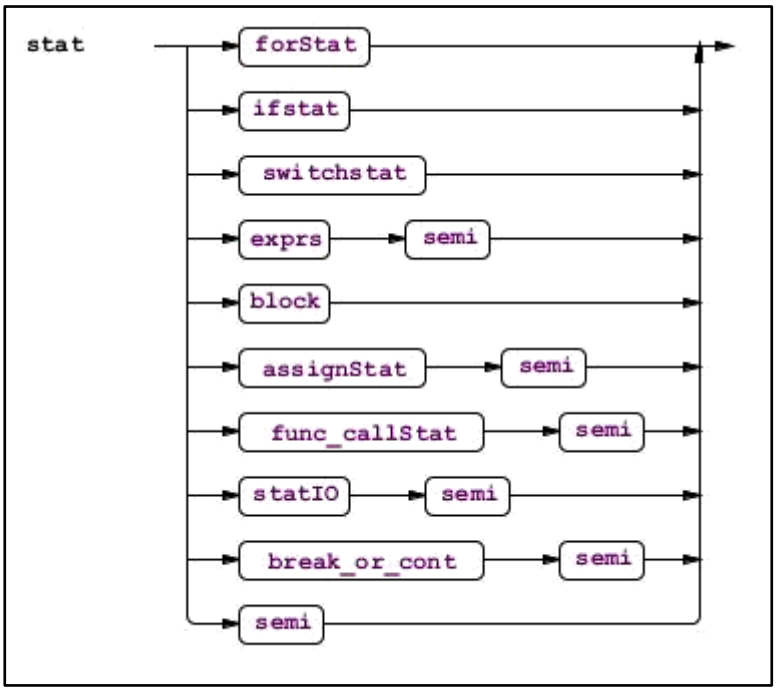
Potongan grammar file yang di buat bernama *algritmik.g* seperti dalam gambar 7 berikut, yang selengkapnya ada di lampiran :

```
grammar Algoritmik;
options {
      backtrack=true;
      memoize=true;
      k=2;
      language=Java;
      output=template;
}

program
    : declaration+
    ;
.....
forStat    :
  ID* 'Traversal' ('[' s=logCond range e=logCond ']')*
      ('Step' exprs_or_assignStat)* 'Do' block ';'
  |   'Repeat' block 'Until' '('(logCond)*')' ';'
  |   'While' '('(logCond)*')' 'Do' block ';'
;
.....
WS
      :     (' ' | '\r'|'\t'|'\u000C'|'\n') {$channel=HIDDEN;}
      ;
COMMENT
      :     '/*' (options {greedy=false;} : . )* '*/'
{$channel=HIDDEN;}
      ;
LINE_COMMENT
      :     '//' ~('\n'|'\r')* '\r'? '\n' {$channel=HIDDEN;}
      ;
```

Gambar 7. Potongan grammar Algoritmik.g

Tipe grammar yang di pilih dalam penelitian ini adalah gabungan parser dan lexer dengan output template, semua rule yang di tulis juga dapat di visualisasikan dalam syntax diagram, yang akan di pakai dalam editor translator, gambar 8 berikut merupakan salah satu contoh syntax diagram dalam rule statemen,



Gambar 8. Syntax diagram rule statement

Sementara itu string template juga di tulis dengan tangan (hand code) dan di beri nama file *algoritmik.stg* dan potongan filenya seperti terlihat pada gambar 9 berikut,

```
group Algoritmik;
program(libs,globals,functions,mainfunctions) ::=<<
<libs; separator="\n">
<globals; separator="\n">
<functions; separator="\n">
<mainfunctions; separator="\n">
>>
/*library*/
setLib(lib)::="#include\<<setFile(lib)>\>"
setFile(name)::="<name>"
getTypedeclarator(list)::= <<<list;separator=",">
>>
/*main program*/ //int argc,char *argv[]
mainfunct(arg,locals,stats)::=<<int main(){
  <locals; separator="\n">
  <stats; separator="\n">
  return 0;
}
>>
....
.....
```

Gambar 9. String Template Notasi Algoritmik

Grammar dan string template di atas merupakan kernel dari translator notasi algoritmik yang berfungsi mengenali input (source code) sesuai grammar (lexer dan parser) sehingga setelah valid akan di translasikan sesuai string template yang di buat, kernel lain yang di pakai adalah library ANTLR yang di integrasikan dalam class tersendiri yang di tulis dalam java. Model grammar menyesuaikan dengan model notasi algoritmik yang di pilih untuk penelitian ini (Liem, 2007, Wijanarto, 2012) dengan beberapa modifikasi yang bertujuan untuk memudahkan implementasi model notasi algoritmik ke dalam grammar.

5.1.1.b. Implementasi RAD dalam kerangka MVC

Hasil implementasi metode RAD dalam kerangka MVC di tulis dengan paradigma object oriented dan akan di paparkan pada bagian di bawah ini.

A. Fase Planning

B.1 Tujuan

Menghasilkan aplikasi atau tool (*editor teks*) yang mampu mentranslasikan notasi algoritmik standar dalam bahasa C. Sehingga alat ini akan membantu mahasiswa atau orang yang tertarik di bidang pemrograman untuk fokus belajar bagaimana menyelesaikan masalah dengan notasi algoritmik standar yang sederhana tanpa memikirkan bahasa pemrograman yang terkesan rumit dan membingungkan.

B.2 Fungsional

Fungsi sistem ini adalah sebagai media yang menjembatani pengajar/dosen melakukan pendampingan kepada pembelajar/mahasiswa yang melakukan proses pembelajaran notasi algoritmik. Tabel 7 berikut dapat lebih detail menjelaskan kebutuhan fungsional aplikasi yang di bangun.

Tabel 7 Fungsi dan Fasilitas Translator Notasi Algoritmik

Kode	**Keterangan**
Fungsi Dasar	
D.1	Memberikan fasilitas untuk menulis notasi algoritmik baru
D.2	Memberikan fasilitas untuk membuka file yang menyimpan notasi algoritmik
D.3	Menyediakan fasilitas untuk run program
D.4	Menyediakan fasilitas untuk build & compile

D.5	Menyediakan fasilitas untuk translate notasi algoritmik ke bahasa C
Fungsi Grammar Constructor	
C.1	Menyediakan fasilitas untuk menulis Algorithmic Notation Grammar
C.2	Menyediakan fasilitas untuk menulis String Template for C Language

B.3 Ruang Lingkup

Rancangan sistem yang dibuat merupakan sebuah sistem yang mampu mentranslasikan notasi algoritmik standar dalam bahasa C, berisi modul utama program berupa *text editor* yang menjadi pengantar muka proses pembuatan notasi algoritmik sampai proses translasi menjadi code program standar dalam bahasa C.

Translator Notasi Algoritmik ini memberikan fasilitas untuk menulis notasi algoritmik baru, membuka notasi yang sudah tersimpan, melakukan *run program*, *build & compile* dan fasilitas untuk mentranslasikan notasi algoritmik ke bahasa C.

B. Fase User Design

B.1 Use Case Diagram

Diagram *Use-case* di sini dijabarkan secara grafis yang menggambarkan interaksi antara sistem, sistem eksternal, dan pengguna dari aplikasi Translator Notasi Algoritmik yang dibangun. Diagram *Use-case* berikut menggambarkan siapa saja yang akan menggunakan aplikasi Translator Notasi Algoritmik, dan bagaimana cara pengguna berinteraksi dengan sistem yang dibuat.

Use-case Naratif akan menjelaskan sekuensi langkah-langkah dari setiap interaksi yang terjadi, sesuai dengan metode yang digunakan maka langkah langkah yang dilakukan dalam membuat *use-case* diagram adalah sebagai berikut:

a) **Mengidentifikasikan Pelaku Bisnis**

Disini pelaku bisnis diartikan sebagai pemakai, user atau aktor yang akan terlibat dalam sistem ini, daftar aktor bisa dilihat pada tabel 8 berikut :

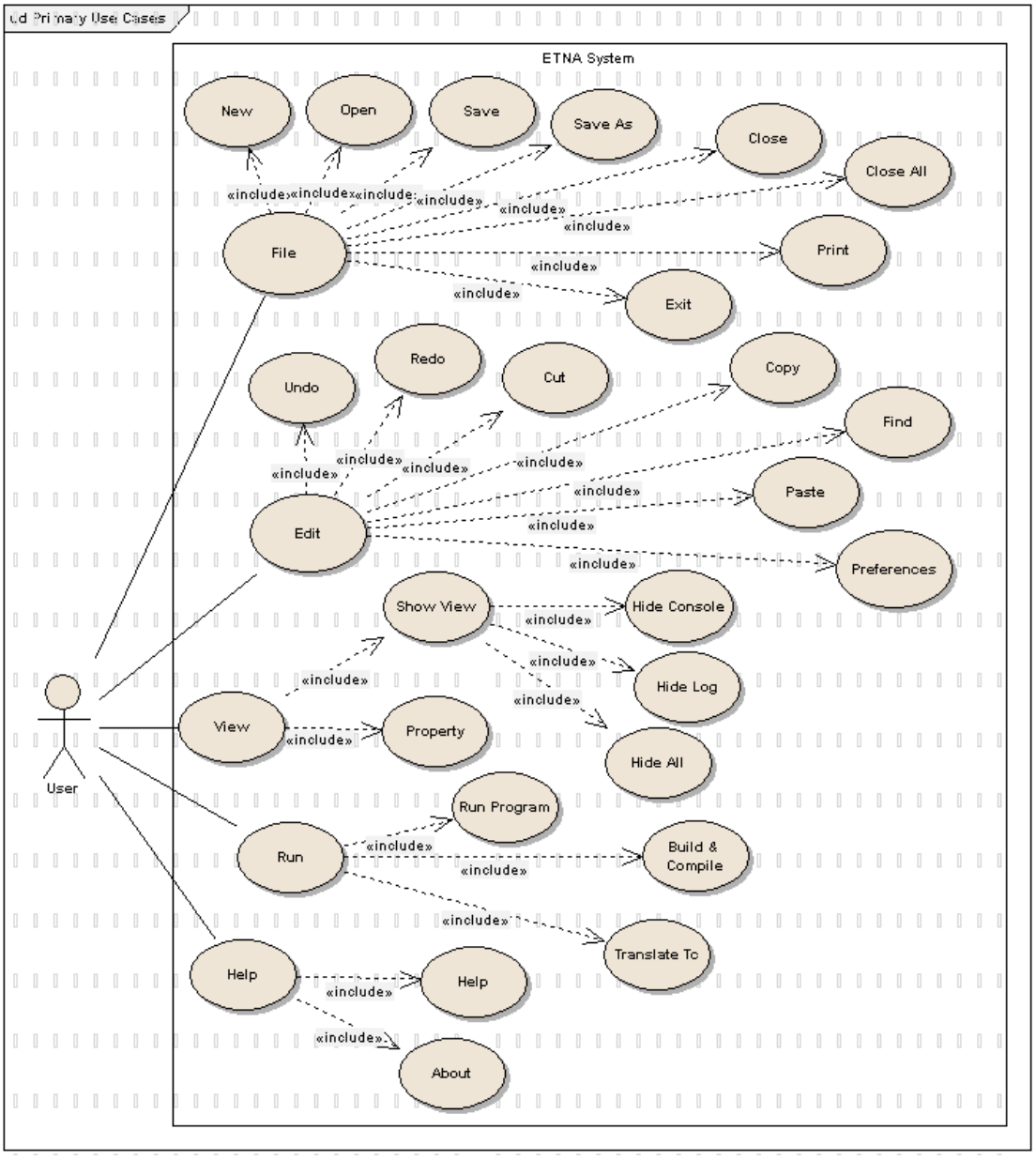
Tabel 8. Identifikasi Pelaku Bisnis

Istilah	Sinonim	Deskripsi
1. Grammar Constructor	Grammar Constructor	Individu atau orang yang membuat Algoritmic Notation Grammar dan membuat String Template for C Language

2. User	Pengguna	Individu atau orang yang memakai program Translator Notasi Algoritmik

b) **Diagram model *Use-case* ETNA System**

Editor translator notasi algoritmik ini akan di beri nama ETNA yang merupakan singkatan dari Editor Translator Notasi Algoritmik, dan gambar 10 di bawah ini merupakan model utama dari system ETNA yang di bangun.

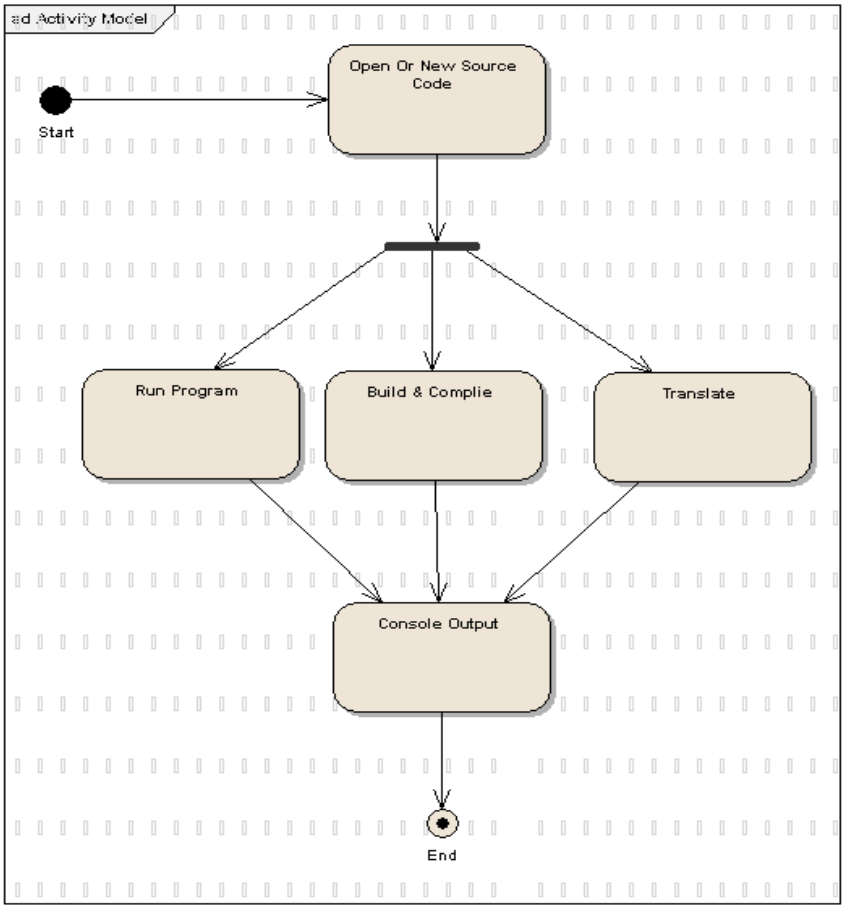


Gambar 10. Model Use case ETNA.

Gambar di atas menjelaskan hubungan antara user/pengguna dengan *use case*. User/pengguna melakukan proses *file, edit, view, run, help*. Dalam proses *file* terdapat fasilitas untuk *new, open, save, save as, close, close all, print* dan *exit*. Dalam proses *edit* terdapat fasilitas untuk *undo, redo, copy, cut, find, paste,* dan *preferences*. Di dalam proses view terdapat fasilitas untuk *show view* dan *property*. Di dalam proses *run* terdapat fasilitas untuk *run program, build & compile* dan *translate To c*. Dan diproses *help* terdapat fasilitas *help* dan *about*.

B.2 Activity Diagram

Diagram aktifitas menggambarkan berbagai alir aktifitas perancangan sistem, bagaimana alir berawal, keputusan yang mungkin terjadi, dan bagaimana berakhir. Grambar 11 di bawah ini menunjukan diagram aktifitas yang di lakukan oleh user saat menggunakan ETNA.

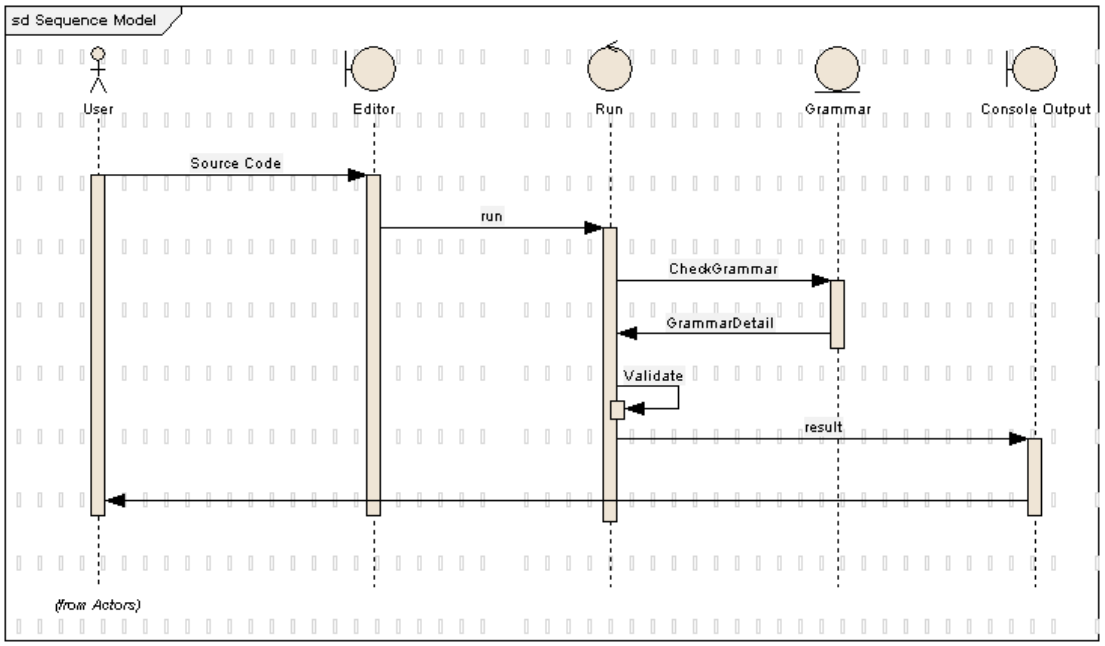


Gambar 11. Diagram Aktifitas ETNA

Gambar 11 di atas menjelaskan pengguna/user akan masuk ke *text editor* kemudian menuliskan notasi algoritmik atau membuka file yang sudah ada, selanjutnya user/pengguna dapat melakukan *run program* atau *build & complie* atau *translate,* hasil dari ke tiga proses tersebut di tampilkan di *console output atau console system*

B.3 Sequence Diagram

Diagram *Sequence* menggambarkan perilaku pada sebuah scenario dapat di lihat pada gambar 12 di bawah ini.

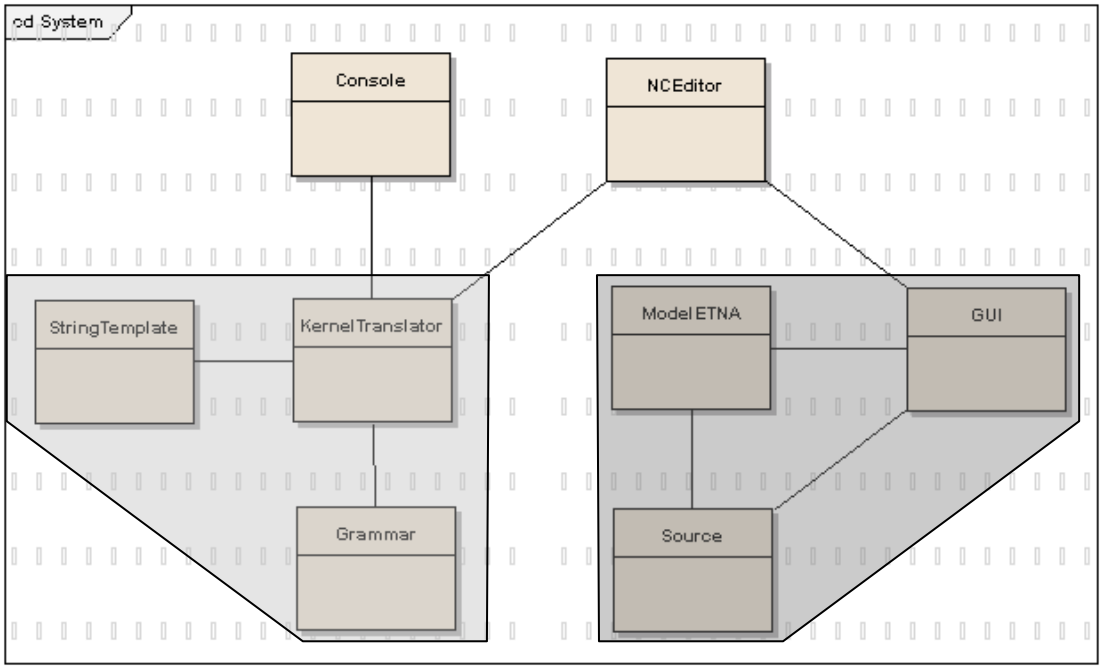


Gambar 12. Diagram Sekuen ETNA

Proses *run program, user* memasukkan notasi algoritmik, notasi algoritmik diverifikasi dengan mengecek *grammar,* bila notasi algoritmik benar atau tidak sesuai dengan *grammar* maka hasil ditampilkan di *console output*.

B.4 Class Diagram

Diagram *class* di sini menggambarkan hubungan antar *class* dan hubungan *class* dengan *class* pendukungnya. Gambar 13 diagram *class* dapat dilihat pada gambar berikut, sebagai catatan tidak semua class di gambarkan dalam laporan ini mengingat class ETNA sangat besar karena melibatkan library dari luar dan yang penting saja yang di tampilkan dalam gambar 13 :

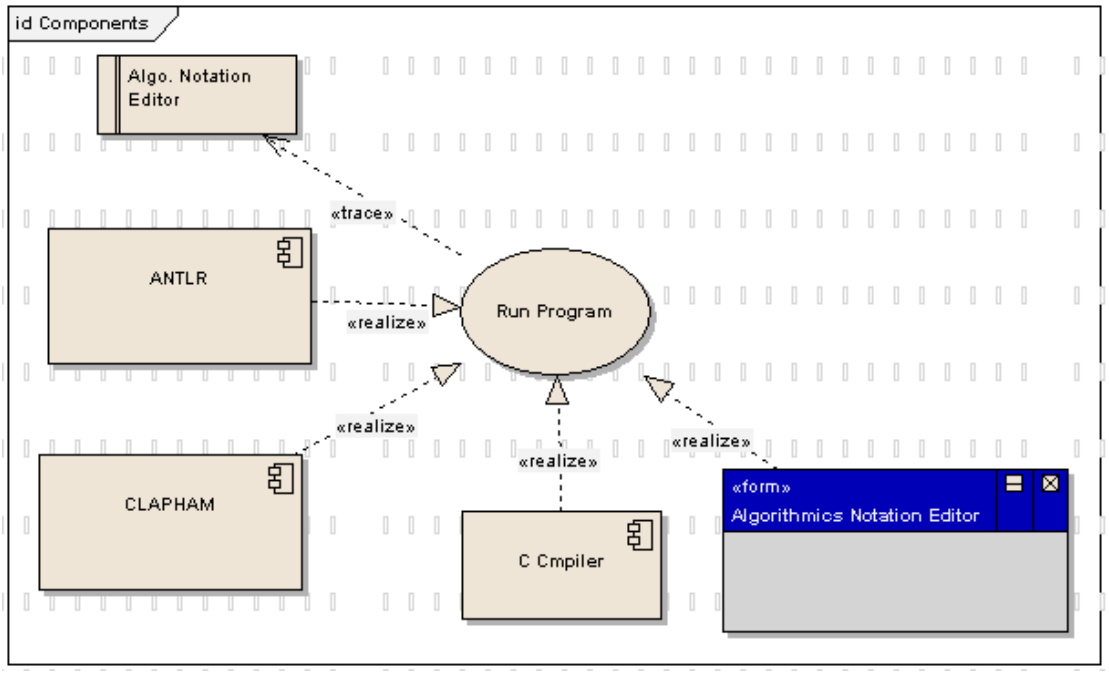


Gambar 13. Class Diagram ETNA

Terlihat pada gambar di atas, spot bidang abu-abu muda di bagian kiri, merupakan kernel utama dari ETNA yang di tulis dengan tangan (hand coded), sementara pada spot bidang abu-abu tua di sebelah kanan merupakan implementasi moel MVC yang di ekspresikan dengan metode RAD.

B.5 Implementasi Diagram

Diagram implementasi menggambarkan hubungan antar komponen yang terlibat dalam sistem. Gambar 14 di bawah menunjukan fungsi utama dalam kernel (menu Run) dalam GUI ETNA serta beberapa tool yang di pakai dalam kernel (ANTLR, C Compiler, Clapham) serta kernel utama (grammar dan string template).

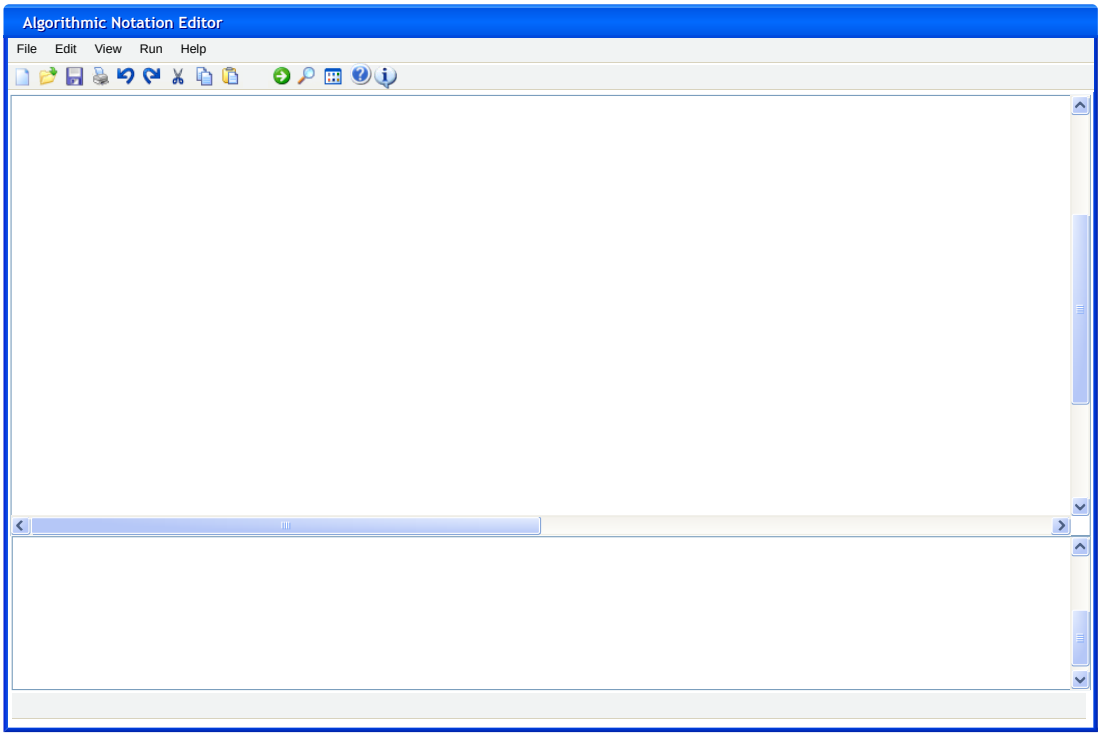


Gambar 14 : Implementation Diagram ETNA System

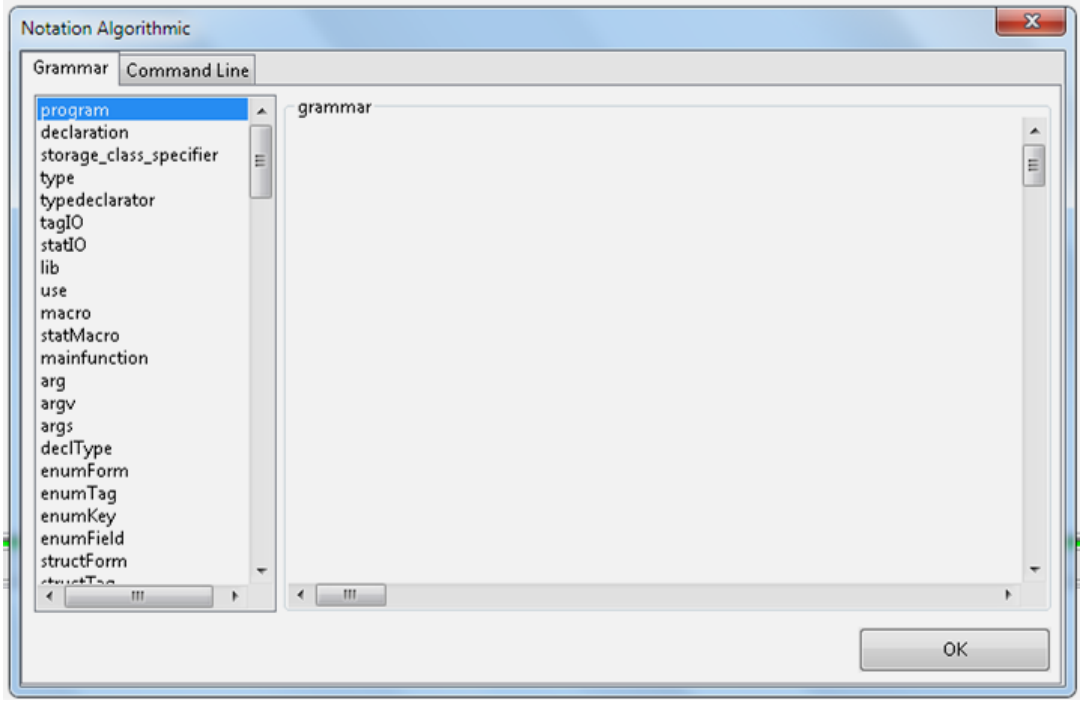
Gambar di atas menjelaskan saat *run* program, sistem memanggil *external tool for parsing input* ANTRL, memanggil *external tool for generate image syntax tree* CLAPHAM, memanggil C *Compiler* yang direalisasikan di *Algorithmic Notation Editor*.

B.6 Disain Interface

Disain interface ETNA, seperti pada editor umumnya, hanya terdiri dari workspace untuk menulis notasi algoritmik, yang di lengkapi dengan menu pendukungnya, serta console output yang di pakai sebaga tempat menampilkan hasil translasi, pesan kesalahan, proses translasi, proses building dan proses running notasi, seperti di tunjukan pada gambar 15 di bawah ini. Tidak semua interface akan di tampilkan dalam laporan ini karena alasan fokus utama interface merupakan workspace untuk menulis notasi, sedangkan interfacea yang lain hanya merupakan mendukung untuk melakukan format editor yang tidak penting untuk di tampilkan dalam penelitian ini. Sementara gambar 15 adalah interface untuk membantu pemahaman user dalam menulis notasi yang di wujudkan dalam syntax diagram tree dari rule yang di buat dalam grammar.



Gambar 15. Disain Interface Utama ETNA



Gambar 16. Disain Bantuan Syntax Tree Untuk User

C. Fase Construction

Pada tahap konstruksi ini rancangan translator notasi algoritmik di implementasikan dalam bentuk editor teks dengan menggunakan bahasa pemrograman *Java SE 1.7 dengan Integrated Deveopment Environtment Eclipse Juno*. Berikut hasil aplikasi ETNA yang di bangun berdasarkan beberapa pendukung baik hardware maupun software sebagai berikut :

Perangkat lunak pendukung yang dibutuhkan adalah :

1. ANTLR 3.4 ke atas
2. Clapham 1.0 ke atas
3. MinGW (GCC Compiler) atau GCC 4.6.2 ke atas
4. Jdk-7-windows-i586 atau Jdk-7-linux32-i586 atau Jdk-7-linux64-i586

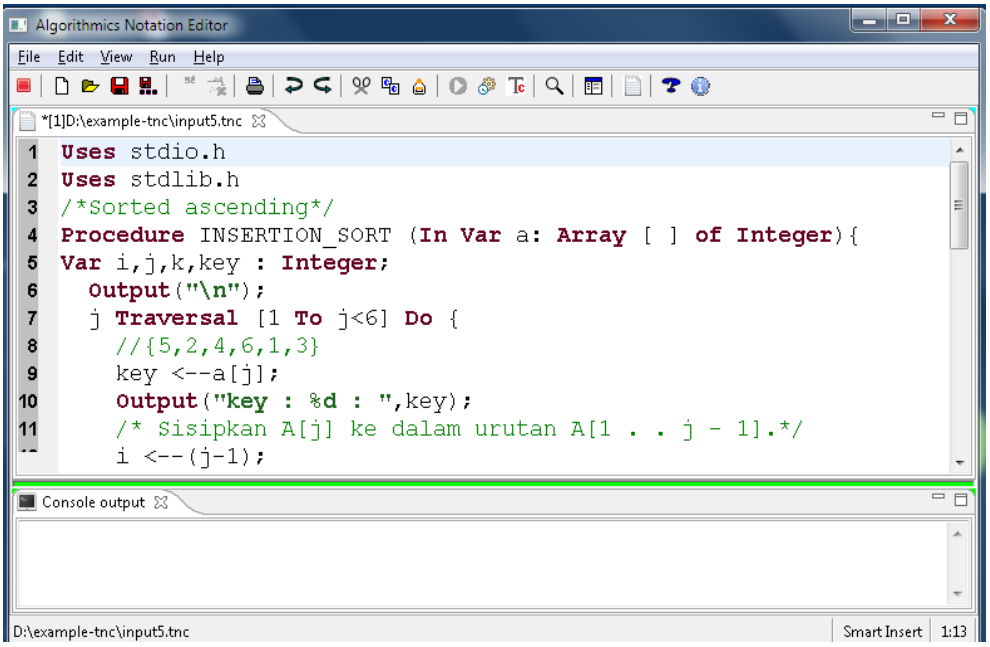
Spesifikasi *hardware* untuk implementasi sistem sebagai berikut :

1. Intel Pentium® Dual-Core
2. RAM 3 GB
3. Resolusi Monitor 1280 x 720

ETNA di disain dapat berjalan pada sistem operasi yang digunakan MS Windows XP Edition SP 3, Windows 7 32 bit, Linux Ubuntu 32 dan 64 bi. Berikut ini hasil implementasi aplikasi ETNA dengan nama file eksekusi Nattoc32Win seperti terlihat pada beberapa gambar 17 berikut :

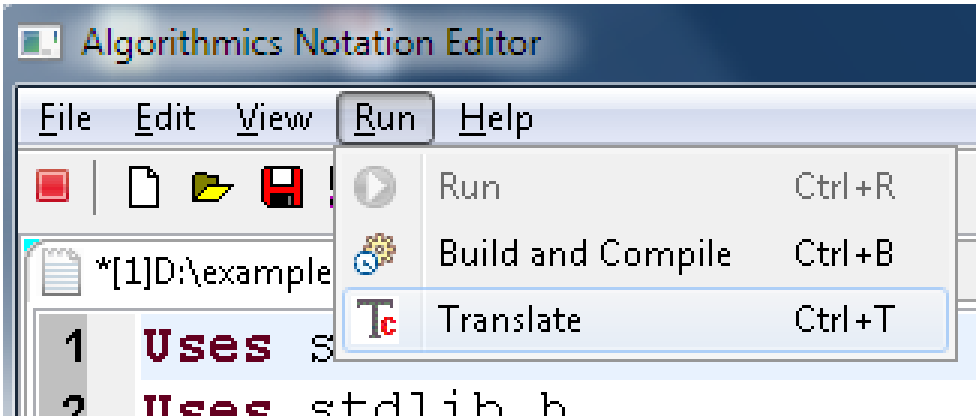
Gambar 17. Start Up Aplikasi ETNA

Gambar 17 merupakan start up aplikasi, sekaligus melakukan inisialisasi terhadap library yang di butuhkan oleh ETNA, termasuk melakukan generate syntax tree untuk bantuan user. Setelah tahap ini selesai maka ETNA siap menerima input (File lama dalam format *.tnc atau user menulis langsung ke editor), gambar 18 menunjukan aplikasi ETNA sedang dalam keadaan menerima file yang sedang terbuka.



Gambar 18. ETNA dengan file notasi aktif

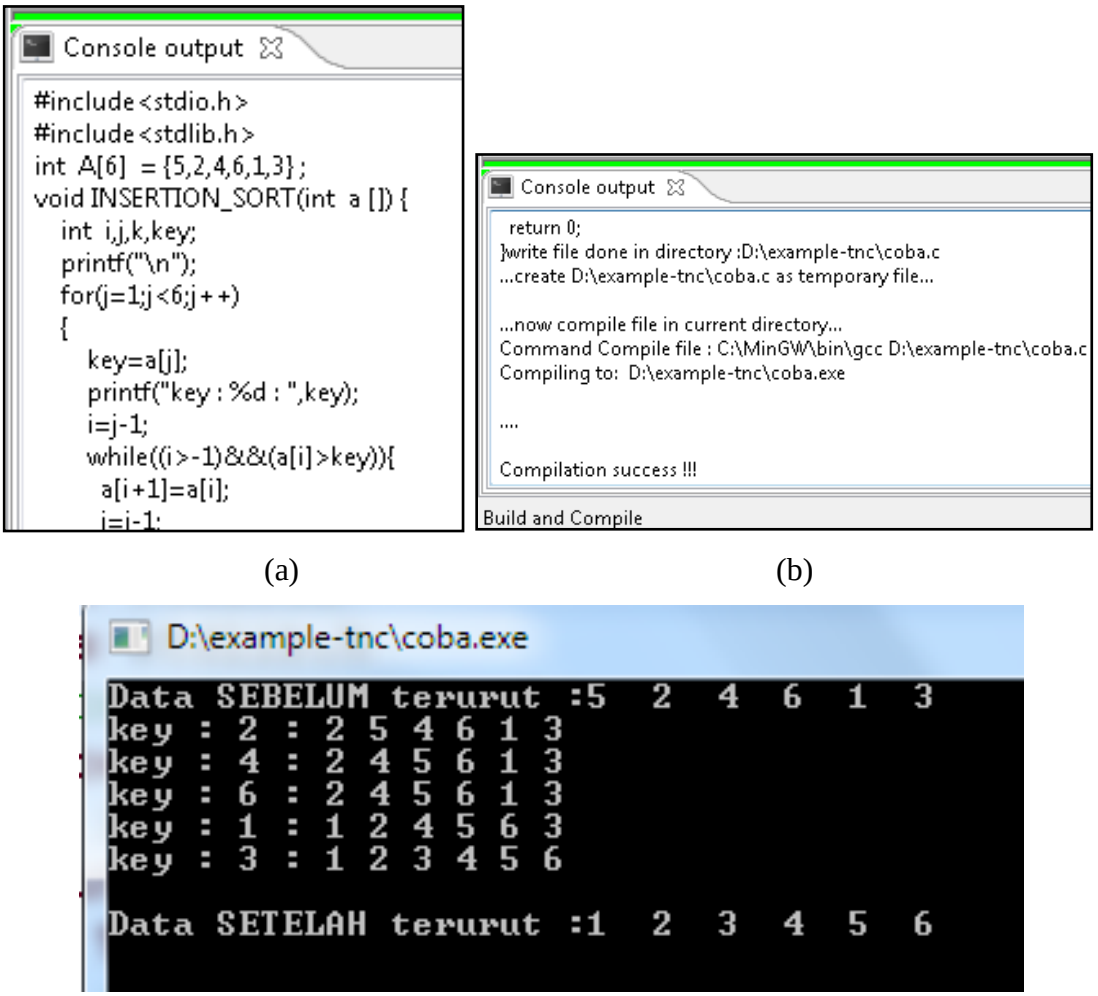
Perlu juga di tampilkan fungsi utama ETNA yang merupakan kernel, yang di implementasikan dalam bentuk menu dalam ETNA seperti terlihat pada gambar 19 berikut.



Gambar 19. Menu Utama ETNA

Kernel utama ETNA terdiri dari translate, build and compile serta run, dimana urutan tersebut mennjukan hirarki eksekusi, notasi harus di translasikan lebih dahulu sebelum dapat di build and compile dan setelah dilakukan kompilasi, maka menu run dapat di jalankan untuk memeriksa apakah notasi yang terkompilasi sudah sesuai yang di inginkan.

ETNA juga dapat memberikan output berupa hasil translasi, proses build adn compile serta hasil running program baik dalam console aplikasi atau console sistem seperti terlihat berturut -turut a, b dan c dalam gambar 20 berikut,

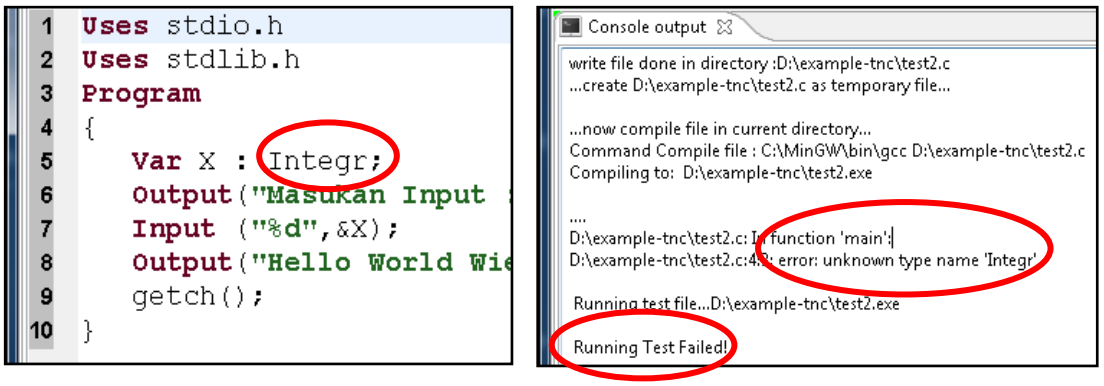


(a) (b)

(c)

Gambar 20. Output ETNA (a) hasil translas, (b) hasil kompilasi (c)hasil running

Output console ETNA juga dapat menampilkan kesalahan yang terjadi jika penulisan notasi salah maka saat di jalankan akan salah pula, seperti terlihat pada gambar 21 di bawah ini.



(a) (b)

Gambar 21. Output ETNA (a) kesalahan notasi (b) hasil kesalahan

D. Fase Cutover

Pada tahap ini hasil aplikasi Translator Notasi Algoritmik di uji coba ke pengguna/user untuk mengetahui dan mengevaluasi hasil aplikasi apakah sudah sesuai dengan rancangan dan tujuan dari pembuatan aplikasi. Pada tahap ini akan di lakukan uji hipotesa dengan analisa kuantitatif yang akan di paparkan pada bagian lain dari laporan penelitian ini. Penelitian awal sudah di lakukan, yaitu berupa kegiatan persiapan , pengambilan sampel penelitian , penentuan alat dan teknik pengumpulan data hingga uji validitas dan reliabilitas instrumen, seperti urutan berikut berikut :

1. Membuat dan menentukan paradigma penelitian
2. Menentukan variabel penelitian
3. Menentukan tempat dan waktu penelitian
4. Menentukan populasi dan sampel penelitian
5. Menentukan Alat dan Teknik Pengumpulan data
6. Melakuka uji validitas instrumen
7. Melakukan Uji reliabiitas instrumen

Namun kegiatan di atas akan dilaporkan setelah sajikan dalam laporan akhir supaya keterurutannya menjadi lebih jelas dan mudah di baca.

BAB 6. RENCANA TAHAPAN BERIKUTNYA

Dari hasil yang di capai, maka rencana tahap berikutnya dalam penelitian ini akan melakukan pengujian terhadap ETNA, untuk membuktikan bahwa ETNA akan memberi dampak yang signifikan terhadap kemampuan dan efektifitas pembelajaran pemrograman dasar. Dengan bantuan program SPSS, berturt-turut akan di lakukan langkah-langkah penelitian eksperimen untuk mengetahui perbedaan mahasiswa yang di perlakukan dengan menggunakan ETNA dan yang tidak menggunakan ETNA dengan melakukan prosedur penelitian sebagai berikut :

a. Uji Normalitas Data
b. Uji Homogenitas Data
c. Analisa Data Statistik
d. Uji Hipotesa Statistik
e. Menarik kesimpulan dan membuat saran

Tahap-tahap di atas akan di laksanakan saat laporan kemajuan ini selesai hingga batas waktu laporan akhir di buat.

BAB 7. KESIMPULAN DAN SARAN

Setelah melakukan penelitian perekayasaan guna membuat suatu aplikasi, maka hasil sementara yang dapat di simpulkan dalam penelitian ini adalah :

1. Pemilihan model notasi algoritmik dapat di implementasikan ke dalam bentuk grammar yang mudah di implementasikan menjadi suatu aplikasi.
2. Aplikasi yang di bangun merupakan hasil implementasi dari model notasi dan grammar yang disesuaikan berhasil di kembangkan menjadi bentuk editor notasi algoritmik yang di beri nama ETNA (Editor Translator Notasi Algoritmik).
3. Hasil pengujian aplikasi, menunjukan bahwa ETNA berhasil melakukan translasi notasi algoritmik menjadi kode bahasa c yang dapat di kompilasi dan di jalankan dengan sempurna.
4. Aplikasi dapat memberikan respon bila terjadi kesalahan penulisan notasi yang salah dan memberi pesan kesalahan pada user.

Kedepan peelitian ini dapat di kembangkan untuk di beri fitur tambahan dan kemampuan yang lebih cerdas sebagai berikut :

1. Editor di lengkapi dengan teknik *error detection* yang *real time*, artinya jika terdapat kesalahan pada format penulisan notasi oleh user, ETNA di harapkan mampu mendeteksi secara dini dan member pesan kesalahan saat berpindah baris.
2. Hasil deteksi kesalahan dapat di visualisasikan dalam bentuk ekuivalensi kesalahan penulisan notasi dengan syntax tree diagram yang di spesifikasikan oleh grammar yang di pakai.
3. Diharapkan ETNA tidak hanya mampu menjadi translator untuk satu bahasa prosedural saja, tapi dapat menjadi multi translator pada bahasa dengan paradigma prosedural.

DAFTAR PUSTAKA

Alfred V Aho, Monica S Lam, Ravi Sethi , Jeffrey D Ullman, 2007, Compilers : principles, techniques, and tools Second Edition. New York : Pearson Education Addison Wesly.

Alverd V Aho, Jeffery D Ullman, 1973, The Theory of Parsing, Translation and Compiling. New York : Prentice Hall Englewood Cliffs, 1973. 0-13-914564-8.

Arikunto, Suharsimi. 2006. Prosedur Penelitian suatu Pendekatan Praktek Edisi Revisi V. Yogyakarta: Rineka Cipta.

Andrew W Appel, Maia Ginsburg, 1998, Modern Compiler Implementation In C. New York : CAMBRIDGE UNIVERSITY PRESS.

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

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.

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.

David A Watt, Deryck F Brown, 2000, Programming Language Processors in Java, Compiler and Intepreter. New York : Pearson Education, Addison Wesly.

David Harel, Yishai A. Feldman, 2004 , Algorithmics: the spirit of computing, Edition 3, Pearson Education, ISBN 0-321-11784-0.

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.

Ian Somerville, 2011, Software engineering, 9th edition, Pearson Education, Addison-Wesly, Boston, Massachusetts.

Kruskal J. B, Jr., 1956, On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedingsof the American Mathematical Society, 7, pp. 48-50.

Liem, Inggriani, 2007, Draft Diktat Dasar Pemrograman (Bagian Prosedural), ITB , Bandung.

Nasution, 2007, Metode Research, Jakarta, Bumi Aksara.

Nurgiyantoro, Burhan. 2009. *Penilaian dalam Pengajaran Bahasa dan Sastra*. Yogyakarta: BPFE UGM.

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

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

Stanchfield, Scott, Advanced MVC Patterns. JavaDude. [Online] 1996-2009. diakses: 10-10- 2012 http://javadude.com/articles/vaddmvc1/mvc1.htm

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

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

Sukmadinata, Nana, Saodih, 2007, Metode Penelitian Pendidikan, Bandung, Rosdakarya.

Terence Parr, 2007, The Definitive Guide ANTLR reference, Building Domain-Specific Language. Raleigh, North Carolina Dallas, Texas : The Pragmatic Bookshelf.

Terence Parr, 2010, Language Implementation Patterns Create Your Own Domain-Specific and General Programming Languages. Raleigh, North Carolina Dallas, Texas : The Pragmatic Bookshelf.

Terence Parr, Kathleen Fisher, 2011, LL(*): the foundation of the ANTLR parser generator. s.l. : Vol 11 ACM SIGPLAN Notices - PLDI.

Wikipedia, 2013, diakses 6-01-2013, http://en.wikipedia.org/wiki/Text_editor

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

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

Lampiran 1. Biodata Peneliti

BIODATA KETUA PENELITI

A. Identitas Diri

1	Nama Lengkap (dengan gelar)	Wijanarto, S.Sos.,M.Kom.
2	Jenis Kelamin	Laki-laki
3	Jabatan Fungsional	Asisten Ahli
4	NIP/NIK/Identitas lainnya	0686.11.2009.354
5	NIDN	0628027003
6	Tempat dan Tanggal Lahir	Yogyakarta, 28-02-1970
7	E-mail	wijanarto@dosen.dinus.ac.id
8	Nomor Telepon/HP	081328635965
9	Alamat Kantor	Jl. Nakulo I 5 – 11 Semarang 50131
10	Nomor Telepon/Faks	024-3520165
11	Lulusan yang Telah Dihasilkan	S1 = 10 Orang
12. Mata Kuliah yg Diampu		1. Dasar Pemrograman
		2. Algoritma Dan Pemrograman
		3. Struktur Data
		4. Strategi Algoritma

B. Riwayat Pendidikan

	S-1	S-2	S-3
Nama Perguruan Tinggi	Universitas Brawijaya	Universitas Gajah Mada	
Bidang Ilmu	Ilmu Administrasi	Imlu Komputer	
Tahun Masuk-Lulus	1990-1995	2004-2006	
Judul Skripsi/Tesis/Disertasi	Aspek Kultural Jawa Dalam Birokrasi Indonesia 1965-1992	Restorasi Citra Digital Dengan Algoritma Inpainting	
Nama Pembimbing/Promotor	Prof. Drs. Ismani, MPA. Drs. Irwan Noor MA.	Drs. Agus Harjoko, MSc.,Ph.D.	

C. Pengalaman Penelitian Dalam 5 Tahun Terakhir

No	Tahun	Judul Peneitian	Pendanaan	
			Sumber*	Jml (Juta Rp)

D. Pengalaman Pengabdian Kepada Masyarakat dalam 5 Tahun Terakhir

No	Tahun	Judul Pengabdian Kepada Masyarakat	Pendanaan	
			Sumber*	Jml (Juta Rp)

1	2009	Campaign Olimpiade Peserta OSN SMA SEMESTA	SMA Semesta	
2	2010	Diklat Pranata Komputer Kejaksaan Tinggi Jateng	Kejaksaan Tinggi Jateng	
3	2013	Pembnaan OSK SMA 3 Semarang	SMA 3 Semarang	

E. Publikasi Artikel Ilmiah Dalam Jurnal dalam 5 Tahun Terakhir

No	Judul Artikel Ilmiah	Nama Jurnal	Volume/Tahun
1	Restorari Citra Digital Dengan Algoritma Inpainting	Techno-Com	Vol. 8 No.1/ 2009
2	Image Retrieval Berdasarkan Properti Statistik Histogram	Techno-Science	Vol. 38 No.2/2009
3	Vulnerabilitas Program Buffer Overflow	Dian	Vol. 10 No.1/2010
4	Solusi Pencarian N-Puzzle Dengan Langkah Optimal : Suatu Aplikasi Pendekatan Fungsional	Techno-Com	Vol. 10. No.3/2011
5	Simulasi Dan Visualisasi Algoritma Greedy Pemilihan Koin Dalam Bentuk Game	Dian	Vol.11 No.3/2011
6	Perancangan Dan Pembangunan Aplikasi Perangkingan Penerimaan Peserta Didik Smp Hasanuddin 04 Semarang Dengan Promethee Method	Techno-Com	Vol. 11 No. 2/2012
7	Portabilitas Aplikasi Perangkingan Seleksi Penerimaan Siswa Baru Dengan Metode Promethee	Techno-Com	Vol. 11 No.4 2012
8	Model Translator Notasi Algoritmik Ke Bahasa C	KOMMIT Gunadharma	Vol. 7/2012

***F*. Pemakalah Seminar Ilmiah (*Oral Presentation*) dalam 5 Tahun Terakhir**

No	Nama Pertemuan Ilmiah / Seminar	Judul Artikel Ilmiah	Waktu dan Tempat
1	KOMMIT	Model Translator Notasi Algoritmik Ke Bahasa C	8 Oktober 2012, Universitas Gunadarma, Jakarta

G. Karya Buku dalam 5 Tahun Terakhir

No	Judul Buku	Tahun	Jumlah Halaman	Penerbit
1	Teori Pengolahan Citra Digital	2009	255	Andi Offset
2	Strategi Dan Analisis Algoritma	2010	147	Universitas Dian Nuswantoro

H. Perolehan HKI dalam 5–10 Tahun Terakhir

No	Judul/Tema HKI	Tahun	Jenis	Nomor P/ID

I. Pengalaman Merumuskan Kebijakan Publik/Rekayasa Sosial Lainnya dalam 5 Tahun Terakhir

No	Judul/Tema/Jenis Rekayasa Sosial Lainnya yang Telah Diterapkan	Tahun	Tempat Penerapan	Respon Masyarakat

J. Penghargaan dalam 10 tahun Terakhir (dari pemerintah, asosiasi atau institusi lainnya)

No	**Jenis Penghargaan**	**Institusi Pemberi Penghargaan**	**Tahun**

Semua data yang saya isikan dan tercantum dalam biodata ini adalah benar dan dapat dipertanggungjawabkan secara hukum. Apabila di kemudian hari ternyata dijumpai ketidaksesuaian dengan kenyataan, saya sanggup menerima sanksi.

Demikian biodata ini saya buat dengan sebenarnya untuk memenuhi salah satu persyaratan dalam pengajuan Hibah Penelitian Dosen Pemula.

.

Semarang, 13-03-2013
Pengusul,

Wijanarto, S.Sos., M.Kom

BIODATA ANGGOTA PENELITI

A. Identitas Diri

1	Nama Lengkap (dengan gelar)	Ajib Susanto, M.Kom.
2	Jenis Kelamin	L
3	Jabatan Fungsional	Asisten Ahli
4	NIP/NIK/Identitas lainnya	-
5	NIDN	0615127404
6	Tempat dan Tanggal Lahir	Bojonegoro, 15-12-1974
7	E-mail	ajibsusanto@gmail.com
8	Nomor Telepon/HP	0818455527
9	Alamat Kantor	Jl. Nakula I 5 – 11 Semarang 50131
10	Nomor Telepon/Faks	024-3520165
11	Lulusan yang Telah Dihasilkan	D3 = 7, S1 = 32 Orang
12. Mata Kuliah yg Diampu		1. Pemrograman Berorientasi Obyek
		2. Pemrograman Web
		3. Pemrograman Client Server
		4. Pemrograman Aplikasi Bisnis

B. Riwayat Pendidikan

	S-1	S-2	S-3
Nama Perguruan Tinggi	Universitas Dian Nuswantoro	Universitas Dian Nuswantoro	
Bidang Ilmu	Teknik Informatika	Teknik Informatika	
Tahun Masuk-Lulus	2002-2004	2005-2008	
Judul Skripsi/Tesis/Disertasi	Pemanfaatan Type Data Bertipe Blob dalam File Binary untuk Pengaksesan File Melalui Streaming SQL pada Server Database	Rekayasa Sistem Pengelolaan Pembelajaran Elektronik Berbasis Web (eLMS)	
Nama Pembimbing/Promotor	Dr-Ing. Vincent Suhartono	Dr.Eng. Yuliman Purwanto, M.Eng	

C. Pengalaman Penelitian Dalam 5 Tahun Terakhir

No	Tahun	Judul Peneitian	Pendanaan	
			Sumber*	Jml (Juta Rp)
1	2011	Rekayasa Model ”Supermuseum” Batik Online Untuk Mengenalkan Keaneka Ragaman Motif Batik Di Indonesia Dalam Upaya Meningkatkan Pemasaran Batik Produk Usaha Kecil Dan Home Industry	Penelitian Strategi Nasional, Dirjen DIKTI Jakarta.	90
2	2010	Perancangan Sistem Informasi Perhitungan Angka Kredit Dosen	LPP Universitas Dian Nuswantoro	3,5

D. Pengalaman Pengabdian Kepada Masyarakat dalam 5 Tahun Terakhir

No	Tahun	Judul Pengabdian Kepada Masyarakat	Pendanaan	
			Sumber*	Jml (Juta Rp)
1	2008	Pelatihan E-Learning dengan MOODLE bagi Guru SMA Negeri 1 Semarang sebagai Instruktur	SMA Negeri 1 Semarang	-
2	2009	Diklat Pranata Komputer Kejaksaan RI sebagai Instruktur	Kejaksaan Tinggi Jateng	-
3	2010	Pelatihan Aplikasi Perkantoran Open Source PNS Kota Semarang	Universitas Dian Nuswantoro	-
4	2010	Juri Javakanmu ”The art of Java Programming for Education” Tingkat Jateng dan DIY	Universitas Dian Nuswantoro	-
5	2010	Pembuat Soal Komputerisasi dalam Seleksi Pengandaan CPNSD Pemerintah Provinsi dan Kabupaten Kota di Jawa Tengah	Universitas Dian Nuswantoro Semarang	-
6	2010	Diklat Pranata Komputer Kejaksaan RI sebagai Instruktur	Kejaksaan Tinggi Jateng	-
7	2010	Juri pada Lomba Pemilihan Guru Berprestasi Dalam Pembuatan Bahan Ajar Mandiri Berbasis Multimedia	Lembaga Penjaminan Mutu Pendidikan (LPMP) Jawa Tengah	-
8	Tahun pelajaran 2010/2011	Exsternal Assesor pada Ujian Praktik Kejuruan Animasi dan Multimedia	SMK Negeri 11 Semarang SMK Negeri 3 Jepara	-
9	2011	Juri pada Lomba Pemilihan Guru Berprestasi Dalam Pembuatan Bahan Ajar Mandiri Berbasis Multimedia	Lembaga Penjaminan Mutu Pendidikan (LPMP) Jawa Tengah	-
10	Tahun Pelajaran 2011/2012	Exsternal Assesor pada Ujian Praktik Kejuruan Animasi dan Multimedia	SMK Negeri 11 Semarang	-
11	2012	Juri Lomba Pengayaan Sumber Belajar (LPSB) Berbasis Blog Guru Dikdas dan Dikmen Tingkat Provinsi Jawa Tengah	BPITKP Dinas Pendidikan Provinsi Jawa Tengah	-
12	2012	Juri Lomba Multimedia Pembelajaran Guru Tingkat SD/MI, SMP/MTS, SMU/SMK	LPMP Jateng	-
13	Tahun Pelajaran 2012/2013	Exsternal Assesor pada Ujian Praktik Kejuruan Animasi dan Multimedia	SMK Perdana, SMK Robi Rodliyah Semarang	-

E. Publikasi Artikel Ilmiah Dalam Jurnal alam 5 Tahun Terakhir

No	Judul Artikel Ilmiah	Nama Jurnal	Volume/Tahun
1	Rekayasa Sistem Pengelolaan Pembelajaran Elektronik Berbasis Web	Majalah Ilmiah DIAN, Udinus Semarang, ISSN 1412-3088	Vol.9/ No. 2/ Mei 2009

2	Rekayasa E-commerce Berbasis Web pada PT. Samwon Busana Indonesia,	Majalah Ilmiah DIAN, Universitas Dian Nuswantoro Semarang, ISSN 1412-3088	Vol.9/ No. 3/ September 2009
3	Perancangan dan Implementasi Sistem Kunci Elektronik pada Locker dengan Media Bluetooth	Jurnal Techno Science, FT UDINUS, ISSN 1978-9793	Vol 3/ No. 2/ Oktober 2009
4	Perancangan dan Implementasi Mobile Siadin (M-Siadin) pada Universitas Dian Nuswantoro Semarang Berbasis J2ME	Majalah Ilmiah DIAN, Universitas Dian Nuswantoro Semarang, ISSN 1412-3088	Vol. 10/ No.2/ Mei 2010
5	Rancang Bangun Peta Jalur Fiber Optik di Pt. Indonesia Commets Plus Regional Jawa Tengah dan Daerah Istimewa Yogyakarta secara Online	Jurnal Teknologi Informasi, Techno.COM, ISSN 1412-2693	Vol. 10/No. 4 November /2011
6	Kombinasi Algoritma RSA dan Algoritma Cipher Transposisi untuk Keamanan Database	Jurnal DIAN, Universitas Dian Nuswantoro, ISSN 1412-3088	Vol. 11/No.3/ September 2011
7	Rancang Bangun Aplikasi RMI (Remote Method Invocation) untuk Menghubungkan Sistem Pembayaran Udinus dengan Bank Jateng	Jurnal Teknologi Informasi Techno.COM ISSN 1412-2693	Vol. 11/No.2, Mei 2012
8	Teknik Proteksi SQL Injection dengan Konsep AMNESIA pada Aplikasi Web	Journal of Intellgent Systems and Business Intellegence ISSN 2302-268X	Vol. 1, No.2, September 2012
9	Rancang Bangun Aplikasi Penjadwalan Praktikum di Laboratorium Komputer Universitas Dian Nuswantoro dengan Pendekatan Algoritma Genetika	Majalah Ilmiah DIAN, Universitas Dian Nuswantoro Semarang, ISSN 1412-3088	Vol. 12, No.3, September 2012
10	Rancang Bangun Mobile GIS (Geographic Information System) Pencarian Lokasi ATM BNI Di Semarang Pada Media Ponsel Berbasis Android	SNASTIKOM MEDAN	2013

***F*. Pemakalah Seminar Ilmiah (*Oral Presentation*) dalam 5 Tahun Terakhir**

No	Nama Pertemuan Ilmiah / Seminar	Judul Artikel Ilmiah	Waktu dan Tempat
1	Seminar Jurnal Techno Science	Perancangan dan Implementasi Sistem Kunci Elektronik pada Locker dengan Media Bluetooth	2009 Fak. Teknik Udinus
2	COWISBI Pasca Sarjana Udinus	Teknik Proteksi SQL Injection dengan Konsep AMNESIA pada Aplikasi Web	2012 Pascasarjana Udinus
3	Lomba Mading Digital	Cloud Computing	2013 Udinus

***G*. Karya Buku dalam 5 Tahun Terakhir**

No	Judul Buku	Tahun	Jumlah Halaman	Penerbit
1	Pemrograman Jaringan VB 6.0 & MySQL	2012	162	Widya Karya Semarang

H. **Perolehan HKI dalam 5–10 Tahun Terakhir**

No	Judul/Tema HKI	Tahun	Jenis	Nomor P/ID

I. **Pengalaman Merumuskan Kebijakan Publik/Rekayasa Sosial Lainnya dalam 5 Tahun Terakhir**

No	Judul/Tema/Jenis Rekayasa Sosial Lainnya yang Telah Diterapkan	Tahun	Tempat Penerapan	Respon Masyarakat

J. **Penghargaan dalam 10 tahun Terakhir (dari pemerintah, asosiasi atau institusi lainnya)**

No	Jenis Penghargaan	Institusi Pemberi Penghargaan	Tahun

Semua data yang saya isikan dan tercantum dalam biodata ini adalah benar dan dapat dipertanggungjawabkan secara hukum. Apabila di kemudian hari ternyata dijumpai ketidaksesuaian dengan kenyataan, saya sanggup menerima sanksi.

Demikian biodata ini saya buat dengan sebenarnya untuk memenuhi salah satu persyaratan dalam pengajuan Hibah Penelitian Dosen Pemula.

.

Semarang, 13-03-2013
Anggota Pengusul,

(Ajib Susanto, M.Kom)

Lampiran 2. Active Submission

KNIF 2013
KONFERENSI NASIONAL INFORMATIKA
INFORMATIKA DI INDONESIA: ANTARA POTENSI, PROSPEK DAN TA

BERANDA TENTANG KNIF USER HOME KIRIM MAKALAH
FORMAT MAKALAH PENCARIAN PENGUMUMAN

Beranda > User > Author > Active Submissions

Active Submissions

ACTIVE ARCHIVE

ID	MM-DD SUBMIT	TRACK	AUTHORS	TITLE	STATUS
45	09-24	GEN	wijanarto	PROTOTYPE TRANSLATOR NOTASI ALGORITMIK UNTUK PEMBELAJARAN...	Awaiting assignment

1 - 1 of 1 Items

Pengiriman makalah dapat dilakukan melalui link berikut.
KIRIM MAKALAH

USER
Anda login sebagai...
wijanarto
- Profil
- Log Out

AUTHOR
Submissions
- Active (1)
- Archive (0)
- Kirim Makalah

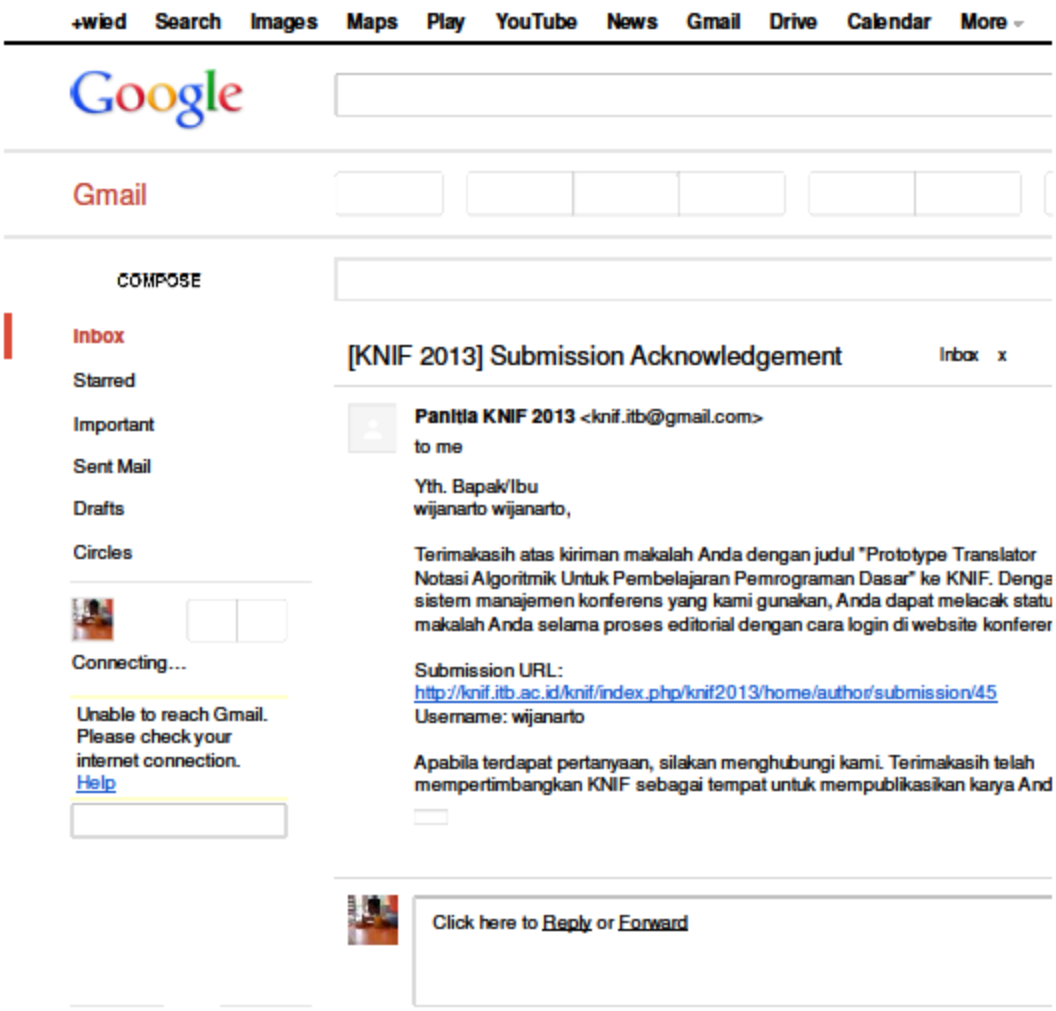
NOTIFICATIONS
- View (1 new)
- Manage

FONT SIZE

CONFERENCE CONTENT
Pencarian
All
Search

Conference Information
- Overview
- Call for Papers (May 27, 2013 - October 15, 2013)
- Jadwal Konferensi
- Registrasi
- Tanggal Penting

Lampiran 3. Submission Acknowledgement



Lampiran 4. Artikel Proceeding

Prototype Translator Notasi Algoritmik Untuk Pembelajaran Pemrograman Dasar

Wijanarto
Fakultas Ilmu Komputer
Universitas Dian Nuswantoro
Semarang
wijanarto.udinus@gmail.com

Ajib Susanto
Fakultas Ilmu Komputer
Universitas Dian Nuswantoro
Semarang
ajibsusanto@gmail.com

***Abstract*—Pemrograman dasar merupakan pondasi utama seseorang atau mahasiswa yang ingin belajar membuat program untuk menyelesaikan suatu masalah tertentu. Algoritma merupakan salah satu teknik untuk memecahkan masalah di bidang pemrograman yang di ekspresikan dalam bahasa pemrograman. Kesulitan utama seseorang dalam membuat solusi dalam bentuk bahasa formal merupakan masalah tersendiri, selain pemilihan alat atau aplikasi yang tepat untuk membantunya, bahkan untuk orang dengan latar belakang ilmu komputer. Paper ini mencoba menghasilkan Domain Specific Language (DSL) untuk pengajaran pemrograman dasar, dalam suatu rancangan aplikasi untuk mempermudah penyelesaian masalah dibidang pengajaran pemrograman dasar berbasis notasi algoritmik. Model notasi algoritmik yang di pilih merupakan model yang sudah pernah diterapkan dan diajarkan di perguruan tinggi. Grammar dihasilkan dengan bantuan ANTLR dan string template, yang di sesuaikan dengan model yang di pilih. Rancangan editor translator notasi algoritmik (ETNA), di bangun dengan model MVC dengan teknik RAD yang diperuntukan bagi mahasiswa di tahun pertama, yang dapat mentranslasikan notasi algoritmik ke bahasa c standar. Alat ini diharapkan membantu seseorang atau mahasiswa untuk dapat mendisain solusi dalam bentuk notasi algoritmik, tanpa memikirkan kerumitan dalam bahasa yang di pilihnya.**

Keywords—Translator; Notasi Algoritmik; Pemrograman; Domain Spesific Language

I. PENDAHULUAN

Pemrograman dasar merupakan pondasi utama sesorang atau mahasiswa yang ingin belajar membuat program untuk menyelesaikan suatu masalah tertentu. Sesederhana apapun, masalah yang harus di pecahkan harus dilakukan secara terstruktur dan ilmiah. Dalam dunia ilmu komputer atau teknik informatika langkah-langkah pemecahan masalah atau metode yang logis, terstruktur dan berhingga di sebut sebagai algoritma [1,4]. Seperti diketahui algoritma merupakan metode penyelesaian masalah yang umum dan banyak di lakukan hampir di seluruh bidang ilmu, seperti penentuan DNA [3], Teori graph dalam menentukan lintasan terpendek [7] dan masih banyak lagi.

Dalam studi yang pernah dilakukan di Afrika Selatan [2], keberhasilan pembelajaran pemrograman dasar di pengaruhi oleh, (1) lingkungan belajar (alat atau aplikasi) yang mendukung notasi yang sederhana, yang dapat mengkonstruksi notasi umum untuk bahasa pemrograman, (2) penampilan visual dari struktur program harus memungkinkan mahasiswa pemrograman dasar dapat memahami semantik konstruksi program dan (3) lingkungan kerja aplikasi harus melindungi mahasiswa untuk tidak melakukan interpretasi dan pemahaman yang salah. Di lain pihak pemahaman mahasiswa atau orang yang tertarik mempelajari pemrograman sering terkendala oleh bagaimana menggunakan bahasa itu sendiri. Artinya kesulitan utama mempelajari pemrograman di karenakan kesulitan bagaimana memahami semantik dari suatu bahasa pemrograman, seperti di jelaskan dalam [2]. Di Indonesia studi mengenai pembelajaran pemrograman dasar sangat sedikit, apalagi yang menyangkut alat penunjang atau ketepatan penggunaan aplikasinya. Dalam penelitian yang di lakukan Hidayanti [5], 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 [12], dalam matakuliah sejenis yaitu komputer dasar, menyimpulkan (masih dari aspek pedagogik) bahwa metode belajar

Penelitian paper ini di danai oleh direktorat jendaral perguruan tinggi dengan skim penelitian dosen pemula tahun 2013.

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, sehingga dapat mempermudah pemahanan serta mengurangi interpretasi yang salah dalam rangka menyelesaikan masalah dalam bidang pemrograman.

Model sederhana yang dipakai merupakan suatu translator notasi algoritmik yang secara otomatis dapat menghasilkan suatu bahasa pemrograman tingkat tinggi yang umum [11]. Sementara notasi algoritmik yang standar yang diberikan merupakan notasi yang sudah di ajarkan di perguruan tinggi [7]. Paper ini akan mencoba menghasilkan prototype translator notasi algoritmik ke dalam bahasa C standard untuk pengajaran pemrograman yang di buat dengan metode pengembangan system Rapid Application Development dengan Model View Controller dalam pemodelan aplikasinya.

II. MODEL DAN ARSITEKTUR TRANSLATOR NOTASI ALGORITMIK

A. Model Translator Notasi Algoritmik

Menentukan model standar standar notasi algoritmik merupakan jantung dari penelitian ini, di karenakan model ini merupakan kerangka utama dari aplikasi yang akan di hasilkan. Model notasi yang di pilih merupakan model notasi dalam [11]. Secara umum arsitektur model grammar yang di pakai adalah seperti dalam Fig. 1 sebagai berikut.

Fig. 1. Model Translator Notasi Algoritmik ke Bahasa C

Model diatas, secara umum membutuhkan suatu masukan berupa kode sumber dalam bentuk notasi algoritmik sesuai model yang di pilih dalam [8], dengan sedikit modifikasi, translator akan mengenali notasi berdasarkan grammar yang sudah di buat dan string template yang di tentukan dalam format bahasa c, sehingga secara otomatis translator akan mengenerate notasi menjadi kode sumber dalam bentuk bahasa c standar yang valid.

B. Arsitektur Translator Notasi Algoritmik

Suatu sistem aplikasi di kembangkan dengan suatu metode atau cara yang beragam, paper ini akan menggunakan dua pendekatan dalam mengembangkan aplikasi yaitu Rapid Application Development (RAD) dan Model View Contrller (MVC). Adapun rancangan arsitektur secara umum sebagai kerangka pikirnya adalah seperti Fig. 2 sebagai berikut .

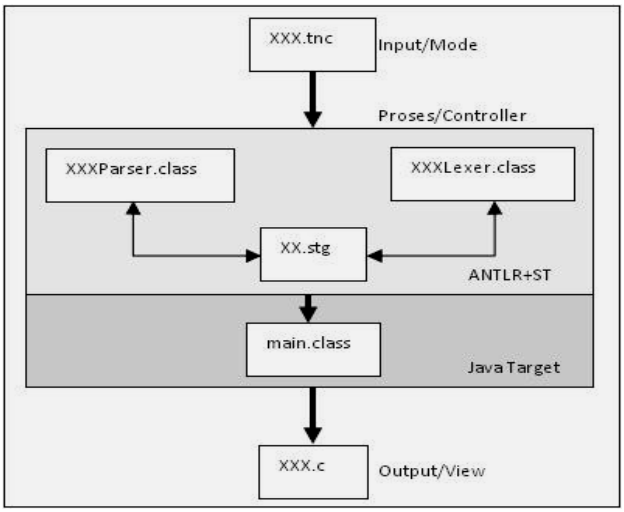


Fig. 2. Arsitektur Translator Notasi Algoritmik berbasis MVC

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. Model terdiri dari 3 buah langkah yaitu Notasi Algoritmik [11], yang berupa bahasa yang mudah di pahami manusia (natural) untuk mengekspresikan disain solusi suatu masalah pemrograman yang merupakan input yang akan di proses oleh translator dan akan menghasilkan (menggenerate) bahasa formal (bahasa C). Pendekatan yang di pakai dalam arsitektur ini adalah MVC (Model View Controller) berbasis pada paradigma object oriented. Model atau pendekatan ini pertama kali di sajikan dalam suatu laporan teknis yang di keluarkan oleh Xerox [8,9] dan dalam perkembangannya pendekatan ini banyak di pakai dalam pengembangan sistem khususnya yang berbasis pada paradigma obyek oriented [10].

Sementara itu teknik pengembangan sistem yang di pakai adalah *Rapid Application Development* (RAD). Disamping karena kemudahannya, teknik ini juga sangat cepat dalam membangun sistem skala menengah ke atas. Fase pengembangan sistem dengan metode RAD di bagi menjadi: (1) Fase *Planning*, untuk menentukan tujuan, fungsionalitas dan scope yang akan di kerjakan, (2) Fase *User Design*, yaitu menentukan interface dan bagaimana system akan bekerja dalam bentuk prototype, (3) Fase *Construction*, Prototype di konversi menjadi aplikasi yang sudah berfungsi, dengan pengkodean dan pengembangan fungsionalitas aplikasi, (4) Fase *Cutover*, merupakan fase terakhir dimana kegiatan utamnya adalah mencoba pada pemakai dan mendidik para pemakai [6].

III. HASIL DAN PERCOBAAN

Prototype ini masih dalam tahap pengembangang lebih lanjut, namun secara umum sudah mendekati tujuan yang di inginkan, yaitu dapat menerima masukan yang valid, mentranslasikan ke bahasa yang di maksud, mengkompilasi dan mencobanya secara terintegrasi dalam satu lingkungan kerja.

A. Use Case dan Block Diagram

Use case diagram terdiri dari dua entitas utama (grammar creator dan user), grammar creator membuat grammar dengan ANTLR dan menulis string template untuk bahasa c. Sedangkan user atau student yang menjalankan aplikasi sistem atau editor translator notasi algoritmik dengan menulis (dapat juga membuka file yang sudah ada) source notasi algoritmik. Selanjutnya terdapat tiga fungsi utama, selain fitur editor text yang umum (*syntax highlight*, *code completion*, dan sebagainya), *pertama*, fungsi *translate*, dimana aplikasi hanya melakukan translasi notasi algoritmik ke dalam bahasa C standar di dalam *console output* editor, *kedua*, fungsi *build and compile*, yang melakukan translasi notasi dan menyimpannya ke dalam *output* file source code bahasa c sekaligus melakukan kompilasi menjadi file *executable* dan *ketiga*, fungsi *run*, yang akan mengeksekusi file *executable* ke dalam *console* aplikasi atau *console system*, seperti terlihat pada Fig 3. Fungsi translate bergantung pada grammar (lexer dan parser) serta string template yang di hasilkan oleh konstruktor grammar. Sementar itu Compile dan build bergantung pada keberhasilan fungsi translate jika tidak terjadi kesalahan penulisan notasi algoritmik, untuk di kompilasi menggunakan kompiler C. Fungsi Run bergantung pada keberhasilan Compile and Build, untuk menjalankan program dengan system call atau dalam konsol ETNA. Kesalahan yang mungkin terjadi baik saat mentranslasikan (kesalahan notasi) atau mengkompilasikan (kesalahan syntax bahasa), membuat user untuk melakukan perbaikan seperlunya dalam ETNA, demikian skenario yang di buat untuk sistem ETNA pada Fig. 3.

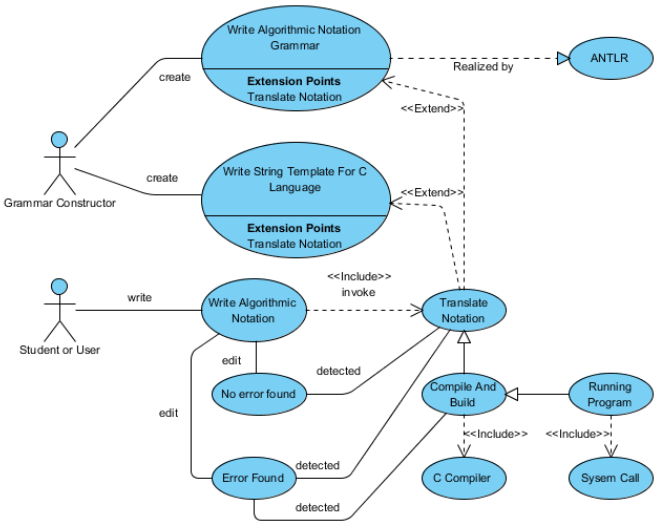


Fig. 3. Use Case Diagram Sederhana ETNA

Block Diagram disini di gunakan untuk menjelaskan detail ETNA, yang terdiri dari kumpulan paket dan kelas yang terintegrasi, seperti tampak pada Fig. 4 berikut

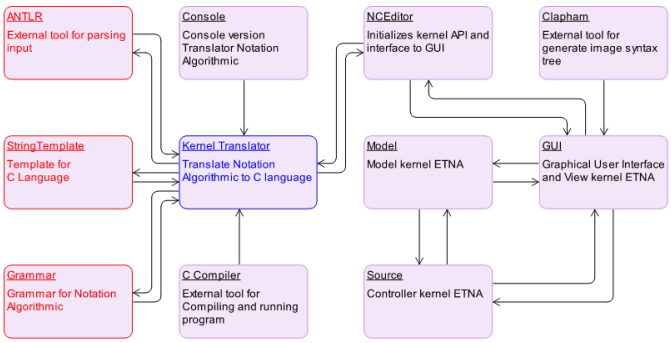


Fig. 4. Block Diagram Kernel Transator

Seperti terlihat, Translator (kernel ETNA) ditandai dengan warna biru, saling berkomunikasi dengan ANTLR, String Template dan grammar di tandai warna merah, 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 (disesuaikan kebutuhan), dipakai oleh kernel selama ETNA berjalan. GUI sebagai interface ETNA dan user memakai kernel saat diperlukan. Console merupakan translator dalam versi command line yang memakai kernel serta kompiler c sebagai tool luar untuk menghasilkan file eksekusi juga di pakai oleh kernel. Interaksi kernel dan GUI (ETNA), melalui NCEditor, ditandai warna magenta, akan di jelaskan selanjutnya seperti terlihat pada Fig. 5. Saat aplikasi dimulai kernel akan di inisialisasikan oleh NCEditor bersama-sama GUI sekaligus sebagai viewer ditandai warna biru, model dokumen serta source controller di tandai warna orange, sebagai implementasi model MVC.

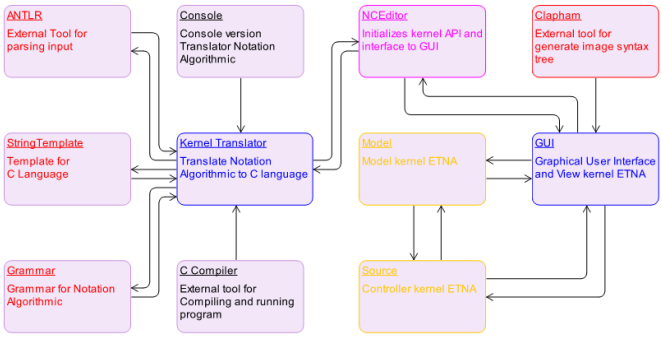


Fig. 5. Use case Diagram

Sementara tool dari luar Clapham di pakai 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). Seperti terlihat Kernel dan GUI tidak berkomunikasi secara langsung, tapi melalui Model dan Source yang di hubungkan oleh NCEditor untuk berkomunikasi dengan Kernel, dimana parser, lexer serta string template juga melalui kernel dan NCEditor untuk berkomunikasi dengan GUI sebagai interface user.

B. Editor Translator Notasi Algoritmik (ETNA)

Start-up ETNA, akan mengenerate *syntax tree* untuk keperluan bantuan visual bagi user yang baru belajar, sekaligus men-setup semua keperluan library untuk melakukan translasi (ETNA membuat temporary file library parser, lexer dan template karena ETNA dalam format terkompres jar). Setelah semua library siap ETNA dapat menerima source notasi algoritmik untuk dapat di edit sesuai keperluan, disini syntax notasi langsung dapat di kenali sebagai notasi algoritmik yang di modelkan, karena kebutuhan standar editor teks sudah seperti pada Fig. 6.

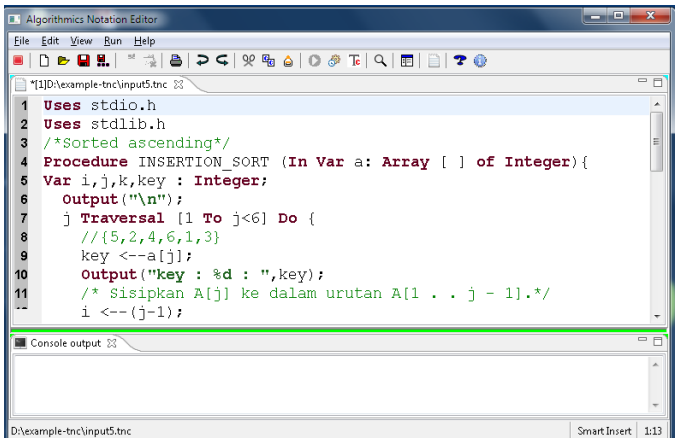


Fig. 6. Tampilan ETNA dengan file notasi algoritmik aktif

Fungsi utama dari ETNA, *run, build and compile* dan *translate* siap di jalankan jika pemakai ingin segera melakukan translasi, kompilasi dan eksekusi notasi yang telah di tulis dalam editor, Fig. 5 di bawah menunjukan 3 fungsi utama ETNA.

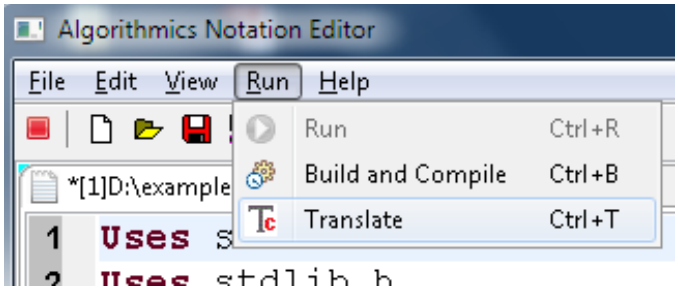
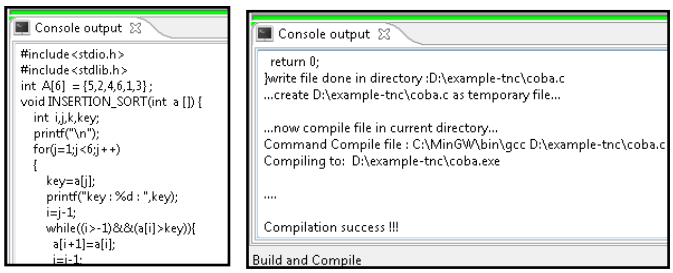
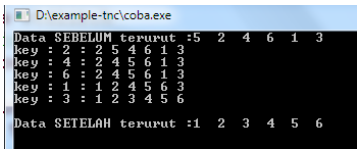


Fig. 7. Tiga fungsi atau menu utama ETNA

Berturut-turut di tampilkan hasil dari percobaan ETNA telah melakukan translasi, kompilasi dan eksekusi program hasil translasi notasi algoritmik ke bahasa c oleh ETNA terlihat pada Fig. 6 (a) merupakan hasil translasi notasi algoritmik dalam konsol ETNA dalam hal ini tidak ada penyimpanan, (b) merupakan hasil kompilasi, dimana file hasil translasi notasi akan di simpan dahulu dalam format c dan kemudian dilakukan kompilasi pada direktori yang sama dan (c) merupakan hasil eksekusi dari file yang telah terkompilasi untuk di jalankan di dalam atau diluar dalam lingkungan terpadu ETNA (konsol ETNA atau konsol system dari ETNA). Selain penggunaan sistem operasi linux, ETNA memerlukan kompiler c yang terpasang dan diset sebelumnya.



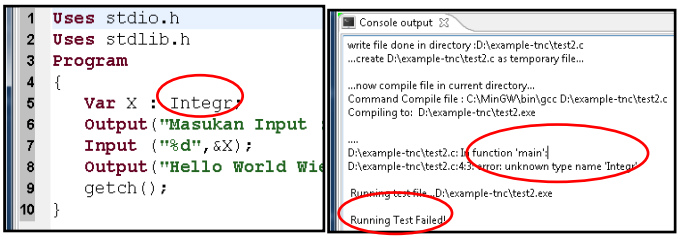
(a) (b)



(c)

Fig. 8. (a). Hasil translasi, (b) Hasil kompilasi, (c). Hasil eksekusi

Sementara itu, jika hasil translasi tidak valid karena terjadi kesalahan dalam notasi, maka ETNA memberi respon seperti Fig. 7 berikut



(a) (b)

Fig. 9. (a). Kesalahan Notasi; (b) Hasil Eksekusi dari notasi yang salah

Saat Notasi di tulis dengan format yang salah, ETNA saat ini belum dapat mendeteksi, tapi teta melakukan translasi ke bahasa c dengan sempurna, namun saat di lakukan kompilasi dan atau eksekusi, ETNA memberikan pesan kesalahan.

IV. PENUTUP

Dari hasil implementasi prototype sistem translator notasi algoritmik untuk pengajaran pemrograman dasar, penulis dapat menyimpulkan sementara bahwa dari model translator yang di pilih dapat membantu pemakai (mahasiswa) untuk memecahkan masalah pemrograman dasar dalam bentuk notasi, tanpa perlu memahami bahasa yang di pakai. pembelajaran pemrograman dasar. *Prototype* editor translator sudah memiliki fitur yang cukup untuk membantu dalam menulis notasi (*code completion*, *syntax highlight*, *error correction*). Kemampuan mentranslasikan, mengkompilasi dan dapat menjalankan program *on the fly* juga lebih tepat dapat memberi informasi mengenai kegagalan atau keberhasilan solusi yang di tulis dengan notasi algoritmik. Kedepan *prototype* ini perlu di lengkapi representasi *visual syntax tree* mengenai notasi algoritmik, yang muncul ketika di temukan kesalahan kode dan memunculkan grafik notasi syntax tree tepat di mana letak kesalahan di temukan serta bagaimana notasi yang benar, sehingga pengguna dapat memahami *syntax* yang benar dan segera membetulkan kesalahannya.

REFERENCES

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

[2] 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.

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

[5] 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.

[6] Ian Somerville, 2011, *Software engineering, 9th edition*, Pearson Education, Addison-Wesly, Boston, Massachusetts.

[7] Kruskal J. B, Jr., 1956, *On the shortest spanning subtree of a graph and the traveling salesman problem*. Proceedings of the American Mathematical Society, 7, pp. 48-50.

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

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

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

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

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

[13] 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.

Lampiran 5. File Algoritmik.g

```
grammar Algoritmik;
options {
      backtrack=true;
      memoize=true;
      k=2;
      language=Java;
      output=template;
}

program
      :     declaration+
    ;
declaration
    : lib
    | macro
    | declType
    | variable
    | constant
    | function
    | mainfunction
    ;
storage_class_specifier
      :     'Extern'
      |     'Static'
      |     'Auto'
      |     'Register'
      ;
type
    : 'Integer'
    |   'Character'
    |   'Real'
    | 'String'
    | 'Short'
      |     'Long'
      |     'Double'
      |     'Signed'
      |     'Unsigned'
    | structTag
    | enumTag
      |     declarator
    ;
typedeclarator
      : p+=ID (',' p+=ID)*
      ;
tagIO
      :     'Input'
      |     'Output'
      ;
statIO
      :     tagIO '(' STRING_LITERAL (',' paramE_or_PE )* ')'

      ;
lib
      :     'Uses' use
      ;
use
      :     postfix_expression
      ;
```

```
macro
      :     statMacro
      ;
statMacro
      :     'Def' (type) ('As' constSwitch)*
      |     'IfNotDef' (type)
      |     'EndDef'
      |     'ElseDef'
      ;
mainfunction

      : 'Program'
            block
      ;
arg
      :     (argv','args)
      ;
argv
      :     ('int' 'argv')
      ;
args
      :     ('char' '*' 'args' '[]')
      ;
/*=====Tipe bentukan=====*/
declType
      :     'Type' (declarator* pointer* '='* advtype* type* ':'*
structForm* enumForm*)
                        ;
/*=====Tipe enum=====*/
enumForm
      :      '(' constSwitch(','  constSwitch)*')'
      ;
enumTag
      :     enumKey ID pointer*     ;
enumKey
      :     'Enum'
      ;
enumField
      :     val_array
      ;
/*=====bentuk struktur=====*/
structForm
      :     contents
      ;
structTag
      :     structKey ID pointer*
      ;
structKey
      :     'Struktur'
      |     'Union'
      ;
tagName
      :     ID
      ;
getTag
      :     listID*
      ;
listID
      :     (options{backtrack=true;}:p+=ID (',' p+=ID)*)
```

```
      ;
contents
      :     '<' field(','  field)*'>'
      ;

field
      :     variable
      ;
block
    :  '{'
       ( variable   )*
       ( stat  )*
       LINE_COMMENT*
       COMMENT*
       WS*
       '}'
    ;
declarator
    :   ident
    ;

ident
      : ID (',' ID)*
      ;
type_qualifier
      :     'Constant'
      |     'Volatile'
      ;

pointer
      :
            //'@'  pointersymbol+
            p+=pointersymbol
(options{backtrack=false;memoize=false;}:p+=pointersymbol)*
      ;
pointersymbol
      :     '^'
      ;
varOrStruct returns[String s]
      :     'Var' {$s="Var";}
      |     'Struk' {$s="Struktur";}
      ;
isStruktur returns [boolean b]
    :       varOrStruct {if($varOrStruct.s.equals("Var")) $b=true;
else $b=false;}
    ;
initvar
    : (exprs  | '{' ( val_array (','  val_array)*)? '}' )
    ;

val_array
      :     DECIMAL_LITERAL (',' DECIMAL_LITERAL)*
      |     FLOATING_POINT_LITERAL (',' FLOATING_POINT_LITERAL)*
      |     CHARACTER_LITERAL (',' CHARACTER_LITERAL)*
      |     STRING_LITERAL (',' STRING_LITERAL)*
      ;
advtype
      :     'Array' '[' (  argArray (','  argArray )* )? ']' 'of'
```

```
      ;
argArray
      :     startArray 'To' endArray
      ;

startArray
      :     exprs
      ;
endArray
      :     exprs
      ;
function

    :   funcType+ ID
        '('  (  formalParameter ( ','  formalParameter )* )? ')'
((':' type)* pointer*)
        block

    ;
funcType
      :     'Function'
      |     'Procedure'
      ;

inout
      :'Out'
      |'In'
      ;
formalParameter
    :   inout* 'Var' ( paramName (','  paramName)*)? ':' type
          ;
paramName
scope {
    String p;
}
      :     ID pointer* {$paramName::p=$pointer.text;}
      ;
funcName
      :
      (     ID
      |     alokasi
      |     retStat
      )
      ;
retStat
      :     'Return'
      ;
alokasi
      :     'Alokasi'
      |     'CAlokasi'
      |     'Dealokasi'
      ;
func_callStat
      :     funcName '(' (actualParameters)? ')'
                  ;
actualParameters
      :     paramE_or_PE (',' paramE_or_PE)*

      ;
```

```
paramE_or_PE
      :     exprs
      |     postfix_expression
      |     func_callStat
      |     primary_expression
      |     ID
      ;
constant
scope {
    String atype;
}
      //storage_class_specifier
    :  'Constant'  declarator(pointer)* ':' (advtype)*  type  '='
initvar ';'
      {$constant::atype=$advtype.text;}
    ;
variable
scope {
    String initial;
    String atype;
}
      //storage_class_specifier
    :   varOrStruct isStruktur* declarator pointer* ':' advtype*
type*  ('<==' initvar )* ';'

      {$variable::initial=$initvar.text;$variable::atype=$advtype.
text;}
     ;
stat

    :       forStat
    | ifstat
    | switchstat
    |       exprs semi
    |       block
    |       assignStat semi
    | func_callStat semi
    | statIO semi
    | break_or_cont semi
    |       semi
    ;
forStat

                  //d=exprs masalah utk assignSat
    :       ID* 'Traversal' ('[' s=logCond range e=logCond ']')*
('Step' exprs_or_assignStat)* 'Do'
               block ';'
    |       'Repeat' block 'Until' '('(logCond)*')' ';'

      |     'While' '('(logCond)*')' 'Do' block ';'
      ;
exprs_or_assignStat
      :     logCond
      |     lvalue assignOpr casting* rvalue

      ;
opt returns[String s]
      :     'To' {$s="To";}
      |     'DownTo' {$s="DownTo";}
```

```
      ;
range returns [boolean b]
    :       opt {if($opt.s.equals("To")) $b=true; else $b=false;}
    ;
ifstat
      :     'If' '('(logCond)')' 'Then' thenstat ('Else'
elsestat)*
      ;
logCond
      :     exprs
      |     postfix_expression
      |     rvalue
      |     constSwitch
      |     primary_expression
      |     ID
      ;
break_or_cont
      :     'Break'
    | 'Continue'
      ;
thenstat
      :     stat
      ;

elsestat
      :     stat
      ;
switchstat
      :     'Depend On' '('ID')' '<*'
                  (switchPart (',' switchPart)*)?
                  elseswithcpart*
             '*>'       ;
constSwitch
      :
      (     ID pointer*
      |     CHARACTER_LITERAL
      |     STRING_LITERAL
      |     DECIMAL_LITERAL
      |     FLOATING_POINT_LITERAL
      )+
      ;
switchPart
      :     (constSwitch ':' stat)
      ;
elseswithcpart
      :     ('else' constSwitch ':' stat)
            ;
assignStat
    : lvalue assignOpr casting* rvalue
    ;
rvalue
      :     postfix_expression
      |     exprs
      |     unary_opr (exprs|postfix_expression)+

      |     func_callStat
      ;
conditional_expression
      :     '?' '('b=exprs')' ':' '('c=exprs')'
```

```
      ;
lvalue
      :     unary_expression
      |     postfix_expression
      ;
unary_expression
      :     postfix_expression
      |     'SizeOf' (type_or_unexp)
      ;
casting
      :     '('type pointer*')'
      ;
type_or_unexp
      :     casting
      |     '('unary_expression')'
      ;
postfix_expression
      :     primary_expression
      (     '[' exprs ']'
      |     '[' ID ']'
      |     dot_or_rarrow+
      |     '(' argument_expression_list ')'
      |     conditional_expression
      |     func_callStat
      )*

      ;

argument_expression_list
      :     assignStat (',' assignStat)*
      ;
dot
      :     '.'   postfix_expression
      ;
rarrow
      :     '->' postfix_expression
      ;
dot_or_rarrow
      :     dot
      |     rarrow
      ;
primary_expression
      :     ID pointer*
      |     constexpression
      |     '(' exprs ')'
      ;
constexpression
      :     HEX_LITERAL
      |     OCTAL_LITERAL
      |     DECIMAL_LITERAL
      |     CHARACTER_LITERAL
      |     STRING_LITERAL
      |     FLOATING_POINT_LITERAL
      ;
assignFunction
      :     (ID|postfix_expression) pointer* assignOpr
func_callStat semi*

      ;
```

```
unary_opr
      :     '&'
      //|   '*'
      |     '(+)'
      |     '(-)'
      |     'Not'
      |     'Negasi'
      |     'Inc'
      |     'Dec'
      ;

assignOpr
      :     '<--'
      |     'SumPlus'
      |     'SumMin'
      |     'SumMul'
      |     'SumDiv'
      |     'SumMod'
      |     'SumShl'
      |     'SumShr'
      |     'SumAnd'
      |     'SumIncl'
      |     'SumExcl'
      ;
term
      :     lvalue
      |     postfix_expression
      |     '(' exprs ')'
      |     CHARACTER_LITERAL
      |     STRING_LITERAL
      |     DECIMAL_LITERAL
      |     FLOATING_POINT_LITERAL
      |     func_callStat
    ;

postfix
      :     term (unary_opr)*
      ;
negation
      :     (unary_opr)* postfix
      ;
unary
      :     (unary_opr)* negation
      ;
mult
      :     a=unary ((oprmult) b=unary)*
      ;
oprmult
      :     '*'
      |     'Div'
      |     'Mod'
      ;
add
      :     a=mult ((opradd) b=mult)*
      ;
opradd
      :     '+'
      |     '-'
      ;
```

```
relation
      :     a=add ((oprrel) b=add)*
      ;
oprrel
      :     '='
      |     '<>'
      |     '<'
      |     '<='
      |     '>'
      |     '>='
      ;
shiftbit
      :     a=relation ((oprshift) b=relation)*
      ;
oprshift
      :     'shl'
      |     'shr'
      ;
exprs
      :     a=shiftbit  ((oprexpr) b=shiftbit )*

      ;
oprexpr
      :     'And'
      |     'Or'
      ;

semi
      :     ';'
      ;
ID
      :     LETTER (LETTER|'0'..'9')*
      ;
fragment LETTER
      :     '$'
      |     '.'
      |     'A'..'Z'
      |     'a'..'z'
      |     '_'
      ;
CHARACTER_LITERAL
      :     '\'' (EscapeSequence | ~('\'' | '\\')) '\''
      ;
STRING_LITERAL
      :     '"' (EscapeSequence | ~('\\'|'"'))* '"'
      ;
HEX_LITERAL
      :     '0' ('x'|'X') HexDigit+ IntegerTypeSuffix?
      ;
DECIMAL_LITERAL
      :     ('0'|'1'..'9' '0'..'9'*) IntegerTypeSuffix?
      ;
OCTAL_LITERAL
      :     '0' ('0'..'7')+ IntegerTypeSuffix?
      ;
fragment HexDigit
      :     ('0'..'9'|'a'..'f'|'A'..'F')
      ;
fragment IntegerTypeSuffix
```

```
      :     ('u'|'U')* ('l'|'L')
//    |     ('u'|'U')  ('l'|'L')*
      ;

FLOATING_POINT_LITERAL
      :     ('0'..'9')+ '.' ('0'..'9')* FloatTypeSuffix?
//Exponent? FloatTypeSuffix?
//    |     '.' ('0'..'9')+ Exponent? FloatTypeSuffix*
//    |     ('0'..'9')+ Exponent FloatTypeSuffix*
//    |     ('0'..'9')+ Exponent* FloatTypeSuffix
      ;
fragment Exponent
      :     ('e'..'E') ('+'|'-')? ('0'..'9')+
      ;
fragment FloatTypeSuffix
      :     ('f'|'F'|'d'|'D')
      ;
fragment EscapeSequence
      :     '\\' ('b'|'t'|'n'|'f'|'r'|'\"'|'\''|'\\')
      |     OctalEscape
      ;
fragment OctalEscape
      :     '\\' ('0'..'3')('0'..'7') ('0'..'7')
      |     '\\' ('0'..'7') ('0'..'7')
      |     '\\' ('0'..'7')
      ;
fragment UnicodeEscape
      :     '\\' 'u' HexDigit HexDigit HexDigit HexDigit
      ;
WS
      :     (' ' | '\r'|'\t'|'\u000C'|'\n') {$channel=HIDDEN;}
      ;
COMMENT
      :     '/*' (options {greedy=false;} : . )* '*/'
{$channel=HIDDEN;}
      ;
LINE_COMMENT
      :     '//' ~('\n'|'\r')* '\r'? '\n' {$channel=HIDDEN;}
      ;
```

Lampiran 6. File Algoritmik.stg

```
group Algoritmik;
program(libs,globals,functions,mainfunctions) ::=<<
<libs; separator="\n">
<globals; separator="\n">
<functions; separator="\n">
<mainfunctions; separator="\n">
>>
/*library*/
setLib(lib)::="#include\<<setFile(lib)>\>"
setFile(name)::="<name>"
getTypedeclarator(list)::= <<<list;separator=",">
>>
/*main program*/ //int argc,char *argv[]
mainfunct(arg,locals,stats)::=<<int main(){
  <locals; separator="\n">
  <stats; separator="\n">
  return 0;
}
>>

pointer(a)::="<a:{n|<n>}>"
castexprs(t,p)::="(<t> <p>)"
/*macro*/
mdef(type,val)::="#define <type> <if(val)><val><endif>"
mifndef(type)::="#ifndef <type>"
mendif()::="#endif"
melse()::="#else"
getArg(v,s)::="(<v><s>)"
nameMacro(id,type)::="<id><type>"
setMacro(id,expr)::="#define<id> <expr>"
//deklarasi tipe  data primitive dan array
struktur(field)::="<getVar(field)>"
getVar(v)::=<< <v;separator="\n">
>>
struct(point,name,idx)::=<<
<if(idx)>struct <pointer(point)> <name>{
   <struktur(idx)>
};<endif>
>>
getName(name)::="<typeName(name)>"
typeName(id)::=<< <id;separator=",">
>>
//getEndArray(idx)
getArrayForm(idx)::=<<
<if(idx)><idx;separator="][""><else>[]<endif>
>>
getEndArray(startArray,endArray)::=<<
<startArray,endArray:{s,e|<e>}>
>>
//definisi variabel primitive dan array
variable(v,point,type,name,idx) ::= <<
<if(!type)><if(!v)>struct <endif><pointer(point)> <name><if(idx)>{
<idx>
};<endif><endif>
<if(type)><type> <pointer(point)>
<name><if(idx)>[<idx>]<endif>;<endif>
>>
//definisi variabel primitive serta inisialisasinya
```

```
initVar(point,type,name,idx,init)::="<type> <pointer(point)>
<name><if(idx)>[<idx>]<endif> <if(init)> <assigninit(init)> ;
<else> ; <endif>"
//assignment inisialisasi variabel
assigninit(init)::="= <init>"
/*-------tipe array-----*/
//definisi variabel primitive dan array
constant(pointer,type,name,idx) ::="const <type> <pointer>
<name><if(idx)>[<idx>]<endif>;"
//definisi variabel primitive serta inisialisasinya
initConst(pointer,type,name,idx,init)::="const <type> <pointer>
<name><if(idx)>[<idx>]<endif> <if(init)> <assigninit(init)> ;
<else> ; <endif>"
/*-------tipe array-----*/

//definisi variabel array size index serta inisialisasinya
initarray(type,name,size,init)::="<type> <name>[<size>] <if(init)>
<assignarray(init)> ; <endif> ; <endif>"
//assignment inisialisasi variabel array
assignarray(init)::= "= {<init>}"

function(type,funcType,name,args,locals,stats,s) ::= <<
<if(s)><funcType><type> <name>(<args ; separator=",">);<else>
<funcType><type> <name>(<args ; separator=",">) {
    <locals; separator="\n">
    <stats; separator="\n">
}<endif>
>>
//(vars={$getTag.st},point={$pointer.st},atype={$type.st},btype={$
advtype.st},form={$structForm.st},ctype={$enumForm.st})
//update scalar belum
declType(vars,point,atype,btype,form,ctype,stype)::= <<
<if(vars)>typedef
<if(ctype)>enum{<ctype>}<point><vars>;<elseif(btype)><atype>
<vars>[<getArrayForm(btype)>];<elseif(atype)><atype>
<point><vars>;<else>struct{
<form>
}<vars>;<endif><endif><if(!vars)><if(ctype)><atype>{<ctype>};<else
><atype>{
 <form>
};<endif><endif>
>>
structTag(key,tag,point)::="<key> <tag><point>"
enumTag(key,tag,point)::="<key> <tag><point>"
structForm(cont)::= <<
{
  <cont>
}
>>
tagName(tag)::=<< <tag;separator=",">
>>
getenumVal(v)::=<< <v;separator=",">
>>
getTag(tag)::="<tagName(tag)>"
getenum(e)::=<<
<e;separator=",">
>>
getenumfield(f)::=<<<f;separator=",">
>>
```

```
/*primitive type*/
inputstat()::="scanf"
outputstat()::="printf"
type_int() ::= "int"
type_char() ::= "char"
type_float() ::= "float"
type_pointer()::= "*"
type_string()::= "char *"
type_void()::= "void"
type_short()::= "short"
type_long()::= "long"
type_double()::= "double"
type_signed()::= "signed"
type_unsigned()::= "unsigned"
type_user_object(name)::="<name>"
/*=====operator======*/
postfix(opr,term)::="<if(opr)><term><opr><else><term><endif>"
//postfix(opr,term)::="<if(opr)><term><opr><else><term><endif>"
negasi(opr1,term)::="<if(opr1)><opr1><endif><term>"
unary(opr1,unary)::="<if(opr1)><opr1><endif><unary>"

mult(op1,opr,op2)::="<if(opr)><op1><opr><op2><else><op1><endif>"
add(op1,opr,op2)::="<if(opr)><op1><opr><op2><else><op1><endif>"
relation(op1,opr,op2)::="<if(opr)><op1><opr><op2><else><op1><endif
>"
referensi(op1,opr,op2)::="<op1><if(opr)><opr><op2><endif>"
shiftbit(op1,opr,op2)::="<if(opr)><op1><opr><op2><else><op1><endif
>"
//exprs(l,r,op1,opr,op2)::="<if(l)><l><if(opr)><op1><opr><op2><els
e><op1><endif><r><else><if(opr)><op1><opr><op2><else><op1><endif><
endif>"
exprs(op1,opr,op2)::="<if(opr)>(<op1>)<opr>(<op2>)<else><op1><endi
f>"
assignEq()::="="
parameter(a,type,tinout,name) ::=<<
<name:{n|<type> <tinout> <n> <a>};separator=", ">
>>
paramPE(pe)::=<<<pe:{n|<n>};separator=",">
>>
paramE(ap)::=<<<ap:{n|<n>};separator=",">
>>
fterm(f)::="<f>"
procedurecall(f)::="<f>;"
functioncall(f,ap)::="<if(ap)><f>(<ap>)<else><f>()<endif>"
getfParam(p)::=<< <p;separator=",">
>>
statProc(p)::="<p>;"
getAP(f,r)::="<f><if(r)><r><endif>"
getAParam(p)::=<< <p;separator=",">
>>
statement(expr) ::="<expr>;"
statementList(locals,stats) ::= <<
{
    <locals; separator="\n">
    <stats; separator="\n">
}
>>
statIO(tag,arg,exp)::="<tag>(<arg><if(exp)>,<exp><endif>);"
```

```
//for(<id>=<start>;<if(!range)><id>\>=<end>;<if(step)><id>=<id>-
<step><else><id>--
<endif><endif><if(range)><id>\<=<end>;<if(step)><id>=<id>+<step><e
lse><id>++<endif><endif>)
//range=true=Inc
forLoop(id,range,start,end,step)::= <<
for(<if(id)><id>=<start>;<if(range)><if(!step)><end>;<id>++)<else>
<end>;<step>)<endif><else><if(!step)><end>;<id>--
)<else><end>;<step>)<endif><endif><else>;;)<endif>
{
    <locals; separator="\n">
    <stats; separator="\n">
}//endfor
>>
ifstat(log,t,e2)::= <<if(<log>)<t><if(e2)><e2><endif>
>>
//ifreg(e1,e2,s0,s)::=<<if(<e1>)<s0><if(e2)><e2><endif><if(s)><s><
endif>;//endif>>
//getelseifstat(e,s)::="else if(<e>)<s>"
getthenstat(s)::="<s>"
getelsestat(s)::="else <s>"
switchstat(id,spart,elseswitch)::=<<
switch(<id>){
  <spart:{s|<s>};separator="\n">
     <if(elseswitch)>  default:<elseswitch;separator="\n">
};<else>
};<endif>//endswitch
>>
getSwitchPart(id,stat)::=<<
case <id>:<stat>
>>
getElseswitch(id,stat)::="  <stat>"
setbreakcont(bc)::="<bc>;"
whilestat(e)::=<<
while(<if(e)><e><else>1<endif>){
  <locals; separator="\n">
  <stats; separator="\n">
};//endwhile
>>
repeatstat(e)::=<<
do{
  <locals; separator="\n">
  <stats; separator="\n">
}while(<if(e)><e><else>1<endif>);//enddo
>>
decar(id,ue)::="<id><ue>"
assignref(p,c,d)::= "<p><c><d>"
unary_exprs(a,e,pe)::="<a><if(e)><e><else><pe><endif>"
setDot(id)::=<<<id>
>>
setArrow(id)::="-><id>"
refArrow(a)::="<setArrow(a)>"
CastInc(uop,c)::="<uop> <c>"
sizeExprs(e)::="sizeof<e>"
//postexprs(pe,e,id,ref)::="<if(e)><pe>[<e>]<elseif(id)><pe>[<id>]
<elseif(ref)>[<pe><ref>]<else><pe><endif>" //<elseif(op)><op><pe>
postexprs(pe,e,id,dr,et,ce,f)::="<if(e)><pe>[<e>]<elseif(id)><pe>[
<id>]<elseif(dr)><pe><dr><elseif(et)><pe><et><elseif(ce)><pe><ce><
elseif(f)><pe><f><else><pe><endif>"
```

```
argExprsList(al)::="<al>"
assign(lval,asg,e,c)::= "<lval><asg><if(c)><c><endif><e>;"
assignLoop(lval,asg,e,c)::= "<lval><asg><if(c)><c><endif><e>"
assignfunc(lhs1,pe,point,asg1,rhs1,s)::=
"<if(!s)><if(pe)><pe><else><lhs1><endif><point> <asg1>
<rhs1><else>;<endif>"
refArrayTerm(id,arr)::="<id><arr>"
refTerm(pe,id)::="<id><pe>"
refVar(id)::="<id>"
declsuff(id,e)::="[<id><e>]"
termdeclsuff(a,d)::="<a><d>"
refVariabel(id,point,ds)::="<if(!point)><id><ds><elseif(!ds)><poin
t><id><else><point><id><ds><endif><endif>"
getParam(id)::=<<
<id;separator=",">
>>
getParamPoint(id,point) ::= <<
<id:{n|<point><n>};separator=",">
>>
iconst(value) ::= "<value>"
cconst(value) ::= "<value>"
sconst(value) ::= "<value>"
sconstpas(value)::="<value>"
fconst(value) ::= "<value>"
globalVariable::=initVar
globalConstant::=initConst
//globalTypedef::=typeDecl
globalStruct::=struct
globaldeclType::=declType

//constant expresseion
hexconstexpr(hex)::="<hex>"
octconstexpr(oct)::="<oct>"
decconstexpr(dec)::="<dec>"
chrconstexpr(chr)::="<chr>"
strconstexpr(str)::="<str>"
strconstexprpas(str)::="<str>"
floconstexpr(flo)::="<flo>"
//new edition
ifCond(e,ce)::="<if(ce)><e><ce><else><e><endif>"
condExprs(b,c)::="? <b> : <c>"

castexp(t,c,ue)::="<if(ue)><ue><else><t> <c><endif>"
unaryexp(pe,a,uo,ce,s)::="<if(pe)><pe><elseif(a)><a><elseif(uo)><u
o><ce><else><s><endif>"
unInDec(a,ue)::="<a><ue>"
sizeOF(p,ue,t)::="<if(ue)><p><ue><else><p><t><endif>"
postexp(pe,e,blank,arg,a)::=<<
<if(e)><pe><e><elseif(blank)><pe><blank><elseif(arg)><pe><arg><els
e><pe><a><endif>
>>
refdot(id)::=".<id>"
refpoint(id)::="-\><id>"
innerexpr(e)::="(<e>)"
idp(id,p)::="<if(p)>(<p><id>)<else><id><endif>"
//operator
oprunNeg()::="!"
oprunNot()::="~"
oprassignEq()::="="
```

```
oprmultdiv()::="/"
oprmultmod()::="%"
oprreleq()::="=="
oprrelneq()::="!="
oprshl()::="\<\<"
oprshr()::="\>\>"
oprexprsor()::="||"
oprexprsand()::="&&"
returnStat(id)::="<id>"
pointersymbol()::="*"
advtypetag(a,b)::="<a> <b>"
```

Lampiran 7. Laporan Penggunaan Dana

URAIAN PENGGUNAAN DANA

Dibiayai oleh Universitas Dian Nuswantoro Melalui LP2M dengan No. Kontrak : 009/A.35-02/UDN.09/IX/2013

Honor					
Honor	**Honor/Jam (Rp)**	**Waktu**	**Minggu**	**Honor Per Tahun**	
Ketua	8850	5	30	1327500	
Anggota	7375	4	25	737500	
			Sub Total (Rp)	**2065000**	
Peralatan Penunjang					
Material	**Justifikasi Pemakaian**	**Kuantitas**	**Harga Satuan (Rp)**	**Total Biaya**	
USB Flash Disk 8 GB	Copy Data & Aplikasi Lab	2 Buah	100000	200000	
MMC Camera 8 GB	Dokumentasi	1 Buah	75000	100000	
Upgrade Memori 2 GB	Menambah Memori Komputer	1 Buah	974000	974000	
Harddisk Eksternal 1 TB	Backup Data	1 Buah	1250000	1250000	
Modem CDMA	Koneksi Internet	1 Buah	950000	950000	
			Sub Total (Rp)	**3474000**	
Bahan Habis Pakai					
Material	**Justifikasi Pemakaian**	**Kuantitas**	**Harga Satuan (Rp)**	**Total Biaya**	
Pembelian ATK	Total Pembelian ATK			1620000	
Pulsa Internet 3	Total Pulsa Data	2 Paket	125000	250000	
Pulsa Telepon	Total Pulsa Telpon	1 Paket	100000	100000	
Konsumsi	Total Konsumsi		755500	755500	
			Sub Total (Rp)	**2725500**	
Perjalanan					
Material	**Justifikasi Perjalanan**	**Kuantitas**	**Harga Satuan (Rp)**	**Total Biaya**	
Bahan bakar	Perjalanan dalam kota	30.76 liter	6500	200000	
Sewa Mobil	Perjalanan Dalam kota	1 hari	250000	250000	
			Sub Total (Rp)	**450000**	
Lain-lain					
Kegiatan	**Justifikasi**	**Kuantitas**	**Harga Satuan (Rp)**	**Total Biaya**	
PPH(2%)		1	184545	184545	
PPH 21(5%/6%)		1	145000	145000	
PPN(10%)		1	922727	922727	
			Sub Total (Rp)	**1252272**	
			Total anggaran yang telah di gunakan	9966772	