ISSN: 2302-3805

Membangun Chatbot Berbasis AIML dengan Arsitektur Pengetahuan Modular

Bayu Setiaji [1] , Ema Utami [2], Hanif Al Fatta [3]
[1] *Dosen Program Studi Teknik Informatika STMIK AMIKOM Yogyakarta*
[2, 3] *Dosen Magister Teknik Informatika Program Pasca Sarjana STMIK AMIKOM Yogyakarta*
Jl. Ring Road Utara Condong Catur Depok Sleman Yogyakarta
Telp: (0274) 884201-207, Fax: (0274) 884208 Kodepos: 55283
email: bayusetiaji@hotmail.co.id [1], ema.u@amikom.ac.id [2], hanif.a@amikom.ac.id [3]

Abstrak

Chatbot adalah salah satu sistem cerdas yang dihasilkan dari Pemrosesan Bahasa Alami atau Natural Language Processing (NLP) yang merupakan salah satu cabang dari Kecerdasan Buatan atau Artificial Intelligence (AI). Chatbot memungkinkan manusia dapat berkomunikasi dengan mesin menggunakan perantaraan bahasa alami. Bentuk komunikasi yang terjadi adalah melalui percakapan menggunakan media tulisan. Percakapan dengan chatbot dapat berupa obrolan biasa atau obrolan pada tema – tema tertentu yang melibatkan disiplin ilmu yang lain.

Artificial Intelligence Markup Language (AIML) adalah bahasa yang dapat digunakan untuk menyusun logika chatbot. AIML berisi kumpulan pola dan respon yang dapat digunakan oleh chatbot untuk penelusuran jawaban setiap kalimat yang diberikan. Interpreter AIML diperlukan untuk menerima input dan melakukan penelusuran jawaban pada dokumen AIML. Saati ini tersedia banyak interpreter AIML dalam berbagai bahasa pemrograman sehingga proses pembuatan chatbot dapat terfokus pada penyusunan dokumen AIML. Selain itu juga saat ini di internet banyak tersedia dokumen AIML siap pakai untuk berbagai bidang percakapan. Dengan AIML dapat dibuat interpreter yang memungkinkan pengetahuan chatbot bersifat modular di mana pengetahuan chatbot terpisah dari chatbot utama.

Hasil penelitian adalah sebuah chatbot yang berupa service yang dapat digunakan untuk percakapan melalui program client. Service chatbot terhubung dengan modul – modul pengetahuan tertentu yang bersifat independen.

Kata kunci :
AIML, chatbot, interpreter, parser, modul

1. Pendahuluan

Chatbot adalah salah satu sistem cerdas yang saat ini mulai banyak dibuat. Pembuatan *chatbot* memiliki banyak tujuan mulai dari sekedar untuk percakapan biasa sampai pada sistem pendukung *customer service* pada perusahaan. Salah satu metode pembuatan *chatbot* adalah dengan menggunakan bahasa AIML yang berisi sekumpulan pola (*pattern*) dan *template* yang digunakan *chatbot* untuk penelusuran jawaban dari kalimat yang masuk. Proses ini memerlukan *interpreter* AIML sebagai mesin utama penerima input dan pencari pasangan *pattern-template* yang kemudian akan menghasilkan respon untuk diberikan sebagai jawaban.

Saat ini banyak tersedia *interpreter* yang ditulis dalam berbagai bahasa pemrograman sehingga pembuatan *chatbot* dapat terfokus pada penyusunan dokumen AIML. Pembuatan *chatbot* untuk tema percakapan tertentu juga dipermudah dengan adanya ribuan dokumen AIML yang dapat diunduh sehingga dapat langsung diintegrasikan dengan *interpreter*. Beberapa dokumen AIML dengan tema yang berbeda juga dapat diintegrasikan langsung sehingga memungkinkan 1 *chatbot* memiliki banyak tema percakapan. Semakin banyak dokumen AIML yang dimasukkan dalam interpreter akan membuat chatbot semakin pintar. Namun, kondisi tersebut berpengaruh pada kinerja *interpreter* yang akan semakin melambat karena harus melakukan penelusuran *pattern-template* lebih banyak.

Berdasar latar belakang di atas dapat dibuat sebuah *chatbot* dengan pengetahuan yang terpisah di mana *interpreter* AIML hanya bertugas melayani percakapan biasa dan melempar kalimat yang mengandung tema tertentu ke pemroses pengetahuan yang sesuai. Untuk fleksibilitas *chatbot* dibuat dalam bentuk *service* sehingga memungkinkan banyak program *client* dengan *platform* beragam dapat berinteraksi.

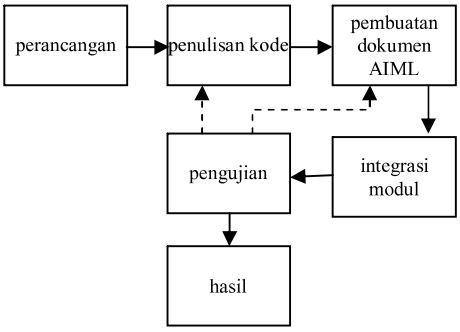
Tujuan penelitian ini adalah membuat sebuah *service chatbot* dengan modul – modul pengetahuan terpisah. Agar lebih terfokus penelitian ini dibatasi pada hal – hal berikut:

1. *Chatbot* melayani percakapan dalam bahasa Indonesia
2. Kode interpreter dan semua pendukungnya ditulis dalam bahasa PHP
3. Dokumen AIML sebagai pengetahuan linguistik disimpan dalam basis data MySQL untuk mempermudah proses penelusuran
4. *Chatbot* tidak dapat melakukan pembaruan pengetahuan secara otomatis

ISSN: 2302-3805

5. Modul yang diintegrasikan dalam *chatbot* adalah modul yang sudah ada, yaitu faraid (perhitungan waris) dan math (matematika dasar)
6. Menggunakan program *client* berbasis konsol sebagai sarana interaksi dengan chatbot

Secara umum metode penelitian digambarkan dalam alur seperti berikut:



Gambar 1.1 Alur penelitian

Gambar 1.1 di atas menjelaskan bahwa penelitian dimulai dari perancangan alur kerja *chatbot* yang diikuti penulisan kode untuk *interpreter* dan pendukungnya. Kemudian dilanjutkan dengan penyusunan dokumen AIML dalam basis data MySQL. Pada tahap ini *chatbot* sudah dapat digunakan untuk melakukan percakapan biasa. Langkah selanjutnya adalah melakukan integrasi modul pengetahuan yang disusul dengan pengujian apakah *chatbot* sudah dapat memetakan percakapan biasa dan percakapan pada tema pengetahuan tertentu. Hasil pengujian dapat dijadikan bahan untuk perbaikan kode maupun susunan dokumen AIML. Hasil akhir berupa *service chatbot* dengan program *client* berbasis konsol sebagai media interaksinya.

2. Tinjauan Pustaka

Berikut ini adalah eberapa penelitian berkaitan dengan *chatbot* dan NLP pada umumnya yang sudah pernah dilakukan sebelumnya:

1. Aplikasi BotQA, sebuah aplikasi *chatbot offline* dan *stand alone* yang dilengkapi dengan animasi ekspresi wajah. Aplikasi ini menggunakan model *pattern matching* dalam pemrosesannya dengan daftar pola pertanyaan-jawaban tersimpan dalam *file* teks. Arsitektur pengetahuannya sudah bersifat modular di mana pola pertanyaan-jawaban untuk tiap topik pembicaraan tersimpan dalam *file* yang berbeda. Walaupun demikian pemrosesan penelusuran masih terpusat pada interpreter utama. [4]
2. Aplikasi NL-Faraid, sebuah aplikasi penghitung pembagian waris yang menggunakan bahasa alami sebagai input. Input berupa kalimat atau frase yang kemudian akan diekstraksi menjadi kata kunci dan bilangan kunci sebagai nilainya. Output aplikasi berupa hasil perhitungan yang disajikan dalam kalimat berbahasa Indonesia . Aplikasi ini tidak dapat digunakan untuk percakapan biasa. [3]
3. Program-O (www.program-o.com), sebuah *interpreter chatbot* berbasis AIML yang ditulis menggunakan bahasa PHP. *Interpreter* ini dilengkapi dengan panel kontrol untuk botmaster dalam mengelola pengetahuan chatbot yang meliputi penambahan pengetahuan dari file AIML eksternal, pengeditan pengetahuan, dan lain – lain. Aplikasi ini belum mendukung interaksi dengan modul pemroses pengetahuan tambahan.

Berikut adalah beberapa teori yang dapat dijadikan landasan atau sumber referensi penelitian ini:

1. *Chatbot*

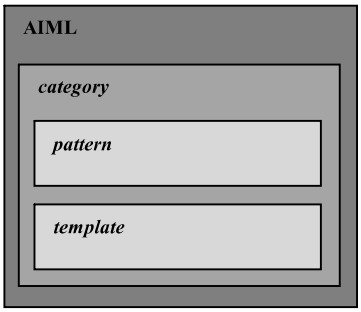
   Secara harfiah *chatbot* berasal dari dua kata yaitu *chat* dan *bot*. Dalam dunia komputer *chat* dapat diartikan sebagai kegiatan komunikasi yang menggunakan sarana tulisan. Sedangkan *bot* merupakan program yang memiliki sejumlah data yang bila diberi input akan menghasilkan output sebagai jawaban. Dari dua istilah di atas dapat diartikan bahwa *chatbot* adalah program komputer yang dapat melakukan percakapan melalui media tulisan. Percakapan dapat terjadi dengan manusia atau *chatbot* yang lain [4]

2. AIML [3]

   Artificial Intelligence Markup Language (AIML) adalah sebuah bahasa yang mendeskripsikan objek data dan perilaku program komputer yang memprosesnya. AIML sendiri merupakan turunan dari *Extensible Markup Language* (XML).

   Objek AIML tersusun atas unit – unit yang disebut *topics* dan *categories*, berisi data yang sudah ter-parsing maupun belum ter-parsing. Data yang ter-parsing berisi karakter – karakter, beberapa di antaranya berupa data karakter, yang lainnya dapat berupa elemen AIML. Elemen AIML mengkaspulasi pengetahuan dalam bentuk *stimulus-response* di dokumen.

ISSN: 2302-3805



Gambar 2.1 Struktur dokumen AIML

Penjelasan gambar 2.1 di atas adalah sebagai berikut:

a. aiml

Tag yang menandai awal dan akhir dokumen AIML.

b. category

Tag yang menandai unit pengetahuan dalam basis pengetahuan *chatbot*.

c. pattern

Digunakan untuk memuat sebuah pola sederhana yang mungkin cocok dengan input user kepada *chatbot*. Di dalam tag pattern dapat berisi *tag – tag* lain yang digunakan untuk *handling* saat *pattern matching*.

d. template

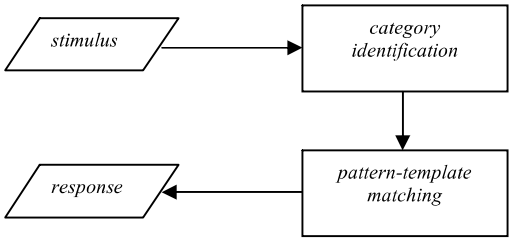
Berisi respon terhadap input user.

Selain *tag – tag* di atas masih terdapat *tag* lain yang semuanya terdefinisikan dalam *AIML Working Draft* [1].

3. *Interpreter* AIML

Interpreter AIML adalah perangkat lunak yang dapat membaca set AIML, mencocokkan input *user* dengan *category*, memproses isi *template* dalam *category* dan mengembalikannya kepada user [2].

Sebuah *interpreter* AIML dapat menggunakan *service XML processor* tetapi tidak boleh melanggar batasan – batasan yang sudah ditetapkan didalamnya.

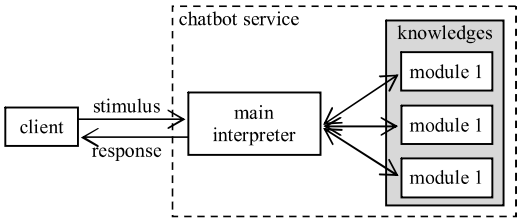


Gambar 2.2 Alur kerja interpreter AIML

Gambar 2.2 di atas menjelaskan bahwa *interpreter* AIML akan menerima input dari user (stimulus) kemudian mengidentifikasi kategorinya. Kemudian dilanjutkan lagi dengan pencarian template yang sesuai.

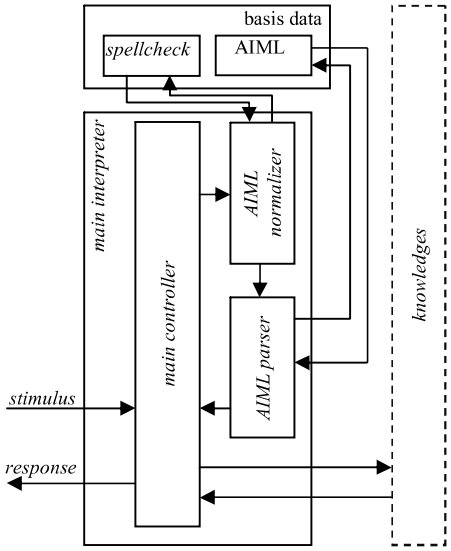
3. Metode Penelitian

Sesuai dengan alur kerja yang sudah digambarkan pada Gambar 1.1 di atas maka yang dilakukan pertama adalah membuat rancangan untuk *chatbot*. Secara umum rancangan *chatbot* digambarkan dalam Gambar 3.1 berikut:



Gambar 3.1 Gambaran umum *chatbot*

Fokus perancangan adalah pada *main interpreter* yang didalamnya terdapat beberapa sub proses. Gambar 3.2 berikut ini adalah detail dari *main interpreter*:



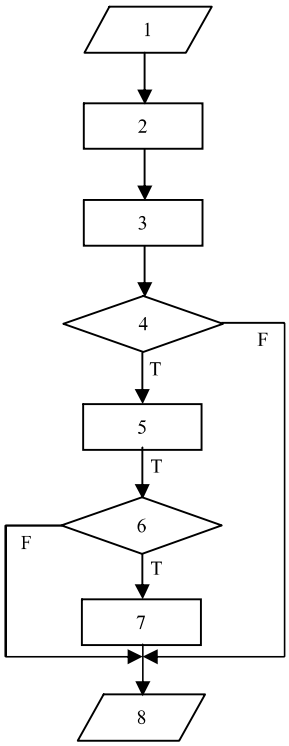
Gambar 3.2 Detail rancangan main interpreter

Gambar 3.2 menjelaskan tentang detail rancangan *main interpreter* yang terdiri dari:

1. *Main controller*

Sebagai *access point* yang bertugas menerima *stimulus* dari *client*, menyiapkan *response* dan mengirimkannya kembali kepada *client*. Bila *response* berupa perintah untuk menghubungi modul tertentu maka *main controller* harus meneruskan *stimulus* ke modul yang dimaksud dan menerima hasilnya untuk kemudian dikirimkan kepada *client*. Gambar 3.3 berikut ini menjelaskan alur kerja *main controller*:

ISSN: 2302-3805



Gambar 3.3 *Flowchart main controller*

Flowchart pada Gambar 3.3 di atas dijelaskan dalam *pseudocode* berikut:

```
1. stimulus
2. NORMALIZE stimulus
3. response ← GET template
4. IF response = route:[module]
5. response ← response FROM [module]
6. IF response = NULL
7. response ← GET response ON FAIL
8. RESULT ← response
```

Baris nomor 4 dan 5 menjelaskan jika terdapat perintah *route* ke modul [module] maka *main controller* harus menghubungi modul [module] untuk melakukan pemrosesan lebih lanjut dan menghirimkan kembali hasilnya.

2. AIML *normalizer*

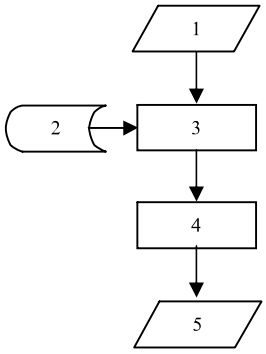
AIML *normalizer* bertugas menormalkan *stimulus* sebelum diproses lebih lanjut oleh AIML *parser*. Dalam AIML *normalizer* terdapat dua sub proses yaitu:

a. *Mark removal*, yaitu penghilangan tanda baca. Prosesnya menggunakan operasi *regular expression* sebagai berikut:

```
REPLACE {?|.|,} IN stimulus
WITH ""
REPLACE {\s\s+} IN stimulus
WITH {\s}
```

Operasi di atas akan menghilangkan tanda baca tanya (?), koma (,), dan titik (.) serta menggantikan ekstra spasi dengan spasi tunggal.

b. *Spell correction*, yaitu pembetulan ejaan. Prosesnya dijelaskan dalam gambar 3.4 berikut:



Gambar 3.4 *Flowchart spell correction*

Flowchart pada gambar 3.4 di atas dijelaskan dalam *pseudocode* berikut:

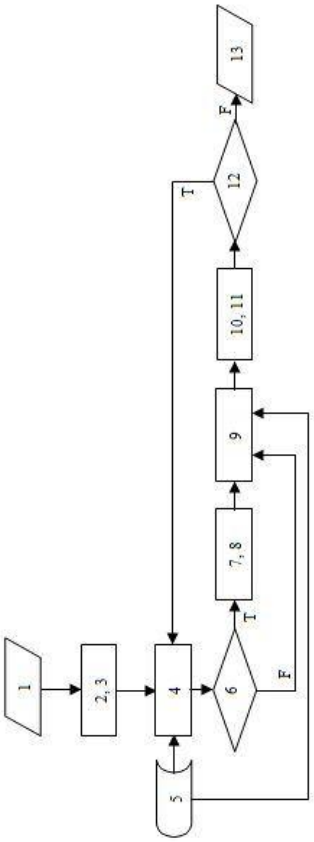
```
1. stimulus
2. spellcheck
3. words ← SELECT misspelling,
   correction FROM spellcheck
4. REPLACE ALL words[misspelling]
   IN stimulus WITH ALL
   words[correction]
5. RESULT ← stimulus
```

Bagian ini berhubungan dengan tabel *spellcheck* di basis data yang berisi daftar kata salah dan pembetulannya. Baris nomor 3 menjelaskan proses pengambilan daftar pasangan kata salah dan pembetulannya. Baris 4 menjelaskan tentang penggantian kata salah yang ada di *stimulus* sesuai daftar hasil proses baris nomor 3.

3. AIML *parser*

Stimulus yang sudah dinormalkan di AIML *normalizer* kemudian dilemparkan ke AIML *parser* untuk diproses lebih lanjut. AIML *parser* bertugas mencari *template* yang cocok dengan *pattern* yang dimasukkan. Dalam operasinya AIML *parser* berhubungan dengan tabel aiml di basis data. Tabel aiml ini berisi pasangan *pattern-template* yang merupakan representasi dari dokumen AIML. *Flowchart* proses *parsing* dijelaskan dalam gambar 3.5 berikut:

ISSN: 2302-3805



Gambar 3.5 *Flowchart* AIML *parser*

Flowchart pada gambar 3.5 di atas dijelaskan dalam *pseudocode* berikut:

```
1. stimulus
2. pattern ← stimulus
3. star ← ""
4. patt ← pattern_star[]
5. aiml
6. IF patt[star] <> "" AND patt[pattern]
   <> ""
7. star ← patt[star]
8. pattern ← patt[pattern]
9. rec ← SELECT template FROM aiml WHERE
   pattern LIKE '%pattern%'
10. template ← (REPLACE "*" IN
   rec:template WITH patt[star])
11. srai ← LOAD srai IN template
12. IF srai <> ""
13. RESULT ← template
```

Inti dari proses parsing terletak pada operasi nomor 9, yaitu pencarian *template* di basis data. Proses *parsing* akan dilakukan secara rekursif ketika *template* yang ditemukan mengandung *tag srai*. Pemeriksaan *tag srai* terjadi pada operasi nomor 12.

4. Tabel *spellcheck*

   Digunakan untuk menyimpan pasangan kata yang kemungkinan salah ketik beserta pembetulannya. Tabel 3.1 berikut ini menjelaskan struktur tabel *spellcheck*:

Tabel 3.1 Struktur tabel *spellcheck*

Kolom	Tipe	Keterangan
id	Int	Kode
misspelling	varchar	Kata yang salah
correction	varchar	Koreksi kesalahan

5. Tabel aiml

   Tabel ini berisi beberapa kolom sebagai representasi dokumen AIML yang dituliskan dalam file teks. Tabel 3.2 berikut ini menjelaskan struktur tabel aiml:

Tabel 3.2 Struktur tabel aiml

Kolom	Tipe	Keterangan
id	int	Kode
pattern	varchar	*Pattern stimulus*
template	varchar	*Template response*
topic	varchar	Topik

4. Hasil dan Pembahasan

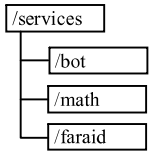
Proses berikutnya adalah penulisan kode dalam bahasa PHP yang dilanjutkan dengan pembuatan dokumen AIML yang kemudian dimasukkan dalam tabel aiml di basis data. Selain *pattern-template* untuk percakapan biasa, ada beberapa pasangan *pattern-template* yang dimasukkan sebagai perintah untuk routing ke modul pengetahuan eksternal. Beberapa pasangan tersebut dijelaskan dalam tabel 3.3 berikut ini:

Tabel 4.1 Pattern-template untuk routing

Pattern	template	Keterangan
hitung *	route:math	routing ke modul math
berapa *	route:math	routing ke modul math
hitung waris *	route:faraid	routing ke modul faraid
berapa pembagian waris bila *	route: faraid	routing ke modul faraid

Berdasar tabel 3.3 di atas maka dalam penelitian ini melibatkan 2 modul pengetahuan tambahan sebagai contoh yaitu modul *math* untuk perhitungan matematika dan modul *faraid* dari aplikasi NL-Faraid untuk penghitung pembagian harta waris.

Proses selanjutnya adalah pengintegrasian modul sekaligus instalasi. Instalasi dilakukan di *web server* dengan struktur direktori seperti pada gambar 3.6 berikut:



Gambar 4.2 Struktur direktori *chatbot service*

Proses pengujian *service chatbot* memerlukan program *client* agar dapat berinteraksi. Gambar 4.3 berikut ini menunjukkan tampilan program *client*:

ISSN: 2302-3805

Gambar 4.3 Program *client*

Tabel 4.1 berikut ini menunjukkan percakapan sekaligus hasil pengujian terhadap fungsionalitas *chatbot*:

Tabel 4.1 Pengujian

No.	Test case	Respon
1	assalamu'alaikum	wa'alaikum salam
2	apa kbr?	alhamdulillah, kabar baik
3	boleh saya bertanya sesuatu?	silakan, dengan senang hati
4	yg bener?	ah kamu..
5	hitung 2+3*5-8	7
6	berapa 4!8x6	hitung saja sendiri..
7	Hitung waris bila orang menginggal dengan hartia 100000000 2 anak lelaki 2 1 anak perempuan dan 1 istri,?	hasil perhitungan: - anak lelaki (2): 35000000 - anak perempuan (1): 17500000 - istri (1): 12500000
8	hitung	apanya yang dihitung?
9	kita akhiri obrolan kita	masa?
10	bye	terima kasih

Test case nomor 4 dan 9 menunjukkan *chatbot* tidak menemukan pola yang tepat sehingga respon berupa kalimat *random*. *Test case* nomor 5 dan 7 menunjukkan bahwa *interpreter* berhasil memanggil modul *math* dan *faraid*. *Test case* nomor 6 menunjukkan bahwa *interpreter* berhasil memanggil modul *math* tetapi menghasilkan respon *nihil* karena operasi matematika tidak dikenali.

5. Kesimpulan dan Saran

Dari pembahasan yang telah diuraikan sebelumnya dapat ditarik beberapa kesimpulan sebagai berikut:

1. Pengetahuan modular *chatbot* disusun sebagai *service* terpisah.
2. Pemanggilan modul pengetahuan dilakukan dengan proses *routing* ke service modul.
3. Proses routing didefinisikan dalam *pattern-template* dokumen AIML dalam *chatbot*.
4. Kemampuan penelusuran jawaban termasuk *routing* sangat ditentukan oleh banyaknya ragam *pattern-template* di dokumen AIML-nya.

Berikut ini beberapa saran yang dapat diberikan untuk pengembangan selanjutnya:

1. Ditambahkan kemampuan untuk belajar mandiri dari hasil percakapan
2. Ditambahkan panel kontrol bagi *botmaster* dalam mengelola isi AIML dan pengelolaan modul tambahan.

Daftar Pustaka

[1] Bush, Noel., 2001, *Artificial Intelligence Markup Language (AIML) Version 1.0. 1,* A.L.I.C.E AI Foundation Working Draft 25 June 2001 (rev 006)

[2] Sullivan, Kim. 2009. *An AIML Interpreter*. Tesis, Department of Computer Science and Enginreering, Faculty of Technical Engineering, Czech Technical University in Prague, Prague

[3] Setiaji, Bayu, S.Kom. 2012. *Pemanfaatan Bahasa Alami Sebagai Antar Muka Aplikasi NL-Faraid*, Jurnal Teknologi Informasi Respati Volume VII Bulan November 2012, Universitas Respati Yogyakarta, Yogyakarta

[4] Utami, Ema, Dr., S.Si., M.Kom. 2007. *Aplikasi BotQA Untuk Meningkatkan Cara Interaksi Manusia dan Mesin*, Seminar Nasional Teknologi Informasi 2007 (SNATI 2007), 16 Juni 2007, Yogyakarta

Biodata Penulis

Bayu Setiaji, M.Kom, memperoleh gelar Sarjana Komputer (S.Kom), Program Studi Teknik Informatika STMIK AMIKOM Yogyakarta, lulus tahun 2006. Tahun 2012 memperoleh gelar Magister Komputer (M.Kom) dari Program Studi Magister Teknik Informatika STMIK AMIKOM Yogyakarta. Saat ini sebagai dosen STMIK AMIKOM Yogyakarta.

Dr. Ema Utami, S.Si, M.Kom, memperoleh gelar Sarjana Sains (S.Si) dari Program Studi Ilmu Komputer UGM pada tahun 1997. Tahun 2002 memperoleh gelar Magister Komputer (M.Kom) dengan predikat cumlaude dari Program Pascasarjana Ilmu Komputer UGM. Tahun 2010 memperoleh gelar Doktor dari Program Doktor Ilmu Komputer UGM. Sejak 1998 menjadi Staff Pengajar di STMIK AMIKOM Yogyakarta dan sejak 2010 menjadi Wakil Direktur I Bidang Akademik Program Pascasarjana STMIK AMIKOM Yogyakarta.

Hanif Al Fatta, M.Kom, memperoleh gelar Sarjana Komputer (S.Kom) Program Studi Ilmu Komputer UGM tahun 2002 dan memperoleh gelar Magister Komputer (M.Kom) Program Pasca Sarjana Ilmu Komputer UGM tahun 2007. Saat ini aktif sebagai dosen tetap dan Ketua Jurusan Program Diploma III Teknik Informatika STMIK AMIKOM Yogyakarta.