BAB II

TINJAUAN PUSTAKA

2.1 Bahasa Alami dan Bahasa Formal

Bahasa menurut kamus *Websters* adalah *"the body of words and methods of combining words used and understood by a considerable community"*, sedangkan menurut Kamus Besar Bahasa Indonesia (KBBI), bahasa adalah:

1) Sistem lambang bunyi yang arbitrer, yang digunakan oleh anggota suatu masyarakat untuk bekerja sama, berinteraksi, dan mengidentifikasikan diri.
2) Percakapan (perkataan) yang baik, tingkah laku yang baik, sopan santun, perkataan-perkataan yang dipakai oleh suatu bangsa.

Definisi di atas mengungkapkan bahwa suatu bahasa adalah kalimat atau perkataan. Kalimat dalam sebuah bahasa dibentuk dengan menggabungkan satu atau lebih kata-kata. Perhatian kita dalam pembentukan kalimat hanya tertuju pada sintaks atau bentuk kalimat, dan bukan pada *semantic* atau makna kalimatnya. Bahasa komunikasi yang digunakan oleh manusia, yaitu bahasa ucap seperti bahasa Indonesia, Inggris, Jerman, Spanyol dan sebagainya disebut sebagai bahasa alami atau bahasa natural (*natural language*). Sintaks bahasa alami sangat rumit dan tidak mungkin untuk menspesifikasiksan semua

aturan sintaksnya. Bahasa yang kaidah sintaksnya dapat dispesifikasikan secara matematis dengan baik disebut bahasa formal. Bahasa formal dapat didefinisikan secara abstraks sebagai "sistem matematik". Kaidah sintaks dalam teori bahasa formal tidak hanya bermanfaat untuk studi *linguistic* bahasa alami seperti penterjemahan secara otomatis dari suatu bahasa ke bahasa lain, tetapi juga berguna untuk studi bahasa pemograman. (Merliana, 2005).

2.2 Tata Bahasa

Tata bahasa disebut juga *Grammar,* yang didefinisikan secara formal sebagai kumpulan dari himpunan-himpunan *variabel*, simbol-simbol terminal, simbol awal yang dibatasi oleh aturan tata bahasa. Suatu tata bahasa dapat menghasilkan sejumlah string dengan menerapkan aturan tata bahasa. Contoh tata bahasa:

$\alpha \rightarrow \beta$ dapat dibaca: α menghasilkan β

Tata bahasa berfungsi untuk menentukan kebenaran dalam penulisan suatu *statement*, sesuai dengan aturan yang terdapat pada suatu program.

Tata bahasa merupakan salah satu bagian penting dalam pembuatan implementasi ini. Masukan yang tidak sesuai dengan tata bahasa yang telah ditetapkan menyebabkan proses tidak dapat dilakukan. Tata bahasa bebas konteks (*Context Free Grammar*) digunakan dalam implementasi ini. Tata bahasa bebas konteks yang digunakan menghasilkan aturan produksi. Aturan produksi ini merupakan pusat dari tata bahasa yang menspesifikasikan

bagaimana suatu tata bahasa melakukan transformasi suatu string ke bentuk lain (Merliana, 2005).

2.3 Hirarki Tata Bahasa Menurut Chomsky

Hierarki tata bahasa menurut *Noam Chomsky* (dalam Hamzah, 2009), adalah 4 penggolongan tingkat bahasa yang ditampilkan dalam suatu hierarki, penggolongan tersebut adalah sebagai berikut:

a. Tipe 0 : *Phrase-Structure Grammar* / PSG atau *Unrestricted grammar (natural language).*

   Batasan pada tipe ini ialah ruas kiri (α) minimal mempunyai satu simbol non terminal, sedangkan ruas kanan (β) tidak mempunyai batasan dalam aturan produksinya, sebagai contoh:

   Abc → aa

   Be → aBaB

b. Tipe 1 : *Context Sensitive grammar* / CSG

   Batasan pada tipe ini yaitu, panjang string pada ruas kiri (α) lebih kecil atau sama dengan ruas kanan (β). Seperti terlihat pada contoh berikut:

   AC → Ed

   AB → aDF

c. Tipe 2 : *Context-Free Grammar* / CFG

Batasan pada tata bahasa bebas konteks yaitu ruas kiri (α) tepat mempunyai satu simbol non terminal, sedangkan ruas kanan (β) tidak dibatasi, sebagai contoh:

A → Bcde

B → CFcgh

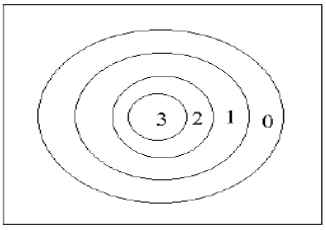
d. Tipe 3 *: Regular Grammar* / RG

Batasan untuk tipe *regular* adalah ruas kiri (α) tepat mempunyai satu simbol non terminal dan ruas kanan maksimal mempunyai satu simbol non terminal yang terletak paling kanan, jadi ruas kanan bisa memiliki simbol terminal yang tidak terbatas, sebagai contoh:

A → aaa

B → aaB

Diagram Hierarki Tata Bahasa (*Grammar*) menurut Noam Chomsky (dalam Hamzah, 2009) ditunjukkan pada Gambar 1.



Gambar 1. Hierarki Tata Bahasa (*Grammar*) Menurut Noam Chomsky (Hamzah, 2009).

2.4 Tata Bahasa Bebas Konteks (*Context Free Grammar*)

Tata bahasa bebas kontext (*Context Free Grammar*), biasa disingkat dengan CFG memiliki batasan sebagai berikut:

Tata-bahasa Tipe 2 (*Context Free Grammar*)**:** G (Σ, N, S, P), adalah tata bahasa tipe-1 yang memiliki aturan produksi: $\alpha \rightarrow \beta$ dengan tambahan batasan:

α: hanya terdiri dari 1 simbol non terminal saja, atau $\alpha \in N$

β: tidak dibatasi, atau : $\beta \in \{(\Sigma \cup N)^*$

Perbedaan antara bahasa bebas konteks dengan bahasa *regular* adalah pada string sisi kanan tanda panah untuk bahasa *regular* harus satu terminal tunggal atau terminal tunggal diikuti non terminal sedangkan untuk bahasa bebas konteks tidak dibatasi. Contoh:

- *Grammar* dengan aturan :

  S→a|aA

  A→bB

  B→ ε

  Adalah tata bahasa *regular*

- *Grammar* dengan aturan:

  S→a|aA

  A→bB | AA

  B→ ε

  Adalah tata bahasa bebas konteks karena ada aturan produksi: A→AA

(Hamzah, 2009).

2.5 Penyederhanaan

Tujuan dari penyederhanaan adalah melakukan pembatasan sehingga tidak menghasilkan pohon penurunan yang memiliki kerumitan yang tidak perlu atau aturan produksi yang tidak berarti.

Contoh 1:

S → AB | a

A→ a

- Aturan produksi S → AB tidak berarti karena B tidak memiliki penurunan

Contoh 2:

S→A

A→B

B→C

C→D

D→ a | A

- Memiliki kelemahan terlalu panjang jalannya padahal berujung pada S→a,
- produksi D → A juga menyebabkan kerumitan.

Cara Penyederhanaan:

1. Penghilangan produksi *useless* (tidak berguna).

2. Penghilangan produksi *unit*.

3. Penghilangan produksi *epsilon* (ε).

(Rustamaji, 2004).

2.5.1 Penghilangan Produksi *Useless*

Produksi *useless* didefinisikan sebagai:

- Produksi yang memuat simbol *variabel* yang tidak memiliki penurunan yang menghasilkan terminal-terminal seluruhnya.
- Produksi yang tidak pernah dicapai dengan penurunan apapun dari simbol awal, sehingga produksi itu redundan (berlebih).

Contoh:

S → aSa | Abd | Bde

A → Ada

B→ BBB | a

Maka:

1) Simbol *variabel* A tidak memiliki penurunan yang menuju terminal, sehingga bisa dihilangkan.
2) Konsekuensi no (1), aturan produksi S → Abd tidak memiliki penurunan.

Penyederhanaan menjadi:

S → aSa | Bde

B → BBB | a

(Rustamaji, 2004).

2.5.2 Penghilangan Produksi *Unit*

- Produksi dimana ruas kiri dan kanan aturan produksi hanya berupa satu simbol *variabel*, misalkan: A → B, C → D.
- Keberadaannya membuat tata bahasa memiliki kerumitan yang tak perlu.
- Penyederhanaan dilakukan dengan melakukan penggantian aturan produksi *unit*.

Contoh:

S → Sb

S → C

C → D

C → ef

D → dd

Penggantian dilakukan berturutan mulai dari aturan produksi yang paling dekat menuju ke penurunan terminal-terminal (‘=>’ dibaca ‘menjadi’):

C → D => C → dd

S → C => S → dd | ef

Aturan produksi setelah penyederhanaan:

S → Sb

S → dd | ef

(Rustamaji, 2004).

2.5.3 Penghilangan Produksi *Epsilon* (ε)

Produksi ε adalah produksi dalam bentuk

$$\alpha \rightarrow \varepsilon$$

atau bisa dianggap sebagai produksi kosong. Penghilangan produksi ε dilakukan dengan melakukan penggantian produksi yang memuat *variabel* yang bisa menuju produksi ε, atau biasa disebut *nullable.*

Prinsip penggantiannya bisa dilihat kasus berikut:

$$S \rightarrow bcAd$$

$$A \rightarrow \varepsilon$$

A *nullable* serta A→ ε satu-satunya produksi dari A, maka *variabel* A bisa ditiadakan, hasil penyederhanaan tata bahasa bebas konteks menjadi:

$$S \rightarrow bcd$$

Tetapi bila kasusnya:

$$S \rightarrow bcAd$$

$$A \rightarrow bd \mid \varepsilon$$

A *nullable,* tapi A→ ε bukan satu-satunya produksi dari A.

Hasil penyederhanaan:

$$S \rightarrow bcAd \mid bcd$$

$$A \rightarrow bd$$

(Rustamaji, 2004).

2.6 Bentuk Normal *Chomsky*

Bentuk normal *Chomsky / Chomsky Normal Form* (CNF) merupakan salah satu bentuk normal yang sangat berguna untuk tata bahasa bebas konteks (CFG). Bentuk normal *Chomsky* dapat dibuat dari sebuah tata bahasa bebas konteks yang telah mengalami penyederhanaan yaitu penghilangan produksi *useless*, *unit*, dan ε. Suatu tata bahasa bebas konteks dapat dibuat menjadi bentuk normal *Chomsky* dengan syarat tata bahasa bebas konteks tersebut:

- Tidak memiliki produksi *useless*
- Tidak memiliki produksi *unit*
- Tidak memiliki produksi *epsilon* (ε)

Aturan produksi dalam bentuk normal *Chomsky* ruas kanannya tepat berupa sebuah terminal atau dua *variabel*. Misalkan:

$$A \rightarrow BC$$

$$A \rightarrow b$$

$$B \rightarrow a$$

$$C \rightarrow BA \mid d$$

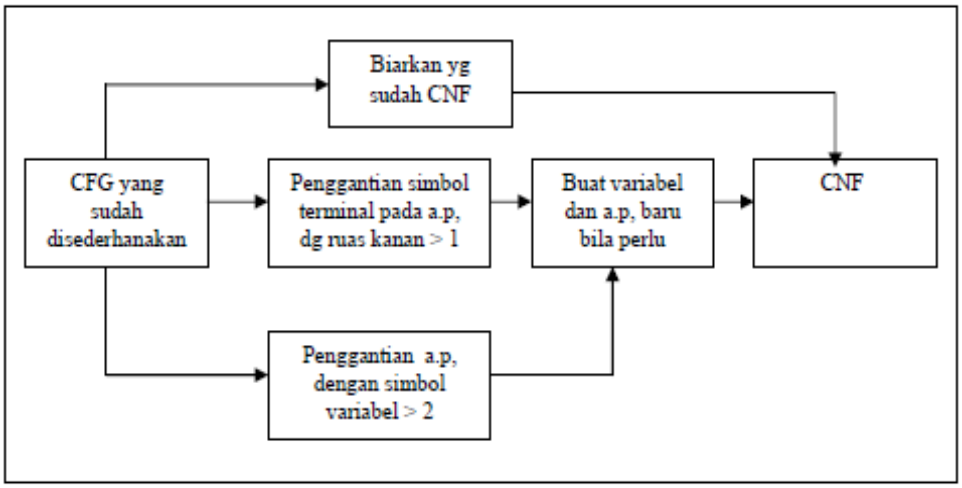
2.6.1 Proses Pembentukan Normal *Chomsky*

Langkah-langkah pembentukan bentuk normal *Chomsky* secara umum sebagai berikut:

- Biarkan aturan produksi yang sudah dalam bentuk normal *Chomsky*.
- Lakukan penggantian aturan produksi yang ruas kanannya memuat simbol terminal dan panjang ruas kanan > 1.

- Lakukan penggantian aturan produksi yang ruas kanannya memuat > 2 simbol *variabel*.
- Penggantian-penggantian tersebut bisa dilakukan berkali-kali sampai akhirnya semua aturan produksi dalam bentuk normal *Chomsky.*
- Selama dilakukan penggantian, kemungkinan akan muncul aturan-aturan produksi baru, dan juga memunculkan simbol-simbol *variabel* baru. (Rustamaji, 2004).

Tahapan-tahapan dalam pembentukan bentuk normal *Chomsky* terdapat pada Gambar 2.



Gambar 2. Tahapan-Tahapan Pembentukan Bentuk Normal *Chomsky* (Rustamaji, 2004).

Contoh, tata bahasa bebas konteks (kita anggap tata bahasa bebas konteks ini sudah mengalami penyederhanaan):

$$S \rightarrow bA \mid aB$$

$$A \rightarrow bAA \mid aS \mid a$$

$$B \rightarrow aBB \mid bS \mid b$$

Aturan produksi yang sudah dalam bentuk normal *Chomsky:*

$$A \rightarrow a$$

$$B \rightarrow b$$

Penggantian aturan produksi yang belum bentuk normal *Chomsky*

(‘=>’ bisa dibaca berubah menjadi):

$S \rightarrow bA \Rightarrow S \rightarrow P1A$

$S \rightarrow aB \Rightarrow S \rightarrow P1B$

$A \rightarrow bAA \Rightarrow A \rightarrow P1AA \Rightarrow A \rightarrow P1P3$

$A \rightarrow aS \Rightarrow A \rightarrow P2S$

$B \rightarrow aBB \Rightarrow B \rightarrow P2BB \Rightarrow B \rightarrow P2P4$

$B \rightarrow bS \Rightarrow B \rightarrow P1S$

Aturan produksi dan simbol *variabel* baru yang terbentuk:

$P1 \rightarrow b$

$P2 \rightarrow a$

$P3 \rightarrow AA$

$P4 \rightarrow BB$

Hasil akhir aturan produksi dalam bentuk normal *Chomsky:*

$$A \rightarrow a$$

$$B \rightarrow b$$

$$S \rightarrow P_1A$$

$$S \rightarrow P_2B$$

$$A \rightarrow P_1P_3$$

$$A \rightarrow P_2S$$

$$B \rightarrow P_2P_4$$

$$B \rightarrow P_1S$$

$$P_1 \rightarrow b$$

$$P_2 \rightarrow a$$

$$P_3 \rightarrow AA$$

$$P_4 \rightarrow BB$$

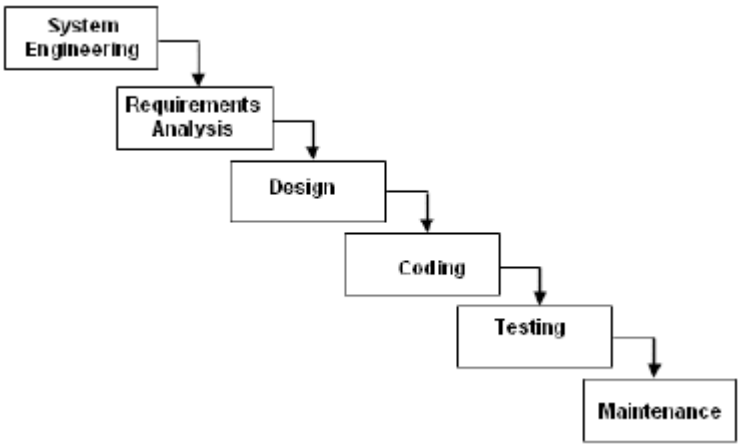
(Rustamaji, 2004).

2.7 Metode Pengembangan Sistem

Metodologi pengembangan sistem adalah metode-metode, prosedur-prosedur, konsep-konsep pekerjaan, aturan-aturan dan postulat-postulat yang akan digunakan untuk mengembangkan suatu sistem informasi. Pengembangan sistem didefinisikan sebagai aktivitas untuk menghasilkan sistem informasi berbasis komputer untuk menyelesaikan persoalan *(problem)* organisasi atau memanfaatkan kesempatan yang timbul.

Model air terjun *(waterfall)* biasa juga disebut siklus hidup perangkat lunak. Metode pengembangan sistem *waterfall* mengambil kegiatan dasar seperti spesifikasi, pengembangan, validasi, evolusi dan merepresentasikannya sebagai fase-fase proses yang berbeda seperti spesifikasi persyaratan, perancangan perangkat lunak, implementasi, pengujian dan seterusnya.

Tahapan-tahapan dalam metode pengembangan sistem *waterfall* terdapat pada Gambar 3 (Jogiyanto, 2010).



Gambar 3. Tahapan Metode *Waterfall* (Jogiyanto, 2010).

Keterangan Menurut gambar di atas alur dari Model *Waterfall* sebagai berikut:

1. Rekayasa perangkat lunak (*system engineering*), melakukan pengumpulan data dan penetapan kebutuhan semua elemen sistem.
2. *Requirements analysis*, melakukan analisis terhadap permasalahan yang dihadapi dan menetapkan kebutuhan perangkat lunak, fungsi performsi.
3. *Design*, menetapkan domain informasi untuk perangkat lunak, fungsi dan *interfacing*.
4. Implementasi, pengkodean yang mengimplementasikan hasil desain ke dalam kode atau bahasa yang dimengerti oleh mesin komputer dengan menggunakan bahasa pemprograman tertentu.
5. Pengujian, kegiatan untuk melakukan pengetesan program yang sudah dibuat apakah sudah benar atau belum di uji dengan cara manual, jika *testing* sudah benar maka program boleh digunakan.

6. Perawatan, menangani perangkat lunak yang sudah selesai supaya dapat berjalan lancar dan terhindar dari gangguan-gangguan yang dapat menyebabkan kerusakan.

2.8 PHP (*Hypertext Preprocessor*)

PHP (*Hypertext Preprocessor*) merupakan bahasa pemrograman yang berbentuk *scripting*, sistem kerja dari program ini adalah sebagai *interpreter* bukan sebagai *compiler* (Nugroho, 2004). *Interpreter* menurut Kadir adalah perintah menerjemahkan *statement* program, sedangkan bahasa *compiler* adalah semua perintah di dalam program yang diterjemahkan terlebih dahulu, baru kemudian semuanya dijalankan" (Kadir, 2005).

PHP adalah salah satu bahasa pemograman skrip yang dirancang untuk membangun aplikasi *web.* Program yang ditulis dengan PHP akan di-*parsing* di dalam *web server* oleh *interpreter* PHP dan diterjemahkan ke dalam dokumen HTML ketika dipanggil *web browser*, yang selanjutnya akan ditampilkan kembali ke *web browser*. PHP dikatakan sebagai bahasa sisi *server (server side)* karena pemrosesan program PHP dilakukan di lingkungan *web server*. Kode PHP tidak akan terlihat pada saat *user* memilih perintah "*View Source"* pada *web browser* yang mereka gunakan (Raharjo dkk, 2014).

PHP adalah bahasa yang dirancang untuk mudah diletakkan didalam kode HTML. Kode PHP yang menyatu dengan kode HTML banyak sekali dijumpai. Kode PHP diawali dengan *tag* <?php dan diakhiri dengan *tag* ?>. *Tag* tersebut

dapat diganti dengan <? dan >? dengan cara melakukan konfigurasi terhadap *file php.ini* untuk mengijinkan pengguna *tag* pendek *(short tag)* dengan mengubah nilai *short_open_tag* menjadi *on.*

Contoh kode PHP yang sangat sederhana.

```
<?php
     echo "Hello World";
?>
```

Perintah *echo* di dalam PHP berguna untuk mencetak nilai, baik teks maupun numerik, ke layar *web browser,* selain *echo,* dapat juga menggunakan perintah *print,* sehingga kode diatas dapat juga ditulis seperti berikut.

```
<?php
     print "Hello World";
?>
```

Setiap perintah atau *statement* di dalam kode PHP harus diakhiri dengan tanda titik koma atau *semicolon* (;) (Raharjo dkk, 2014).

Kelebihan yang dimiliki oleh pemrograman PHP adalah dapat disisipkan ke dalam *tag-tag* HTML, namun dengan kelebihan yang dimiliki, PHP juga mampu berdiri sendiri tanpa berada di sela-sela program lain. Contoh program PHP yang berada pada *tag* HTML sebagai berikut:

```
<html>
<head>
     <title>Kode PHP dalam kode HTML</title>
</head>
<body>
<p>Paragraf 1: Teks dari kode HTML</p>

<!-- Menyisipkan kode PHP dalam kode HTML -->
<?php
echo "<p>Paragaraf 2: Teks dari kode PHP</p>";
?>
<!-- akhir kode PHP -->

<p>Paragraf 3: Teks dari kode HTML</p>

</body>
</html>
```

(Raharjo dkk, 2014).