# BAB 3 PERANCANGAN

Pada bab ini akan dijelaskan perancangan *web service* yang menyediakan layanan analisis morfologi kata-kata bahasa Indonesia dengan menggunakan program *Morphological Analyzer* yang telah dikembangkan sebelumnya oleh Femphy Pisceldo dan Ruli Manurung (Pisceldo, 2008). Program *Morphological Analyzer* ini masih berupa sebuah *offline desktop application* yang hanya dapat diakses oleh *workstation* di mana program ini telah diinstalasi di dalamnya. Diawali dari melihat keterbatasan ini, muncullah sebuah ide untuk membuat program ini dapat diakses secara luas melalui *web*.

Untuk mewujudkan ide tersebut, dimanfaatkanlah teknologi *web service* untuk membungkus program *Morphological Analyzer* tersebut. Dengan membungkus program ini ke dalam *web service* dan menaruh *web service* tersebut di sebuah *server* yang terhubung dengan Internet, program ini dapat dimanfaatkan oleh semua orang yang memiliki Internet tanpa harus terlebih dahulu melakukan instalasi program ini di dalam *workstation* mereka. Selain itu, dengan adanya *web service* ini, dapat dibuat *web service* lain yang merupakan sebuah *composite service* yang menggunakan *service* yang telah tersedia.

Dalam rangka pembuatan *web service* yang akan membungkus program *Morphological Analyzer* ini, ada beberapa tahap yang harus dilakukan, yaitu tahap-tahap berikut.

1. Mengembangkan *web application* pada *server side* yang menghubungkan program *Morphological Analyzer* yang ada dengan *web service* yang dibuat. Tahap ini akan dijelaskan pada Subbab 3.1.
2. Membuat WSDL yang akan mendeskripsikan layanan yang diberikan oleh *web service*, termasuk informasi mengenai fungsi yang diimplementasikan pada *web application* pada *server side*. Tahap ini akan dijelaskan pada Subbab 3.2.

Setelah kedua tahap di atas dilakukan, perlu dikembangkan pula sebuah *web application* pada *client side* yang menghubungkan permintaan *client* dengan *web service* yang dibuat. *Web application* ini dibuat untuk keperluan uji coba dari penelitian ini. Perancangan dari *web application* pada *client side* ini akan dijelaskan pada Subbab 3.3.

### 1.1 Pengembangan *Web Application* pada *Server Side*

*Web application* pada *server side* ini adalah sebuah aplikasi yang menghubungkan program *Morphological Analyzer* dengan *web service* yang akan dibangun. Fungsi yang dibuat pada aplikasi ini adalah fungsi yang nantinya dapat dipanggil oleh *web application* pada *client side*. Informasi mengenai fungsi yang diimplementasikan pada aplikasi ini akan disediakan oleh *file* WSDL yang akan dibuat kemudian.

Fungsi yang akan diimplementasikan pada aplikasi ini adalah fungsi **`applyUp`**, **`applyDown`**, **`applyUpAll`**, **`applyDownAll`**, dan **`doAnalyze`**. Spesifikasi dari keempat fungsi tersebut akan dijelaskan pada Subbab 3.1.1 hingga Subbab 3.1.5.

#### 1.1.1 Fungsi `applyUp`

Fungsi ini menerima masukan berupa sebuah *string* yang merupakan sebuah kata dalam Bahasa Indonesia yang ingin dianalisis morfologi katanya, dan menghasilkan keluaran berupa sebuah *string* yang merupakan hasil analisis morfologi dari *string* masukan. Hasil analisis morfologi yang dihasilkan berturut-turut adalah kata dasar, kelas kata (Contoh: *noun*, *verb*), dan, jika ada, *tags* lain yang berisi informasi sintaksis dan semantik (Contoh: AV, PASS, UV) yang dipisahkan oleh karakter "+". Jika terdapat lebih dari satu analisis morfologi yang memenuhi, maka hanya analisis morfologi yang pertama yang akan menjadi keluaran dari fungsi ini. Jika *string* masukan bukanlah sebuah kata dasar atau kata berimbuhan yang ada dalam Bahasa Indonesia, fungsi ini akan menghasilkan keluaran berupa *string* kosong.

Proses yang dilakukan di dalam fungsi ini adalah sebagai berikut.

1. Mengimplementasikan sebuah *process builder* untuk menjalankan program *Morphological Analyzer*, serta memberikan masukan dan mengambil keluaran dari program tersebut.
2. Memberikan masukan pada program *Morphological Analyzer* melalui *process builder* yang telah dibuat. Masukan yang diberikan pada program *Morphological Analyzer* adalah perintah "applyUp" diikuti dengan *string* masukan dari fungsi **`applyUp`** ini.
3. Mengambil keluaran dari program *Morphological Analyzer* melalui *process builder* yang telah dibuat. Jika terdapat keluaran yang dihasilkan, *string* keluaran tersebut akan dijadikan keluaran dari fungsi ini. Jika tidak, *string* kosong yang akan menjadi keluaran dari fungsi ini.

Sebagai contoh, *string* yang dikembalikan dari pemanggilan fungsi **`applyUp("makan")`** adalah "**`ma+Verb`**", karena kata "makan" dalam Bahasa Indonesia dapat dibentuk dari kata "ma" dengan akhiran "-kan". Sebenarnya terdapat beberapa hasil analisis morfologi lain seperti "**`makan+BareVerb+UV`**", tetapi pada fungsi **`applyUp`** hanya akan dikeluarkan hasil analisis morfologi yang pertama saja.

### 1.1.2 Fungsi `applyDown`

Pada dasarnya, fungsi ini adalah kebalikan dari fungsi **`applyUp`**. Fungsi ini menerima masukan berupa sebuah *string* yang berisi kata dasar dalam Bahasa Indonesia diikuti dengan kelas kata (Contoh: *noun*, *verb*) dan, jika ada, *tags* lain yang berisi informasi sintaksis dan semantik dari kata tersebut (Contoh: AV, PASS, UV) yang dipisahkan oleh karakter "+", dan menghasilkan keluaran berupa sebuah *string* yang merupakan kata dasar atau kata berimbuhan dalam bahasa Indonesia yang memiliki kata dasar, kelas kata, dan *tags* lain yang berisi informasi sintaksis dan semantik yang sesuai dengan masukan. Jika terdapat lebih dari satu kata yang sesuai,

hanya kata yang pertama saja yang akan menjadi keluaran dari fungsi ini. Jika tidak ada kata yang sesuai, fungsi ini akan mengeluarkan sebuah *string* kosong.

Proses yang dilakukan di dalam fungsi ini adalah sebagai berikut.

1. Mengimplementasikan sebuah *process builder* untuk menjalankan program *Morphological Analyzer*, serta memberikan masukan dan mengambil keluaran dari program tersebut.
2. Memberikan masukan pada program *Morphological Analyzer* melalui *process builder* yang telah dibuat. Masukan yang diberikan pada program *Morphological Analyzer* adalah perintah "applyDown" diikuti dengan *string* masukan dari fungsi **`applyDown`** ini.
3. Mengambil keluaran dari program *Morphological Analyzer* melalui *process builder* yang telah dibuat. Jika tidak terdapat keluaran yang dihasilkan, *string* kosong yang akan menjadi keluaran dari fungsi ini.

Sebagai contoh, *string* yang dikembalikan dari pemanggilan fungsi **`applyDown("pukul+Verb+AV")`** adalah "**`mengepukulkan`**". Kata "mengepukulkan" merupakan kata dalam Bahasa Indonesia yang memiliki kata dasar "pukul", bentuk kata "Verb" atau kata kerja, dan *tags* sintaksis dan semantik lain "AV" atau *active voice*. Sebenarnya terdapat beberapa kata di dalam Bahasa Indonesia yang memiliki morfologi yang sama, sebagai contoh adalah kata "memukul", tetapi yang menjadi keluaran dari fungsi **`applyDown`** ini hanyalah kata yang pertama saja.

### 1.1.3 Fungsi `applyUpAll`

Fungsi ini serupa dengan fungsi **`applyUp`** yang telah dijelaskan pada Subbab 3.1.1. Perbedaannya adalah, jika pada fungsi **`applyUp`** keluaran yang dihasilkan hanyalah hasil analisis morfologi yang pertama saja, pada fungsi **`applyUpAll`** ini keluaran yang dihasilkan adalah semua hasil analisis morfologi yang dipisahkan dengan karakter ' ' (spasi).

Sebagai contoh, *string* yang dikembalikan dari pemanggilan fungsi **applyUpAll("makan")** adalah "**ma+Verb mak+Noun makan+BareNoun makan+BareVerb+UV**". Keempat hasil keluaran tersebut merupakan hasil analisis morfologi dari kata "makan".

### 1.1.4 Fungsi `applyDownAll`

Fungsi ini serupa dengan fungsi **applyDown** yang telah dijelaskan pada Subbab 3.1.2. Perbedaannya adalah, jika pada fungsi **applyDown** keluaran yang dihasilkan hanyalah kata yang pertama saja yang memenuhi morfologi yang diberikan pada masukan, pada fungsi **applyDownAll** ini keluaran yang dihasilkan adalah semua kata yang memenuhi morfologi yang diberikan pada masukan, yang dipisahkan dengan karakter ' ' (spasi).

Sebagai contoh, *string* yang dikembalikan dari pemanggilan fungsi **applyDownAll("pukul+Verb+AV")** adalah "**mengepukulkan menerpukulkan memperpukul memperpukulkan memperpukuli memberpukulkan memukul memukulkan memukuli**". Semua kata hasil keluaran tersebut merupakan kata dalam bahasa Indonesia yang memiliki kata dasar "pukul", bentuk kata "Verb" atau kata kerja, dan *tags* sintaksis dan semantik lain "AV" atau *active voice*.

### 1.1.5 Fungsi `doAnalyze`

Fungsi ini merupakan implementasi dari *abstract method* pada *abstract class* **AbstractMorphologicalAnalysisService** yang akan dijelaskan pada Subbab 3.4.2. Fungsi ini menerima masukan berupa **Language** yang mendeskripsikan kode bahasa dari *text* yang akan dianalisis morfologinya dan sebuah *string text*. Setiap kata yang dipisahkan karakter ' ' (spasi) pada *string text* akan dianalisis morfologinya. Hasil dari analisis morfologi yang dilakukan adalah berupa kumpulan dari **Morpheme**[1], di mana satu **Morpheme** merepresentasikan hasil analisis dari satu kata

[1] **Morpheme** adalah sebuah obyek yang memiliki tiga buah atribut, yaitu **word**, **lemma**, dan **partOfSpeech**. Atribut **lemma** berisi kata dasar dari atribut **word**, dan atribut **partOfSpeech**

pada masukan. Urutan **`Morpheme`** yang merupakan hasil analisis pada keluaran disesuaikan dengan urutan kata pada masukan. Kumpulan **`Morpheme`** tersebut diperoleh dari pemanggilan fungsi "applyUp" yang telah dijelaskan pada Subbab 3.1.1.

Dalam penelitian ini, hanya digunakan empat buah kemungkinan *part of speech*, yaitu *noun*, *verb*, *adjective*, *other*, dan *unknown*. Kata yang memiliki kelas kata yang mengandung kata "Noun", "Verb", dan "Adjective" akan dikategorikan masing-masing ke dalam *part of speech noun*, *verb*, dan *adjective*. Kata yang memiliki kelas kata "Etc" akan dikategorikan ke dalam *part of speech other*. Kata-kata lainnya akan dikategorikan ke dalam *part of speech unknown*.

Sebagai contoh, keluaran yang dihasilkan dari pemanggilan fungsi **`doAnalyze("memukul meja")`** adalah dua buah **`Morpheme`**, yang masing-masing merupakan hasil analisis morfologi kata "memukul" dan "meja".

## 1.2 Pembuatan *File* WSDL

Setelah mengembangkan *web application* pada *server side*, hal berikutnya yang perlu dilakukan adalah pembuatan *file* WSDL. *File* WSDL ini akan mendeskripsikan *web service* yang akan dibuat.

Informasi yang ada di dalam *file* WSDL ini meliputi informasi mengenai operasi atau fungsi yang diimplementasikan pada aplikasi *server side* dan dapat dipanggil oleh *web application* pada *client side* dan informasi mengenai lokasi dari *web service* tersebut. Informasi mengenai fungsi yang diimplementasikan pada aplikasi *server side* meliputi nama fungsi dan tipe data dari setiap argumen pada fungsi tersebut. Setelah *web service* ini selesai dibangun, *web service* ini dapat diakses melalui lokasi yang diberikan di dalam *file* WSDL yang mendeskripsikan *web service* tersebut.

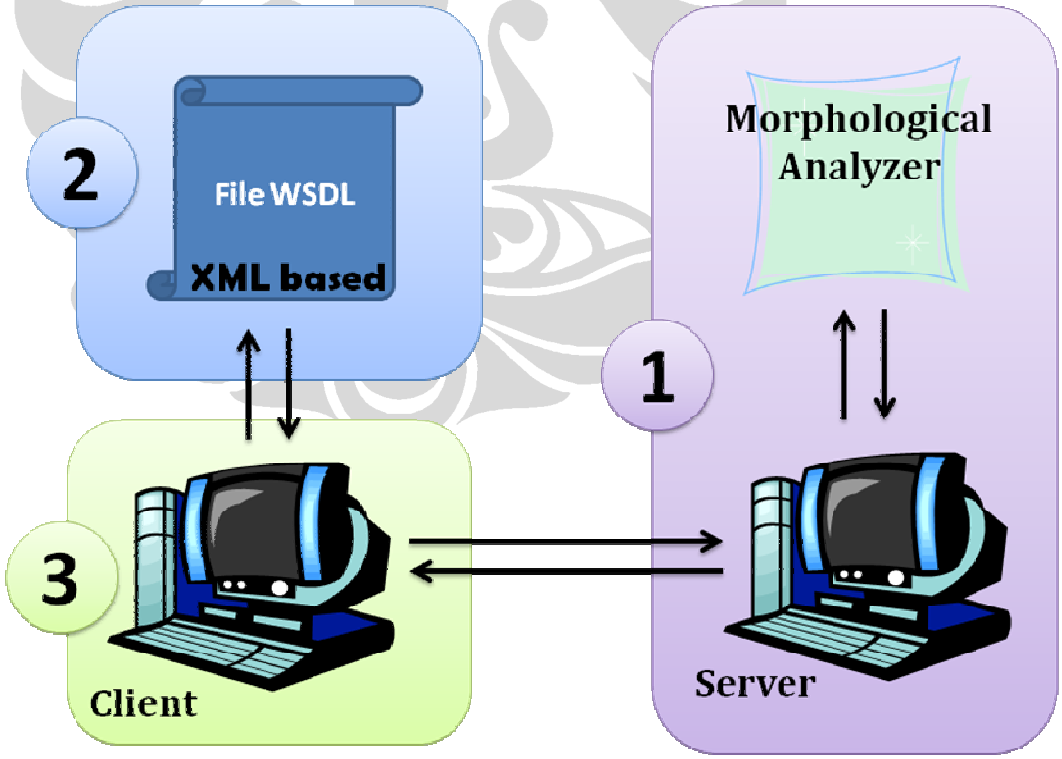
---

berisi obyek **`PartOfSpeech`** yang mendefinisikan kelas kata dari **`word`**. Kelas kata yang tersedia dapat dilihat pada Subbab 3.4.2.

### 1.3 Pengembangan *Web Application* pada *Client Side*

Setelah *web application* pada *server side* dan *file* WSDL yang mendeskripsikan *web service* telah selesai diimplementasikan, tahap terakhir yang perlu dikembangkan adalah *web application* pada *client side*. Aplikasi ini adalah sebuah aplikasi berbasis *web* yang berfungsi untuk mengantarkan *request* dari *client* atau pengguna kepada *web application* pada *server side* dengan menggunakan informasi yang ada pada *file* WSDL yang telah dibuat sebelumnya. Dengan kata lain, aplikasi ini akan berkomunikasi dengan *web service* yang menyediakan layanan yang dibutuhkan oleh pengguna (dalam hal ini adalah layanan *Morphological Analyzer* Bahasa Indonesia).

Protokol yang digunakan untuk berkomunikasi dengan *web service* ini adalah SOAP. Protokol ini membuat *web application* pada *client side* dan *server side* dapat saling berkomunikasi melalui HTTP.




Gambar 0.1 Hubungan *web application* pada *server side*, *file* WSDL, dan *web application* pada *client side*

Pada Gambar 0.1, digambarkan hubungan dari *web application* pada *server side*, *file* WSDL, dan *web application* pada *client side*, yang perancangannya telah dijelaskan pada Subbab 3.1 hingga Subbab 3.3. Ada pun tahapan dari penggunaan *web service* ini adalah sebagai berikut.

1. Pada tahap pertama, *client* mengirimkan *request* untuk menggunakan *web service Morphological Analyzer*. *Client* menggunakan SOAP sebagai protokol untuk berkomunikasi dengan *web service* tersebut. Hal pertama yang akan dilakukan *client* adalah memeriksa fungsi atau operasi yang disediakan oleh *web service Morphological Analyzer* dengan melihat isi dari WSDL yang berisi deskripsi dari *web service* tersebut.
2. Pada tahap kedua, *client* telah memperoleh informasi mengenai fungsi yang disediakan pada *web service Morphological Analyzer*, atau lebih tepatnya fungsi yang diimplementasikan pada *web application* pada *server side*, yaitu fungsi **`applyUp`** dan fungsi **`applyDown`**.
3. Pada tahap ketiga, *client* akan mengirimkan *request* kepada *web application* pada *server side* dengan menggunakan informasi yang telah didapatkan pada WSDL pada tahap sebelumnya. Setelah itu, *request* dari *client* ini akan diproses oleh *web application* pada *server side* dengan memanggil fungsi yang diminta dalam *request client* (fungsi **`applyUp`** atau fungsi **`applyDown`**). Hasil keluaran dari pemanggilan fungsi tersebut akan dikembalikan lagi kepada *client*.
4. Pada tahap keempat, hasil keluaran yang diberikan oleh *web application* pada *server side* akan ditampilkan pada *client* melalui *web application* pada *client side*.

## 1.4 *Wrapping* Program *Morphological Analyzer* pada Language Grid

Pada **Error! Reference source not found.**telah dijelaskan mengenai proyek Language Grid yang bertujuan untuk mempersatukan *language resources* dan *language processing functions* yang saat ini dikembangkan secara individual ke

dalam sebuah *platform* yang memungkinkan *language resources* dan *language processing functions* tersebut diakses secara mudah dan cuma-cuma oleh orang yang ada di seluruh dunia. Program *Morphological Analyzer* adalah salah satu contoh *language processing function* yang dikembangkan secara individual. Program ini ingin dimasukkan ke dalam Language Grid. Untuk mewujudkan hal tersebut, program ini perlu dibungkus menjadi sebuah *web service* yang sesuai dengan standar Language Grid. Proses pembungkusan inilah yang disebut dengan *wrapping*. Informasi mengenai *wrapping* pada Language Grid ini bisa didapatkan pada dokumen "Wrapping Manual for Language Resources and Language Processing Functions" (NICT Language Grid Project, 2008).

### 1.4.1 Definisi *Wrapper Morphological Analyzer*

Berdasarkan (NICT Language Grid Project, 2008), *wrapper Morphological Analyzer* adalah sebuah program yang membuat program *Morphological Analyzer* beserta data yang digunakan oleh program tersebut dapat diakses melalui *web service*, dan yang menyesuaikan spesifikasi masukan dan spesifikasi keluaran dari *language resources* tersebut dengan spesifikasi masukan atau keluaran menggunakan "NICT Language Service Interface". *Wrapper* ini akan ditaruh pada "Language Grid Service Node", dan akan menerima *request* dari "Language Grid Core Node".

Saat "Language Grid Service Node" menerima *request* dari "Language Grid Core Node", "Language Grid Service Node" akan mengakses *language resource* yang ada di dalam *wrapper*, yaitu program *Morphological Analyzer* beserta data yang digunakan oleh program tersebut. Peran dari *language service wrapper* ini dapat dilihat pada **Error! Reference source not found.**.

### 1.4.2 Perancangan *Wrapper Morphological Analyzer*

Untuk merancang *wrapper* dari sebuah *Morphological Analysis Service*, digunakan *abstract class* `AbstractMorphologicalAnalysisService` (NICT

Language Grid Project, 2008). Informasi mengenai *abstract class* tersebut dapat dilihat pada Tabel 0.1.

Tabel 0.1 Abstract Class untuk Morphological Analysis Service (NICT Language Grid Project, 2008)

| | |
|---|---|
| Package | `jp.go.nict.langrid.wrapper.ws_1_2.morphologicalanalysis` |
| Class | `AbstractMorphologicalAnalysisService` |
| Constructor | `public AbstractMorphologicalAnalysisService();` |
| Abstract Method | `protected abstract Collection<Morpheme> doAnalyze(`<br>`  Language language, String text`<br>`  )`<br>`  throws InvalidParameterException`<br>`  , ProcessFailedException;` |

Pada *wrapper Morphological Analyzer* akan diimplementasikan *method* doAnalyze sesuai dengan *abstract method* yang didefinisikan pada *abstract class* **`AbstractMorphologicalAnalysisService`**. *Method* ini berisi proses untuk menjalankan program *Morphological Analyzer*. Spesifikasi masukan dan keluaran dari *method* ini dapat dijelaskan sebagai berikut.

Masukan dari *method* ini terdiri dari dua buah argumen. Argumen pertama berisi kode bahasa dan argumen kedua berisi sebuah *string* yang merupakan kumpulan kata-kata yang akan dianalisis morfologinya.

Keluaran dari *method* ini adalah kumpulan obyek **`Morpheme`** yang memiliki tiga atribut, yaitu atribut *word*, *original*, dan *part of speech*. Setiap obyek **`Morpheme`** yang dikeluarkan oleh *method* ini merepresentasikan hasil analisis morfologi dari sebuah kata pada masukan. Urutan hasil analisis yang dikeluarkan sesuai dengan urutan kata pada masukan.

Pada obyek **`Morpheme`**, atribut *word* berisi *string* yang merupakan kata yang menjadi masukan. Atribut *original* berisi *string* kata dasar dari kata yang menjadi masukan. Atribut *part of speech* berisi kode *part of speech* dari kata yang menjadi masukan. *Part of speech* dari sebuah kata adalah kelas kata tersebut. Contoh dari *part*

*of speech* adalah *verb*, *noun*, *adjective*, dan lain-lain. Adapun kode *part of speech* yang disediakan oleh Language Grid adalah sebagai berikut.

1. PartOfSpeech.noun_proper
2. PartOfSpeech.noun_common
3. PartOfSpeech.noun
4. PartOfSpeech.noun_other
5. PartOfSpeech.verb
6. PartOfSpeech.adjective
7. PartOfSpeech.adverb
8. PartOfSpeech.unknown
9. PartOfSpeech.other

Sebagai contoh, untuk pengurai morfologi bahasa Indonesia, jika diberikan masukan *text* berupa "adik terlalu bersemangat menendang bola hingga kaca menjadi pecah", maka keluaran yang dihasilkan adalah kumpulan **`Morpheme`** seperti yang direpresentasikan pada Tabel 0.2.

Tabel 0.2 Contoh representasi kumpulan **`Morpheme`** yang menjadi keluaran *method* **`doAnalyze`**

| word | original | part of speech |
|---|---|---|
| adik | adik | noun |
| terlalu | lalu | adjective |
| bersemangat | semangat | verb |
| menendang | tendang | verb |
| bola | bola | noun |
| hingga | hingga | noun |
| kaca | kaca | noun |
| menjadi | jadi | verb |
| pecah | pecah | noun |

**UNIVERSITAS INDONESIA**