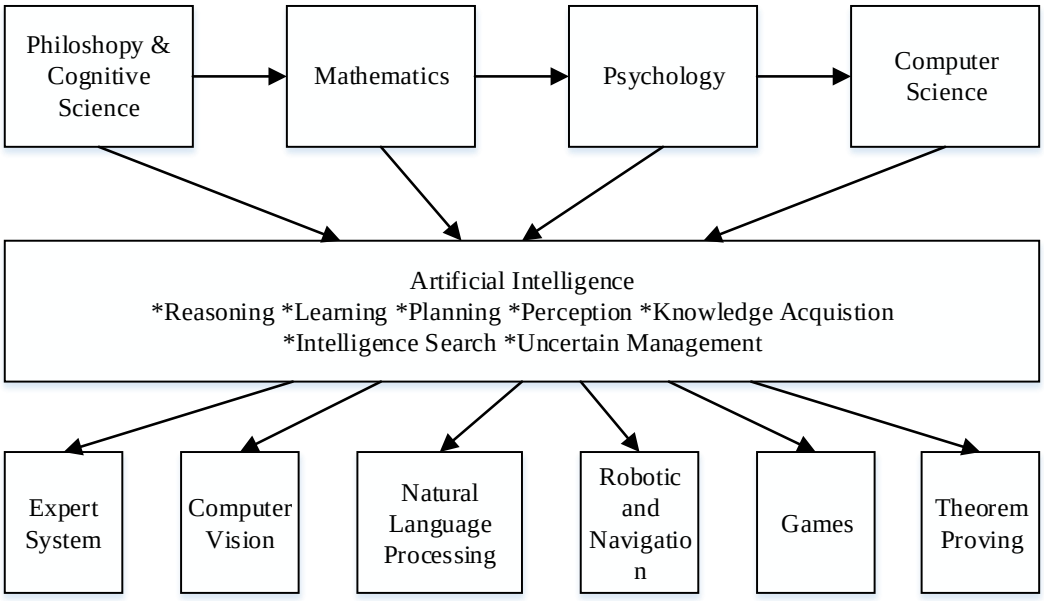
BAB 2

LANDASAN TEORI

2.1 Artificial Intelligence

Artificial Intelligence (AI) atau Kecerdasan Buatan merupakan bidang ilmu komputer yang mempunyai peran penting di era kini dan masa akan datang. Bidang ini telah berkembangsangat pesat di 20 tahun terakhir seiring dengan pertumbuhan kebutuhan akan perangkat cerdas pada industri dan rumah tangga. Kata *Intelligence* berasal dari bahasa latin *intelliga* yang berati saya "saya paham". Jadi, *Intelligence* adalah kemampuan memahami dan melakukan aksi. McCarthy mendefinisikan AI sebagai,"AI merupakan cabang dari ilmu komputer yang berfokus pada pengembangan komputer untuk dapat memiliki kemampuan dan berprilaku seperti manusia" [6].



Gambar 2. 1 Domain Area AI [6]

Tujuan dari sistem kecerdasan buatan dibagi dalam 4 kategori [6]:

1. Sistem yang dapat berpikir seperti manusia
2. Sistem yang dapat berpikir secara rasional
3. Sistem yang dapat beraksi seperti manusia
4. Sistem yang dapat beraksi secara rasional

2.2 Morphological Analysis

Morphological Analysis adalah proses dimana setiap kata yang berdiri sendiri (individual word) dianalisis kembali ke komponen pembentuk dan token nonword seperti tanda baca dan sebagainya dipisahkan dari kata tersebut. Hasil akhir dari proses ini adalah proses Parsing. Parsing adalah proses mengkonversikan daftar kata yang berbentuk kalimat ke dalam bentuk struktur yang mendefinisikan unit yang diwakili oleh sebuah daftar [6]. Pada tabel 2.1 dapat dilihat beberapa karakter (token nonword)yang harus dipisahkan dari kata.

Tabel 2. 1 Tabel Karakter (Token *Nonwrod*)

Karakter			
!	~	+	/
@	&	+	\
#	*	{	“
$	(	}	‘
%	)	[	:
^	-	]	:
`	_	|	.
,	<	>	?
White space (tab, spasi, enter)			

2.3 Stopwords Removal

Stopword removal adalah sebuah proses untuk menghilangkan kata yang ‘tidak relevan’ pada hasil parsing sebuah dokumen teks dengan cara membandingkannya dengan *stoplist*. *Stoplist* berisi sekumpulan kata yang ‘tidak relevan’, namun sering sekali muncul dalam sebuah dokumen. Pada tabel 2.2 merupakan daftar *stoplist* yang digunakan didalam sistem.

Tabel 2. 2 Daftar *Stoplist* [7]

Stoplist				
'yang'	‘untuk’	‘ini’	‘telah’	‘begitu’
‘pada’	‘ke’	‘karena’	‘dari’	‘maka’
‘menurut’	‘namun’	‘kepada’	‘di’	‘lagi’
‘antara’	‘dia’	‘oleh’	‘serta’	‘tentang’
‘ia’	‘dua’	‘saat’	‘bagi’	‘demi’
‘seperti’	‘tidak’	‘harus’	‘sekitar’	‘dimana’
‘jika’	‘dan’	‘sementara’	‘kami’	‘kemana’
‘sehingga’	‘kembali’	‘setelah’	‘belum’	‘sampai’
‘sebagai’	‘ada’	‘mereka’	‘anda’	‘sedangkan’
‘masih’	‘juga’	‘sudah’	‘itulah’	‘selagi’
‘hal’	‘akan’	‘saya’	‘daripada’	‘sementara’
‘ketika’	‘dengan’	‘terhadap’	‘yakni’	‘sebelum’
‘adalah’	‘kita’	‘secara’	‘yaitu’	‘tetapi’
‘itu’	‘hanya’	‘agar’	‘kenapa’	‘apakah’
‘dalam’	‘atau’	‘lain’	‘mengapa’	‘supaya’
‘bisa’	‘bahwa’	‘anda’	‘begitu’	‘dll’

2.4 Stemming & Lemmatization

Stemming merupakan sebuah proses yang bertujuan untuk mereduksi jumlah variasi dalam representasi dari sebuah kata. Resiko dari proses stemming adalah hilangnya informasi dari kata yang di-stem. Hal ini menghasilkan menurunnya akurasi atau presisi. Sedangkan, keuntungannya adalah, proses stemming dapat meningkatkan kemampuan untuk melakukan recall.

Tujuan dari stemming sebearnya adalah meningkatkan performance dan mengurangi penggunaan resource dari system dengan mengurangi jumlah unique word yang harus diakomodasikan oleh sistem. Jadi, secara umum algoritma stemming mengerjakan transformasi dari sebuah kata menjadi sebuah standar representasi morfologi (yang dikenal sebagai stem).

Lemmatization adalah sebuah proses untuk menemukan bentuk dasar dari sebuah kata. Ada sebuah teori yang menjelaskan bahwa lemmatization adalah proses yang bertujuan untuk melakukan normalisasi pada teks atau kata berdasarkan bentuk dasar yang merupakan bentuk lemma-nya. Normalisasi disini dalam artian mendefinisikan dan menghapus sebuah prefix serta suffiks dari sebuah kata. Lemma adalah bentuk dasar dari sebuah kata yang memiliki arti tertentu berdasarkan pada kamus.

2.5 Algoritma Stemming Nazief dan Adriani

Algoritma *stemming* Nazief dan Adriani (1996) dikembangkan berdasarkan aturan morfologi Bahasa Indonesia yang mengelompokkan imbuan menjadi awalan (*prefix*), sisipan (infix), akhiran (*suffix*) dan gabungan awalan-akhiran (*confixes*). Algoritma ini menggunakan kamus kata dasar dan mendukung *recoding*, yakni penyusunan kembali kata-kata yang mengalami proses *stemming* berlebih.

Aturan *morfologi* Bahasa Indonesia mengelompokkan imbuhan ke dalam beberapa kategori sebagai berikut :

1) *Inflection suffixes* yakni kelompok akhiran yang tidak merubah bentuk kata dasar. Sebagai conth, kata “duduk” yang diberikan akhiran “-lah” akan menjadi “duduklah”.

   Kelompok ini dibagi menjadi dua :

   a. *Particle* (P) atau partikel, yakni termasuk di dalamnya “-lah”, “-kah”, “-tah”, dan “-pun”.
   b. *Possessive Pronoun* (PP) atau kata ganti kepunyaan, termasuk di dalamnya adalah “-ku”, “-mu”, dan “-nya”.

2) *Derivation Suffixes* (DS) yakni kumpulan akhiran asli Bahasa Indonesia yang secara langsung ditambahkan pada kata dasar yaitu akhiran “-i”, “-kan”, dan “-an”.
3) *Derivation Prefixes* (DP) yakni kumpulan awalan yang dapat langsung diberikan pada kata dasar murni, atau pada kata dasar yang sudah

mendapatkan penambahan sampai dengan 2 awalan. Termasuk di dalamnya adalah :

a. Awalan yang dapat ber*morfologi* (“me-”, ”be-”, “pe-”, dan “te-”)
b. Awalan yang tidak ber*morfologi* (“di-”, “ke-”, dan “se-”)

Aturan untuk pemenggalan kata awalan pada algoritma *stemmer* Nazief dan Adiani dapat dilihat pada Tabel 2.3.

Tabel 2. 3 Aturan Pemenggalan Awalan *Stemmer* Nazief Dan Adriani [7]

Aturan	Format Kata	Pemenggalan
1	berV…	ber-V…| be-rV…
2	berCAP…	ber-CAP… dimana C!=’r’ & P!=’er’
3	berCAerV…	ber-CaerV… dimana C!=’r’
4	belajar	bel-ajar
5	beC1erC2…	be-C1erC2.. dimana C1!={‘r’|’1’}
6	terV…	ter-V… | te-rV…
7	terCerV…	ter-CerV… diaman C!=’r’
8	terCP…	ter-CP... dimana C!=’r’ dan P!=’er’
9	teC1erC2...	te-C1erC2... dimana C1!=’r’
10	me{l|r|w|y}V...	me-{l|r|w|y}V...
11	mem{b|f|v}...	mem-{b|f|v}...
12	mempe{r|l}...	mem-pe...
13	mem{rV|V}...	me-m{rV|V}... | me-p{rV|V}...
14	men{c|d|j|z}...	men-{c|d|j|z}...
15	menV...	me-nV... | me-tV
16	meng{g|h|q}...	meng-{g|h|q}...
17	mengV...	meng-V... | meng-kV...
18	menyV...	meny-sV…
19	mempV...	mem-pV... dimana V!=’e’
20	pe{w|y}V...	pe-{w|y}V...
21	perV...	per-V... | pe-rV...

23	perCAP…	per-CAP... dimana C!=’r’ dan P!=’er’
24	perCAerV...	per-CAerV... dimana C!=’r’
25	pem{b|f|V}...	pem-{b|f|V}...
26	pem{rV|V}...	pe-m{rV|V}... | pe-p{rV|V}...
27	pen{c|d|j|z}...	pen-{c|d|j|z}...
28	penV...	pe-nV... | pe-tV...
29	peng{g|h|q}	peng-{g|h|q}...
30	pengV...	peng-V... | peng-kV...
31	penyV...	peny-sV…
32	pelV...	pe-lV... kecuali ‘pelajar’ yang menghasilkan ‘ajar’
33	peCerV...	per-erV... dimana C!={r|w|y|l|m|n}
34	peCP...	pe-CP... dimana C!={r|w|y|l|m|n} dan P!=’e’

Keterangan simbol huruf :

C : huruf konsonan

V : huruf vokal

A : huruf vokal atau konsonan

P : partikel atau fragmen dari suatu kata, misalnya “er”

2.6 Rekomendasi Nilai

Rekomendasi nilai merupakan hasil dari *komputasi similarity.* Rekomendasi nilai diambil berdasarkan hasil kecocokkan kata antara jawaban siswa dan kunci jawaban yang ada di *database.* Rekomendasi nilai nantinya melihat persentase kesamaan antara jawaban siswa yang telah dimasukkan dengan kunci jawaban.

$$\boldsymbol{total\ nilai = \frac{jumlah\ nilai\ benar}{jumlah\ soal}} \qquad \textbf{(2.1)}$$

2.7 Jawaban Esai

Ujian dapat dilakukan dalam tiga jenis, yang pertama yaitu ujian pilihan ganda, ujian isian singkat dan ujian esai. Ujian esai merupakan evaluasi pembelajaran dalam bentuk soal esai yang mempunyai jawaban lebih bervariasi dibandingkan soal pilihan ganda. Variasi jawaban tersebut memberikan kesulitan tersendiri bagi guru dalam menilai jawaban.

Esai yang dinilai dibagi menjadi dua bagian, yang pertama adalah esai yang merupakan karangan siswa (bisa berupa eksposisi, deskripsi, argumentasi, atau narasi). Yang kedua adalah jawaban esai pendek yang biasa terdapat pada bagian evaluasi buku teks pelajaran atau ujian.

2.8 Automated Essay Scoring (AES)

Automated essay scoring (AES) dalam bahasa indonesia berarti penilaian esai otomatis didefinisikan sebagai tehnologi komputer yang mengevaluasi dan memberikan nilai pada tulisan. Sistem AES dibuat untuk membantu guru dalam menilai ujian tertulis yang dilakukan dikelas seperti latihan dan ujian akhir semester. Sistem AES memiliki fungi utama untuk membantu menghemat waktu, biaya dalam memeriksa hasil ujian tertulis, dan beberapa masalah dalam penilaian ujian tertulis [8].

Sistem AES mendapatkan kritikan berupa kekurangan hubungan antar manusia, mudah untuk dimanipulasi, dan memerlukan data *training* yang besar untuk melatih sistem AES. Dibalik semua kekurangan itu, sistem AES tetap menarik perhatian para peneliti, pendidik baik di unversitas, sekolah dan perusahaan yang bergerak dibidang pengetesan [8].

2.8.1 Vector Space Model (VSM)

Vector space model (VSM) adalah representasi kumpulan dokumen sebagai vektor dalam sebuah ruang vektor. VSM merupakan teknik dasar dalam perolehan informasi yang dapat digunakan untuk penilaian relevansi dokumen terhadap kata kunci pencarian (*query*) pada mesin pencari, klasifikasi dokumen, dan

pengelompokan dokumen [3]. Dalam *Vector Space Model*, koleksi dokumen direpresentasikan sebagai sebuah matrik *term-document* (matrik *term-frequency*). Setiap sel dalam matrik bersesuaian dengan bobot yang diberikan dari suatu *term* dalam dokmen yang ditentukan. Nilai nol berarti bahwa *term* tersebut tidak hadir dalam dokumen [4].

D1 : Saya mahasiswa Ilmu Komputer
D2 : Saya menimba ilmu di Fakultas Ilmu Komputer
D3 : Mahasiswa Fakultas Ilmu Komputer banyak

	D1	D2	D3
Banyak	0	0	1
Di	0	1	0
Fakultas	0	1	1
Ilmu	1	2	1
Komputer	1	1	1
Mahasiswa	1	0	1
Menimba	0	1	0
Saya	1	1	0

Gambar 2. 2 Contoh Dokumen Dan Matriks Kata-Dokumen

Melalui *vector space model* dan TF *weighting* maka akan didapatkan representasi nilai numerik dokummen sehingga kemudian dapat dihitung kedekatan antar dokumen. Semakin dekat dua vektor di dalam suatu VSM, maka semakin mirip dua dokumen yang diwakili vektor tersebut. Terdapat empat fungsi untuk mengukur kemiripan (*similarity measure*) yang dapat digunakan untuk model ini :

1. *Cosine distance* / *cosine similarity*
2. *Inner similarity*
3. *Dice similarity*
4. *Jaccard similarity*

Salah satu ukuran kemiripan teks yang popular adalah *cosine similarity*. Ukuran ini menghitung nilai cosinus sudut antara dua vektor. Jika terdapat dua vektor dokumen d dan *query* q, serta t *term* diekstrak dari koleksi dokumen maka nilai cosinus antara d dan q didefinisikan sebagai berikut :

$$\boldsymbol{sim\ (Di, Q) = \frac{\sum Di * Q}{\sqrt{\sum D\text{^}2} * \sqrt{\sum Q\text{^}2}}} \qquad \textbf{(2.2)}$$

2.8.2 Term Frequency-Inverse Document Frequency (TF-IDF) Weighting

Metode pembobotan yang paling sederhana terhadap suatu *term* (*term wighting*) adalah dengan menggunakan frekuensi kemunculan *term* (kata) / *term frequency* (TF) yang bersangkutan pada suatu dokumen. *Inverse Document Frequency* (IDF) adalah logaritma dari rasio jumlah keseluruhan dokumen yang diproses dengan jumlah dokumen yang memiliki *term* bersangkutan. Lalu Salton bereksperimen untuk mengkombinasikan kedua metode pembobotan tersebut, dengan mempertimbangkan frekuensi inter-dokumen dan frekuensi intra-dokumen dari suatu *term*. Dengan menggunakan frekuensi *term* pada suatu dokumen dan distribusinya pada keseluruhan dokumen, yakni kemunculan pada dokumen-dokumen lain (IDF). Salton menarik kesimpulan melalui eksperimennya bahwa *term-term* dengan total frekuensin menengah, lebih berguna dalam *retrieval* jika dibandingkan dengan *term-term* yang total frekuensinya terlalu tinggi atau terlalu rendah. Konsep intra-dokumen dan inter-dokumen ini kemudian dikenal sebagai metode TF-IDF.

Rumus yang digunakan untuk menyatakan bobot (*w*) masing-masing dokumen terhadap kata kunci adalah :

$$\boldsymbol{Wd,t\ =\ tf\ d,t\ \times\ \text{IDF}t} \qquad \textbf{(2.3)}$$

Dimana :

d = dokumen ke-d

t = kata ke-t dari kata kunci

Wd,t = bobot dokumen ke-d terhadap kata ke-t

Untuk penelitian saat ini, pembobotan kata yang akan digunakan adalah pembobotan kata berdasarkan *Term Frequency* (TF) dari matriks kata-dokumen dan vektor *query*.

2.9 Pemograman Berorientasi Objek

Metologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diperlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didsarkan pada penerapan prinsip-prinsip pengolahan kompleksitas. Metode beorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemograman berorientasi objek, dan pengujian berorientasi objek [9].

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut :

1. Meningkatkan produktivitas
   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).
2. Kecepatan pengembangan
   Karena sistem yang dibangun dengan caik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.
3. Kemudahan pemeliharaan
   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang munkin sering berubah-ubah.
4. Adanya konsistensi
   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

5. Meningkatkan kualitas perangkat lunak

   Karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.9.1 Konsep Dasar Berorientasi Objek

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek :

1. Kelas (*class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (relantioship) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru. Secara teknis kelas adalah sebuah struktur tertentu dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.

2. Objek (*object*)

   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusi, suatu organisasi, tempat, kejadian, struktur, status, atau hal – hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada staus objeknya. Objek dilihat dari segi teknis adalah elemen pada saat *runtime* yang akan diciptakan, dimanipulasi, dan dihancurkan saat eksekusi sehingga sebuah objek hanya ada saat sebuah program dieksekusi, jika masih dalam bentuk kode, disebut sebagai kelas. Jadi pada saat *runtime* (saat sebuah program diekekusi), yang kita punya adalah objek, di dalam teks program yang kita lihat hanyalah kelas.

3. Metode (*method*)

   Poreasi atau metode atau *method* pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau oprasi. Metode atau operasi yang berfungsi untuk memanipulasi objek iu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Meotde atau operasi berasal dari *event,* aktivitas atau aksi kedaaan, fungsi, atau kelakuan dunia nyata. Contoh metode atau operasi misalnya *read, write, move, copy* dan sebagainya.

4. Atribut (*attribute*)

   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek, misalnya berat, jenis, nama, dan sebaginya.

5. Abstraksi (*abstraction*)

   Prinsip untuk merepresentasi dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*encapsulation*)

   Pembungkus atribut data dan layanan (operasi-operasi) yang mempunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

7. Pewarisan (*inheritance*)

   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dririnya.

8. Antarmuka (*interface*)

   Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

9. *Reusability*

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi Antar Objek

    Komunikasi antarobjek dilakukan lewat pesan (*message*) yang dikirim dari satu objek ke objek lainnya.

12. Polimorfisme (*poymorphism*)

    Kemampuan suatu objek digunakan di banyak tujuan yang berbeda dengan nama yang sehingga menghemat baris program.

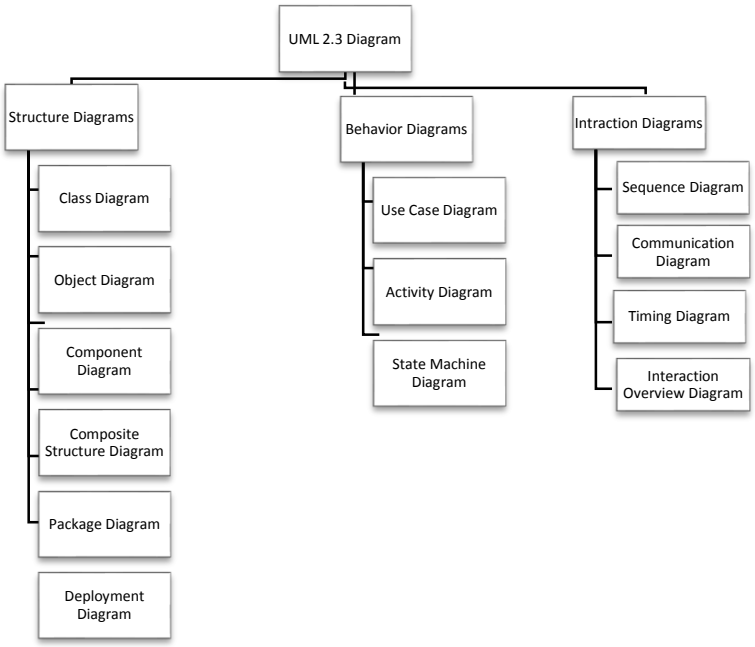
13. *Package*

    Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.9.2 Pengenalan UML

Pada perkembangan teknik pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks - teks pendukung. UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek. UML terbaru adalah

UML 2.3. Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian ketegori didalam UML dapat dilihat pada gambar berikut ini :



Gambar 2. 3 Diagram UML 2.3 [9]

Berikut ini penjelasan dari pembagian kategori didalam UML :

1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada suatu sistem.
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

2.10 Database

Database atau di dalam bahasa Indonesia disebut basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat [9]. Basis data terdiri atas dua kata, basis dan data. Basis dapat diartikan sebagai markas atau

gudang, tempat bersarang/berkumpul. Sedangkan data adalah representasi fakta dunia nyatayang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan dan sebagainya, yang diwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya [10]. Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Sitem yang mengelola basis data disebut dengan Sistem Manajemen Basis Data atau DBMS (*Database Management System*). Spesialis yang menangani DBMS disebut sebagai DBA (*Database Administrator*) [9].

2.11MySql

MySql adalah salah satu DBMS (*Database Management System*) komersil yang paling banyak digunakan. DBMS atau dalam bahasa Indonesia sering disebut Sistem Manajemen Basis Data adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data. Sedangkan SQL (*Structure Query Language*) adalah bahasa yang digunakan untuk mengelola data pada RDBMS (*Relational Database Management System*). SQL mulai berkembang pada tahun 1960an. SQL mulai digunakan sebagai standar yang resmi pada tahun 1986 oleh ANSI (*American National Standards Institute*) dan pada tahun 1987 oleh ISO (*International Organization for Standardization*) dan disebut sebagai SQL-86 [9].

2.12 Teori Pengujian

Pengujian adalah proses menelusuri dan mempelajari sebuah program dalam rangka menemukan kesalahan pada perangkat lunak sebelum diserahkan kepada pengguna. Pengujian perangkat lunak dalam penelitian ini dibagi menjadi dua metode pengujian yaitu metode pengujian *white box* dan metode pengujian *black box*.

2.12.1 Black Box Testing

Black-box testing adalah metode pengujian perangkat lunak yang tes fungsionalitas dari aplikasi yang bertentangan dengan struktur internal atau kerja pengetahuan khusus dari kode aplikasi / struktur internal dan pengetahuan pemrograman pada umumnya tidak diperlukan. Uji kasus dibangun di sekitar spesifikasi dan persyaratan, yakni, aplikasi apa yang seharusnya dilakukan. Menggunakan deskripsi eksternal perangkat lunak, termasuk spesifikasi, persyaratan, dan desain untuk menurunkan uji kasus. Tes ini dapat menjadi fungsional atau non-fungsional, meskipun biasanya fungsional. Perancang uji memilih input yang valid dan tidak valid dan menentukan output yang benar. Tidak ada pengetahuan tentang struktur internal benda uji itu. Metode uji dapat diterapkan pada semua tingkat pengujian perangkat lunak: unit, integrasi, fungsional, sistem dan penerimaan.Ini biasanya terdiri dari kebanyakan jika tidak semua pengujian pada tingkat yang lebih tinggi, tetapi juga bisa mendominasi unit testing juga [5].

Ujicoba *blackbox* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan interface
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan performa
5. kesalahan inisialisasi dan terminasi

2.12.2 White Box Testing

White Box Testing merupakan cara pengujian dengan melihat ke dalam modul untuk meneliti kode-kode program yang ada, dan menganalisis apakah ada kesalahan atau tidak. Jika ada modul yang menghasilkan output yang tidak sesuai dengan proses bisnis yang dilakukan, maka baris-baris program, variabel, dan parameter yang terlibat pada unit tersebut akan dicek satu persatu dan diperbaiki, kemudian di lakukan *compile* ulang [5].

Dengan menggunakan *white box* akan didapatkan kasus uji yang :

1. Menguji semua keputusan logikal
2. Menguji seluruh *Loop* yang sesuai dengan batasannya
3. Menguji seluruh struktur data internal yang menjamin validitas

1. CYCLOMATIC COMPLEXITY

Cyclomatic Complexity merupakan suatu sistem pengukuran yang menyediakan ukuran kuantitatif dari kompleksitas logika suatu program. Pada Basis Path Testing, hasil dari *cyclomatic complexity* digunakan untuk menentukan banyaknya *independent paths. Independent path* adalah sebuah kondisi pada program yang menghubungkan *node* awal dengan node akhir [5].

Terdapat 2 persamaan yang digunakan, yaitu:

$$\boldsymbol{V(G) = E - N + 2} \qquad \textbf{(2.4)}$$

atau

$$\boldsymbol{V(G) = P + 1} \qquad \textbf{(2.5)}$$

Keterangan:

V(G)= *cyclomatic complexity* untuk flow graph G

E=Jumlah *edge*(panah)

N=Jumlah *node*(lingkaran)

P=Jumlah *predicate node*

2. Graph Metrik

Graph metrik merupakan software yang dikembangkan untuk membantu uji coba basis path atau struktur data. Graph metrik adalah matrik empat persegi yang mempunyai ukuran yang sama dengan umlah node pada *flowgraph*. Masing-masing baris dan kolom mempunyai hubungan dengannode yang telah ditentukan dan pemasukan data matrik berhubungan dengan hubungan (*edge*) anta node.

Hubungan bobot menyediakan tambahan informasi tentang aliran kontrol. Secara simpel hubungan bobot dapat diberi nilai 1 jika ada hubungan antara node atau nilai 0 jika tidak ada hubungan. Dapat juga hubungan bobot diberi tanda dengan :

1. Kemungkinan link (*edge*) dikerjakan
2. Waktu yang digunakan untuk proses selama traversal dari link
3. Memori yang diperlukan selama traversal link
4. Sumber daya yang diperlukan.