BAB 2
LANDASAN TEORI

Landasan teori merupakan definisi, konsep yang telah disusun secara sistematis dan dasar yang kuat dalam sebuah penelitian. Landasan teori yang digunakan dalam penelitian ini adalah mengenai *Natural Language Processing* (NLP), pembangkit pertanyaan otomatis, peringkas dokumen teks otomatis, metode *lexical chain, Object Oriented Programming* (OOP), *Unified Modeling Language* (UML).

2.1. *Natural Language Processing* (NLP)

Pada prinsipnya bahasa alami adalah suatu bentuk representasi dari suatu pesan yang ingin dikomunikasikan antar manusia. Bentuk utama representasinya adalah berupa suara/ucapan (*spoken language*), tetapi sering pula dinyatakan dalam bentuk tulisan. Bahasa dapat dibedakan menjadi bahasa alami, dan bahasa buatan. Bahasa alami adalah bahasa yang biasa digunakan untuk berkomunikasi antar manusia, misalnya Bahasa Inggris, Bahasa Indonesia, dan sebagainya. Bahasa buatan adalah bahasa yang dibuat secara khusus untuk memenuhi kebutuhan tertentu, misalnya bahasa pemodelan atau bahasa pemrograman komputer.

Chomsky adalah orang yang pertama kali merepresentasikan bahasa sebagai rangkaian simbol. Chomsky berhasil memperlihatkan bahwa bahasa apapun dapat direpresentasikan dengan suatu cara yang universal. Pemikiran Chomsky yang merepresentasikan bahasa sebagai kumpulan simbol-simbol dan aturan yang mengatur susunan simbol-simbol tersebut telah membuka peluang untuk melakukan pemrosesan bahasa secara simbolik dengan teknologi komputer, sehingga melahirkan bidang ilmu Natural Language Processing (NLP) [6].

Dipandang dari sisi implementasi teknologinya, pemrosesan bahasa lisan dan tulisan adalah sangat berbeda. Bahasa lisan lebih banyak melakukan pemrosesan bunyi atau suara, sedangkan bahasa tulisan lebih banyak melakukan pemrosesan

simbol-simbol tertulis. Sebagai akibatnya, penelitian di bidang bahasa lisan pada awalnya lebih banyak dilakukan dalam bidang ”*signal processing*”, sedangkan penelitian-penelitian untuk pemrosesan bahasa tulisan lebih banyak dilakukan dalam bidang ”*artificial intelligence*” atau kecerdasan buatan yang pada prinsipnya melakukan *symbolic processing*.

Saat ini, teknologi yang berkaitan dengan pemrosesan bahasa alami ini sering disebut sebagai ”*speech and language technology*”, “*natural language processing technology*”, “*human language technology*”, atau dalam beberapa pertemuan ilmiah para peneliti di bidang ini di Indonesia, menyepakati penggunaan istilah ”teknologi bahasa” untuk menyebut teknologi ini. Dari segi keilmuan, bidang ini dikenal sebagai bidang ”*natural language processing*” atau ”*computational linguistic*”.

Suatu sistem pemrosesan bahasa alami secara lisan dapat dibentuk dari tiga sub-sistem, yaitu sebagai berikut.

1. Sub-Sistem Natural Language Processing (NLP), berfungsi untuk melakukan pemrosesan secara simbolik terhadap bahasa tulisan. Beberapa bentuk aplikasi subsistem ini adalah translator bahasa alami (misalnya dari Bahasa Inggris ke Bahasa Indonesia), sistem pemeriksa sintaks bahasa, sistem yang dapat ”menyimpulkan” suatu narasi, dan sebagainya.
2. Sub-sistem Text to Speech (TTS), berfungsi untuk mengubah text (bahasa tulisan) menjadi ucapan (bahasa lisan).
3. Sub-Sistem Speech Recognition (SR), merupakan kebalikan teknologi Text to Speech, yaitu sistem yang berfungsi untuk mengubah atau mengenali suatu ucapan (bahasa lisan) menjadi text (bahasa tulisan). Berbeda dengan kebanyakan sistem lain yang bersifat generik, teknik-teknik yang digunakan dalam pemrosesan bahasa alami bersifat sangat ‘language dependent’. Suatu sistem atau teknik yang berlaku untuk suatu bahasa tidak mudah diterapkan untuk bahasa lainnya [7].

2.2. Question Generator

Question Generation (QG) merupakan proses yang dilakukan untuk membangkitkan pertanyaan yang dapat diterima oleh manusia dari sebuah input dengan bentuk terstruktur (database) ataupun tidak terstruktur (text). Tugas Pembangkit pertanyaan berisi beberapa subarea [2]. Biasanya, pendekatan yang diambil untuk QG tergantung pada tujuan aplikasi QG. Secara umum, sistem QG dapat membantu dalam bidang-bidang berikut:

a. *Intelligent Tutoring*. QG dapat mengajukan pertanyaan berdasarkan materi pembelajaran untuk memeriksa prestasi peserta didik atau membantu mereka fokus pada kunci dari materi pembelajaran. QG juga bisa membantu tutor untuk mempersiapkan pertanyaan yang ditujukan untuk pelajar atau mempersiapkan pertanyaan potensial dari peserta didik.

b. *Closed-domain Question Answering* (QA). Beberapa sistem tertutup domain QA menggunakan *predefined* pasangan tanya jawab untuk menyediakan layanan QA.

c. Sistem sumarisasi Bahasa alami. QG dapat membantu untuk menghasilkan pertanyaan dari sumber informasi dalam rangka memberikan daftar calon pertanyaan yang sering diajukan (FAQ).

Pembangkitan pertanyaan dapat dilakukan dengan memanfaatkan sekumpulan dokumen yang ada di internet (*open domain*) atau sekumpulan dokumen yang ada di dalam sistem itu sendiri (*closed domain*). Secara umum, pertanyaan dapat diklasifikasikan menjadi lima kategori, yaitu pertanyaan yang bersifat *factoid* (*factoid question*), pertanyaan yang bersifat *nonfactoid* (*non-factoid question*), *list question*, *yes/no question*, dan *opinion question*. Hal ini menandakan bahwa pertanyaan yang dibangkitkan oleh sistem pembangkit pertanyaan otomatis dapat berupa salah satu, beberapa, atau semua kategori pertanyaan [8]. Pertanyaan yang bersifat *non factoid* adalah pertanyaan dengan jawaban fakta yang cukup panjang yang biasanya terdiri dari satu klausa atau lebih.

Dalam pembuatan pertanyaan berdasarkan template yang telah ditentukan sebelumnya, terdapat tiga klasifikasi kata khusus yang nantinya akan mengacu pada

template mana yang akan dipakai untuk membangkitkan pertanyaan. Berikut ini adalah kata khusus yang digunakan untuk membangkitkan pertanyaan.

Tabel 2. 1 Pengelompokan kata khusus

Kategori	Kata Khusus Sebelum Target Kata	Kata Khusus Setelah Target Kata
Definisi	disebut, dikenal, dinamakan, mendefinisikan	adalah, yaitu, ialah, merupakan, diartikan
Alasan	oleh sebab itu, jadi, memungkinkan adanya, , dengan demikian, maka, dikatakan, penyebab terjadinya, sehingga, mengapa, walau demikian, namun demikian	sebab, karena, bertujuan
Metode	dengan cara	berfungsi untuk, berguna untuk

2.3. Automatic Text Summarization

Automatic Text Summarization adalah proses meringkas dokumen teks menggunakan bantuan program komputer dengan tetap mempertahankan inti dokumen asli [9]. Pada umumnya, automatic summarization memiliki tiga teknik peringkasan, yaitu *extraction-based summarization, abstraction-based summarization,* dan *maximum entropy-based summarization*. Teknik yang sering digunakan dalam kasus meringkas teks ini yaitu extraction-based summarization. Teknik ini memiliki fungsi untuk memberikan hasil ringkasan yang lebih pendek dari dokumen aslinya. Teknik selanjutnya adalah teknik abstraksi yang merupakan cara mengambil intisari dari teks sumber kemudian membuat ringkasan dengan

menciptakan kalimat-kalimat baru yang merepresentasikan intisari teks sumber dalam bentuk berbeda.

Text mining dapat didefinisikan sebagai keilmuan di mana pengguna berinteraksi dengan koleksi dokumen dari waktu ke waktu menggunakan *tools* analisis. Dengan menganalogikan kedalam data mining, teks mining digunakan untuk mengekstrak informasi yang bermanfaat dari sumber data melalui identifikasi dan mengeksplorasi pola yang menarik. Dalam kasus teks mining, sumber data adalah koleksi dokumen, dan pola yang menarik yang ditemukan tidak di antara catatan database formal tetapi dalam Data tekstual terstruktur di dokumen dalam koleksi ini. Tahapan-tahapan yang dilakukan pada proses text mining adalah sebagai berikut: [6]

2.3.1. Tahap Segmentasi Kalimat

Tahap segmentasi kalimat yaitu proses memisahkan string teks dokumen menjadi kumpulan kalimat - kalimat [6]. Proses pemisahan dokumen menjadi kalimat-kalimat ini menggunakan fungsi split(), dengan tanda titik ".", "?","!" sebagai delimiter untuk memotong string dokumen. Berikut ini adalah contoh segmentasi kalimat dari sebuah paragraf.

Tabel 2. 2 Segmentasi Kalimat

Paragraf	Segmentasi Kalimat
Hubungan manusia dengan sungai jelas tak bisa dipisahkan. Di tepian beberapa sungai, manusia membuat rumah terapung. Di Kalimantan sebutannya lanting sementara di sepanjang sungai Musi, Palembang, disebut rakit.	Hubungan manusia dengan sungai jelas tak bisa dipisahkan
	Di tepian beberapa sungai, manusia membuat rumah terapung
	Di Kalimantan sebutannya lanting sementara di sepanjang sungai Musi, Palembang, disebut rakit

2.3.2. Tahap *Tokenization*

Tahap *tokenization* merupakan tahap pemotongan string input menjadi kumpulan kata yang menyusunnya [6]. Contoh *tokenization* adalah sebagai berikut:

Tabel 2. 3 Hasil *Tokenization*

Kalimat	Hasil *Tokenization*
Hubungan manusia dengan sungai jelas tak bisa dipisahkan	Hubungan
	Manusia
	Dengan
	Sungai
	Jelas
	Tak
	Bisa
	dipisahkan

2.3.3. Tahap Menghapus *Stop Words*

Tahap *removing stop words* merupakan tahap penghilangan kata-kata yang tidak terlalu penting [6]. Setiap kata hasil dari *tokenization* dilakukan pengecekan dengan *stop list* yang berisi kata-kata yang tidak terlalu penting. Jika ditemukan kata pada hasil tokenization sama dengan kata yang ada di dalam *stop list*, maka kata tersebut akan hilang.

2.3.4. Tahap *Stemming*

Tahap *stemming* merupakan tahap mengubah kata dari hasil penghapusan *stop words* menjadi kumpulan kata dasar [6]. Tahap ini tidak jauh berbeda dengan hasil stop words karena jika hasil kata dasar sudah ada dan tidak ada perubahan maka hasilnya akan sama.

2.3.5. Tahap Ekstraksi Kalimat

Extraction feature merupakan proses pengolahan kalimat menggunakan beberapa fitur sehingga menghasilkan nilai-nilai yang nantinya akan digunakan untuk meringkas dokumen. Extraction feature yang digunakan meliputi panjang kalimat, bobot kalimat, posisi kalimat, dan kemiripan antar kalimat [10].

a. Panjang Kalimat

Panjang kalimat merupakan proses untuk mengetahui rasio dalam jumlah kata yang terjadi dalam kalimat atas jumlah kata yang terjadi dalam kalimat dokumen terpanjang. Rumus panjang kalimat adalah sebagai berikut:

$$S_F1 = \frac{Jumlah\ Kata\ dalam\ Kalimat}{Kalimat\ Terpanjang} \quad \text{.....(2.1)}$$

b. Bobot Kalimat

Bobot kata merupakan proses menghitung frekuensi kejadian di dalam dokumen yang telah sering digunakan untuk menghitung kata yang terdapat di dalam kalimat. Bobot yang dilakukan adalah dengan menggunakan perhitungan tf-idf (*term frequency*, *inverse document frequency*). Rumus bobot kata adalah sebagai berikut:

$$S_F2 = \mathbf{wi} = \mathbf{TF} \times (\mathbf{log}\frac{\mathbf{N}}{\boldsymbol{ni}}) \quad \text{.....(2.2)}$$

dimana, tf = frekuensi banyak kata di dalam dokumen

N = jumlah kalimat

ni = jumlah kalimat yang mengandung kata

c. Kemiripan Antar Kalimat

Kemiripan antar kalimat merupakan kata yang muncul di dalam sebuah kalimat sama dengan kata yang muncul di dalam kalimat lain. Rumus kemiripan antar kalimat adalah sebagai berikut:

$$S_F4(K) = \frac{\Sigma Sim(Ki,Kj)}{Max(\Sigma Sim(Ki,Kj))} \quad \text{.....(2.4)}$$

dimana,

Ki = Kalimat ke – i

Kj = Kalimat ke – j

Sim(Ki,Kj) = kemiripan antara kalimat ke-i dan kalimat ke-j

ΣSim(Ki,Kj) = Jumlah kemiripan antara kalimat ke-i dan kalimat ke-j

Dengan Ki,Kj diperoleh dari rumus jarak antar kalimat yaitu:

$$Ki, Kj = \frac{\Sigma Wit \times Wij}{\sqrt{\sum Wit^2} \times \sqrt{\sum Wij^2}} \quad \text{.....(2.5)}$$

dimana,

$\Sigma Wit \times Wij$ = jumlah perhitungan skalar dari pembobotan kalimat

$\sqrt{\sum Wit^2} \times \sqrt{\sum Wij^2}$ = jumlah perhitungan vektor dari pembobotan kalimat

2.4. Lexical Chains

Lexical chains adalah sebuah teknik penting dalam pengolahan bahasa alami. *Lexical chains* berkaitan dengan kalimat yang berelasi dengan kata benda, kata kerja, kata sifat [11]. Jika kalimat tidak berkaitan satu sama lain di dalam teks, maka relasi antar leksikal berelasi satu sama lain. Hal ini merupakan konsep yang penting. Skor dari sebuah rantai leksikal secara langsung berkaitan dengan panjang rantai yang hubungannya dengan kata benda. Peringkas menggunakan *lexical chains* untuk memilih kalimat yang paling representatif.

Langkah-langkah untuk membangun *lexical chains* adalah sebagai berikut:

1. Memilih serangkaian calon kata-kata.
2. Untuk setiap calon kata, menemukan sebuah chain yang bergantung pada keterkaitan antar anggota dari chain.
3. Jika ditemukan, sisipkan kata di dalam *chain* dan perbaharui sesuai hal tersebut. Perhitungan *lexical chains* diperoleh dari rumus sebagai berikut [12]:

$$CStrength = Clength \times \{\Sigma \frac{\Sigma((\text{Tf}-\text{Isfwi} \times \text{Swi})) + (\text{Tf}-\text{Isfwj} \times \text{Swj})}{\text{distance between wi and wj}}\} + TotalNumberOfPairs$$

(2.6)

2.5. Logika *Fuzzy*

Logika *fuzzy* pertama kali diperkenalkan oleh Prof. Lotfi A. Zadeh pada tahun 1965. Dasar logika *fuzzy* adalah teori himpunan *fuzzy*. Pada teori himpunan *fuzzy*, peranan derajat keanggotaan sebagai penentu keberadaan elemen dalam suatu himpunan sangatlah penting. Logika *fuzzy* dapat dianggap sebagai kotak hitam yang menghubungkan antara ruang *input* menuju ruang *output* [13].

Menurut Cox, beberapa alasan menggunakan logika *fuzzy* sebagai berikut:

1. Konsep logika *fuzzy* mudah dimengerti, karena logika *fuzzy* menggunakan dasar teori himpunan, maka konsep matematis yang mendasari penalaran *fuzzy* tersebut cukup mudah untuk dimengerti.
2. Logika *fuzzy* sangat fleksibel, artinya mampu beradaptasi dengan perubahan-perubahan dan ketidakpastian yang menyertai permasalahan.
3. Logika *fuzzy* memiliki toleransi terhadap data yang tidak tepat, artinya jika diberikan sekelompok data yang cukup homogeny, dan kemudian ada beberapa data yang eksklusif, maka logika *fuzzy* memiliki kemampuan untuk menangani data eksklusif tersebut.
4. Logika *fuzzy* mampu memodelkan fungsi-fungsi nonlinear yang sangat kompleks.
5. Logika *fuzzy* dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan.
6. Logika *fuzzy* dapat bekerja sama dengan teknik-teknik kendali secara konvensional.
7. Logika *fuzzy* didasarkan pada bahasa alami, maksudnya menggunakan bahasa sehari-hari sehingga mudah dimengerti.

2.5.1. Himpunan *Fuzzy*

Ada beberapa hal yang perlu diketahui dalam sistem *fuzzy*, diantaranya:

a. Variabel *Fuzzy*

Variabel *Fuzzy* adalah variabel yang hendak dibahas dalam suatu sistem variabel *fuzzy*. Contoh : umur, temperatur, permintaan, dsb.

b. Himpunan *fuzzy*

Himpunan Variabel *fuzzy* adalah suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel *fuzzy*. Himpunan *fuzzy* memiliki dua atribut, yaitu:

1. Linguistik, yaitu penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami.
   Contoh : MUDA, PAROBAYA, TUA.
2. Numeris, yaitu suatu nilai yang menunjukkan ukuran dari suatu variabel.
   Contoh : 40, 25, 50, dsb.

c. Semesta Pembicaraan

Semesta pembicaraan adalah himpunan bilangan *real* yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai semesta pembicaraan dapat berupa bilangan positif atau negatif.

d. Domain

Domain adalah himpunan bilangan *real* yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai domain dapat berupa bilangan positif atau negatif.

2.5.2. Fungsi Keanggotaan

Logika *fuzzy* memiliki fungsi keanggotaan. Fungsi keanggotaan adalah suatu kurva yang menunjukkan pemetaan titik-titik *input* data ke dalam nilai keanggotaannya (derajat keanggotaan) yang memiliki interval antara 0 sampai 1 [10]. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi.

Beberapa fungsi yang dapat digunakan adalah pemetaan *input* ke derajat keanggotaannya digambarkan sebagai suatu garis lurus. Bentuk ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas. Ada 2 keadaan himpunan *fuzzy* yang linear. Pertama, representasi linear naik merupakan kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol (0) bergerak ke kanan menuju ke nilai domain

yang memiliki derajat keanggotaan lebih tinggi. Berikut ini merupakan fungsi keanggotaannya:

$$\mu(x) = \begin{cases} 0; & x \leq a \\ \frac{(x-a)}{(b-a)}; & a \leq x \leq b \\ 1; & x \geq b \end{cases} \quad (2.7)$$

Kedua, representasi linear turun merupakan kebalikan yang pertama. Garis lurus dimulai dari nilai domain dengan derajat keanggotaan tertinggi pada sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah. Berikut ini merupakan fungsi keanggotaannya:

$$\mu(x) = \begin{cases} \frac{(b-x)}{(b-a)}; & a \leq x \leq b \\ 0; & x \geq b \end{cases} \quad (2.8)$$

2.6. Korelasi Antar Kalimat

Korelasi antar kalimat merupakan hubungan antara satu kalimat dengan yang lainnya dimana kalimat kalimat saling berkaitan. Apabila sebuah kalimat memiliki keterkaitan dengan kalimat yang lainnya maka kalimat tersebut bisa menjadi induk klaimat ataupun anak kalimat bagi kalimat sebelum dan sesudahnya. Anak kalimat berguna untuk menjelaskan kalimat induk sehingga mempersempit arti dari kalimat induk tersebut. Keterkaitan antar kalimat ditandai dengan kata penghubung sebagai berikut;

Agaknya, ...	Dengan demikian, ...	Namun, ...
Akan tetapi, ...	Dengan kata lain, ...	Oleh karena itu, ...
Akhinya, ...	Disamping itu, ...	Oleh sebab itu, ...
Akiatnya, ...	Disatu pihak, ...	Pada dasarnya, ...
Artiya, ...	Dipihak lain, ...	Pada hakikatnya, ...
Biarun begitu, ...	Jadi,...	Pada prinsipnya, ...
Biarun demikian, ...	Jika demikian, ...	Sebagai kesimpulan, ...
Berkaitan dengan itu, ...	Kalau begitu, ...	Sebaiknya, ...
Dalam hal ini, ...	Kalau tidak salah, ...	Sebaliknya, ...
Dalam hubungan ini, ...	Kecuali itu, ...	Sebelumnya, ...
Dalam konteks ini, ...	Lagi pula, ...	Sebenarnya, ...
	Meskipun begitu, ...	Sebetulnya, ...
	Mekipun demikian, ...	

2.7. Pemrograman Berorientasi Objek (*OOP*)

Pemrograman berorientasi objek (*Object-Oriented Programming*) yang sering disingkat OOP merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Dibaandingkan dengan logika pemrograman terstruktur, setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

Adapun beberapa istilah dalam pemrograman berorientasi objek [14] adalah sebagai berikut:

1. *Class*

Class merupakan kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh '*class of dog*' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah *class* adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek. Sebuah *class* secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

2. *Object*

Object (objek) dapat disebut membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program computer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

3. Abstraksi

Abstraksi merupakan kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

4. Enkapsulasi

Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak. Hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses *interface* yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

5. Polimorfisme

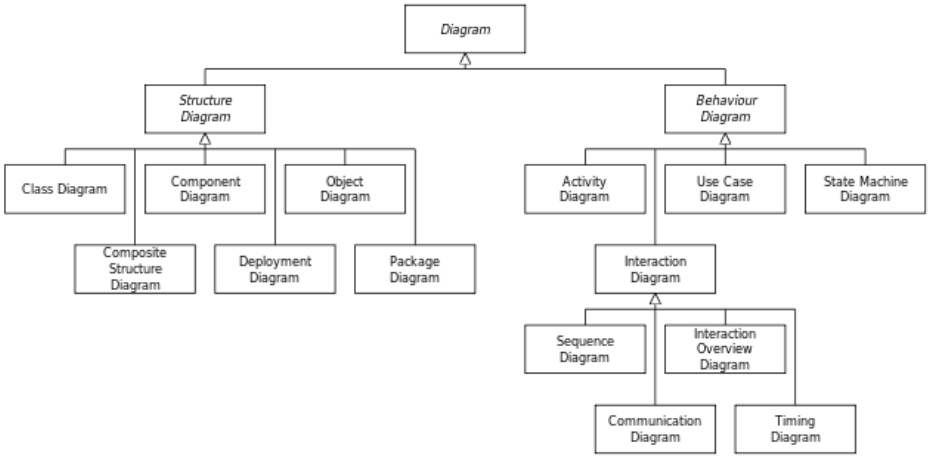
Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini

berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

2.8. *Unified Modelling Language* (UML)

Unified Modeling Language (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya [15]. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh *Object Management Group,* sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat (*tool*) desain berorientasi objek dari IBM.

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem.



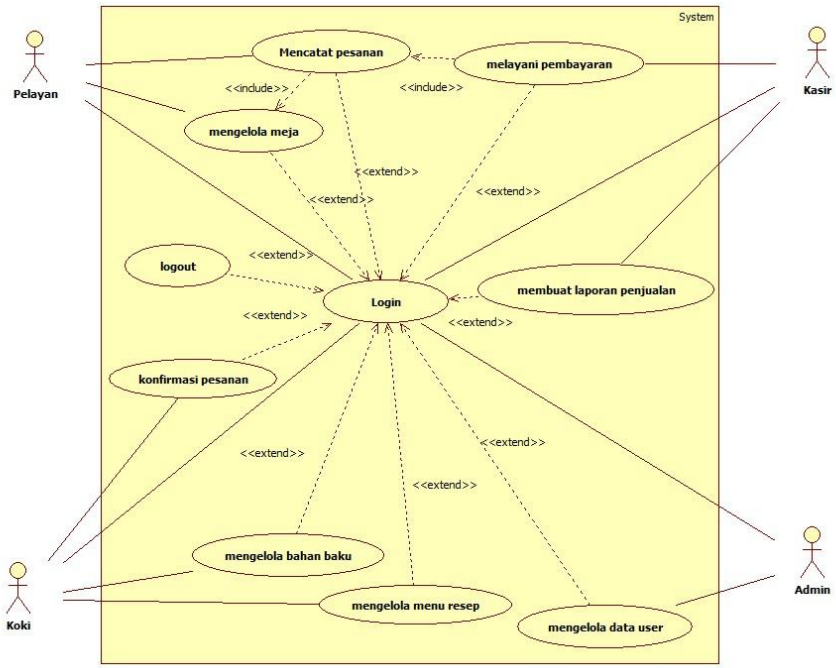
Gambar 2. 1 *Unified Modeling Language*

UML menyediakan beberapa macam diagram untuk memodelkan aplikasi berorientasi objek. Adapun beberapa diagram yang sering digunakan adalah sebagai berikut:

1. *Use Case*

Use case diagram digunakan untuk memodelkan proses bisnis berdasarkan perspektif pengguna sistem. *Use case diagram* terdiri atas diagram untuk *use case* dan *actor*. *Actor* merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi.

Use case merepresentasikan operasi-operasi yang dilakukan oleh *actor*. *Use case* digambarkan berbentuk elips dengan nama operasi dituliskan di dalamnya. *Actor* yang melakukan operasi dihubungkan dengan garis lurus ke use case.



Gambar 2. 2 Use Case Diagram

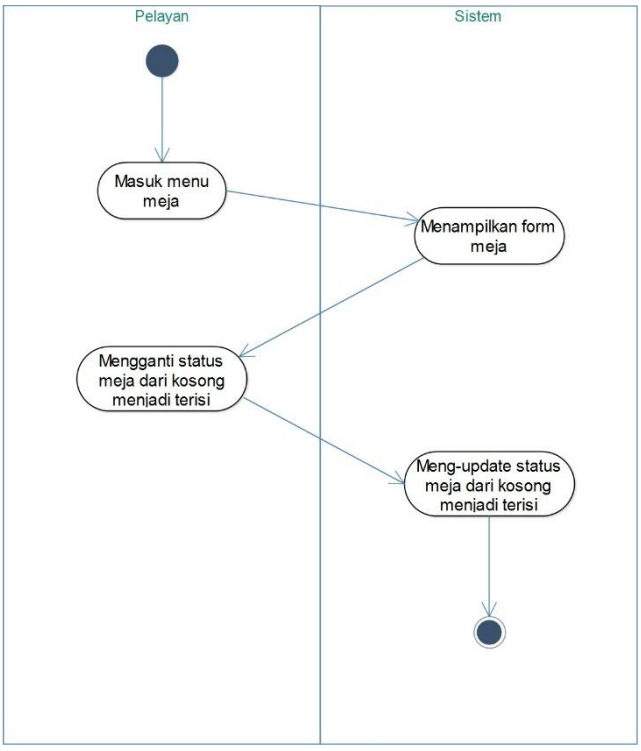
Pada gambar 2.2 bisa dilihat bahwa ada 4 user dan 10 *use case* dimana seluruh user memiliki hak akses untuk dapat menjalankan fungsional dari sistem yang dibangun.

2. *Activity Diagram*

Diagram aktivitas atau dalam bahasa inggris *activity diagram* adalah representasi grafis dari seluruh tahapan alur kerja. Diagram ini mengandung aktivitas, pilihan tindakan, perulangan dan hasil dari aktivitas tersebut. Pada

pemodelan UML, diagram ini dapat digunakan untuk menjelaskan proses bisnis dan alur kerja operasional secara langkah demi langkah dari komponen suatu sistem.

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan state diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu activity diagram tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Struktur diagram ini mirip *flowchart* atau *Data Flow Diagram* (DFD) pada perancangan terstruktur. Sangat bermanfaat apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan. Activity diagram dibuat berdasarkan sebuah atau beberapa *use case* pada *use case diagram*.

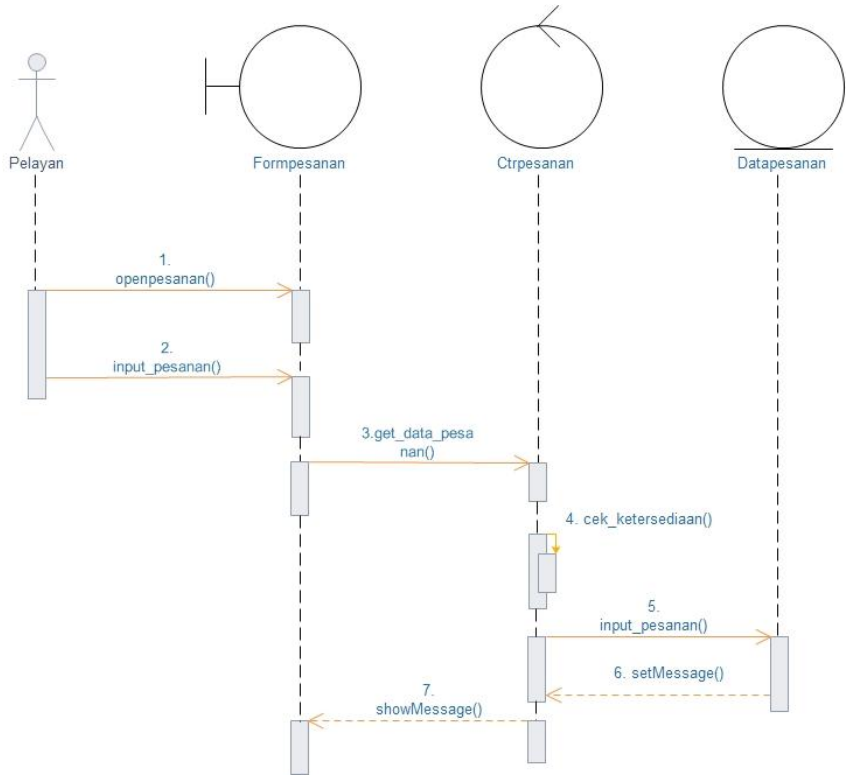


Gambar 2. 3 *Activity Diagram*

Dari 2.3 menjelasakan diagram *activity* sebuah sistem informasi restoran untuk mengelola meja. Terdapat 4 *activity* didalamnya mulai dari pelayan memasuki menu meja, sistem menampilkan form meja, kemudian mengganti status meja dari kosong menjadi terisi, dan sistem melakukan update status meja.

3. Sequence Diagram

Diagram sequence merupakan salah satu diagram interaksi (*interaction diagram*) yang menjelaskan bagaimana suatu operasi itu dilakukan dan *message* (pesan) apa yang dikirim dan kapan pelaksanaannya [15]. Diagram ini diatur berdasarkan waktu. Obyek-obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut [14].



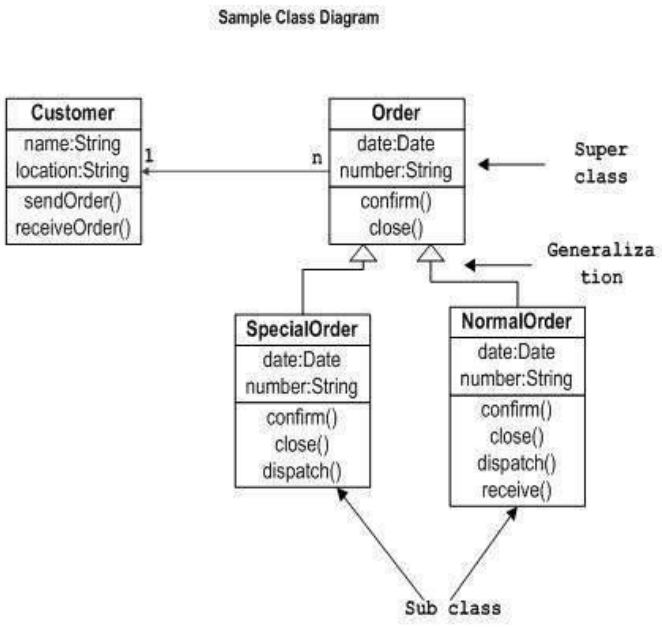
Gambar 2. 4 *Sequence Diagram*

Dari gambar 2.4 menggambarkan sebuah diagram sequence pada sistem informasi restoran mengenai bagaimana pelayan menginputkan pesanan kedalam sistem. Terdapat pelayan sebagai aktor melakukan interaksi input pesanan kedalam sistem melalui form pesanan kemudian inut akan diterima oleh sistem dan control

pesanan mengecek ketersediaan data apabila tersedia dan selanjutnya selanjutnya pesanan dimasukkan ke database dan database memberikan *feedback* berupa pesan terhadap user.

4. Class Diagram

Class diagram menggambarkan struktur statis *class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara: *associated* (terhubung satu sama lain), *dependent* (satu class tergantung/menggunakan class yang lain), *specialed* (satu class merupakan spesialisasi dari class lainnya), atau *package.*



Gambar 2. 5 *Class Diagram*

2.9. Bahasa Pemrograman C#

C# (dibaca: *C Sharp*) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka *.NET Framework.* Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat

pada bahasa-bahasa pemrograman lainnya seperti *Java, Delphi, Visual Basic*, dan lain-lain) dengan beberapa penyederhanaan.

Bahasa C# juga telah di standarisasi secara internasional oleh ECMA. Seperti halnya bahasa pemrograman yang lain, C# bisa digunakan untuk membangun berbagai macam jenis aplikasi, seperti aplikasi berbasis *Windows* (desktop) dan aplikasi berbasis web serta aplikasi berbasis *web services*.

2.10. Microsoft Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi *Windows*, ataupun aplikasi Web. Visual Studio mencakup *compiler*, SDK, *Integrated Development Environment* (IDE), dan dokumentasi (umumnya berupa *MSDN Library*).

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed code* (dalam bentuk *Microsoft Intermediate Language* di atas *.NET Framework*). Selain itu, *Visual Studio* juga dapat digunakan untuk mengembangkan aplikasi *Silverlight*, aplikasi *Windows Mobile* (yang berjalan di atas *.NET Compact Framework*).

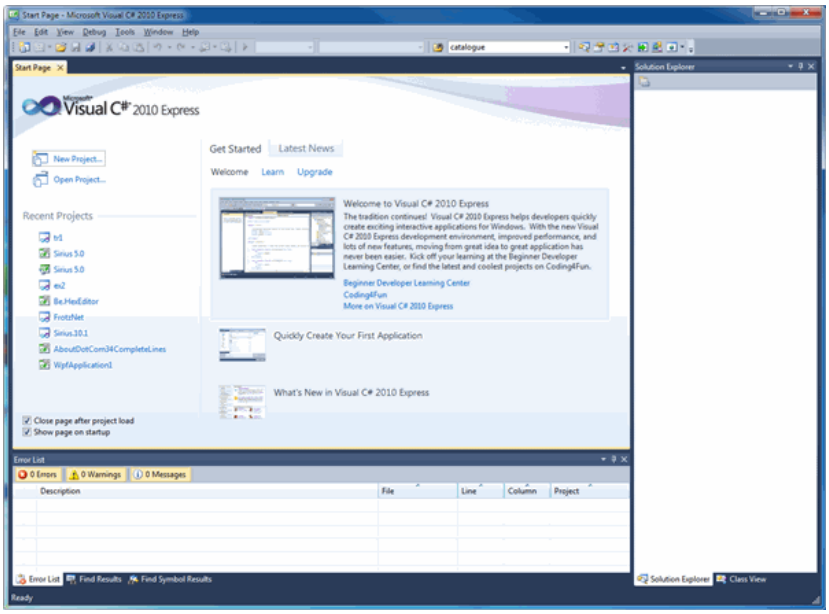


Gambar 2. 6 *Microsoft Visual Studio*

2.6.1. Microsoft Visual C#

Microsoft Visual C# adalah sebuah program alat bantu pemrograman yang dibuat oleh Microsoft Corporation dan dapat digunakan untuk membuat program berbasis grafis dengan menggunakan bahasa pemrograman mirip *C++*. Program ini telah dimasukkan ke dalam produk *Microsoft Visual Studio*, bersama-sama dengan *Visual C++, Visual Basic, Visual FoxPro serta Visual J#.* Sejauh ini, program ini merupakan program yang paling banyak digunakan oleh para programmer untuk membuat program dalam bahasa *C#*.

Microsoft Visual C# adalah sebuah bahasa pemrograman yang bersifat *OOP* (*Object Oriented Programing*) yang dikembangakan oleh *Microsoft Corporation* guna menggantikan *Visual J#* yang terhalah lisensi dengan pemilik *Java* pada saat itu *Sun Microsystem*, yang akhirnya membuat *Microsoft* untuk merancang sebuah bahasa pemrograman baru yang lebih dinamis.



Gambar 2. 7 *Microsoft Visual C#*

2.11. Skenario Pengujian

Pengujian dilakukan terhadap pertanyaan yang telah dibangkitkan oleh sistem melalui kuisioner yang diberikan kepada 10 orang Guru IPA tingkat SD. Pertanyaan yang dinyatakan dapat diterima harus memenuhi salah satu ataupun

sebagian dari kategori yang telah ditentukan. Kategori yang dapat menyatakan sebuah pertanyaan dapat diterima atau tidak, jika pertanyaan tersebut memenuhi yaitu:

1. Pertanyaan yang dibangkitkan sistem dituliskan dengan benar.
2. Maksud dari pertanyaan yang dibangkitkan sistem dapat dimengerti.
3. Pertanyaan yang dibangkitkan sistem sesuai dengan konteks.
4. Jika pembaca membuat pertanyaan dari teks yang diberikan, pertanyaan yang pembaca buat sama seperti pertanyaan yang telah dibangkitkan sistem.
5. Pertanyaan yang dibangkitkan sistem merupakan pertanyaan yang bermanfaat.