II. TINJAUAN PUSTAKA

2.1. Kanji

Kanji (漢字), secara harfiah berarti "aksara dari Han", adalah aksara Tionghoa yang digunakan dalam bahasa Jepang. Kanji adalah salah satu dari empat set aksara yang digunakan dalam tulisan modern Jepang selain kana (katakana, hiragana) dan romaji.

Kanji dulunya juga disebut *mana* (真名) atau *shinji* (真字) untuk membedakannya dari kana. Aksara kanji dipakai untuk melambangkan konsep atau ide (kata benda, akar kata kerja, akar kata sifat, dan kata keterangan). Sementara itu, hiragana (zaman dulu katakana) umumnya dipakai sebagai *okurigana* untuk menuliskan infleksi kata kerja dan kata-kata yang akar katanya ditulis dengan kanji, atau kata-kata asli bahasa Jepang. Selain itu, hiragana dipakai menulis kata-kata yang sulit ditulis dan diingat bila ditulis dalam aksara kanji. Kecuali kata serapan, aksara kanji dipakai untuk menulis hampir semua kosakata yang berasal dari bahasa Tionghoa maupun bahasa Jepang.

2.1.1 Sejarah

Secara resmi, aksara Tionghoa pertama kali dikenal di Jepang lewat barang-barang yang diimpor dari Tionghoa melalui Semenanjung Korea mulai abad ke-5 Masehi. Sejak itu pula, aksara Tionghoa banyak dipakai untuk menulis di Jepang, termasuk untuk prasasti dari batu dan barang-barang lain.

Sebelumnya di awal abad ke-3 Masehi, dua orang lelaki bernama Achiki dan Wani datang dari Baekje (Korea) di masa pemerintahan Kaisar Ōjin. Keduanya konon menjadi pengajar aksara Tionghoa bagi putra kaisar. Wani membawa buku *Analek* karya Kong Hu Chu dan buku pelajaran menulis aksara Tionghoa untuk anak-anak dengan judul Seribu Karakter Klasik. Walaupun demikian, orang Jepang mungkin sudah mengenal aksara Tionghoa sejak abad ke-1 Masehi. Di Kyushu ditemukan stempel emas bertanda tahun 57 Masehi yang diterima sebagai hadiah dari Tiongkok untuk Kaisar Wa (Jepang).

Dokumen tertua yang ditulis di Jepang menurut perkiraan ditulis oleh keturunan imigran dari Tiongkok. Istana Jepang mempekerjakan keturunan imigran dari Tiongkok bekerja di istana sebagai juru tulis. Mereka menuliskan bahasa Jepang kuno yang disebut *Yamato Kotoba* dalam aksara Tionghoa. Selain itu, mereka juga menuliskan berbagai peristiwa dan kejadian penting.

Sebelum aksara kanji dikenal orang Jepang, bahasa Jepang berkembang tanpa bentuk tertulis. Pada awalnya, dokumen bahasa Jepang ditulis dalam bahasa Tionghoa, dan dilafalkan menurut cara membaca bahasa Tionghoa. Sistem *kanbun* (漢文) merupakan cara penulisan bahasa Jepang menurut bahasa Tionghoa yang dilengkapi tanda diakritik. Sewaktu dibaca, tanda diakritik membantu penutur bahasa Jepang mengubah susunan kata-kata, menambah partikel, dan infleksi sesuai aturan tata bahasa Jepang.

Selanjutnya berkembang sistem penulisan *man'yōgana* yang memakai aksara Tionghoa untuk melambangkan bunyi bahasa Jepang. Sistem ini dipakai dalam antologi puisi klasik *Man'yōshū*. Sewaktu menulis *man'yōgana*, aksara Tionghoa ditulis dalam bentuk kursif agar menghemat waktu. Hasilnya adalah hiragana yang merupakan bentuk sederhana dari *man'yōgana*. Hiragana menjadi sistem penulisan yang mudah dikuasai wanita. Kesusastraan zaman Heian diwarnai karya-karya besar sastrawan wanita yang menulis dalam hiragana. Sementara itu, katakana diciptakan oleh biksu yang hanya mengambil sebagian kecil coretan dari sebagian karakter kanji yang dipakai dalam *man'yōgana*.

2.1.2. Cara Pengucapan

Satu aksara kanji bisa memiliki cara membaca yang berbeda-beda. Selain itu tidak jarang, satu bunyi bisa dilambangkan oleh aksara kanji yang berbeda-beda. Aksara kanji memiliki dua cara pengucapan, ucapan Tionghoa (on'yomi) dan ucapan Jepang (kun'yomi).

a. Ucapan Tionghoa (on'yomi)

On'yomi (音読み) atau ucapan Tionghoa adalah cara membaca aksara kanji mengikuti cara membaca orang Tionghoa sewaktu karakter tersebut diperkenalkan di Jepang. Pengucapan karakter kanji menurut bunyi bahasa Tionghoa bergantung kepada zaman ketika karakter tersebut diperkenalkan di Jepang. Akibatnya, sebagian besar karakter kanji memiliki lebih dari satu *on'yomi*. Kanji juga dikenal orang Jepang secara bertahap dan tidak langsung dilakukan pembakuan.

On'yomi dibagi menjadi 4 jenis:

i. ***Go-on*** (**呉音**, "ucapan Wu") adalah cara pengucapan dari daerah Wu di bagian selatan zaman Enam Dinasti Tiongkok (420-589). Walaupun tidak pernah ditemukan bukti-bukti, ucapan Wu diperkirakan dibawa masuk ke Jepang melalui Semenanjung Korea dari abad ke-5 hingga abad ke-6. Ucapan Wu diperkirakan berasal dari cara membaca literatur agama Buddha yang diwariskan secara turun temurun sebelum diketahui cara membaca *Kan-on* (ucapan Han). Semuanya cara pengucapan sebelum *Kan-on* digolongkan sebagai *Go-on* walaupun mungkin saja berbeda zaman dan asal-usulnya bukan dari daerah Wu.

ii. ***Kan-on*** (**漢音**, "ucapan Han") adalah cara pengucapan seperti dipelajari dari zaman Nara (710-794) hingga zaman Heian (794-1185) oleh utusan Jepang ke Dinasti Tang dan biksu yang belajar ke Tiongkok. Secara

khusus, cara pengucapan yang ditiru adalah cara pengucapan orang Chang'an.

iii. ***Tō-on*** (唐音, "ucapan Tang") adalah cara pengucapan karakter seperti dipelajari oleh biksu Zen antara zaman Kamakura (1185-1333) dan zaman Muromachi (1338-1573) yang belajar ke Dinasti Song, dan perdagangan dengan Tiongkok.

iv. ***Kan'yō-on*** (慣用音, "ucapan populer") adalah cara pengucapan *on'yomi* yang salah (tidak ada dalam bahasa Tionghoa), tapi telah diterima sebagai kelaziman.

Dari keterangan di atas, dapat digambarkan dalam sebuah tabel seperti pada tabel 2.1 berikut ini.

Tabel 2.1. Cara baca On

Kanji	**Arti**	**Go-on**	**Kan-on**	**Tō-on**	**Kan'yō-on**
明	terang	*Myō* (明星 myōjō)	*mei* (明暗 meian)	(*min*)* (明国 minkoku)	—
行	Pergi	*gyō* (行列 gyōretsu)	*kō* (行動 kōdō)	(*an*)* (行灯 andon)	—
京	ibu kota	*kyō* (京都 Kyōto)	*kei* (京阪 Keihan)	*kin* (南京 Nankin)	—
青	biru, hijau	*shō* (緑青 rokushō)	*sei* (青春 seishun)	*chin* (青島 Chintao)	-
清	murni	*shō* (清浄 shōjō)	*sei* (清潔 seiketsu)	(*shin*)* (清国 Shinkoku)	—
輸	mengirim	(*shu*)*	(*shu*)*	—	*yu* (運輸 un-yu)

眠	Tidur	(*men*)*	(*ben*)*	—	*min* (睡眠 suimin)

**Ucapan yang tidak umum*

b. Ucapan Jepang (kun'yomi)

Kun'yomi (訓読み) atau **ucapan Jepang** adalah cara pengucapan kata asli bahasa Jepang untuk karakter kanji yang artinya sama atau paling mendekati. Kanji tidak diucapkan menurut pengucapan orang Tionghoa, melainkan menurut pengucapan orang Jepang. Bila karakter kanji dipakai untuk menuliskan kata asli bahasa Jepang, okurigana sering perlu ditulis mengikuti karakter tersebut.

Seperti halnya, *on'yomi* sebuah karakter kadang-kadang memiliki beberapa *kun'yomi* yang bisa dibedakan berdasarkan konteks dan okurigana yang mengikutinya. Beberapa karakter yang berbeda-beda sering juga memiliki kun'yomi yang sama, namun artinya berbeda-beda. Selain itu, tidak semua karakter memiliki *kun'yomi*.

Kata "*kun*" dalam *kun'yomi* berasal kata "*kunko*" (訓詁) (pinyin: xungu) yang berarti penafsiran kata demi kata dari bahasa kuno atau dialek dengan bahasa modern. Aksara Tionghoa adalah aksara asing bagi orang Jepang, sehingga *kunko* berarti penerjemahan aksara Tionghoa ke dalam bahasa Jepang. Arti kanji dalam bahasa Tionghoa dicarikan padanannya dengan kosakata asli bahasa Jepang.

Sebagai aksara asing, aksara Tionghoa tidak dapat diterjemahkan semuanya ke dalam bahasa Jepang. Akibatnya, sebuah karakter kanji mulanya dipakai untuk melambangkan beberapa *kun'yomi*. Pada masa itu, orang Jepang mulai sering membaca tulisan bahasa Tionghoa (*kanbun*) dengan cara membaca bahasa Jepang. Sebagai usaha membakukan cara membaca kanji, satu karakter ditetapkan hanya memiliki satu cara pengucapan Jepang (*kun'yomi*). Pembakuan ini merupakan dasar bagi tulisan campuran Jepang dan Tiongkok (*wa-kan konkōbun*) yang merupakan cikal bakal bahasa Jepang modern.

c. Jūbakoyomi dan yutōyomi

Gabungan dua karakter sering tidak mengikuti cara membaca *on'yomi* dan *kun'yomi* melainkan campuran keduanya yang disebut *jūbakoyomi* (**重箱読み**). Karakter pertama dibaca menurut *on'yomi* dan karakter kedua menurut *kun'yomi*, misalnya:

1. 重箱 (*jūbako*)
2. 音読み (*on'yomi*)
3. 台所 (*daidokoro*)
4. 役場 (*yakuba*)
5. 試合 (*shiai*)
6. 団子 (*dango*).

Sebaliknya dalam *yutōyomi* (湯桶読み), karakter pertama dibaca menurut *kun'yomi* dan karakter kedua menurut *on'yomi*, misalnya:

1. 湯桶 (*yutō*)
2. 合図 (*aizu*)
3. 雨具 (*amagu*)
4. 手帳 (*techō*)
5. 鶏肉 (*toriniku*) [15].

2.2. JLPT

Japanese Language Proficiency Test JLPT (日本語能力試験 *Nihongo Nōryoku Shiken*) adalah salah satu ujian kemampuan berbahasa Jepang yang bergengsi dikhususkan bagi para penutur asing bahasa Jepang. Situs web JLPT dapat diakses dengan alamat www.jlpt.jp.

JLPT dilangsungkan pertama kali pada tahun 1984 dengan jumlah peserta sebanyak 7.000 orang. Ujian ini diselenggarakan pada bulan Desember tiap tahun dengan empat tingkat kesulitan (1, 2, 3, 4). Akan tetapi, sejak tahun 2010, sistem tersebut diubah menjadi lima tingkat (N1, N2, N3, N4, N5) dan diselenggarakan dua kali dalam satu tahun (khusus untuk tingkat N1, N2, dan N3), yaitu pada bulan Juli (semua) dan Desember (N1, N2, N3).

Ujian ini terbagi menjadi tiga bagian, seperti yang terlihat pada tabel 2.2

Tabel 2.2. Tabel ujian JLPT

Ujian	Keterangan
文字･語彙 (*moji, goi*)	Menguji kemampuan peserta dalam memahami penggunaan kosa kata serta perbendaharaan kata dalam bahasa Jepang.
聴解 (*chōkai*)	Menguji kemampuan peserta dalam mendengar dan memahami dialog dalam bahasa Jepang.
読解･文法 (*dokkai, bunpō*)	Menguji kemampuan peserta dalam memahami artikel dalam bahasa Jepang.

Sistem Penilaian

Sebelum tahun 2010, sistem penilaian JLPT adalah menjumlahkan nilai dari masing-masing bagian dan harus mencapai batas nilai tertentu (standar nilai masing-masing tingkat bervariasi). Sejak tahun 2010 sistem penilaian ini diubah, setiap bagian memiliki standar nilai sendiri sehingga untuk lulus para peserta harus mampu mencapai nilai tertentu untuk masing-masing bagian (tidak berlaku sistem penilaian kumulatif seperti sebelumnya).

Penilaian dilakukan dengan menjumlahkan nilai masing-masing jawaban yang benar. Jawaban yang salah atau tidak diisi tidak akan mengurangi jumlah nilai peserta.

Syarat Tingkat Kompetensi

Syarat kompetensi yang disarankan untuk dimiliki oleh para peserta ujian adalah sebagai berikut :

N1

Peserta mampu memahami bahasa Jepang dalam berbagai situasi berikut :

1. Peserta mampu membaca artikel nonfiksi dengan tingkat kesulitan yang tinggi dan memahami bacaan dalam berbagai tema, seperti tajuk rencana surat kabar, kolom saran dan kritik pada majalah, serta mampu memahami struktur dan isi tiap artikel.
2. Peserta mampu membaca artikel fiksi dengan tema yang beraneka ragam dan mampu memahami dialog serta tujuan penulis yang tertera dalam artikel tersebut secara mendalam.
3. Peserta mampu memahami data lisan dengan tingkat kesulitan yang tinggi seperti siaran berita serta berbagai dialog yang disampaikan dalam tingkat kecepatan tinggi.
4. Menguasai sekitar 2.000 kanji dan hafal, serta mengerti penggunaan 10.000 kata dalam bahasa Jepang
5. Diperuntukkan bagi mereka yang telah mempelajari bahasa Jepang selama kurang lebih 900 jam efektif

N2

Peserta mampu memahami bahasa Jepang yang digunakan pada situasi umum dan kondisi-kondisi tertentu :

1. Peserta mampu membaca artikel nonfiksi dengan tingkat kesulitan yang cukup tinggi dan memahami bacaan sederhana dalam berbagai tema,

seperti surat pembaca dalam surat kabar dan majalah; serta mampu memahami struktur dan isi tiap artikel.

2. Peserta mampu membaca artikel fiksi sederhana dengan tema yang beraneka ragam dan mampu memahami dialog serta tujuan penulis yang tertera dalam artikel tersebut secara mendalam.
3. Peserta mampu memahami data lisan dengan tingkat kesulitan yang cukup tinggi seperti siaran berita serta berbagai dialog
4. Menguasai sekitar 1.000 kanji dan 6.000 kata-kata
5. Level ini biasa dicapai oleh mereka yang telah mempelajari bahasa Jepang selama 600 jam

N3

Peserta mampu memahami bahasa Jepang yang digunakan pada situasi umum dan kondisi-kondisi tertentu :

1. Peserta mampu membaca artikel sederhana dengan tema yang tidak kompleks.
2. Peserta mampu membaca dan memahami ringkasan informasi artikel, seperti judul artikel surat kabar.
3. Peserta mampu membaca artikel fiksi sederhana dengan tema yang beraneka ragam dan mampu memahami dialog serta tujuan penulis yang tertera dalam artikel tersebut secara mendalam.
4. Peserta mampu memahami data lisan dengan tingkat kesulitan yang cukup tinggi serta berbagai dialog yang disampaikan dalam tingkat kecepatan cukup tinggi.

N4

1. bisa memahami percakapan (kaiwa) standar dalam kehidupan sehari-hari, memahami kosa kata dan pola kalimat standar.
2. Menguasai sekitar 300 kanji dan 1.500 kata,
3. N4 biasa dicapai oleh mereka yang telah mempelajari bahasa Jepang selama 300 jam.

N5

1. Bisa memahami percakapan (*kaiwa*) standar dalam kehidupan sehari-hari, memahami kosa kata standar.
2. Menguasai sekitar 100 kanji dan 800 kata dalam bahasa Jepang
3. N5 merupakan level dasar dan diperuntukkan bagi mereka yang telah mempelajari bahasa Jepang selama 150 jam [14].

2.3. Perangkat Lunak

Software adalah perintah (program komputer) yang dieksekusi memberikan fungsi dan petunjuk kerja seperti yang diinginkan. Struktur data yang memungkinkan program memanipulasi informasi secara proporsional dan dokumen yang menggambarkan operasi dan kegunaan program.

Software memiliki dua peran, di satu sisi berfungsi sebagai sebuah produk dan di sisi lain sebagai pengontrol pembuatan sebuah produk. Sebagai

produk, *software* mengantarkan potensi perhitungan yang dibangun oleh *software* komputer. *Software* merupakan *transformer* informasi yang memproduksi, mengatur, memperoleh, memodifikasi, menampilkan atau memancarkan informasi. Hal ini dapat sesederhana bit tunggal atau sekompleks sebuah simulasi multimedia. Sedangkan untuk peran sebagai pengontrol yang dipakai untuk mengantarkan produk, *software* berlaku sebagai dasar untuk kontrol komputer (sistem operasi), komunikasi informasi (jaringan), dan penciptaan serta kontrol dari program-program lain (peranti dan lingkungan *software*) [11].

2.4. Visual Basic

Microsoft Visual Basic (sering disingkat sebagai **VB** saja) merupakan sebuah bahasa pemrograman yang menawarkan *Integrated Development Environment* (IDE) visual untuk membuat program perangkat lunak berbasis sistem operasi Microsoft Windows dengan menggunakan model pemrograman (COM).

Visual Basic merupakan turunan bahasa pemrograman BASIC dan menawarkan pengembangan perangkat lunak komputer berbasis grafik dengan cepat.

Beberapa bahasa skrip seperti Visual Basic for Applications (VBA) dan Visual Basic Scripting Edition (VBScript), mirip seperti halnya Visual Basic, tetapi cara kerjanya yang berbeda.

Para programmer dapat membangun aplikasi dengan menggunakan komponen-komponen yang disediakan oleh Microsoft Visual Basic Program-program yang ditulis dengan Visual Basic juga dapat menggunakan Windows API, tapi membutuhkan deklarasi fungsi luar tambahan.

Dalam pemrograman untuk bisnis, Visual Basic memiliki pangsa pasar yang sangat luas.[1] Sebuah survey yang dilakukan pada tahun 2005 menunjukkan bahwa 62% pengembang perangkat lunak dilaporkan menggunakan berbagai bentuk Visual Basic, yang diikuti oleh C++, JavaScript, C#, dan Java.[16]

2.5. *Database*

Database di dalamnya tersimpan data dalam jumlah besar, misalnya data penduduk dalam suatu negara. *Database* mengandung obyek-obyek yang digunakan untuk mewakili, menyimpan, dan mengakses data dengan mudah. Tanpa *database,* akan membuat sebuah program tidak akan berjalan karena seluruh perintah pasti akan mengambil *resource* data yang ada dalam *database*.

Database Management System (DBMS) adalah suatu sistem atau perangkat lunak yang dirancang untuk mengelola suatu basis data dan menjalankan operasi terhadap data yang diminta pengguna.

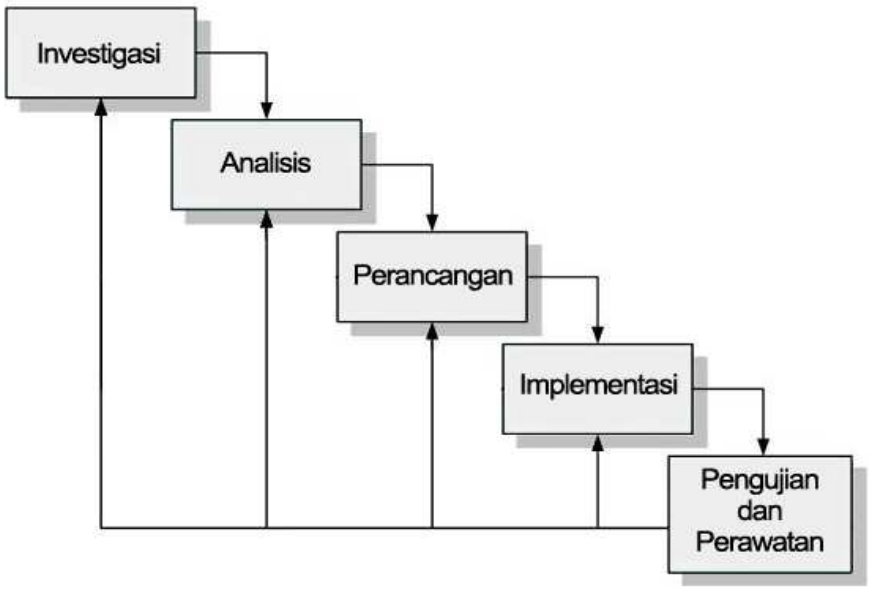
Sedangkan pengertian dari *database* adalah sekumpulan *file-file* yang saling berhubungan salu sama lain atau beberapa kunci penghubung, tersimpan dalam media penyimpanan di luar memori komputer. Media simpan ini dapat

berupa *hard disk. Database* dapat dinyatakan sebagai suatu sistem yang memiliki karakteristik, antara lain :

a. Merupakan suatu kumpulan *"interrelated data"* yang disimpan bersama tanpa mengganggu satu sama lain atau membentuk kerangkapan data.
b. Kumpulan data dalam *database* dapat digunakan oleh sebuah program aplikasi dengan lebih optimal.
c. Penambahan data baru, modifikasi dan pengambilan kembali dari data dapat dilakukan dengan mudah dan terkontrol.[7]

2.6. Pengembangan Perangkat Lunak dengan Model *Modified Waterfall*

Modified waterfall adalah sebuah metode pengembangan *software* yang bersifat sekuensial dan terdiri atas 6 tahap yang saling terkait dan mempengaruhi seperti terlihat pada gambar berikut.



Gambar 2.1. Model *Modified Waterfal.* (Sommerville, 2001)

Pada gambar 2.1 adalah model *modified waterfall* dalam rekayasa perangkat lunak yang muncul karena ketidaksempurnaan pada motode *waterfall* model tradisional. Fase pada model *modified waterfall* ini mirip dengan model tradisional, perubahan utama yang terlihat dalam model *modified waterfall* adalah bahwa fase dalam siklus model *modified waterfall* diizinkan untuk tumpang tindih. Siklus yang tumpang tindih membuat model *modified waterfall* lebih fleksibel dalam rekayasa perangkat lunak.

Keuntungan lain dari model *modified waterfall* adalah bahwa pendekatan yang lebih santai untuk prosedur resmi, dokumen dan ulasan yang dapat menghemat laporan yang akan dibuat, selain itu juga model *modified waterfall* sangat cocok untuk pengembang yang bekerja perorangan.

Terdapat keterkaitan dan pengaruh antar tahap ini karena *output* sebuah tahap dalam model *modified waterfall* merupakan *input* bagi tahap berikutnya, dengan demikian ketidaksempurnaan hasil pelaksanaan tahap sebelumnya adalah awal ketidaksempurnaan tahap berikutnya (Sommerville, 2001). Memperhatikan karakteristik ini, sangat penting bagi pengembang dan perusahaan untuk secara bersama-sama melakukan analisa kebutuhan dan desain sistem sesempurna mungkin sebelum masuk ke dalam tahap penulisan kode program. Berikut adalah penjelasan detail dari masing-masing tahap dalam model *modified waterfall*.

2.6.1. Investigasi

Investigasi awal akan menentukan kebutuhan dan informasi apa saja yang diperlukan bagi sistem informasi yang baru, mendefinisikan masalah, dan memberikan sistem baru yang lebih baik. Investigasi dilakukan dengan cara mengumpulkan data dengan terjun langsung ke lapangan menggunakan teknik observasi, *interview, questionnaire* atau melihat pada dokumen-dokumen yang telah lalu. Hasil dari investigasi akan menjadi *input* dari fase analisis, karena data yang telah didapat pada fase investigasi akan menjadi bahan dasar untuk fase analisis.

2.6.2. Analisis Kebutuhan

Analisis sistem adalah sebuah teknik pemecahan masalah yang menguraikan sebuah sistem menjadi komponen-komponennya dengan tujuan mempelajari seberapa bagus komponen-komponen tersebut bekerja dan berinteraksi untuk meraih tujuan. Analisis kebutuhan juga akan lebih diperjelas pada SRS (*Software Requirement Specifications*) yang akan dibuat. *Software Requirement Specifications* (SRS) adalah dokumen yang menjelaskan tentang berbagai kebutuhan yang harus dipenuhi oleh suatu *software*.

Analisis adalah bagian terpenting dari proses rekayasa perangkat lunak. Karena semua proses lanjutan akan sangat bergantung pada baik tidaknya hasil analisis. Ada satu bagian penting yang biasanya dilakukan dalam tahapan analisis yaitu pemodelan proses bisnis. Model proses adalah model yang memfokuskan pada seluruh proses di dalam sistem yang mentransformasikan data menjadi informasi. Model proses juga

menunjukkan aliran data yang masuk dan keluar pada suatu proses. Biasanya model ini digambarkan dalam bentuk Diagram Alir dan akan lebih diperjelas pada Diagram Arus Data (*Data Flow Diagram* / DFD). Diagram alir merupakan salah satu penyajian algoritma sedangkan DFD meyajikan gambaran apa yang manusia, proses dan prosedur lakukan untuk mentransformasi data menjadi informasi. Hasil dari fase analisis ini akan menjadi *input* pada fase perancangan. DFD yang telah dihasilkan dari fase analisis akan menjadi acuan pada fase perancangan dalam merancang *database*. Ada pula STD (*State Transition Diagram*) adalah diagram yang memodelkan tingkah laku (*behaviour*) sistem berdasarkan pada definisi satu bagian dari keadaan sistem. STD sering dipakai untuk menggambarkan kinerja sistem. Dalam mendefinisikan data yang mengalir di sistem dengan lengkap, dibutuhkan DD (Data Dictionary). DD atau kamus data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi.

a. SRS (*Software Requirement Specifications*)

Software Requirement Specifications (SRS) adalah dokumen yang menjelaskan tentang berbagai kebutuhan yang harus dipenuhi oleh suatu *software*. Dokumen ini dibuat oleh developer (pembuat software) setelah menggali informasi dari calon pemakai *software*. Pembuatannya pun seharusnya mengikuti standar yang ada dan paling diakui oleh para praktisi rekayasa *software* di dunia. Oleh karena itu, standar yang akan dibahas di sini adalah standar dari IEEE.

IEEE membuat standar SRS agar dokumen penting itu tidak ambigu dan tentu saja komplit. Lengkap. Dengan standar itu, si penggguna dapat mencurahkan semua keinginannya terkait *software* tersebut dengan jelas dan akurat sehingga sang *developer* pun dapat memahami apa yang diinginkan pengguna dengan tepat. Bahkan, bagi perorangan, standar ini dapat membantunya dalam mengembangkan outline SRS yang baku khusus untuk perusahaannya, membantunya membuat dokumen SRS dengan format dan isi yang standar (minimal), serta membantunya mengembangkan rincian-rincian pendukung lainnya.

SRS yang baik akan bermanfaat bagi *customer*, *supplier*, ataupun perorangan. Manfaat-manfaat tersebut antara lain:

1. Sebagai bentuk perjanjian antara customer dan supplier tentang *software* apa yang akan dibuat
2. Mengurangi beban dalam proses pengembangan *software*
3. Sebagai bahan perkiraan biaya dan rencana penjadwalan
4. Sebagai dasar validasi dan verifikasi *software* di ujung penyelesaian proyek nantinya
5. Memfasilitasi transfer, semisal *software* tersebut ingin ditransfer ke pengguna atau mesin-mesin yang lain. Customer pun merasa mudah jika ingin mentransfer *software* ke bagian-bagian lain dalam organisasinya. Bahkan, jika terjadi pergantian personil *developer*,

proyek dapat mudah ditransfer ke personil baru dengan memahami SRS ini.

6. Mendasari perbaikan produk *software* di kemudian hari. Jadi, kadang SRS boleh diperbaiki dengan alasan dan mekanisme tertentu serta atas kesepakatan antara customer dan developer [17].

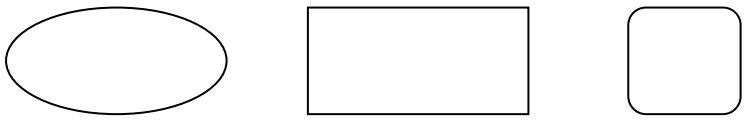
b. DFD (*Data Flow Diagram*)

DFD adalah suatu gambaran *drafts* dari suatu sistem yang menggunakan sejumlah bentuk-bentuk simbol untuk menggambarkan bagaimana data mengalir melalui suatu proses yang saling berkaitan. DFD digunakan untuk menyatakan aliran data mulai dari *external entity* sampai dengan penyimpanan data dalam suatu bentuk *database.* Ada empat simbol utama yang dikenal dalam DFD yaitu:

1. Simbol *Entity,* gambar 2.2 menunjukkan asal/tujuan dari data di sistem. Bisa juga dikatakan *input/output.*

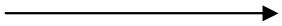
Gambar 2.2. Simbol *Entity*

2. Simbol Proses, gambar 2.3 menunjukkan pemrosesan data yang masuk kearahnya dan mengeluarkan data lainnya.



Gambar 2.3. Simbol Proses

3. Simbol arus data, gambar 2.4 menunjukkan aliran data, dari mana data itu dan kemana tujuannya.



Gambar 2.4. Simbol Arus Data

4. Simbol penyimpanan data, gambar 2.5 menunjukkan suatu termpat penyimpanan data.

Database

Gambar 2.5. Simbol Penyimpanan Data

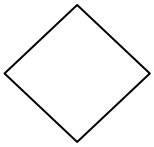
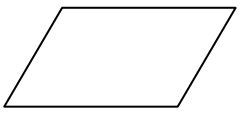
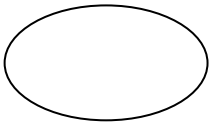
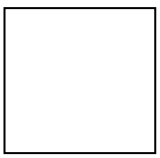
Ada enam panduan yang harus dilakukan dalam merancang DFD :

1. Setiap proses dan *entity* dalam DFD harus memiliki *input* dan *output.*
2. Beri label pada tiap arus data dengan nama yang unik.
3. Jaga agar nama arus data tetap dari satu level ke level berikutnya.
4. Tidak dibedakan antara data dan informasi, semua dianggap data.
5. Hindari garis arus data yang berpotongan, hal ini akan membingungkan seseorang yang membacanya.
6. Setiap *database* hanya boleh menerima *input* dari proses dan juga memberikan *output* ke proses.

c. Diagram Alir

Flowchart atau diagram alir merupakan sebuah diagram dengan simbol-simbol grafis yang menyatakan aliran algoritma atau proses yang menampilkan langkah-langkah yang disimbolkan dalam bentuk kotak, beserta urutannya dengan menghubungkan masing masing langkah tersebut menggunakan tanda panah. Diagram ini bisa memberi solusi selangkah demi selangkah untuk penyelesaian masalah yang ada di dalam proses atau algoritma tersebut [13].

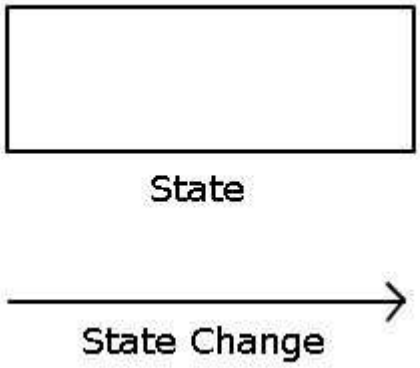
Tabel 2.3. simbol *flowchart* yang umum digunakan

Gambar	Simbol untuk	Keterangan
	Proses / Langkah	Menyatakan kegiatan yang akan ditampilkan dalam diagram alir.
	Titik Keputusan	Proses / Langkah dimana perlu adanya keputusan atau adanya kondisi tertentu. Di titik ini selalu ada dua keluaran untuk melanjutkan aliran kondisi yang berbeda.
	Masukan / Keluaran Data	Digunakan untuk mewakili data masuk, atau data keluar.
	Terminasi	Menunjukkan awal atau akhir sebuah proses.
	Garis alir	Menunjukkan arah aliran proses atau algoritma.
	Kontrol / Inspeksi	Menunjukkan proses / langkah dimana ada inspeksi atau pengontrolan.

d. STD (*State Transition Diagram*)

STD merupakan diagram yang memodelkan tingkah laku (*behaviour*) sistem berdasarkan pada definisi satu bagian dari keadaan sistem. STD sering dipakai untuk menggambarkan kinerja sistem.

Komponen STD dibagi menjadi 4 :



Gambar 2.6. Gambar *state* dan *state change*

1. State

State merupakan kondisi dari suatu sistem. State dapat dikategorikan menjadi 2 macam, yaitu : State Awal dan State Akhir. State Awal hanya boleh berjumlah 1 state, dan State Akhir boleh memiliki jumlah lebih dari satu state.

2. State Change (Tanda Panah)

Menyatakan perubahan state dari sistem.

3. Kondisi

Kondisi menyatakan suatu kejadian pada lingkungan eksternal yang dapat dideteksi oleh sistem, contoh: sinyal.

4. Aksi

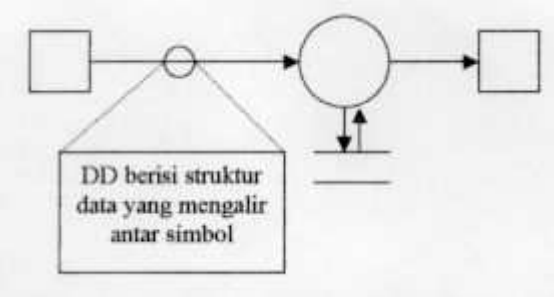
sistem melakukan sesuatu sehingga terjadi perubahan *state* atau merupakan suatu reaksi terhadap kondisi [9].

e. DD (*Data Dictionary*)

Kamus data atau systems data dictionary adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan DD analis sistem dapat mendefinisikan data yang mengalir di sistem dengan lengkap. Pada tahap analisis sistem, DD digunakan sebagai alat komunikasi antara analis sitem dengan pemakai sistem tentang data yang mengalir ke sistem, yaitu tentang data yang masuk ke sistem dan tentang informasi yang dibutuhkan oleh pemakai sistem.

Pada tahap perancangan sistem, DD digunakan untuk merancang input, merancang laporan-laporan dan database. DD dibuat berdasarkan arus data yang ada di DFD (Data Flow Diagram). Arus data di DFD sifatnya adalah global, hanya ditunjukkan nama arus datanya saja. Keterangan lebih lanjut tentang struktur dari suatu arus data di DFD secara lebih terinci dapat dilihat di DD.

Gambar berikut menunjukkan hubungan antara DFD dengan DD.



Gambar 2.7. Hubungan antara Data Flow Diagram (DFD) dengan Data Dictionary (DD).

DD tidak menggunakan notasi grafik sebagaimana halnya DFD. DD berfungsi membantu pelaku sistem untuk mengerti aplikasi secara detil, dan mereorganisasi semua elemen data yang digunakan dalam sistem secara presisi sehingga pemakai dan penganalisa sistem punya dasar pengertian yang sama tentang masukan, keluaran, penyimpanan dan proses. DD mendefinisikan elemen data dengan fungsi sebagai berikut:

1. Menjelaskan arti aliran data dan penyimpanan dalam DFD.

2. Mendeskripsikan komposisi paket data yang bergerak melalui aliran, misalnya alamat diuraikan menjadi kota, kodepos, propinsi, dan negara.

3. Mendeskripsikan komposisi penyimpanan data.

4. Menspesifikasikan nilai dan satuan yang relevan bagi penyimpanan dan aliran.

5. Mendeskripsikan hubungan detil antara penyimpanan yang akan menjadi titik perhatian dalam entity relationship diagram.

3.2 Isi DD

Data dictionary harus dapat mencerminkan keterangan yang jelas tentang data yang dicatatnya. Untuk maksud keperluan ini, maka DD harus memuat hal-hal berikut :

h. Nama arus data.

Karena DD dibuat berdasarkan arus data yang mengalir di DFD, maka nama dari arus data juga harus dicatat di DD, sehingga mereka yang membaca DFD dan memerlukan penjelasan lebih lanjut tentang suatu arus data tertentu di DFD dapat langsung mencarinya dengan mudah di DD.

i. Alias.

Alias atan nama lain dari data dapat dituliskan bila nama lain ini ada. Alias perlu ditulis karena data ayang sama mempunyai nama yang berbeda untuk orang atau departemen satu dengan yang

lainnya, misalnnya bagian pembuat faktur dan langganan menyebut bukti penjualan sebagai faktur, sedang bagian gudang menyebutnya sebagai tembusan permintaan persediaan. Baik faktur dan tembusan permintaan persediaan ini mempunyai struktur data yang sama, tetapi mempunyai struktur yang berbeda.

j. Bentuk data.

Bentuk data perlu dicatat di DD, karena dapat digunakan untuk mengelompokkan DD ke dalam kegunaannya sewaktu perancangan sistem. DD yang mencatat data yang mengalir dalam bentuk dokumen dasar atau formulir akan digunakan untuk merancang bentuk input sistem. DD yang mencatat data yang mengalir dalam bentuk laporan tercetak dan dokumen hasil cetakan komputer akan digunakan untuk merancang output yang akan dihasilkan oleh sistem. DD yang mencatat data yang mengalir dalam bentuk tampilan dilayar monitor akan digunakan untuk merancang tampilan layar yang akan dihasilkan oleh sistem. DD yang mencatat data yang mengalir dalam bentuk parameter dan variabel akan digunakan untuk merancang proses dari program. DD yang mencatat data yang mengalir dalam bentuk dokumen, formulir, laporan, dokumen cetakan komputer, tampilan di layar monitor, variabel dan *field* akan digunakan untuk merancang *database*.

d. Arus data.

Arus data menunjukkan dari mana data mengalir dan ke mana data akan menuju. Keterangan arus data ini perlu dicatat di DD supaya memudahkan mencari arus data ini di DFD.

k. Penjelasan.

Untuk tidak memperjelas lagi tentang makna dari arus data yang dicatat di DD, maka bagian penjelasan dapat diisi dengan keterangan-keterangan tentang arus data tersebut. Sebagai misalnya nama dari arus data adalah tembusan permintaaan persediaan, maka dapat lebih dijelaskan sebagai tembusan dari faktur penjualan untuk meminta barang dari gudang.

l. Periode.

Periode ini menunjukkan kapan terjadinya arus data ini. Periode perlu dicatat di DD karena dapat digunakan untuk mengidentifikasikan kapan input data harus dimasukkan ke sistem, kapan proses dari program harus dilakukakan dan kapan laporan-laporan harus dihasilkan.

m. Volume.

Volume yang perlu dicatat di DD adalah tentang volumen rata-rata dan volume puncak dari arus data. Volume rata-rata menunjukkan banyaknya rata-rata arus data yang mengalir dalam suatu periode tertentu dan volume puncak menunjukkan volume yang terbanyak, Volume ini digunakan untuk mengidentifikasikan besarnya

simpanan luar yang akan digunakan, kapasitas dan jumlah dari alat input, alat pemroses dan alat output.

h. Struktur data.

Struktur data menunjukkan arus data yang dicatat di DD terdiri dari item-item apa saja.

KAMUS DATA

a. Nama arus data : Tembusan permintaan persediaan.
b. Alias : Faktur.
Tembusan jurnal.
Tembusan kredit.
c. Bentuk data : Dokumen cetakan komputer.
d. Arus data : Proses 1.4 - Gudang
Proses 1.5 - Bagian pengiriman
e. Penjelasan : Tembusan dari faktur penjualan untuk meminta barang dari gudang.
f. Periode : Setiap kali terjadi penjualan (harian).
g. Volume : Volume rata-rata tiap hari adalah 100.
Volume puncak adalah 150.
h. Struktur data:Terdiri atas item data :
- o Kode langganan.
- o Nama langganan.
- o Tanggal penjualan.
- o Nomor faktur.
- o Satu sampai dengan maksimum 10 kali :
  - * Kode barang.
  - * Nama barang.
  - * Unit jual.
  - * Harga satuan.
  - * Total harga.
- o Total penjualan.
- o Potongan penjualan.
- o Pajak penjualan.
- o Total dibayar.
- o Jenis penjualan.

Gambar 2.8. Contoh DD

3.3 Simbol DD

Kebanyakan sistem, kadang-kadang elemen data terlalu kompleks untuk didefinisikan. Kekomplekkan tersebut seharusnya diuraikan melalui sejumlah elemen data yang lebih sederhana. Kemudian elemen data yang lebih sederhana tersebut didefinisikan kembali hingga nilai dan satuan relevan dan elementer. Pendefinisian tersebut menggunakan notasi yang umum digunakan dalam menganalisa sistem dengan menggunakan sejumlah simbol, seperti berikut :

Tabel 2.4. Simbol Data Dictionary Sumber (Pengantar Perancangan Sistem, Husni Iskandar Pohan, dkk) [5]

No	Simbol	Uraian
1	=	Terdiri dari, mendefinisikan, diuraikan menjadi, artinya
2	+	Dan
3	()	Opsional (boleh ada atau boleh tidak ada)
4	{}	Pengulangan
5	[ ]	Memilih salah satu dari sejumlah alternatif, seleksi
6	**	Komentar
7	@	Identifikasi atribut kunci

8	|	Pemisah sejumlah alternatif pilihan antara simbol [ ]

2.6.3. Perancangan

Perancangan perangkat lunak adalah tugas, tahapan atau aktivitas yang difokuskan pada spesifikasi rinci dari solusi berbasis komputer. Perancangan perangkat lunak juga bisa diartikan proses di mana analisa diterjemahkan menjadi *blueprint* untuk membangun perangkat lunak. Awalnya, *blueprint* menggambarkan pandangan menyeluruh perangkat lunak. Hasil dari fase perancangan akan menjadi *input* pada fase implementasi. Pada tahap perancangan menghasilkan suatu perancangan *database*, perancangan proses, dan perancangan *user interface*. *Database*, proses dan *user interface* yang telah dirancang pada fase perancangan ini akan diaplikasikan pengkodeannya pada fase implementasi.

a. Perancangan *Database*

Merancang *database* merupakan hal yang sangat penting, karena di sini akan menentukan konsep-konsep apa saja yang akan menjadi *database* dalam suatu sistem, sehingga hasil rancangan tersebut memenuhi kebutuhan anda akan informasi untuk saat ini dan masa yang akan

datang. Pada perancangan *database* disini terfokus pada perancangan *database* secara konsep. Perancangan secara konsep merupakan langkah pertama dalam merancang *database*. Sesuai dengan namanya, pada tahap ini anda hanya menentukan konsep-konsep yang berlaku dalam sistem *database* yang akan dibangun. Pemahaman seorang perancang *database* terhadap sistem yang akan dibangun sangat menentukan baik atau tidaknya hasil rancangan *database*-nya. Dalam tahap ini, setidaknya yang harus diketahui :

1. Prosedur kerja secara keseluruhan yang berlaku pada sistem yang sedang berjalan.
2. Informasi (*output*) apa yang diinginkan dari *database* ?
3. Apa saja kelemahan-kelemahan dari sistem yang sedang berjalan ?
4. Pengembangan sistem di masa yang akan datang.
5. Apa saja *input* yang diperlukan ?

b. Perancangan *User Interface*

Perancangan *user interface* adalah dilakukannya penjabaran komunikasi internal perangkat lunak antara perangkat lunak dengan sistem di luarnya dan antara perangkat lunak dengan *user*-nya.

2.6.4. Implementasi

Implementasi perangkat lunak adalah melaksanakan, eksekusi, atau praktek dari rencana, metode, atau perancangan dalam pengembangan perangkat lunak. Pada tahap ini dilakukan kerja untuk membangun

perangkat lunak berdasarkan analisa dan pemodelan yang telah dilakukan. Sehigga hasil dari tahap ini adalah basis data dan *source code* perangkat lunak. Hasil dari fase implementasi akan menjadi *input* pada fase pengujian dan perawatan.

2.6.5. Pengujian dan Pemeliharaan

a. Pengujian

Setelah *source code* dihasilkan, perangkat lunak harus diuji untuk menemukan sebanyak mungkin kesalahan yang dibuat guna mengetahui terdapat atau tidaknya kesalahan pada sistem yang telah dibuat. Pada umumnya terdapat dua jenis pengujian yaitu, *Black-Box Testing* dan *White-Box Testing.* Perbedaan yang mencolok di antara keduanya adalah pengujinya. *Black-Box* dilakukan oleh pengguna perangkat lunak yang mana hanya memperhatikan input dan outputnya saja. Apabila hasil output telah sesuai dengan input yang diuji, maka perangkat lunak talah lulus uji. Sedangkan *White-Box* testing biasanya dilakukan oleh tim penguji dari pembuat perangkat lunak. Sehingga yang diperhatikan bukan hanya *input* dan *output,* melainkan juga proses yang terjadi yang mengakibatkan perubahan dari input menjadi output.

Salah satu contoh dari pengujian *white-box* adalah *basis-path,* yang memungkinkan desainer *test case* mengukur kompleksitas logis dari

desain procedural dan menggunakannya sebagai pedoman untuk menetapkan basis set dari jalur eksekusi.

b. Pemeliharaan

Pemeliharaan suatu *software* sangat perlu dilakukan, termasuk di dalamnya pengembangan, karena *software* yang dibuat tidak selamanya hanya seperti itu. Ketika dijalankan mungkin saja masih ada *error* kecil yang tidak ditemukan sebelumnya atau bisa disebut *bugs*, atau ada penambahan fitur-fitur yang belum ada pada *software* tersebut. Pengembangan diperlukan ketika adanya perubahan dari eksternal seperti ketika ada pergantian sistem operasi, atau perangkat lainnya. Dalam fase pengujian dan perawatan, jika terdapat kesalahan yang ditemukan pada perangkat lunak yang dibangun, pengembang dapat mengetahui pada fase mana pengembang harus mengulang kegiatannya [8].

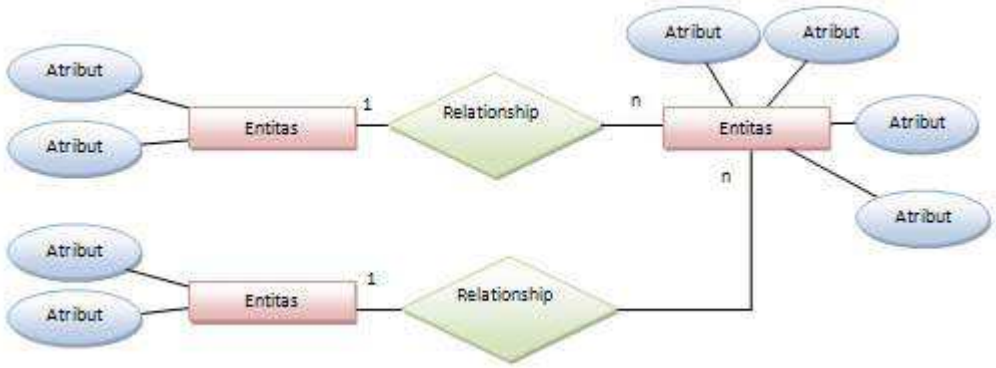
2.7 *Entity Relationship* Diagram (ERD)

Dalam rekayasa perangkat lunak, sebuah *Entity-Relationship Model* (ERM) merupakan abstrak dan konseptual representasi data. *Entity-Relationship* adalah salah satu metode pemodelan basis data yang digunakan untuk menghasilkan skema konseptual untuk jenis/model data

semantik sistem. Di mana sistem seringkali memiliki basis data relasional, dan ketentuannya bersifat *top-down*. Diagram untuk menggambarkan model *Entitiy-Relationship* ini disebut *Entitiy-Relationship diagram*, *ER diagram,* atau *ERD.*

Notasi ERD

Ada sejumlah konvensi mengenai Notasi ERD. Notasi klasik sering digunakan untuk model konseptual. Berbagai notasi lain juga digunakan untuk menggambarkan secara logis dan fisik dari suatu basis data, salah satunya adalah IDEF1X.



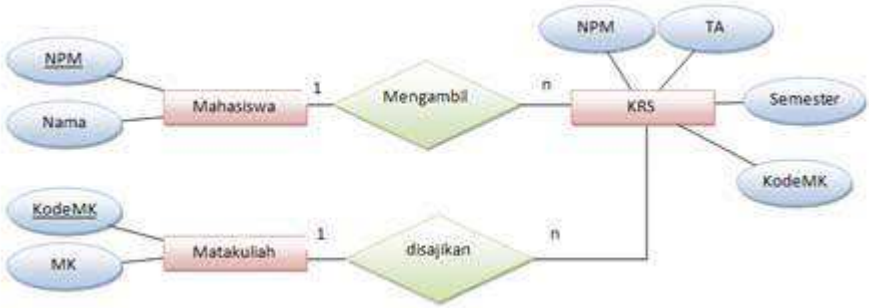
Gambar 2.9. Model ERD

Notasi-notasi simbolik yang digunakan dalam *Entity Relationship Diagram* adalah sebagai berikut :

1. *Entitas*, Adalah segala sesuatu yang dapat digambarkan oleh data. Entitas juga dapat diartikan sebagai individu yang mewakili sesuatu yang nyata

(eksistensinya) dan dapat dibedakan dari sesuatu yang lain (Fathansyah, 1999). Ada dua macam entitas yaitu entitas kuat dan entitas lemah. Entitas kuat merupakan entitas yang tidak memiliki ketergantungan dengan entitas lainnya. Contohnya entitas anggota. Sedangkan entitas lemah merupakan entitas yang kemunculannya tergantung pada keberadaaan entitas lain dalam suatu relasi.

2. ***Atribut***, Atribut merupakan pendeskripsian karakteristik dari entitas. Atribut digambarkan dalam bentuk lingkaran atau elips. Atribut yang menjadi kunci entitas atau key diberi garis bawah.
3. ***Relasi atau Hubungan***, Relasi menunjukkan adanya hubungan diantara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.
4. *Penghubung antara himpunan relasi* dengan himpunan entitas dan himpunan entitas dengan atribut dinyatakan dalam bentuk garis.



Gambar 2.10. Contoh ERD

Derajat relasi atau kardinalitas

Menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Macam-macam kardinalitas adalah:

1. ***Satu ke satu (one to one)***, Setiap anggota entitas A hanya boleh berhubungan dengan satu anggota entitas B, begitu pula sebaliknya.
2. ***Satu ke banyak (one to many)***, Setiap anggota entitas A dapat berhubungan dengan lebih dari satu anggota entitas B tetapi tidak sebaliknya.
3. ***Banyak ke banyak (many to many),*** Setiap entitas A dapat berhubungan dengan banyak entitas himpunan entitas B dan demikian pula sebaliknya.

Tahap ERD

Tahap pertama pada desain sistem informasi menggunakan model ER adalah menggambarkan kebutuhan informasi atau jenis informasi yang akan disimpan dalam *database*. Teknik pemodelan data dapat digunakan untuk menggambarkan setiap ontologi (yaitu gambaran dan klasifikasi dari istilah yang digunakan dan hubungan anatar informasi) untuk wilayah tertentu.

Tahap berikutnya disebut desain logis, di mana data dipetakan ke model data yang logis, seperti model relasional. Model data yang logis ini kemudian dipetakan menjadi model fisik, sehingga kadang-kadang, Tahap kedua ini disebut sebagai “desain fisik”.

Secara umum metodologi ERD sebagai berikut:

Tabel 2.5. Metodologi ERD

1. Menentukan entitas	Menentukan peran, kejadian, lokasi, hal nyata dan konsep dimana penggunaan untuk menyimpan data
2. Menentukan relasi	Menentukan hubungan antar pasangan entitas menggunakan matriks relasi
3. Gambar ERD sementara	Entitas digambarkan dengan kotak, dan relasi digambarkan dengan garis
4. Isi kardinalitas	Menentukan jumlah kejadian satu entitas untuk sebuah kejadian pada entitas yang berhubungan
5. Tentukan kunci utama	Menentukan atribut yang mengidentifikasikan satu dan hanya satu kejadian masing-masing entitas
6. Gambar ERD berdasarkan kunci	Menghilangkan relasi many to many dan memasukkan primary dan kunci tamu pada masing-masing entitas
7. Menentukan atribut	Menentukan field-field yang diperlukan system
8. Pemetaan atribut	Memasangkan atribut dengan entitas yang sesuai
9. Gambar ERD dengan atribut	Mengatur ERD dari langkah 6 dengan menambahkan entitas atau relasi yang ditemukan pada langkah 8
10. Periksa hasil	Apakah ERD sudah menggambarkan system yang akan dibangun?

Contoh Kasus

Sebuah perusahaan mempunyai beberapa bagian. Masing-masing bagian mempunyai pengawas dan setidaknya satu pegawai. Pegawai ditugaskan paling tidak di satu bagian (dapat pula dibeberapa bagian). Paling tidak satu pegawai mendapat tugas di satu proyek. Tetapi seorang pegawai dapat libur dan tidak dapat tugas di proyek.

Menentukan entitas

Entitasnya : pengawas, bagian, pegawai, proyek

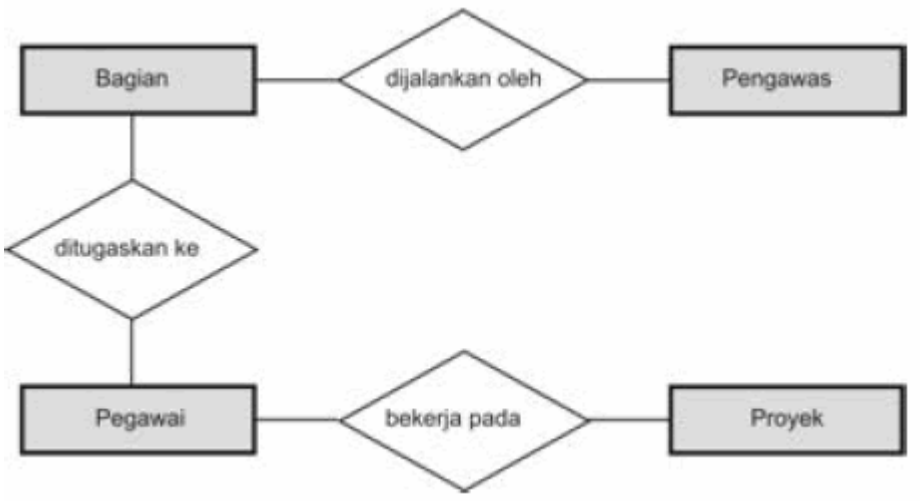
Menentukan relasi dengan matrik relasi

Tabel 2.6. Menentukan Relasi

	Bagian	Pegawai	Pengawas	Proyek
Bagian		ditugaskan ke	dijalankan oleh	
Pegawai	milik			bekerja pada
Pengawas	menjalankan			
Proyek		menggunakan		

Gambar ERD sementara

Hubungkan entitas sesuai dengan matrik relasi yang dibuat

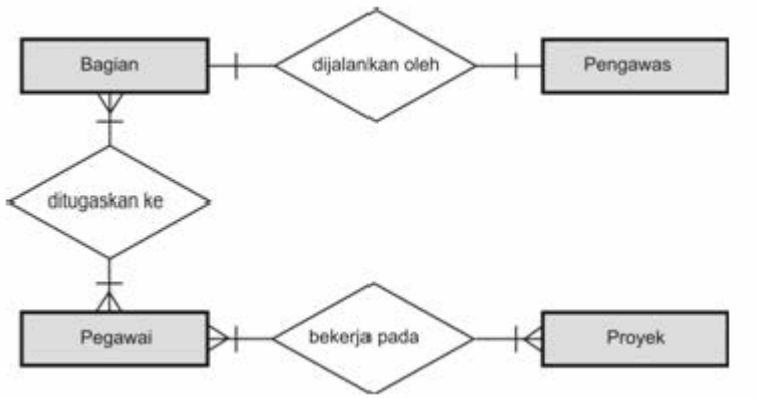


Gambar 2.11.ERD Sementara

Mengisi kardinalitas

Dari gambaran permasalahan dapat diketahui bahwa:

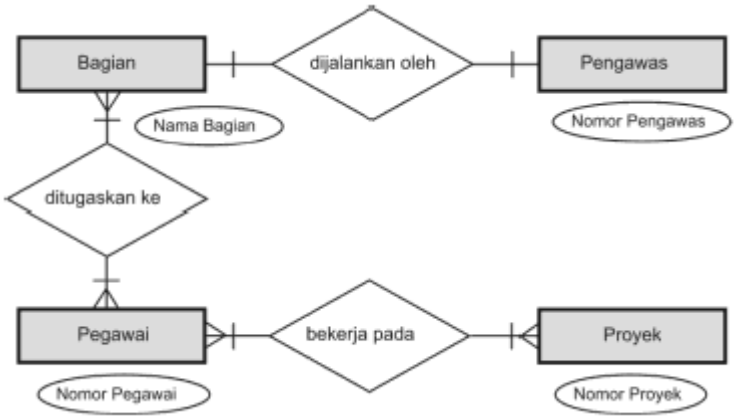
1. masing-masing bagian hanya punya satu pengawas
2. seorang pengawas bertugas di satu bagian
3. masing-masing bagian ada minimal satu pegawai
4. masing-masing pegawai bekerja paling tidak di satu bagian
5. masing-masing proyek dikerjakan paling tidak oleh satu pegawai



Gambar 2.12. Mengisi kardinalitas

Menentukan kunci utama

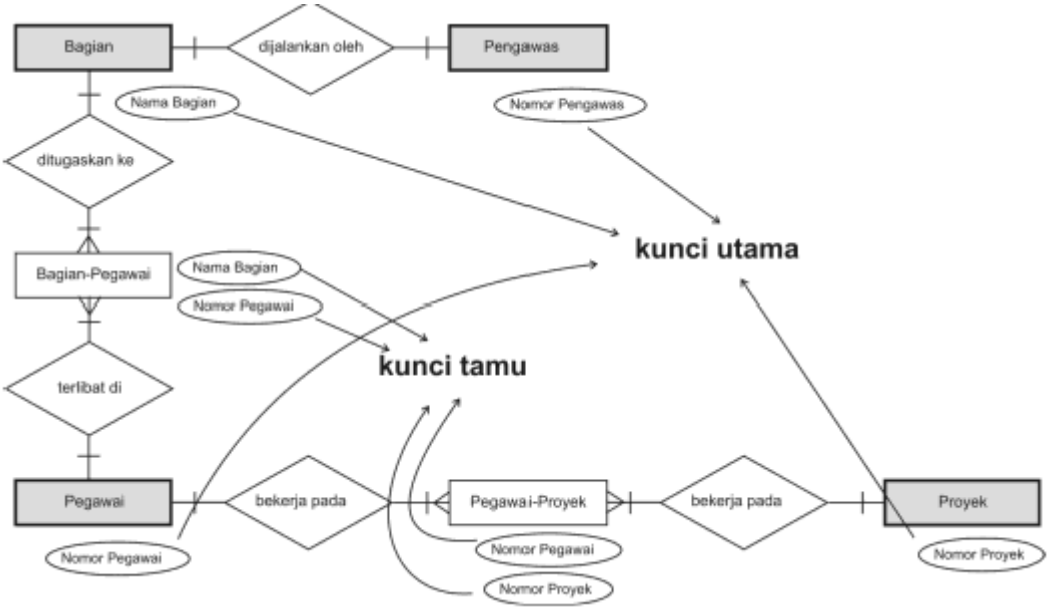
Kunci utamanya: Nomor Pengawas, Nama Bagian, Nomor Pegawai, Nomor Proyek



Gambar 2.13. Menentukan Kunci Utama

Menggambar ERD berdasarkan kunci

Ada dua relasi many to many pada ERD sementara, yaitu antara bagian dengan pegawai, pegawai dengan proyek, oleh sebab itu kita buat entitas baru yaitu bagian -pegawai dan pegawai-proyek Kunci utama dari entitas baru adalah kunci utama dari entitas lain yang akan menjadi kunci tamu di entitas yang baru.



Gambar 2.14. Menggambar ERD berdasarkan kunci

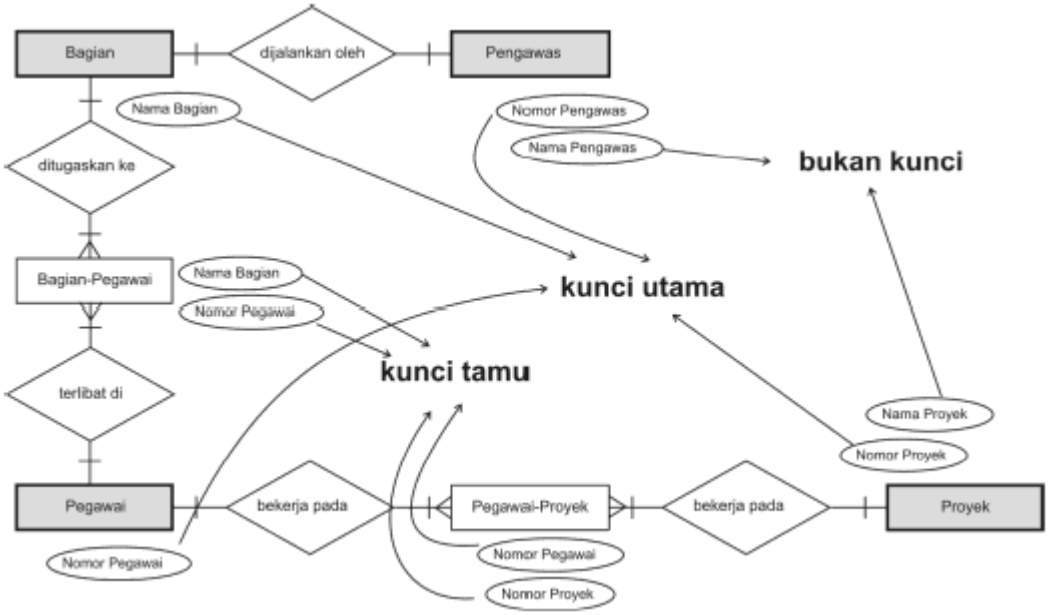
Menentukan atribut

Atribut yang diperlukan adalah: nama bagian, nama proyek, nama pegawai, nama pengawas, nomor proyek, nomor pegawai, nomor pengawas

Memetakan atribut

1. Bagian : Nama bagian
2. Proyek: Nama proyek
3. Pegawai:Nama pegawai
4. Pengawas: Nama pengawas
5. Proyek-Pegawai : Nomor proyek, Nomor pegawai
6. Pengawas: Nomor pengawas

Menggambar ERD dengan atribut



Gambar 2.15. Menggambar ERD dengan atribut

Memeriksa Hasil

Periksa apakah masih terdapat redundasi. ERD akhir: untuk pemodelan data pada sistem [2].