# BAB II

# LANDASAN TEORI

Berdasarkan penjelasan pada BAB I bahwa isi dari BAB II akan membahas mengenai uraian singkat teori-teori yang berhubungan mengenai topik yang akan dibuat. Teori-teori tersebut berdasarkan pada penelitian atau hal-hal lainnya yang didapatkan dari buku, jurnal maupun dari sumber lainnya. Teori-teori yang dijelaskan pada bab ini diantaranya yaitu artikel berita, internet, *website, scraping, text mining, K-Nearest Neighbour, Naïve Bayes, website,* model pengembangan perangkat lunak *linear sequential, context diagram,* DFD *(Data Flow Diagram), database,* dan *black box testing.*

## Artikel Berita

Artkel adalah karangan faktual mengenai suatu hal yang dibuat untuk dipublikasikan melalui media cetak seperti koran, majalah, buletin atapun melalui media elektronik seperti internet. Pembuatan artikel ini bertujuan untuk berbagi informasi kepada masyarakat. Artikel dibuat dengan melakukan penelitian terlebih dahulu terhadap objek yang akan dijadikan sebagai bahan pembahasan.

Berita adalah sebuah laporan yang berisi informasi mengenai suatu kejadian pada saat itu juga. Dalam Wikipedia Indonesia dijelaska bahwa definisi berita adalah informasi baru atau informasi mengenai sesuatu yang sedang terjadi, disajikan lewat bentuk cetak, siaran, internet, atau dari mulut ke mulut [8]. Tidak semua kejadian dapat dikatakan berita. Suatu kejadian dapat dikatakan sebagai berita jika kejadian itu merupakan kejadian yang luar biasa dan baru saja terjadi, selain itu kejadian tersebut juga memiliki dampak bagi masyarakat jika kejadian tersebut disebarluaskan.

Melalui kedua penjelasan diatas maka definisi dari artikel berita adalah suatu karangan atau laporan suatu kejadian tertentu, dimana kejadian tersebut merupakan kejadian yang luar biasa dan baru saja terjadi yang kemudian akan dipublikasikan baik melalui media cetak maupun media elektronik.

## Internet

Internet *(Interconnection Networking)* adalah sekumpulan komputer yang terhubung satu sama lain melalui protokol TCP/IP sehingga memungkinkan adanya pertukaran informasi dari komputer *server* ke komputer *client.* Komputer-komputer tersebut akan terhubung dan membentuk sebuah jaringan melalui jalur telekomunikasi seperti telepon, *wireless,* ataupun satelit.

TCP *(Transmission Control Protocol)* adalah protokol yang bertanggung jawab untuk mengirimkan data dari *host* sumber ke *host* tujuan dengan benar(Sofana, 2011)*.* Selain itu TCP juga bertugas untuk mendeteksi kesalahan atau terjadi kehilangan data saat proses pengiriman data. Jika terjadi hal tersebut maka TCP akan melakukan pengiriman kembali data yang salah atau hilang itu ke *host* tujuan sampai data diterima dengan lengkap. IP *(Internet Protocol)* adalah protokol yang mengatur bagaimana suatu data dapat dikenal dan dikirim dari satu komputer ke komputer lain (Sofana, 2011). IP biasa digunakan untuk pengalamatan internet atau biasa dikenal dengan nama *IP address.*

### Sejarah Internet

Sejarah terbentuknya internet dimulai pada tahun 1969. Pada saat itu, DARPA *(Defence Advance Research)* yang merupakan lembaga riset Departemen Pertahanan Amerika mendanai sebuah proyek untuk mengembangkan jaringan komunikasi data antar beberapa komputer. Tujuannya adalah untuk keperluan militer, yaitu untuk menghubungkan daerah-daerah penting agar bisa terhubung dalam menghadapi masalah jika terjadi serangan nuklir dan gangguan keamanan lainnya. Perkembangan jaringan tersebut menghasilkan ARPNet. Pada tahun 1972, organisasi ini mendemonstrasikan proyek ini di depan peserta *The First Internastional Conference on Computer Communication* dengan memasang 40 titik jaringan.

FTP *(File Transfer Protokol)* merupakan aplikasi internet pertama yang diciptakan, yang berfungsi untuk mentransfer atau mengirimkan file. Jumlah host di internet pada tahun 1984 lebih dari 1000 host dan pada tahun ini pula diperkenalkan DNS *(Domain Name System)* yang digunakan untuk mendefinisikan IP *(Internet Protocol)* agar lebih mudah untuk diingat.

Tahun 1986, U.S.NSF *(National Science Foundation)* yang merupakan Lembaga Ilmu Pengetahuan Nasional Amerika Serikat mendanai pembuatan jaringan TCP/IP yang didanai oleh organisasi NSFNet. Jaringan TCP/IP digunakan untuk menghubungkan universitas-universitas di Amerika Serikat dengan kecepatan transfer file sebesar 56 Kbps. Jaringan TCP/IP merupakan awal lahirnya perkembangan internet sampai saat ini.

Tahun 1987, lahirlah ISP *(Internet Service Provider)* yang diberi nama UUNet, yang merupakan provider utama internet. Tahun 1995, perkembangan internet semakin luas yang ditandai dengan adanya penyedia jasa internet (ISP). NSFNet yang sudah lama menjadi pusat internet kemudian diubah kembali menjadi jaringan untuk keperluan riset. Karena perubahan ini, maka lalu lintas data yang melalui Amerika dialihkan ke jaringan pusat ISP. Sementara itu NSPNet mengembangkan VBNS *(Very High Speed Backbone Network Service)*, yaitu jaringan dengan kecepatan sangat tinggi yaitu sebesar 622 Mbps (OC-12), yang menghubungkan lima komputer super. Seiring dengan berjalannya waktu, aplikasi internet semakin banyak dengan diciptakannya WAIS *(Wide Area Information Servers), Gopher,* dan WWW *(World Wide Web).* (Juju, 2007)*.*

## *Website*

*Website* adalah halaman situs berupa berkas teks *(plainteks)* yang diatur sedemikian rupa melalui instruksi berbasis HTML dan juga bahasa skrip lainnya yang saling berhubungan satu sama lain. Halaman *web* berisi informasi baik berupa teks, gambar, audio, maupun video. Halaman *web* tersimpan dalam *server web* dan terhubung secara global dengan komputer *client* melalui jaringan TCP/IP sebagai protokol pertukaran paket data.

*Website* terbentuk dari beberapa komponen yang membentuk *website* itu sendiri. Komponen-komponen pembentuk *website* diantaranya:

1. HTML

HTML *(HyperText Markup Language)* adalah bahasa pemrograman yang digunakan untuk membuat halaman *web*. Dokumen HTML dapat dijalankan diberbagai sistem operasi. HTML merupakan bahasa pemrograman yang tidak *case sensitive* yang artinya tidak ada perbadaan antara huruf kecil dengan huruf kapital. HTML terdiri dari tag-tag yang berisi elemen pembangun halaman web, seperti untuk menebalkan huruf maka digunakan tag <b> atau <B>.

1. PHP

PHP *(Hypertext Preprocessor)* merupakan bahasa pemrograman *server side* yang biasa digunakan oleh *developer* untuk membangun serta mengembangkan *website.* Untuk menjalankan php itu sendiri yaitu dengan menggunakan *web server.* PHP yang diciptakan oleh Rasmus Lerdorf pada tahun 1994 yang awalnya singkatan dari *Personal Home Page* yang kemudian berubah menjadi *Hypertext Preprocessor.* Versi terbaru PHP saat ini yaitu versi ke 5 atau yang sering kita sebut PHP5.

Untuk melakukan eksekusi program PHP dibutuhkan perangkat lunak *web server* yaitu Apache. Apache menggunakan protokol HTTP untuk melayani fasilitas *web/www.* Apache dapat dijalankan di berbagai sistem operasi seperti Windows, Linux, dan sistem operasi lainnya.

1. CSS

CSS *(Cascading Style Sheet)* adalah bahasa *style sheet* untuk mengatur dokumen HTML, seperti pengaturan *background,* margin, *font, border, teks*, gambar atau yang lainya, sehingga membuat tampilan web menjadi lebih indah. Menurut Juju, CSS secara sederhana adalah sebuah metode yang digunakan untuk mempersingkat penulisan tag HTML, seperti *font, color, text*, dan tabel menjadi lebih ringkas sehingga tidak terjadi pengulangan penulisan. (Juju, 2007). CSS bersifat lintas *platform* dimana *script* ini dapat dibaca oleh berbagai macam sistem operasi dan *browser*. CSS dikembangkan oleh W3C *(World Wide Web Consortium),* yaitu suatu organisasi yang mengembangkan teknologi internet pada tahun 1996. (Jayan, 2010).

1. Javascript

Javascript adalah bahasa pemrograman web berbasis java. Kode javascript disisipkan ke dalam halaman *web* melalui tag-tag yang berisi *script.* Javascript bersifat *case sensitif,* artinya huruf kecil dengan huruf capital berbeda. Javascript juga bersifat *client side scripting* yang eksekusinya berada di sisi *client* yang berupa *browser* seperti Internet Explorer (IE), Mozilla Firefox, Google Chrome, Netscape, dan Opera (Setiawan, 2015).

1. Jquery

Jquery adalah *library* yang berisi kumpulan kode-kode Javascript yang berfungsi untuk mempermudah dan mempercepat dalam pembuatan kode Javascript. Fitur yang terdapat di jquery diantaranya seperti fitur untuk memanipulasi *class* atau *id* pada CSS, efek animasi, dan yang lainnya.

1. MySQL

MySQL *(My Structure Query Language)* adalah sebuah program pembuat dan pengelola *database* atau yang disebut dengan DBMS *(DataBase Management System)*. (Nugroho, 2009). DBMS ini bersifat *open source*, yang artinya para *developer* dapat memperolehnya secara gratis melalui alamat website <http://www.mysql.com>.

## *Scraping*

*Scraping* adalah teknik untuk mengambil isi dari halaman *website* dengan spesifik, seperti mengambil *link website* atau *link image.* Teknik *scraping* dapat dilakukan menggunakan kode-kode pada HTML DOM *(Document Object Model).* DOM adalah objek model standar untuk HTML dan XML yang bersifat *platform independent* (Setiawan, 2015)*.*

## Text Mining

*Text mining* adalah proses menganalisis suatu dokumen berupa teks yang tidak terstruktur dalam jumlah besar untuk mendapatkan informasi. *Text mining* dilakukan dengan pencarian pola pada teks, yaitu dengan mencari kata-kata pada dokumen yang dapat dijadikan sebagai perwakilan dari satu dokumen kemudian akan dianalisis dengan dokumen lainnya dan dilihat keterhubungannya dengan dokumen yang lain. Terdapat beberapa tahapan yang dilakukan dalam *text mining* yaitu sebagai berikut:

1. *Case folding,* adalah proses mengubah semua huruf pada teks menjadi huruf kecil serta menghilangkan semua karakter yang ada dalam teks selain huruf a-z, kecuali tanda pemecah kalimat seperti spasi, tab, dan *newline* (garis baru).
2. *Tokenizing,* adalah proses pemotongan kata-kata dalam teks yang menyusun kalimat/*string.* Pemotongan kata dilakukan dengan melihat pemisah kata seperti spasi, tab, atau *newline* (baris baru). Hasil dari *tokenizing* adalah kata-kata tunggal penyusun *string*/kalimat yang dimasukan.
3. *Filtering,* adalah proses penyaringan terhadap kata-kata hasil dari *tokenizing* dengan menghilangkan *stoplist.* *Stoplist* adalah kata yang tidak relevan atau tidak memiliki keterkaitan dengan tema, seperti kata “dari”, “di”, “yang”, dan lain-lain.
4. *Stemming,* adalah proses mencari akar *(root)* kata dari setiap kata penyusun dokumen yaitu dengan mengembalikan kata berimbuhan ke dalam bentuk kata dasar.
5. *Analizing,* adalah proses untuk menghitung bobot (w) dokumen untuk dilihat tingkat kemiripannya dengan dokumen lain dengan melihat *keyword* penyusun dokumen tersebut.

## *K-Nearest Neighbour*

*K-Nearest Neighbour* atau biasa dikenal dengan KNN adalah metode untuk mengklasifikasikan suatu objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Menurut Jumadi dan Winarko, *K-Nearest Neighbour* merupakan algoritma yang melakukan klasifikasi berdasarkan kedekatan lokasi (jarak) suatu data dengan data yang lain (Jumadi & Winarko, 2015).

Langkah-langkah yang dilakukan dalam *k-nearest neighbor* yaitu:

* + - 1. Inputkan data latih, data uji, dan nilai *k.*
      2. Hitung jarak setiap data latih dengan data uji. Untuk menghitung jarak data latih dengan data uji menggunakan formula jarak *Eucledian,* yaitu:

(1)

Keterangan:

*x* dan *y* adalah titik pada ruang vektor *n* dimensi.

*xi* dan *yi* adalah besaran scalar untuk dimensi ke-*I* dalam ruang vektor *n* dimensi.

* + - 1. Urutkan data latih berdasarkan jarak terdekat.
      2. Ambil data latih sebanyak nilai *k* yang sudah ditentukan.
      3. Tentukan kelas berdasarkan kelas yang paling dominan dari *k* data latih yang sudah diambil.

Penggunaan *k-nearest neighbor* untuk *clustering* dilakuan dengan tahapan sebagai berikut:

Buat himpunan kosong untuk menampung klaster, misalnya klaster *m0.*

Tentukan nilai *k* data latih yang memiliki jarak terdekat dengan data uji.

Baca teks *p* dokumen pada *corpus* data latih. Bentuk klaster baru dengan anggota *p* dari label klaster teks *p.*

Jika tidak ada teks yang dapat dibaca pada *corpus* data latih maka lanjut ke tahap 6. Jika ada teks yang dapat dibaca maka hitung jarak terdekat teks *p* dengan semua *cluster* yang jaraknya terdekat dengan *p.*

Gabungkan kata *p* ke dalam klaster C.

Hentikan proses *clustering*. Ambil hasil *clustering m0 =* {C1, C2, C3, … , Cn}. Setiap kalster dalam *m0* berisi kumpulan kata *p* dan label klaster. Himpunan *m0* adalah model klasifikasi.

Terdapat beberapa kondisi untuk menentukan kemiripan suatu data pada *k-nearest neighbor,* yaitu:

1. Jika terdapat 1 kelas dari 1 data latih sebagai tetangga terdekat (terdekat pertama) dari data uji, maka dijadikan sebagai kelas untuk data uji.
2. Jika dalam *k*-tetangga terdapat 2 kelas yang berbeda, maka diambil kelas dengan jumlah data terbanyak (mayoritas voting).
3. Jika terdapat 2 kelas dimana pertama memiliki jumlah tetangga jauh lebih banyak dan kelas kedua memiliki jumlah tetangga dekat lebih sedikit. Keadaan ini akan menyebabkan distorsi data. Untuk mengatasi mayoritas voting pada kasus ini maka ditambahkan penggunaan bobot untuk menghitung kandidat kelas yang sebaiknya diambil oleh data uji dari *k*-tetangga terdekat. Bobot untuk setiap tetangga terdekat dihitung dengan rumus di bawah ini:

(2)

Formula yang dapat digunakan yaitu:

(3)

*v* merupakan jumlah data yang masuk ke dalam kelas *yi* (Prasetyo, 2012).

## *Naïve Bayes*

*Naïve bayes* adalah metode untuk mengklasifikasikan suatu objek yang menggunakan perhitungan probabilitas. Algoritma *Naïve bayes* dikemukakan oleh seorang ilmuwan asal Inggris, yaitu Thomas Bayes. Aturan dasar *Naïve bayes* adalah suatu hipotesis atau peristiwa dapat diperkirakan berdasarkan beberapa bukti yang diamati. Rumus umum *Naïve bayes* yaitu:

(4)

Keterangan:

P(H | E) : Probabilitas akhir bersyarat *(conditional probability)* suatu hipotesis H terjadi jika diberikan bukti *(evidence)* E terjadi.

P(E | H) : Probabilitas sebuah bukti E terjadi akan mempengaruhi hipotesis H.

P(H) : Probabilitas awal (priori) hipotesis H terjadi tanpa memandang bukti apapun.

P(E) : Probabilitas awal (priori) bukti E terjadi tanpa memandang hipotesis/bukti yang lain.

Rumus *Naïve bayes* untuk klasifikasi atau biasa disebut *Naïve Bayes Clasifier* (NBC) menggunakan formula di bawah ini:

(5)

Keterangan:

P(Y | X) : Probabilitas data dengan vektor X pada kelas Y.

P(Y) : Probabilitas awal kelas Y.

: Probabilitas independen kelas Y dari semua fitur dalam vektor X (Prasetyo, 2012).

Klasifikasi dengan metode *naïve bayes* dilakukan dengan menentukan nilai kategori yang paling tinggi probabilitasnya (VMAP) dari dokumen yang akan diklasifikasikan berdasarkan *term* yang muncul dalam dokumen. Penentuan nilai kategori dilakukan dengan memasukan atribut <a1, a2, a3, … an>. Dari atribut-atribut tersebut akan dicari nilai maksimumnya dengan rumus:

(6)

Berdasarkan teorema bayes pada persamaan (4), maka persamaan (6) ditulis:

(7)

untuk semua *vj* memiliki nilai yang sama, maka persamaan ini dapat ditulis:

(8)

Dengan mengasumsikan bahwa nilai setiap kata dalam atribut <a1, a2, a3, … an> sama, maka nilai dalam persamaan (8) ditulis:

(9)

Sehingga persamaan (8) ditulis:

(10)

dan probabilitas kata *wk* untuk setiap kategori dihitung pada saat pelatihan. Untuk mencari nilai dan dihitung menggunakan rumus:

(11)

(12)

Keterangan:

*|docs j|* merupakan banyaknya dokumen pada kategori j saat pelatihan.

*|Contoh|* merupakan banyaknya dokumen yang digunakan saat pelatihan.

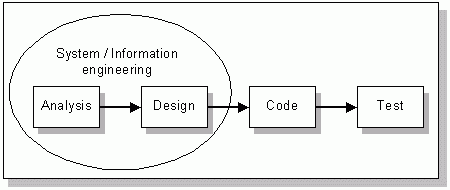
*nk* merupakan frekuensi kemunculan kata *wk* pada kategori *vj.*

*n* merupakan banyaknya kata pada semua dokumen dengan kategori *vj.*

*|kosakata|* merupakan banyaknya kata dalam pelatihan.

## Model Pengembangan Perangkat Lunak *Linear Sequential*

Model pengembangan perangkat lunak *linear sequential* atau biasa dikenal dengan model *waterfall* adalah model pengembangan perangkat lunak dengan pendekatan alur hidup perangkat lunak yang terurut *(sequential)*, dimulai dari analisis, desain, pengodean, pengujian, dan tahap pengujian.



**Gambar 1.2** Model *Sequential Linear*[[[1]](#footnote-1)]

1. Tahap analisis kebutuhan perangkat lunak adalah tahap untuk mengumpulkan semua kebutuhan perangkat lunak yang dilakukan secara intensif. Tahap ini merupakan proses untuk menspesifikasikan kebutuhan perangkat lunak yang akan dibuat sehingga dapat dengan mudah dimengerti oleh *user.* Tahap analisis perlu untuk didokumentasikan.
2. Tahap desain adalah tahap dimana kebutuhan perangkat lunak yang sebelumnya didapatkan pada tahap analisis akan digambarkan ke dalam desain perangkat lunak, seperti struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengkodean. Tahap ini perlu untuk didokumentasikan.
3. Tahap pengkodean adalah tahap menerjemahkan desain perangkat lunak yang dibuat pada tahap desain ke dalam kode-kode bahasa pemrograman. Hasil yang didapatkan dari tahap ini adalah program komputer yang sesuai dengan desain perangkat lunak yang sebelumnya dibuat pada tahap desain.
4. Tahap pengujian *(test)* adalah tahap untuk menguji fungsionalitas perangkat lunak untuk memastikan bahwa setiap bagian dari perangkat lunak tersebut berjalan sebagaiman mestinya. Tahap ini dilakukan untuk meminimalisir kesalahan *(error)*, serta untuk memastikan *output* yang dihasilkan sesuai dengan yang diinginkan.
5. Tahap pemeliharaan *(maintenance)* atau tahap pendukug *(support)* adalah tahap untuk pemeliharaan terhadap perangkat lunak yang sudah dikirimkan kepada *user* jika akan dilakukan perubahan atau perbaikan terhadap bagian perangkat lunak yang tidak berjalan sesuai keinginan *user*. Tahap ini dilakukan dengan mengulangi proses pengembangan yang dimulai dari tahap analisis untuk perubahan perangkat lunak yang sudah ada, tetapi tidak untuk membuat perangkat lunak baru.

## UML (Unified Modeling Language)

UML *(Unified Modeling Language)* adalah bahasa pemodelan standar pada pembangunan perangkat lunak berbasis objek untuk medefinisikan *requirement* perangkat lunak, membuat analisis dan desain perangkat lunak, serta menggambarkan arsitektur perangkat lunak. UML merupakan standar bahasa yang banyak digunakan untuk pengembangan perangkat lunak berorientasi objek.

UML dikeluarkan oleh perusahaan *Rational Software Corporation* pada tahun 1995. UML dibangun dari beberapa gabungan konsep pemodelan, seperti OMT *(Object Modelling Technique)* yang diciptakan oleh Rambaugh dan Booch (1991), CRC *(The Classes, Responsibilities, Collaborators)* oleh Rebecca Wirfs Brock (1990), dan konsep lainnya (Shalahuddin & A.S, 2013). Dari gabungan konsep-konsep tersebut, UML mengambil elemen-elemen yang lebih efektif dan membuang elemen-elemen yang tidak praktis, serta menambahkan elemen-elemen baru yang sebelumnya belum ada pada konsep terdahulu sehingga UML lebih ekspresif dari konsep pemodelan yang lainnya.

### Use Case Diagram

*Use case diagram* adalah pemodelan pada sistem berorientasi objek yang menggambarkan kelakuan *(behaviour)* dari sistem yang akan dibuat. *Use case* mendeskripsikan interaksi antara aktor dengan sistem. Dengan kata lain, *use case* digunakan untuk mengetahui siapa saja yang berhak menggunakan sistem dan fungsi apa saja yang dapat dilakukan oleh sistem.

Terdapat aturan dalam pemberian nama pada *use case,* yaitu *use case* harus didefinisikan sesimpel mungkin dan dapat dipahami oleh para *developer*. Aturan untuk pemberian nama pada aktor harus menggunakan kata benda sedangkan pemberian nama pada *use case* harus menggunakan kata kerja.

*Use case diagram* memiliki beberapa simbol yang digunakan untuk menggambarkan fungsionalitas sistem. Table 2.1 di bawah ini akan menjelaskan mengenai simbol-simbol yang terdapat pada *use case diagram.*

**Tabel 2.1** Simbol *Use Case Diagram*

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Nama** | **Simbol** | **Keterangan** |
| 1 | *Use case* | Description: Description: Description: Description: Description: Description: Description: Description: 2010-06-01_110252 | *Use case* mendefinisikan fungsionalitas yang disediakan oleh sistem. Pemberian nama pada *use case* menggunakan kata kerja di awal frase namanya. |
| 2 | Aktor/*actor* | Description: Description: Description: Description: Description: Description: Description: Description: 2010-06-01_110124 | *Actor* mendefinisikan orang, proses atau sistem lain yang berinteraksi dengan sistem yang akan dibuat. Pemberian nama pada *actor* menggunakan kata benda di awal frase namanya. |
| 3 | Asosiasi */ Association* | Description: Description: Description: Description: Description: Description: Description: Description: 2010-06-01_110406 | Asosiasi mendefinisikan interaksi antara *actor* yang berdengan *use case.* Asosiasi digambarkan dengan sebuah garis lurus yang menghubungkan *actor* dengan *use case.* |
| 4 | *Dependency include* |  | *Dependency extend* mendefinisikan hubungan antara dua *use case* dimana *use case* yang satu akan memanggil *use case* yang lain. |
| 5 | *Dependency extend* |  | *Dependency include* mendefinisikan pemanggilan *use case* lain yang hanya diperlukan pada kondisi tertentu. |
| 6 | Generalisasi |  | Generalisasi mendefinisikan hubungan antara dua *use case* dimana fungsi yang satu merupakan fungsi yang lebih umu dari fungsi yang lainnya. |

### Class Diagram

*Class diagram* adalah pemodelan pada sistem berorientasi objek yang yang menggambarkan struktur sistem yang didefinisikan oleh kelas-kelas yang akan dibuat untuk membangun sistem tersebut. Kelas-kelas pada struktur sistem harus dapat melakukan fungsi sesuai dengan kebutuhan sistem.

*Class diagram* menggambarkan struktur kelas suatu sistem. Susunan struktur kelas yang baik pada *class diagram* memiliki jenis kelas sebagai beritkut:

1. Kelas *main*, merupakan kelas yang paling pertama dieksekusi ketika program dijalankan.
2. Kelas *view*, merupakan kelas yang mengatur tampilan *user interface.*
3. Kelas *controller,* merupakan kelas yang menangani fungsi-fungsi yang sudah didefinisikan sebelumnya pada *use case diagram.*
4. Kelas *model*, merupakan kelas yang menghubungkan sistem dengan *database* sehingga sistem dapat menambahkan, mengubah, atau menghapus data di *database.*

Tabel 2.2 di bawah ini merupakan simbol-simbol yang terdapat pada *class diagram.*

**Tabel 2.2** Simbol *Class Diagram*

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Nama** | **Simbol** | **Keterangan** |
| 1 | *Class* | Description: Description: Description: Description: Description: Description: Description: Description: Class | *Class* menggambarkan struktur perangkat lunak yang berisi kode programyang terkait dengan nama kelas. Class terdiri dari *object name* sebagai nama kelas, *attribute* sebagai variabel-variabel yang terdapat pada kelas, dan *operation* sebagai metode atau fungsi yang terdapat pada kelas. |
| 2 | *Association* | Description: Description: Description: Description: Description: Description: Description: Description: 2010-06-01_104903 | *Association* menggambarkan relasi antara dua kelas. *Association* digambarkan dengan garis lurus yang menghubungkan antara kelas yang satu dengan kelas lainnya. |
| 3 | *Composition* | Description: Description: Description: Description: Description: Description: Description: Description: 2010-06-01_105039 | *Composition* menggambarkan relasi antara satu kelas dengan kelas lain yang merupakan bagian dari kelas tersebut, dalam artian kelas pertama bergantung pada kelas kedua. |
| 4 | *Dependency* | Description: Description: Description: Description: Description: Description: Description: Description: 2010-06-01_105127 | *Dependency* menggambarkan operasi pada satu kelas yang menggunakan kelas lain. |
| 5 | *Aggregation* | Description: Description: Description: Description: Description: Description: Description: Description: 2010-06-01_105208 | *Aggregation* menggambarkan relasi antar kelas dengan makna “semua-bagian” |
| 6 | *Generalization* | Description: Description: Description: Description: Description: Description: Description: Description: 2010-06-01_105257 | *Generallization* menggambarkan relasi antar kelas dengan makna umum-khusus. |

### Activity Diagram

*Activity diagram* adalah pemodelan pada sistem berorientasi objek yang menggambarkan alur kerja *(workflow)* atau aktivitas suatu proses pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa aktivitas yang dilakukan bukanlah aktivitas yang dilakukan aktor terhadapat sistem, tetapi aktivitas yang dapat dilakukan oleh sistem.

Pemodelan pada *activity diagram* diawali dengan *start* dan diakhiri dengan *end.* Pemodelan itu dibuat dengan simbol-simbol *activity diagram.* Table 2.3 di bawah ini menjelaskan simbol-simbol yang ada pada *activity diagram.*

**Tabel 2.3** Simbol *Activity Diagram*

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Nama** | **Simbol** | **Keterangan** |
| 1 | Status awal |  | Status awal aktivitas dari sistem. |
| 2 | Aktivitas |  | Aktivitas/alur kerja yang dilakukan oleh sistem. Penamaan aktivitas diawali oleh kata kerja. |
| 3 | Percabangan / *decision* |  | Asosiasi percabangan digambarkan jika terdapat pilihan aktivitas yang lebih dari satu. |
| 4 | Penggabungan / *join* |  | Asosiasi penggabungan digambarkan jika terdapat beberapa aktivitas kemudian digabungkan menjadi satu. |
| 5 | Status akhir |  | Status akhir aktivitas yang dilakukan sistem. |
| 6 | *swimlane* | Description: Description: Description: Description: Description: Description: Description: Description: I:\SI Enterprise\2010-05-22_114414.png | *Swimlane* menggambarkan pemisahan pada organisasi bisnis yang bertanggung jawab pada aktivitas yang terjadi. |

### Sequence Diagram

*Sequence diagram* adalah pemodelan pada sistem berorientasi objek yang menggambarkan kelakuan *(behaviour)* objek pada *use case* yang diddeskripsikan dengan waktu hidup objek serta *message* yang dikirimkan dan diterima dari objek satu ke objek lainnya. Untuk pembuatan *sequence diagram* maka harus mengetahui objek apa saja yang terlibat pada *use case* serta metode apa saja yang dimiliki oleh kelas yang diinstansiasikan menjadi objek tersebut. Selain itu, dalam pembuatan *sequence diagram* juga perlu melihat scenario yang ada pada *use case.*

Pemodelan *sequence diagram* juga dilakukan menggunakan simbol-simbol yang terdapat pada *sequence diagram.* Table 2.4 di bawah ini akan menjelaskan mengani simbol-simbol yang terdapat pada *sequence diagram.*

**Tabel 2.4** Simbol *Sequence Diagram*

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Nama** | **Simbol** | **Keterangan** |
| 1 | Aktor | Description: Description: Description: Description: Description: Description: Description: Description: 2010-06-01_110124  atau | Orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibangun. Penamaannya menggunakan kata benda di awal frase. |
| 2 | *Lifeline* |  | *Lifeline* menggambarkan kehidupan objek. |
| 3 | Waktu aktif |  | Menyatakan objek dalam keadaan aktif dan berinteraksi dengan sistem. Semua yang terhubung dengan waktu aktif menyatakan tahapan yang dilakukan di dalam sistem. |
| 4 | *Message* |  | *Messasge* menyatakan data atau proses yang dikirimkan atau diterima antar objek. |

### Statechart Diagram

*Statechart diagram* adalah pemodelan pada sistem berorientasi objek yang menggambarkan daur hidup sistem dari awal sistem diinisialisasikan hingga keadaan *destroy. Statechart diagram* menggambarkan perubahan atau transisi suatu sistem atau objek. Jika *sequence diagram* digunakan untuk untuk interaksi antar objek, maka *statechart diagram* digunakan untuk interaksi di dalam sebuah objek (Shalahuddin & A.S, 2013).

*Statechart* diagram juga memiliki simbol-simbol yang digunakan untuk melakukan pemodelan pada sistem yang akan dibuat. Tabel 2.5 di bawah ini menjelaskan mengenai simbol-simbol pada *statecharrt diagram.*

**Tabel 2.5** Simbol *Statechart Diagram*

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Nama** | **Simbol** | **Keterangan** |
| 1 | *Start /* status awal |  | *Start* menggambarkan keadaan awal sistem mulai hidup. |
| 2 | *End /* status akhir |  | *End* menggambarkan keadaan akhir dari alur hidup sistem. |
| 3 | *Event* | Description: Description: Description: Description: Description: Description: Description: Description: C:\Users\aminus\Pictures\event.png | *Event* menggambarkan keadaan yang menyebabkan perubahan status mesin/objek. |
| 4 | *State* |  | *State* menggambarkan keadaan mesin atau objek pada saat tertentu. |

## *Database*

*Database* adalah sekumpulan data yang disusun sedemikian rupa dengan aturan tertentu yang saling berelasi satu sama lain, sehingga memudahkan penggunanya dalam mengelola data serta mendapatkan informasi tersebut. *Database* merupakan kumpulan tabel-tabel yang saling berelasi satu sama lain. Tujuan dibentuknya table-tabel di sini adalah untuk menyimpan data ke dalam table-tabel agar mudah diakses (Shalahuddin & A.S, 2013).

## *Black Box Testing*

*Black box testing* adalah pengujian yang fokus pada kebutuhan fungsionalitas sistem yang dibuat. *Black box testing* yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dank ode program (Shalahuddin & A.S, 2013). Pengujian dilakukan dengan maksud untuk mengetahui apakah fungsi-fungsi yang terdapat pada sistem sudah sesuai dengan spesifikasi yang dibutuhkan atau belum.

*Black box testing* dilakukan dengan membuat studi uji untuk dicoba semua fungsinya dengan memakai sistem tersebut apakah sesuai dengan spesifikasi yang dibutuhkan atau tidak. Studi uji yang dibuat terdiri dari kasus benar dan kasus salah. Contohnya seperti proses *login,* yaitu:

1. Jika *user* menggunakan *username* dan *password* yang benar.
2. Jika *user* menggunakan *username* dan/atau *password* yang salah.

1. Shalahuddin, M., dan A.S, R, **“*Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek”***, Informatika Bandung*,* 2013, hal 29. [↑](#footnote-ref-1)