BAB II

LANDASAN TEORI

II.1. *Barcode Scanner*

Barcode scanner ini dengan sebuah alat canggih dan serbaguna bekerja dengan berbagai jenis *barcode*, perangkat membaca, dan *Interfaces*. It komputer mendiskriminasikan lebih dari dua puluh simbol yang berbeda secara otomatis. *Bercode Scanner* digunakan untuk mengidentifikasi produk, barang atau pengiriman. Perangkat untuk membaca *barcode* yang *all-around*, dalam bentuk pembaca jenis pena, *laser scanner*, atau *scanner* LED. Pembaca berbasis kamera, sebagai jenis baru *barcode reader*, baru-baru ini mendapat banyak perhatian. Kepentingan dalam kamera berbasis pengakuan *barcode* membangun fakta bahwa banyak perangkat *mobile* sudah digunakan, yang menyediakan kemampuan untuk mengambil gambar dari kualitas yang baik. Upaya mengenai pengakuan 1D *barcode* menggunakan kamera ponsel telah dibuat. Al bersih Adelman. telah disajikan dua aplikasi *prototipe*: tampilan informasi literatur tentang buku-buku yang dipindai, dan tampilan informasi bahan tentang makanan dipindai untuk alergi orang. Mereka tidak melaporkan pertunjukan pengakuan, tetapi menunjukkan bukti konsep untuk baru aplikasi. Teknologi *barcode* menjadi banyak digunakan saat ini. Misalnya, di supermarket, ada banyak genggam *scanner* yang secara otomatis membaca *barcode* di merchandise. Membaca *barcode* melalui *scanner* khusus adalah teknologi dewasa. Komersial *berbasis laser*, genggam *barcode scanner* mencapai membaca yang kuat dengan harga

yang wajar. Baru-baru ini, bagaimanapun, telah ada minat dalam mengakses *barcode* dengan ponsel biasa daripada dengan perangkat khusus. Sayangnya, beberapa gambar yang diambil oleh kamera sering berkualitas rendah. Yang sering menghasilkan gambar buram. Hanya beberapa kamera memiliki berkedip sehingga *blur* dan *noise* adalah masalah yang sangat umum untuk gambar yang diambil dalam kondisi cahaya rendah. Semua faktor ini, mungkin dikombinasikan dengan gambar rendah *resolusi*, membuat *barcode* membaca sulit dalam situasi tertentu. Memang, semua yang ada berbasis gambar pembaca *barcode* memiliki kinerja yang terbatas ketika datang ke gambar yang diambil dalam kondisi cahaya yang sulit.

Dengan *string biner*, maka Anda dapat menggunakan aturan pengkodean untuk mendapatkan nomor EAN / UPC, sebuah algoritma untuk *recognize1D barcode*, yang bekerja untuk banyak digunakan standar UPC-A, EAN-13. Algoritma kami menggunakan *analisis citra* dan metode pengenalan pola yang mengandalkan pengetahuan tentang struktur dan penampilan *barcode 1D*. Mengingat komputasi yang daya dan kualitas gambar dari kamera saat ini, kontribusi adalah sebuah algoritma yang baik cepat dan kuat *Threshold* Seleksi, Template, Pencocokan dan *Encoding*. (Vina M. Lomte; 2012 : 59-60)

II.1.1. *Threshold selection*

Parameter kunci dalam proses *thresholding* adalah pilihan nilai ambang batas. yang tahan terhadap *noise* adalah *metode iterasi* contoh di bawah ini.

1. batas awal (T) yang dipilih, hal ini dapat dilakukan secara acak atau sesuai dengan yang lain Metode yang diinginkan.

2. Gambar tersegmentasi menjadi objek dan latar belakang piksel seperti dijelaskan di atas, menciptakan dua set:
    a. G1 = {f (m, n): f (m, n)> T} (objek piksel)
    b. G2 = {f (m, n): f (m, n) pixel yang terletak di
3. Rata-rata dari masing-masing set dihitung.
    a. m1 = nilai rata-rata
    b. m2 = nilai rata-rata
4. ambang baru dibuat yang merupakan rata-rata
    a. T '= (m1 + m2) / 2
5. Kembali ke langkah dua, sekarang menggunakan ambang baru dihitung pada langkah empat, tetap .

II.1.2. *Template Matching*

Teknik yang digunakan dalam mengklasifikasikan gambar terhadap satu sama lain. Contoh *image*. Jika standar deviasi dari gambar template yang dibandingkan dengan sumber gambar kecil cukup, template *matching* mungkin karakter, angka, dan kecil, benda-benda sederhana lainnya. *Template* gambar untuk semua posisi mungkin dalam sumber gambar yang lebih besar dan menghitung *numerik Indeks* yang menunjukkan seberapa baik pada *pixel-by-pixel* dasar. (Vina M. Lomte; 2012 : 62)

II.1.3. *Encoding*

Dalam *barcode*, setiap digit dari *encoding* diwakili oleh tujuh bar hitam atau putih sama ukuran. *Visual*, dua atau lebih berdekatan bar hitam muncul

sebagai bar lebar tunggal; bar putih muncul sebagai pemisah ruang (juga dari ukuran bervariasi) antara bar hitam. Jika kita sebut seperti lebar (hitam umum bar, kemudian, sebagai suatu peraturan, representasi angka selalu terdiri dari empat bar seperti umum, yang oleh kebutuhan, interlace: baik putih-hitam-putih putih. Varian kemungkinan representasi 7-bar akan ditampilkan dalam contoh diagram Gambar II.1. Upc-A / Ean-13 Encoding di bawah.

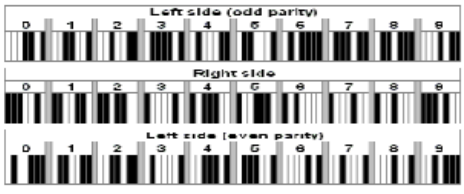


Figure 5 Upc-A Encoding

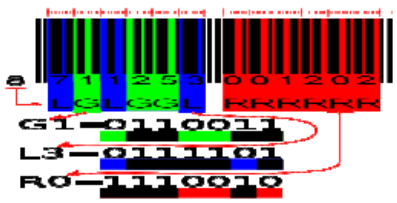


Figure: 6 EAN-13 Encoding

Gambar II.1. Upc-A / Ean-13 Encoding
Sumber: (Vina M. Lomte; 2012 : 59-60)

II.2. Tiket Konser

Tiket konser adalah Satu bagian kunci masuk acara sebuah *event* dan Salah satu indikator kesuksesan penyelenggaraan sebuah event atau acara dapat dilihat dari banyaknya peserta yang mengikuti. Ajang pameran akan disebut sukses besar dan meriah bila pengunjungnya sampai penuh setiap hari. Konser musik tentu akan sangat sukses bila penggemarnya penuh sesak memenuhi arena konser. Begitu pula penyelenggaraan konferensi akan sangat memuaskan, *event organizer* bila pesertanya penuh dan antusias mengikuti acara hingga akhir acara. Akan tetapi tidak mudah untuk bisa mengumpulkan sedemikian banyak peserta dalam setiap *event*. Penyelenggara akan menghadapi bagaimana menerapkan teknik promosi acara yang tepat sehingga menjangkau sasaran. (Ardiansyah ; 2011 : 62)

II.3. *Mobile Application*

Kata *mobile* mempunyai arti bergerak atau berpindah. Sehingga diperoleh pengertian bahwa aplikasi bergerak merupakan aplikasi yang dapat dijalankan walaupun pengguna berpindah atau karena pengguna berpindah. Pemrograman aplikasi bergerak tidak banyak berbeda dengan pemrograman konvensional pada PC. Aspek karakteristik dari perangkat bergerak sering mempengaruhi arsitektur dan implementasi dari aplikasi tersebut. Dalam pemrograman aplikasi bergerak berbagai aspek teknis perangkat lebih menonjol karena memiliki banyak keterbatasan dibandingkan komputer konvensional atau PC. (Sesaria Kiki Tamara; 2011; No. 6; Vol. 3)

II.4. *Android*

Android merupakan sistem operasi yang berisi *middleware* serta aplikasi-aplikasi dasar. Basis sistem operasi *android* yaitu kernel *linux* 2.6 yang telah diperbaharui untuk *mobile device*. Pengembangkan aplikasi *android* menggunakan bahasa pemrograman *java*. Yang mana konsep-konsep pemrograman *java* berhubungan dengan Pemrograman Berbasis Objek *(OOP)*). Selain itu pula dalam pengembangan aplikasi *android* membutuhkan *software development kit* (*SDK*) yang disediakan *android*, *SDK* ini memberi jalan bagi *programmer* untuk mengakses *application programming interface* (*API*) pada *android*. (Arzan Muharom, Rinda Cahyana, H. Bunyamin M.Kom ; 2013 : 2)

II.4.1. Komponen Dasar *Android*

Android UI *framework*, bersama dengan bagian lain dari *Android*, mengandalkan konsep baru disebut sebuah *intent*. Maksud adalah sebuah penggabungan ide-ide seperti pesan *windowing*, tindakan, menerbitkan-dan model berlangganan, antar-proses komunikasi, dan aplikasi pendaftar. *Android* juga memiliki dukungan luas untuk sumber daya, yang meliputi elemen akrab dan *file* seperti *string* dan bitmap, serta beberapa *item* yang tidak begitu akrab seperti *XMLbased* melihat definisi. *Framework* kerja ini membuat penggunaan sumber daya dengan cara baru untuk membuat penggunaannya mudah, intuitif, dan nyaman. Berikut adalah contoh di mana ID sumber daya adalah otomatis dihasilkan untuk sumber daya didefinisikan dalam *file* XML. Setiap ID otomatis dihasilkan di kelas ini sesuai untuk baik elemen dalam file XML atau seluruh *file* itu sendiri. Di mana pun Anda ingin menggunakan definisi tersebut XML, Anda akan menggunakan dihasilkan ID gantinya. Tipuan ini membantu banyak ketika datang ke lokalisasi. Konsep lain yang baru di *Android* adalah penyedia konten. Sebuah penyedia konten adalah abstraksi pada sumber data yang membuatnya tampak seperti emitor dan konsumen tenang jasa. *Database SQLite* mendasari membuat fasilitas penyedia konten kuat alat untuk pengembang aplikasi. (Windu Gata, Grace Gata, Nia Kusuma Wardhani ; 2012 : 66)

II.4.2. *Android* SDK

Android SDK (Software Development Kit) adalah *tools* API *(Aplication Programming Interface)* yang diperlukan untuk mengembangkan aplikasi pada *platform Android* menggunakan bahasa pemrograman *Java*. Beberapa fitur-fitur

Android yang paling penting adalah mesin *Virtual Dalvik* yang dioptimalkan untuk perangkat *mobile*, *integrated browser* berdasarkan *engine open source WebKit*, Grafis yang dioptimalkan dan didukung oleh *libraries grafis* 2D, *grafis* 3D berdasarkan spesifikasi *open* ES 1.0 (Opsional akselerasi perangkat keras), kemudian *SQLite* untuk penyimpanan data (*database*). Fitur-fitur *android* lainnya termasuk media yang mendukung audio, video, dan gambar, juga ada fitur *bluetooth*, EDGE, 3G dan WiFi, dengan fitur kamera, GPS, dan kompas. Selanjutnya fitur yang juga turut disediakan adalah lingkungan *Development* yang lengkap dan kaya termasuk perangkat *emulator*, *tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE *Eclipse*. (Alicia Sinsuw, Xaverius Najoan ; 2013 : 2)

II.4.3. Java

Java memiliki cara kerja yang unik dibandingkan dengan bahasa perograman lainya yaitu bahasa perograman *java* bekerja mengunakan *interptreter* dan juga *compiler* dalam proses pembuatan program, *Interpreter java* dikenal sebagai perograman *bytecode* yaitu dengan cara kerja mengubah paket *class* pada *java* dengan *extensi*. *Java* menjadi *.class*, hal ini dikenal sebagai *class bytecode*, yaitunya *class* yang dihasilkan agar program dapat dijalankan pada semua jenis perangkat dan juga *platform*, sehingga program *java* cukup ditulis sekali namun mampu bekerja pada jenis lingkungan yang berbeda. (Defni, Indri Rahmayun ; 2014 : 64)

II.5. Jaringan Komputer

Jaringan komputer adalah komunikasi data antar komputer, yaitu minimal 2 komputer. Jaringan komputer dapat dilakukan melalui media kabel ataupun nirkabel (*wireless*). Pada sistem antrian rumah sakit ini, jaringan komputer dilakukan melalui media kabel antara 2 komputer. *Interface* yang digunakan adalah DB-25 atau *port* paralel. Data yang dikirimkan antar komputer adalah berupa kode biner 1 dan 0, yang dirancang pada masing-masing program yaitu pada program *server* dan program *client*. Ada tiga tipe jaringan yang umum yang digunakan antara lain : Jaringan *WorkGroup*, Jaringan LAN dan Jaringan WAN

II.5.1. Jaringan Komputer Berdasarkan Skala

1. Jaringan LAN

LAN (*Local Area Network*) adalah suatu kumpulan komputer, dimana terdapat beberapa unit komputer (*client*) dan 1 unit komputer untuk bank data (*server*). Antara masing-masing *client* maupun antara *client* dan *server* dapat saling bertukar *file* maupun saling menggunakan printer yang terhubung pada unit-unit komputer yang terhubung pada jaringan LAN.

2. Jaringan MAN

Metropolitan Area Network atau MAN, merupakan Jenis Jaringan Komputer yang lebih luas dan lebih canggih dari Jenis Jaringan Komputer LAN. Disebut *Metropolitan Area Network* karena Jenis Jaringan Komputer MAN ini biasa digunakan untuk menghubungkan jaringan komputer dari suatu kota ke kota lainnya. Untuk dapat membuat suatu jaringan MAN, biasanya diperlukan adanya operator telekomunikasi untuk menghubungkan antar jaringan komputer.

3. Jaringan WAN

WAN (*Wide Area Network*) adalah kumpulan dari LAN dan/atau *Workgroup* yang dihubungkan dengan menggunakan alat komunikasi modem dan jaringan *Internet*, dari/ke kantor pusat dan kantor cabang, maupun antar kantor cabang. Dengan sistem jaringan ini, pertukaran data antar kantor dapat dilakukan dengan cepat serta dengan biaya yang *relative* murah. Sistem jaringan ini dapat menggunakan jaringan *Internet* yang sudah ada, untuk menghubungkan antara kantor pusat dan kantor cabang atau dengan PC Stand *Alone/Notebook* yang berada di lain kota ataupun negara. (Mardison, al husni ; 2012 : 59)

II.5.2. Jaringan Komputer Bertopologi / Pemasangan

1. Bus

Topologi bus merupakan topologi yang banyak dipergunakan pada masa penggunaan kabel sepaksi menjamur. Dengan menggunakan T-*Connector* (dengan terminator 50ohm pada ujung *network*), maka komputer atau perangkat jaringan lainnya bisa dengan mudah dihubungkan satu sama lain. Instalasi jaringan Bus sangat sederhana, murah dan maksimal terdiri atas 5-7 komputer. Kesulitan yang sering dihadapi adalah kemungkinan terjadinya tabrakan data karena mekanisme jaringan relative sederhana dan jika salah satu node putus maka akan mengganggu kinerja dan trafik seluruh jaringan.

2. *Ring*

Topologi cincin adalah topologi jaringan berbentuk rangkaian titik yang masing-masing terhubung ke dua titik lainnya, sedemikian sehingga membentuk

jalur melingkar membentuk cincin. Pada topologi cincin, komunikasi data dapat terganggu jika satu titik mengalami gangguan.

3. *Star*

Topologi bintang merupakan bentuk topologi jaringan yang berupa *konvergensi* dari *node* tengah ke setiap *node* atau pengguna. Topologi jaringan bintang termasuk topologi jaringan dengan biaya menengah. (Mardison, Al Husni ; 2012 : 59)

4. *Mesh*

Topologi jala atau Topologi mesh adalah suatu bentuk hubungan antar perangkat dimana setiap perangkat terhubung secara langsung ke perangkat lainnya yang ada di dalam jaringan. Akibatnya, dalam topologi *mesh* setiap perangkat dapat berkomunikasi langsung dengan perangkat yang dituju (*dedicated links*).

5. Pohon

Topologi Pohon adalah kombinasi karakteristik antara topologi bintang dan topologi bus. Topologi ini terdiri atas kumpulan topologi bintang yang dihubungkan dalam satu topologi bus sebagai jalur tulang punggung atau *backbone*. Komputer-komputer dihubungkan ke *hub*, sedangkan *hub* lain di hubungkan sebagai jalur tulang punggung. Topologi jaringan ini disebut juga sebagai topologi jaringan bertingkat. Topologi ini biasanya digunakan untuk interkoneksi antar sentral dengan hirarki yang berbeda. Untuk hirarki yang lebih rendah digambarkan pada lokasi yang rendah dan semakin keatas mempunyai

hirarki semakin tinggi. Topologi jaringan jenis ini cocok digunakan pada sistem jaringan komputer. (Mardison, Al Husni ; 2012 : 59)

6. Linier

Jaringan komputer dengan topologi runtut (*linear topology*) biasa disebut dengan topologi bus beruntut, tata letak ini termasuk tata letak umum. Satu kabel utama menghubungkan tiap titik sambungan (komputer) yang dihubungkan dengan penyambung yang disebut dengan Penyambung-T dan pada ujungnya harus diakhiri dengan sebuah penamat (*terminator*). Penyambung yang digunakan berjenis BNC (*British Naval Connector*: Penyambung *Bahari Britania*), sebenarnya BNC adalah nama penyambung bukan nama kabelnya, kabel yang digunakan adalah RG 58 (Kabel *Sepaksi Thinnet*). Pemasangan dari topologi bus beruntut ini sangat sederhana dan murah tetapi sebanyaknya hanya dapat terdiri dari 5-7 komputer.

II.6. *Wifi*

Awalnya *wifi* ditujukan untuk pengunaan perangkat *nirkabel* dan Jaringan Area Lokal (LAN), namun saat ini lebih banyak digunakan untuk mengakses *internet*. Hal ini memungkinan seseorang dengan komputer dengan kartu *nirkabel* (*wireless card*) atau *personal digital assistant* (PDA) untuk terhubung dengan *internet* dengan menggunakan titik akses (*hotspot*) terdekat. *Wifi* dirancang berdasarkan spesifikasi IEEE 802.11. Sekarang ini ada empat variasi dari 802.11, yaitu: 802.11a, 802.11b, 802.11g, and 802.11n. Spesifikasi b merupakan produk pertama *wifi*. Variasi g dan n merupakan salah satu produk yang memiliki

penjualan terbanyak pada 2005. Adapun spesifikasi *wifi* terdapat pada Tabel II. 1. Berikut :

Tabel II.1. Spesifikasi *Wifi*

Spesifikasi Wifi			
Spesifikasi	Kecepatan	Frekuensi Band	Cocok Dengan
802.11b	11 Mb/s	2.4 GHz	b
802.11a	54 Mb/s	5 GHz	A
802.11g	54 Mb/s	2.4 GHz	b,g
802.11n	100 Mb/s	2.4 GHz	B,g,n

Sumber: (Sony Bahagia Sinaga ; 2012 : 47)

Versi *wifi* yang paling luas dalam pasaran AS sekarang ini (berdasarkan dalam IEEE 802.11b/g) beroperasi pada 2.400 MHz sampai 2.483,50 MHz. Dengan begitu mengijinkan operasi dalam 11 *channel* (masing-masing 5 MHz), berpusat di frekuensi berikut :

1. Channel 1 - 2,412 MHz
2. Channel 2 - 2,417 MHz
3. Channel 3 - 2,422 MHz
4. Channel 4 - 2,427 MHz
5. Channel 5 - 2,432 MHz
6. Channel 6 - 2,437 MHz
7. Channel 7 - 2,442 MHz
8. Channel 8 - 2,447 MHz

9. Channel 9 - 2,452 MHz
10. Channel 10 - 2,457 MHz
11. Channel 11 - 2,462 MHz

Secara teknis operasional, *wifi* merupakan salah satu varian teknologi komunikasi dan informasi yang bekerja pada jaringan dan perangkat WLAN (*wireless local area network*). Dengan kata lain, *wifi* adalah sertifikasi merek dagang yang diberikan pabrikan kepada perangkat telekomunikasi (*internet*) yang bekerja di jaringan WLAN dan sudah memenuhi kualitas kapasitas interoperasi yang dipersyaratkan. Teknologi *internet* berbasis *wifi* dibuat dan dikembangkan sekelompok insinyur Amerika Serikat yang bekerja pada *Institute of Electrical and Electronis Engineers* (IEEE) berdasarkan standar teknis perangkat bernomor 802.11b, 802.11a dan 802.16. Perangkat *wifi* sebenarnya tidak hanya mampu bekerja di jaringan WLAN, tetapi juga di jaringan *Wireless Metropolitan Area Network* (WMAN). Karena perangkat dengan standar teknis 802.11b diperuntukkan bagi perangkat WLAN yang digunakan di frekuensi 2,4 GHz atau yang lazim disebut frekuensi ISM (*Industrial*, *Scientific* dan *Medical*). Sedang untuk perangkat yang berstandar teknis 802.11a dan 802.16 diperuntukkan bagi perangkat WMAN atau juga disebut Wi-Max, yang bekerja di sekitar pita frekuensi 5 GHz. (Sony Bahagia Sinaga ; 2012 : 47-48)

II.7. *Client Server*

Client – Server adalah bentuk *distributed computing* dimana sebuah program (*client*) berkomunikasi dengan program lain (*server*) dengan tujuan untuk bertukar informasi, pada umumnya sebuah *client* memiliki tugas sebagai berikut :

1. Menterjemahkan permintaan pengguna ke dalam bentuk *protocol* yang sesuai.

2. Mengirimkan permintaan pengguna ke *server*.

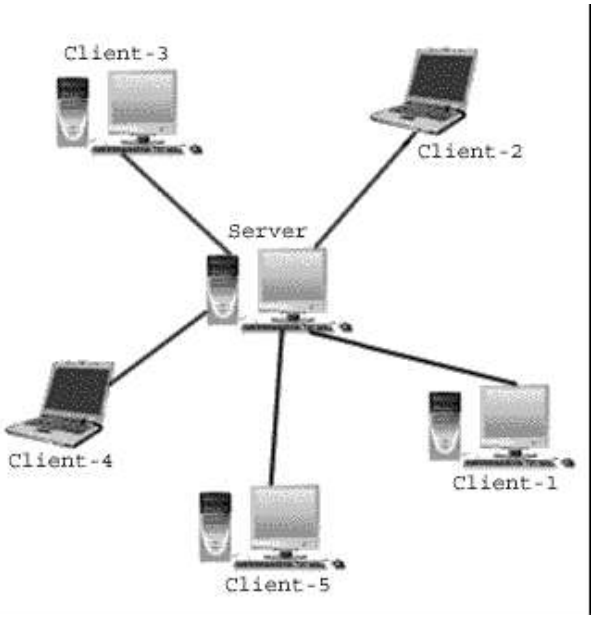
3. Menunggu respon dari *server*.

Kata *client* juga sering disebut dengan kata *host* yang menandakan bahwa *device* tersebut tersambung dalam sebuah jaringan. Sedangkan sebuah *server* memiliki tanggung jawab sebagai berikut :

1. Mendengarkan permintaan dari *client*.

2. Memproses permintaan tersebut.

3. Mengembalikan hasil proses tersebut ke client. (*Painem* ; 2013 : 16-17)



Gambar II.2. Model *Client Server*
Sumber : (Painem ; 2013 : 17)

II.8. UML *(Unified Modelling Language)*

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat

membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock,* dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson*, yang merupakan tiga tokoh yang boleh dikata

metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease *draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. *Booch, Rumbaugh* dan *Jacobson* menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek. (Yuni Sugiarti ; 2013 : 33)

Dalam pembuatan skripsi ini penulis menggunakan diagram *Use Case* yang terdapat di dalam UML. Adapun maksud dari *Use Case* Diagram diterangkan dibawah ini.

1. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include*

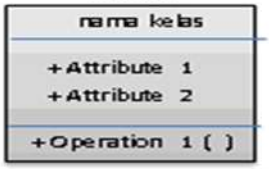
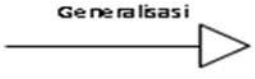
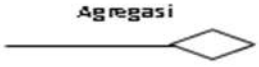
fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend* *use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. (Yuni Sugiarti ; 2013 : 41)

Simbol	Deskripsi
Use Case	fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit dan aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama use case
Aktor nama aktor	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / association	komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor
Extend <<extend>>	relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walaupun tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, arah panah menunjukan pada use case yang dituju contoh : <<extend>>
Include <<include>>	relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini. Ada dua sudut pandang yang cukup besar mengenai include di use case, include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, contoh : <<include>>

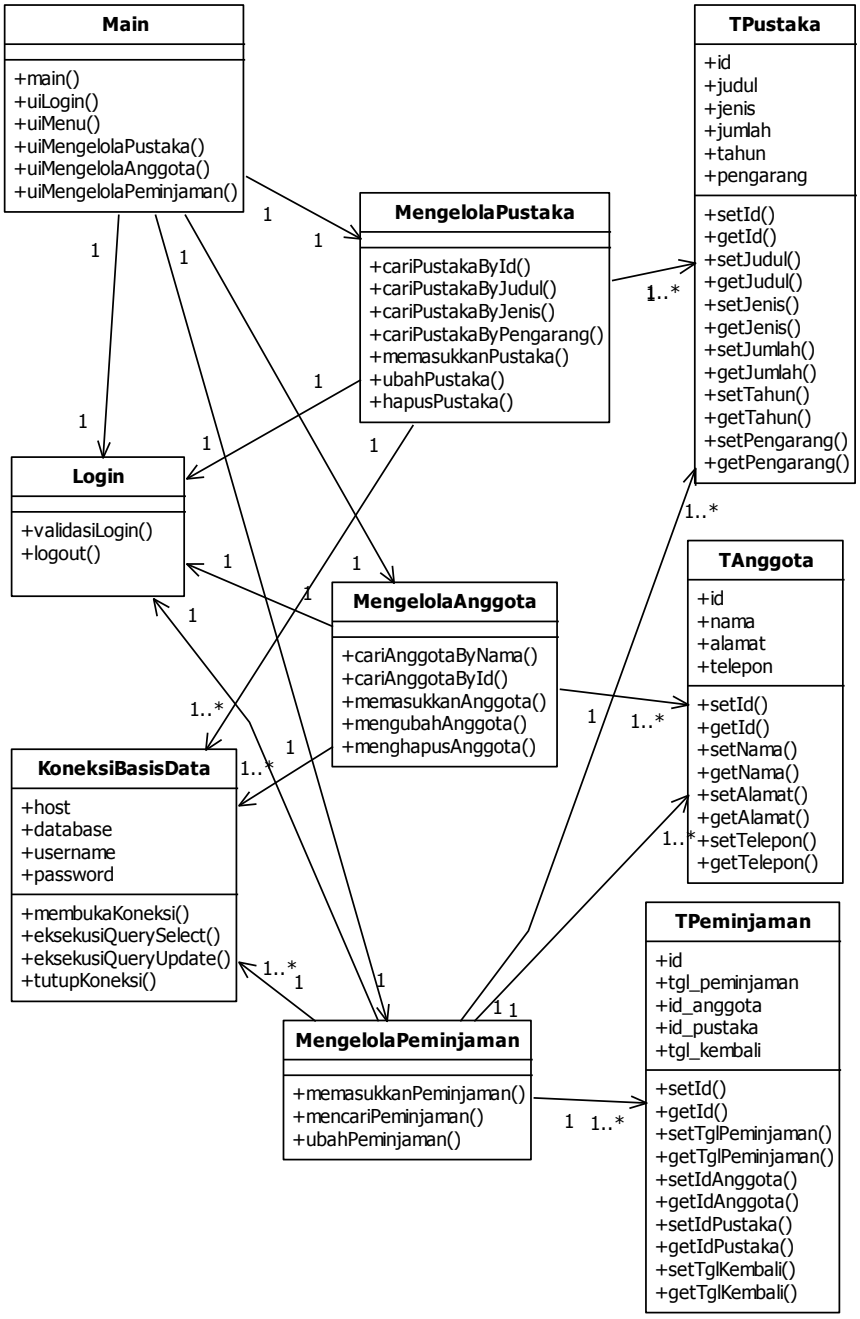
Gambar II.3. *Use Case* Diagram
Sumber : (Yuni Sugiarti ; 2013 ; 42)

2. *Class Diagram*

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah simbol-simbol pada diagram kelas :

Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem
Antarmuka / interface 	sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi 	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
Asosiasi berarah/directed asosiasi	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
Generalisasi 	relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / defedency 	relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi 	relasi antar kelas dengan makna semua-bagian (whole-part)

Gambar II.4. *Class* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 59)



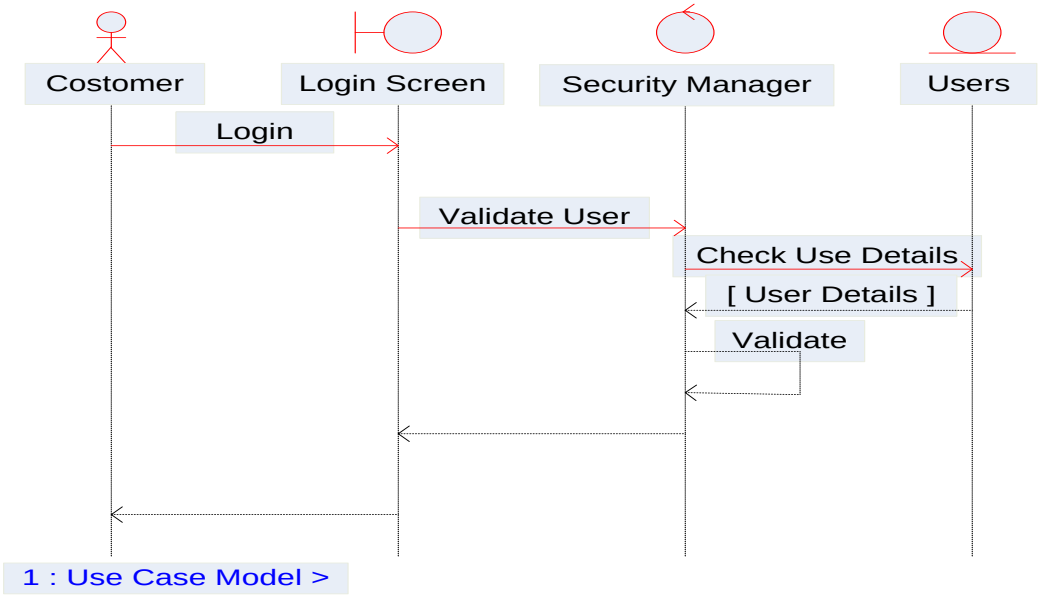
Gambar II.5. Contoh Class Diagram

Sumber : (Yuni Sugiarti ; 2013 : 63)

3. *Sequence Diagram*

Diagram *Sequence* menggambarkan kelakuan/prilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.



Gambar II.6. Contoh *Sequence* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 63)

4. *Activity Diagram*

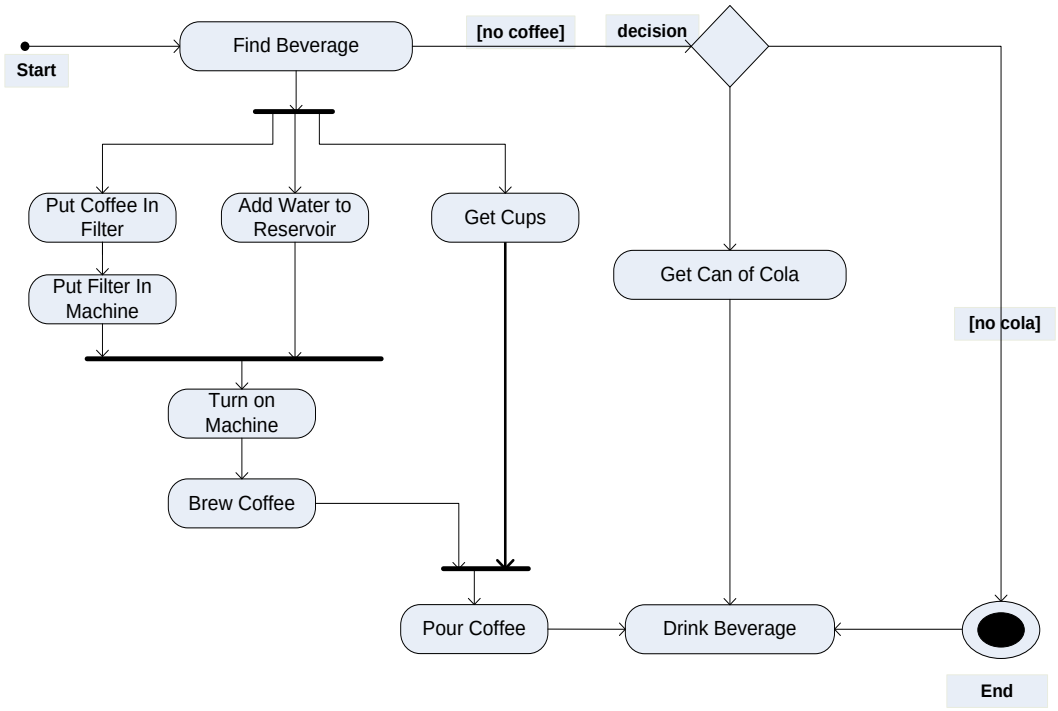
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar II.7. Activity Diagram
Sumber : (Yuni Sugiarti ; 2013 : 76)