

BAB 2

LANDASAN TEORI

2.1 Mixer

Mixer adalah sebuah alat yang digunakan untuk menyatukan, memproses dan menyalurkan kembali sinyal suara yang sudah diolah. Mixer dapat dipergunakan secara luas, dan tidak hanya dipergunakan untuk menggabung-gabungkan suara akan tetapi dapat pula kita pergunakan untuk mengolah suara menjadi terdengar lebih baik [7]. Mixer audio menerima berbagai sumber suara bisa dari microphone, alat music, CD player, tape deck, atau DAT. Dari sini dengan mudah dapat dilakukan pengaturan level masukan dan keluaran mulai dari yang sangat lembut sampai keras. Salah satu syarat terpenting dalam audio mixer yang baik adalah mempunyai input gain yang baik, pengaturan equalizer yang juga baik. Maka dengan demikian akan dapat dilakukan pengaturan yang lebih sempurna dan optimal terhadap setiap input signal, atau apapun yang menjadi sumber suaranya.

Macam-macam mixer ada dua jenis :

1. Mixer Digital

Mixer Digital adalah yang pengolahan sinyal yang melaluinya dalam bentuk digital atau dalam bentuk data digital. Keuntungan utama mixer digital adalah pengaturan menggunakan program dan terdapat recalled untuk memanggil kembali. Kebanyakan mixer digital memiliki beberapa jenis memori yang pengaturan dapat disimpan dan kemudian langsung ingat setiap kali mereka dibutuhkan. Ini bisa menjadi keuntungan luar biasa dalam fasilitas ruang perjamuan misalnya, di mana pengaturan yang diperlukan mungkin sering berubah untuk berbagai jenis acara dan konfigurasi ruangan yang berbeda. Bahkan jika beberapa fine tuning diperlukan, kemampuan untuk hanya mengingat satu set lengkap parameter dasar yang dekat dengan apa yang dibutuhkan dapat secara dramatis mengurangi waktu dan upaya yang diperlukan untuk mengatur untuk sebuah event.

Dan jika terjadi kesalahan, mudah untuk kembali ke pengaturan dasar,

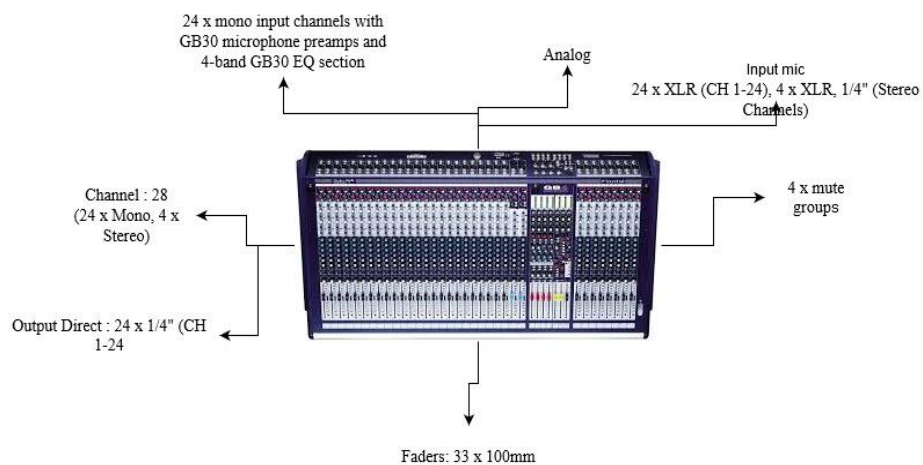
selain itu fitur mixing dan pengolahan langsung di mixer digital. Teknologi digital telah melakukan pengembangan sejumlah fungsi yang tidak tersedia dalam sistem mixer analog. Automatic feedback suppression adalah salah satu contoh. Terdapat dalam “Automated Mix Functions”. Mixer digital dirancang memiliki pemrosesan sinyal seperti efek (reverb, delay, compressor, gate) yang dirancang sesuai dengan aplikasi yang dimaksudkan. Hal ini membuat tidak perlu untuk membeli perangkat pemrosesan sinyal eksternal tambahan, sehingga secara signifikan mengurangi biaya sistem instalasi secara keseluruhan. Ini adalah kelemahan dari sistem mixer analog yang peralatan tambahan harus dibeli dan dipasang untuk menyediakan fungsi pemrosesan sinyal yang dibutuhkan oleh setiap aplikasi individu. Ada mixer analog yang mencakup beberapa fungsi pengolahan built-in, tapi tidak seperti mixer digital di mana semua proses yang diperlukan dapat dilakukan secara langsung dengan menggunakan software yang tersedia didalam mixer digital, kemampuan proses tambahan memiliki pengaruh langsung pada biaya dan ukuran fisik sebuah mixer analog.



Gambar 2.1 Yamaha M7CL Series Digital Mixer

2. Mixer Analog

Mixer Analog adalah mixer yang mengolah sinyal yang melaluinya dalam bentuk analog atau dengan mengubah variabel besaran arus listrik yang melaluinya. Mixer digital tidak selalu bentuk hardware-nya virtual atau hanya dapat kita lihat dalam tampilan layar komputer saja, melainkan banyak pula yang bentuk hardware-nya dapat kita pegang atau kita set [7]. Kekurangan pada audio mixer analog adalah apabila semua sinyal input pada saat yang sama sedang maksimum, maka sinyal pencampuran nya akan menjadi besar sehingga terjadi distorsi akibat clipping, clipping adalah kondisi sinyal yang terpotong akibat amplitudo yang terlalu besar. Keuntungan utama mixer analog adalah turunkan biaya untuk satu set fitur yang terbatas. Jika hanya penggunaan yang dasar, mixer analog sederhana mungkin menjadi pilihan yang lebih ekonomis dari mixer digital, selain itu operasi mudah untuk pengguna pertama kali. Mixer analog biasanya memiliki satu kontrol per fungsi, yang semuanya terlihat dan langsung dapat diakses melalui panel kontrol. Tata letak kontrol logis mengikuti aliran sinyal mixer dan karena itu relatif mudah dipahami. Jenis logis, operasi mudah bisa menjadi keuntungan di aula umum dan sekolah, misalnya, di mana berbagai orang, beberapa memiliki pengalaman sedikit atau tidak ada sebelumnya, mungkin perlu untuk mengoperasikan sistem. Dalam situasi seperti itu mungkin perlu untuk melindungi kontrol dan fungsi yang tidak boleh diubah dengan sampul keamanan.

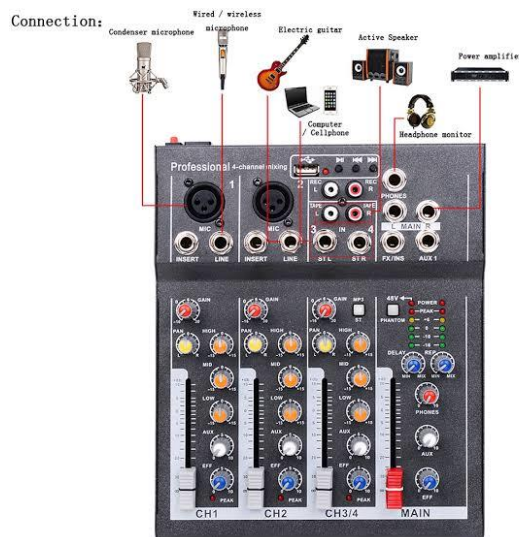


Gambar 2.2 Mixer Analog Soundcraft GB4

2.1.1 Channel

Channel merupakan bagian penting di konsol ini dan jumlah inputnya baik untuk mikrofon dan alat musik atau perangkat lain seperti amplifier, preamp, atau signal processors memengaruhi sinyal audio yang bisa ditangani sebuah mixer. Mixer audio juga tidak selalau menerapkan input channel yang sama, beberapa ada yang menerapkan monaural only (mono) dan ada pula menyematkan stereo.

Jumlah channel juga bisa menjadi patokan awal untuk menentukan mixer yang akan kamu beli atau gunakan.



Gambar 2.3 channel Mixer

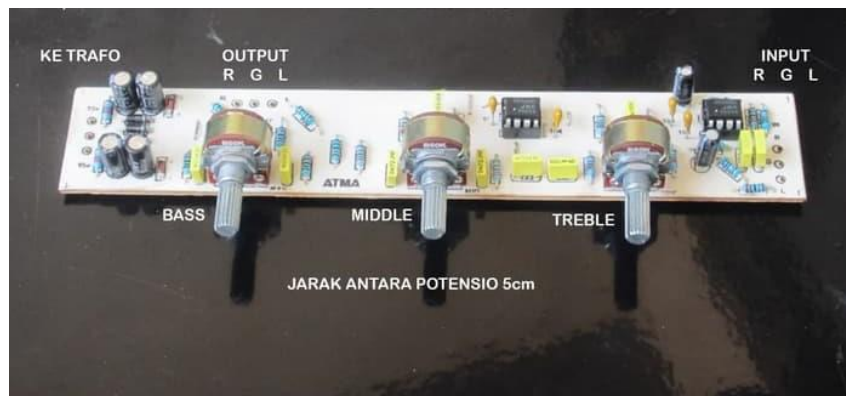
2.1.2 Equalizer (EQ)

EQ adalah kependekan dari equalizer. Fitur ini merupakan perangkat yang digunakan untuk mengubah atau menyesuaikan frekuensi konten atau tone tiap channel atau band. Layaknya switch geser atau knob volume, EQ menjadi fitur yang selalu ada ada di setiap mixer audio, mulai dari yang portabel hingga kelas studio. Namun, yang membedakannya adalah jumlahnya. Beberapa ada yang hanya menggunakan dua band, sementara untuk yang versi lengkap sudah menerapkan EQ tiga band atau lebih yang kadang juga ditunjang dengan MID Sweep. Singkatnya, semakin banyak frekuensi yang band disematkan maka semakin banyak dan detail sound yang bisa diproduksi. Contohnya, bila membutuhkan produksi dengan kualitas audio studio maka membutuhkan mixer multi-band, tapi

bila hanya membutuhkan kemampuan mixing yang simpel maka band untuk mid, high, dan bass juga sudah layak.

2.1.2.1 Bass

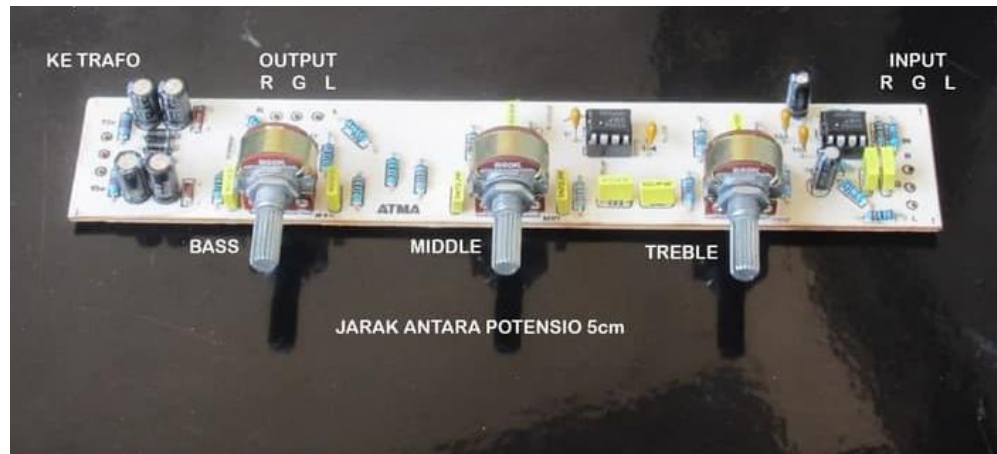
Bass atau biasa disebut “*low*” akan menghasilkan suara yang nge-bass, bass biasanya memiliki frekuensi hingga 150 Hz, atau 150 osilasi perdetik. Suara bass akan dirasakan ketika suara keras biasanya menggunakan speaker tipe subwoofer, penting untuk disadari bahwa peningkatan bass tidak menciptakan suara baru, tetapi justru meningkatkan sensitivitas speaker.



Gambar 2.4 Tone control bass

2.1.2.2 Treble

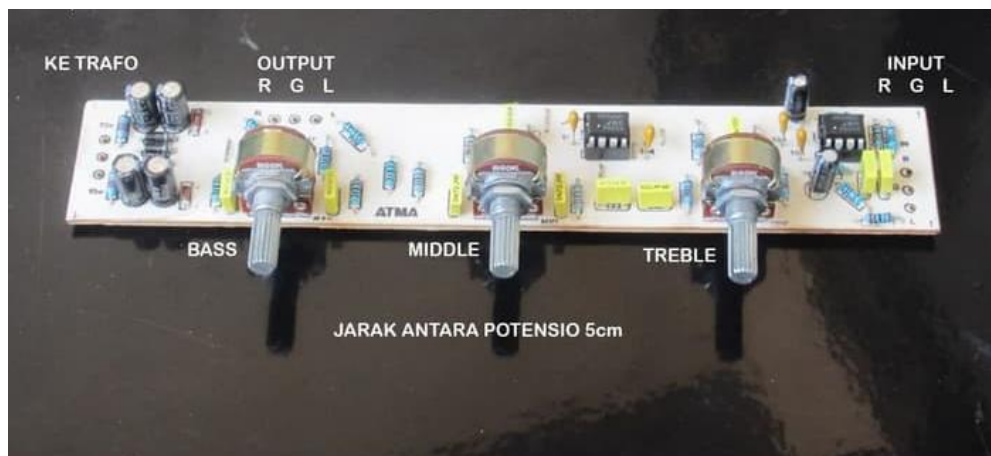
Treble adalah jangkauan suara tinggi yang membentuk karakter suara menghentak/mempertegas. Treble memiliki frekuensi dari 6KHz atau 6000 osilasi per detik hingga batas pendengara sekitar 20 KHz. Kontrol treble akan mengubah sensitivitas sistem ke frekuensi yang lebih tinggi , jadi dengan mengontrol treble membuat segalanya terdengar lebih cerah dan lebih rinci diumpamakan sebagai attack/serangan. menghentak, menyerang, dan mempertegas nada. high tidak boleh terlalu besar dibuka. knob high pada bass difungsikan untuk mempertegas artikulasi nada dan mendukung karakteristik permainan seorang bassist. high yang terlalu besar membuat sound jadi kedengar kasar dan terlalu garing.



Gambar 2.5 Tone control treble

2.1.2.3 Middle

Middle sesuai dengan artinya ‘tengah’ ia akan mengontrol seluruh karakter suara. Middle memiliki frekuensi 200 Hz – 5 KHz, meningkatkan bass dan treble akan membuat frekuensi middle berkurang, mengurangi treble dan bass akan secara efektif meningkatkan frekuensi middle. Sesuai kebutuhan dan selera pemain, middle adalah jangkauan suara menengah yang membentuk karakter suara yang memperjelas artikulasi nada atau clarity dari nada. Mid seperti pelafalan huruf vocal orang yang sedang bicara. harus bisa mendefinisikan artikulasi nada secara jelas.



Gambar 2.6 Tone control middle

2.2 *Internet of Things (IoT)*

Internet of Things (IoT) merupakan sebuah konsep yang bertujuan untuk memperluas manfaat dari konektivitas internet yang tersambung secara terus menerus. Adapun kemampuan seperti berbagi data remote control, dan termasuk juga pada benda dunia nyata. Bahan pangan, elektronik, peralatan apa saja, koleksi termasuk benda hidup, yang semuanya tersambung ke jaringan lokal dan global melalui sensor tertanam dan selalu “on” [8].

Interaksi antara manusia dengan manusia sudah sangat biasa sejak zaman dahulu kala. Interaksi antara manusia dengan mesin sudah biasa pula, sejak adanya penemuan teknologi seperti komputer atau gadget devices lainnya. Namun, Menurut analisa McKinsey Global Institute, *internet of things* adalah sebuah teknologi yang memungkinkan kita untuk menghubungkan mesin, peralatan, dan benda fisik lainnya dengan sensor jaringan dan aktuator untuk memperoleh data dan mengelola kinerjanya sendiri, sehingga memungkinkan mesin untuk berkolaborasi dan bahkan bertindak berdasarkan informasi baru yang diperoleh secara independen. Sebuah publikasi mengenai *Internet of things* in 2020 menjelaskan bahwa *internet of things* adalah suatu keadaan ketika benda memiliki identitas, bisa beroperasi secara intelijen, dan bisa berkomunikasi dengan sosial, lingkungan, dan pengguna.



Gambar 2.7 Ilustrasi dari Internet of Things [8]

Dapat kita simpulkan bahwa *internet of things* membuat suatu koneksi antara mesin dengan mesin, sehingga mesin-mesin tersebut dapat berinteraksi dan bekerja secara independen sesuai dengan data yang diperoleh dan diolahnya secara mandiri. Tujuannya adalah untuk membuat manusia berinteraksi dengan benda dengan lebih mudah, bahkan supaya benda juga bisa berkomunikasi dengan benda lainnya.

Teknologi *internet of things* sangat luar biasa. Jika sudah direalisasikan, teknologi ini tentu akan sangat memudahkan pekerjaan manusia.

Manusia tidak akan perlu lagi mengatur mesin saat menggunakannya, tetapi mesin tersebut akan dapat mengatur dirinya sendiri dan berinteraksi dengan mesin lain yang dapat berkolaborasi dengannya.

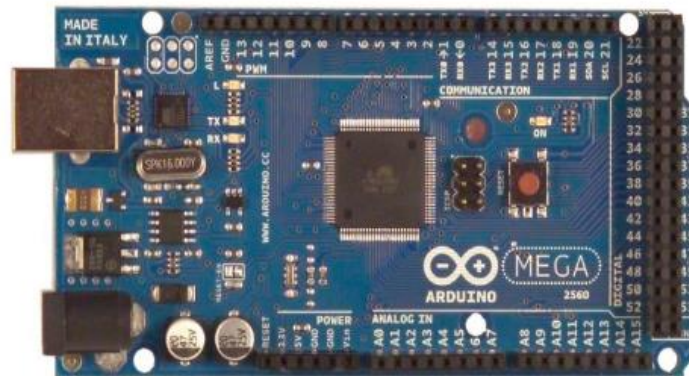
Cara kerja dari *internet of things* cukup mudah. Setiap benda harus memiliki sebuah IP Address. IP Address adalah sebuah identitas dalam jaringan yang membuat benda tersebut bisa diperintahkan dari benda lain dalam jaringan yang sama. Selanjutnya, IP address dalam benda-benda tersebut akan dikoneksikan ke jaringan internet. Saat ini, koneksi internet sudah sangat mudah kita dapatkan. Dengan demikian, kita dapat memantau benda tersebut bahkan memberi perintah kepada benda tersebut. Sebagai contoh, jika ada speaker yang memiliki IP address dan terkoneksi internet di Amerika Serikat, kita dapat memerintahkan speaker tersebut untuk menyalakan musik walaupun kita berada di Indonesia. Yang kita perlukan hanyalah koneksi internet. Lalu apa hubungannya dengan *internet of things*, Setelah sebuah benda memiliki IP address dan terkoneksi dengan internet, pada benda tersebut juga dipasang sebuah sensor. Sensor pada benda memungkinkan benda tersebut memperoleh informasi yang dibutuhkan, benda tersebut dapat mengolah informasi itu sendiri, bahkan berkomunikasi dengan benda-benda lain yang memiliki IP address dan terkoneksi dengan internet juga. Akan terjadi pertukaran informasi dalam komunikasi antara benda-benda tersebut.

Setelah pengolahan informasi selesai, benda tersebut dapat bekerja dengan sendirinya, atau bahkan memerintahkan benda lain juga untuk ikut bekerja. Jadi, dalam *internet of things* manusia akan bertindak sebagai raja dan akan dilayani oleh benda-benda disekitarnya [8].

2.3 Arduino Mega 2560

Arduino Mega 2560 adalah papan pengembangan mikrokontroler yang berbasis Arduino dengan menggunakan chip ATmega2560. Board ini memiliki pin I/O yang cukup banyak, sejumlah 54 buah digital I/O pin (15 pin diantaranya adalah PWM), 16 pin analog input, 4 pin UART (serial port hardware). Arduino Mega 2560 dilengkapi dengan sebuah oscillator 16 Mhz, sebuah port USB, power jack DC, ICSP header, dan tombol reset.

Board ini sudah sangat lengkap, sudah memiliki segala sesuatu yang dibutuhkan untuk sebuah mikrokontroler. [9].



Gambar 2.8 Arduino Mega 2560

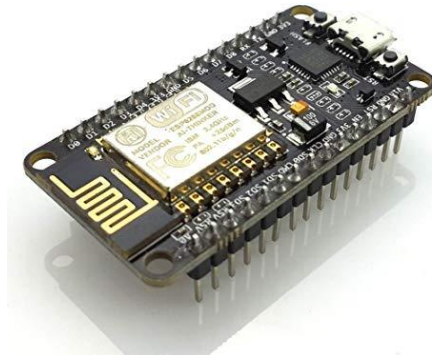
(Sumber : ArduinoMega2560Datasheet.pdf)

Dengan penggunaan yang cukup sederhana, anda tinggal menghubungkan power dari USB ke PC anda atau melalui adaptor AC/DC ke jack DC.

(Sumber : ArduinoMega2560Datasheet.pdf).

2.4 NodeMCU

NodeMCU pada dasarnya adalah pengembangan dari ESP 8266 dengan firmware berbasis e-Lua. Pada NodeMcu dilengkapi dengan micro usb port yang berfungsi untuk pemrograman maupun power supply. Selain itu juga pada NodeMCU di lengkapi dengan tombol push button yaitu tombol reset dan flash. NodeMCU menggunakan bahasa pemrograman Lua yang merupakan package dari esp8266. Bahasa Lua memiliki logika dan susunan pemrograman yang sama dengan c hanya berbeda syntax. Jika menggunakan bahasa Lua maka dapat menggunakan tool Lua loader maupun Lua uploder. Selain dengan bahasa Lua NodeMCU juga support dengan software Arduino IDE dengan melakukan sedikit perubahan board manager pada Arduino IDE [10].



Gambar 2.9 Modul Nodemcu

(Sumber : Dian Mustika P. 2017)

2.5 Konsep Perancangan Berorientasi Objek

Pendekatan perancangan berorientasi objek menganalogikan sistem aplikasi seperti kehidupan yang memiliki elemen-elemen objek didalamnya. Dalam membangun sistem yang berorientasi objek, langkah awal yang dilakukan adalah melakukan proses analisis dan perancangan yang berorientasi pada objek-objek yang terlibat didalam sistem. Tujuan dilakukannya proses tersebut adalah untuk mempermudah pengembang dalam mendesain program dalam bentuk objek-objek dan hubungan antar objek tersebut yang kemudian dimodelkan dalam sistem nyata. Perusahaan software Rational Software, telah membentuk konsorsium dengan berbagai organisasi untuk meresmikan pemakaian *Unified Modelling Language* (UML) sebagai Bahasa standar dalam *Object Oriented Analysis Design* (OOAD) [11].

2.5.1 Unified Modelling Language (UML)

Unified Modeling Language (UML) adalah salah satu alat bantu pembangunan perangkat lunak yang memiliki keunggulan dengan konsep pengembangan sistem yang berorientasi objek. UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem membuat cetak biru atau mekanisme yang efektif untuk berbagi dan mengkomunikasikan rancangan mereka dengan unsur yang lain.

UML dalam sebuah bahasa untuk menentukan visualisasi, konstruksi, dan mendokumentasikan artifact dari sistem software, untuk memodelkan bisnis, dan sistem non-software lainnya. UML merupakan sistem arsitektur yang bekerja dalam OOAD (*Object Oriented Analysis and Design*) dengan satu bahasa yang konsisten untuk menentukan, visualisasi, konstruksi dan mendokumentasikan artifact yang terdapat dalam sistem. Artifact adalah potongan informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa software. Artifact dapat berupa model, deskripsi atau software [11].

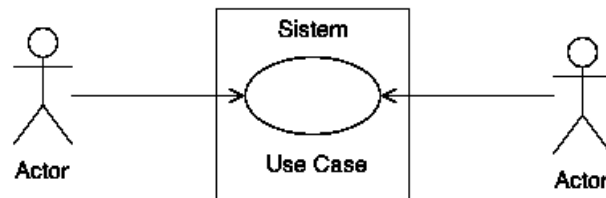
2.5.1.1 Diagram UML

UML dalam mendokumentasikan rancangan menyediakan berbagai macam diagram untuk memodelkan aplikasi perangkat lunak berorientasi objek [11]. Namun, dalam penelitian ini hanya menggunakan empat macam diagram saja untuk memodelkannya. Empat macam diagram yang digunakan diantaranya yaitu:

1. Use Case Diagram

Use case diagram adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara user (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras atau urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh tujuan umum pengguna. *Use case* merupakan alat bantu guna menstimulasi pengguna potensial untuk mengatakan tentang suatu sistem dari sudut pandangnya. Tidak selalu mudah bagi pengguna untuk menyatakan bagaimana mereka bermaksud menggunakan sebuah sistem. Karena sistem pengembangan tradisional sering ceroboh dalam melakukan analisis, akibatnya pengguna seringkali sulit menemukan jawaban tatkala dimintai masukan tentang sesuatu.

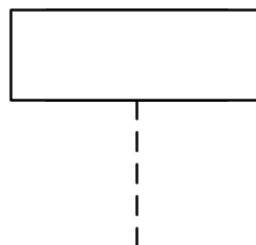
Diagram *use case* menunjukkan 3 aspek dari sistem yaitu : *actor*, *use case* dan *sistem* atau *sub sistem boundary*. *Actor* mewakili peran orang, sistem yang lain atau alat ketika berkomunikasi dengan *use case*.



Gambar 2.10 Use Case Model [11]

2. Sequence Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukkan sejumlah contoh objek dan *message* (pesan) yang diletakan diantara objek-objek ini di dalam *use case*. Komponen utama *sequence diagram* terdiri atas objek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical*. Objek diletakan di dekat bagian atas diagram dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan diagram, istilah objek dikenal juga dengan *participant*, setiap *participant* terhubung dengan garis titik-titik yang disebut *lifeline*. Sepanjang *lifeline* ada kotak yang disebut *activation*, *activation* mewakili sebuah eksekusi operasi dari *participant*. Panjang kotak ini berbanding lurus dengan durasi *activation*.

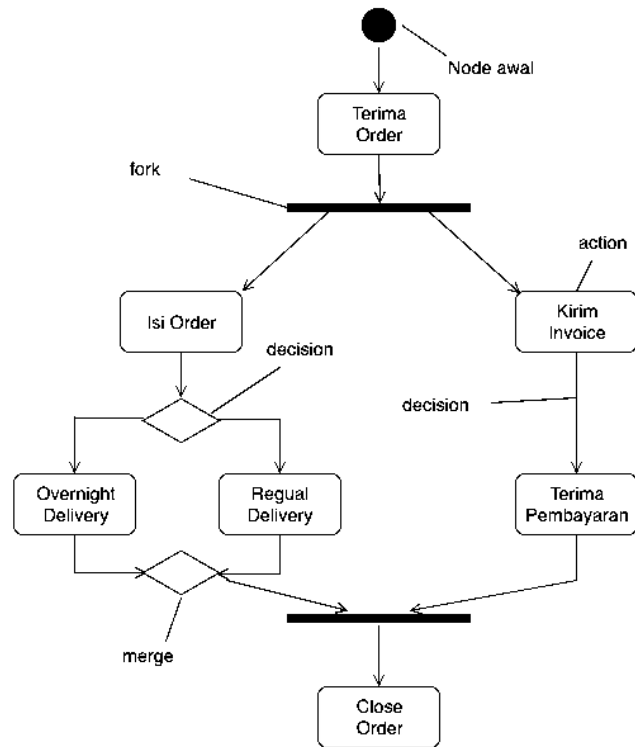


Gambar 2.11 Participant pada sebuah sequence diagram [11]

3. Activity Diagram

Activity diagram adalah bagian penting dari UML yang menggambarkan aspek dinamis dari sistem. Logika prosedural, proses bisnis dan aliran kerja suatu bisnis bisa dengan mudah dideskripsikan dalam *activity diagram*. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa. Tujuan dari *activity diagram* adalah untuk menangkap tingkah laku dinamis dari sistem dengan cara menunjukkan aliran pesan dari satu aktifitas ke aktifitas lainnya. Secara umum *activity diagram* digunakan untuk menggambarkan diagram alir yang terdiri dari banyak aktifitas dalam sistem dengan beberapa fungsi tambahan seperti percabangan, aliran paralel, *swim lane*, dan sebagainya. Sebelum menggambarkan sebuah *activity diagram*, perlu adanya pemahaman yang jelas tentang elemen yang akan digunakan dalam *activity diagram*.

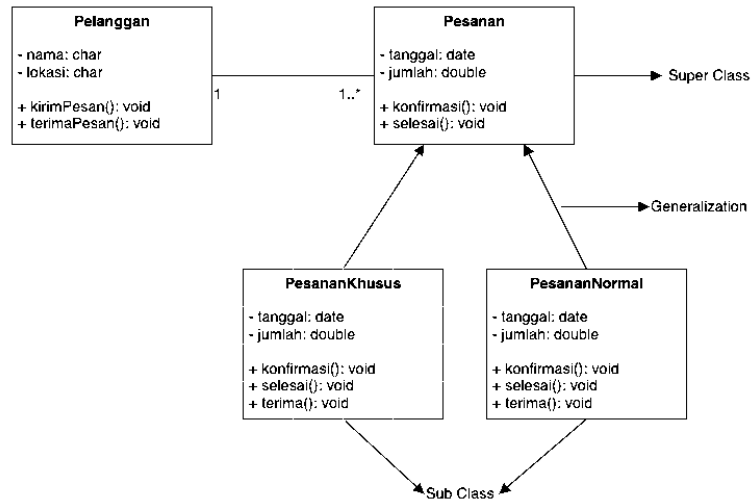
Elemen utama dalam *activity diagram* adalah aktifitas itu sendiri. Aktifitas adalah fungsi yang dilakukan oleh sistem setelah aktifitas teridentifikasi, selanjutnya yang perlu diketahui adalah bagaimana semua elemen tersebut berasosiasi dengan *constraint* dan kondisi lalu perlu penjabaran tata letak dari keseluruhan aliran agar bisa ditransformasikan ke *activity diagram*.



Gambar 2.12 Contoh activity diagram sederhana [11]

4. Class diagram

Class diagram adalah diagram statis. Diagram ini mewakili pandangan statis dari suatu aplikasi. *Class diagram* tidak hanya digunakan untuk memvisualisasikan, menggambarkan dan mendokumentasikan berbagai aspek sistem tetapi juga membangun kode eksekusi dari aplikasi perangkat lunak. *Class diagram* menggambarkan *atribut*, *operation* dan juga *constraint* yang terjadi pada sistem. *Class diagram* banyak digunakan dalam pemodelan sistem OO karena mereka adalah satu-satunya diagram UML yang dapat dipetakan langsung dengan bahasa berorientasi objek. *Class diagram* menunjukkan koleksi *class*, antarmuka, asosiasi, kolaborasi, dan *constraint*, dikenal juga sebagai diagram structural [10].



Gambar 2.13 Contoh class diagram sederhana [11]

2.6 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) merupakan teknik yang digunakan untuk memodelkan kebutuhan data dari suatu organisasi, biasanya oleh *System Analyst* dalam tahap analisis persyaratan proyek pengembangan sistem. ERD merupakan alat peraga yang memberikan dasar untuk desain *database* relasional yang mendasari sistem informasi yang dikembangkan. ERD bersama-sama dengan detail pendukung merupakan model data yang pada gilirannya digunakan sebagai spesifikasi untuk *database* [12].

Dalam *Entity Relationship Diagram* (ERD) terdapat beberapa komponen, yaitu sebagai berikut :

1. Entitas (*Entity*)

Entitas adalah suatu objek yang memiliki hubungan dengan objek lain. Dalam ERD digambarkan dengan bentuk persegi panjang.

2. Hubungan (*Relationship*)

Merupakan keterangan entitas dapat berhubungan dengan entitas lain, hubungan ini disebut dengan *entity relationship* yang digambarkan dengan garis. Terdapat 4 tipe relasi dalam *database* yaitu :

a. One-to-One

Artinya satu data memiliki satu data pasangan di tabel lain.

Jadi, jika dua tabel berelasi one-to-one artinya setiap *record* di entitas pertama hanya akan berhubungan dengan satu *record* di entitas kedua begitu pula sebaliknya. Contohnya relasi antara tabel pegawai dan alamat pegawai. Satu dokumen pegawai hanya berhubungan dengan satu *record* alamat pegawai begitu pula sebaliknya.

b. One-to-Many

Artinya satu data memiliki beberapa data pasangan di tabel lain. Jadi, misalkan terdapat relasi antara tabel satu dan tabel dua yang berarti satu dokumen pada tabel satu boleh berelasi (mempunyai) dengan banyak dokumen pada tabel dua. Namun, satu dokumen pada tabel dua hanya boleh berelasi dengan satu dokumen saja pada tabel satu.

c. Many-to-One

Artinya beberapa data memiliki satu data pasangan di tabel lain. Jadi, misalkan terdapat relasi antara tabel satu dengan tabel dua, dapat diartikan bahwa satu dokumen pada tabel satu hanya boleh berelasi (mempunyai) dengan satu dokumen pada tabel dua. Tetapi, satu dokumen pada tabel dua boleh berelasi dengan banyak dokumen pada tabel satu.

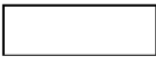



d. Many-to-Many

Artinya beberapa data memiliki beberapa data pasangan di tabel lain. Jadi, jika tabel satu berelasi dengan tabel dua dengan relasi many-to-many berarti bahwa ada banyak dokumen di entitas satu dan entitas dua yang saling berhubungan satu sama lain.

3. Atribut

Atribut adalah elemen dari entitas yang berfungsi sebagai deskripsi karakter entitas dan digambarkan dengan bentuk elips [12].

Tabel 2.1 Entity Relationship Diagram (ERD) [12]

| No | Simbol | Keterangan |
|----|---|------------|
| 1 |  | Entitas |
| 2 |  | Atribut |
| 3 |  | Hubungan |
| 4 |  | Garis |

2.7 Smartphone

Smartphone adalah telepon pintar yang memiliki kemampuan seperti komputer. Smartphone diklasifikasikan sebagai high end mobile phone yang dilengkapi dengan kemampuan mobile computing. Dengan kemampuan mobile computing tersebut, smartphone memiliki kemampuan yang tak bisa dibandingkan dengan ponsel biasa. Smartphone yang pertama kali muncul merupakan kombinasi dari fungsi suatu personal digital assistant (PDA) dengan telepon genggam ataupun telepon dengan kamera. Seiring dengan perkembangannya, kini smartphone juga mempunyai fungsi sebagai media player portable, low end digital compact camera, pocket video camera dan GPS. Smartphone modern juga dilengkapi dengan layar touch screen resolusi tinggi, browser yang mampu menampilkan full web seperti pada PC, serta akses data WiFi dan internet broadband [13].

2.7.1 Karakteristik Smartphone

Beberapa karakteristik yang umum ada pada smartphone yaitu:

1. Mobile OS

Mobile OS yang sering digunakan pada smartphone adalah:

Symbian OS, iPhone OS, Windows Mobile OS, RIM Blackberry, Linux, Palm OS, Android.

2. OpenSource

3. WebFeature

4. Enhanced Hardware

Fitur hardware eksternal seperti layar sentuh lebar dan sensitif, built-in keyboard, resolusi kamera tinggi, sisi kamera depan untuk video conferences.

5. MobilePC Pada umumnya smartphonememiliki prosesor yang cukup tinggi., selain itu memiliki penyimpanan memori yang besar dan memiliki RAM tambahan yang cukup besar seperti sebuah PC desktop atau laptop [13].

2.8 Android

Menurut Nasruddin Safaat H (Pemrograman aplikasi mobile smartphone dan tablet PC berbasis android 2012:1) android adalah sebuah sistem operasi pada handphone yang bersifat terbuka dan berbasis pada sistem operasi Linux. Android bisa digunakan oleh setiap orang yang ingin menggunakannya pada perangkat mereka. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang akan digunakan untuk bermacam peranti bergerak. Awalnya, GoogleInc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia [13].

2.8.1 Versi Android

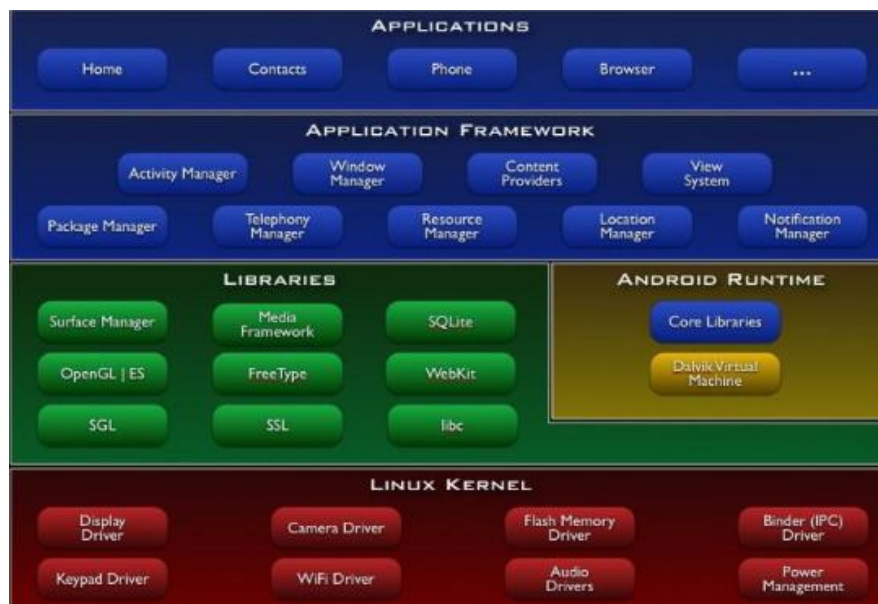
Android memiliki sejumlah pembaharuan semenjak rilis aslinya. Pembaharuan ini dilakukan untuk memperbaiki bug dan menambah fitur-fitur yang baru. Berikut merupakan versi-versi yang dimiliki Android sampai saat ini:

Tabel 2.2 Tabel Versi – versi Android

| Versi Android | API Level | Nickname |
|---------------|-----------|-----------------------|
| Android 1.0 | 1 | |
| Android 1.1 | 2 | |
| Android 1.5 | 3 | Cupcake |
| Android 1.6 | 4 | Donut |
| Android 2.0 | 5 | Eclair |
| Android 2.0.1 | 6 | Eclair |
| Android 2.1 | 7 | Eclair |
| Android 2.2 | 8 | Froyo (Frozen Yogurt) |
| Android 2.3 | 9 | Gingerbread |
| Android 2.3.3 | 10 | Gingerbread |
| Android 3.0 | 11 | Honeycomb |
| Android 3.1 | 12 | Honeycomb |
| Android 3.2 | 13 | Honeycomb |
| Android 4.0 | 14 | Ice Cream Sandwich |
| Android 4.0.3 | 15 | Ice Cream Sandwich |
| Android 4.1 | 16 | Jelly Bean |

2.8.2 Arsitektur Android

Android merupakan kernel Linux yang menyediakan dan mengatur alur proses aplikasi. Gambar 2.1 merupakan struktur dari sistem operasi Android.

**Gambar 2.14** Arsitektur Android

(Sumber : <http://developer.android.com/guide/basics/what-is-android.html>)

Arsitektur Android terdiri atas:

1. Application

Application merupakan bagian yang memuat aplikasi – aplikasi yang dapat digunakan oleh pengguna perangkat Android. Pada bagian Application ini Android memasukkan satu set aplikasi inti yang meliputi email client, program SMS, kalender, peta, browser, dan kontak. Selain aplikasi inti seperti yang terdapat pada arsitektur Android, aplikasi - aplikasi tambahan yang diinstal sendiri oleh pengguna juga akan menempati bagian Application dan memiliki hak akses yang sama terhadap Application Framework. (Developer Android, 2012).

2. Application Framework

Application Framework merupakan bagian yang dapat digunakan oleh pengembang aplikasi dalam membuat aplikasi android. Dengan menyediakan pengembangan platform yang terbuka, Android menawarkan kemampuan pengembangan untuk membangun aplikasi yang sangat kaya dan inovatif. Pengembang diberikan kebebasan untuk mengambil keuntungan dari perangkat keras, mengakses informasi, run background services, set alarm, serta menambahkan pemberitahuan ke status bar. Dalam mengembangkan aplikasi, pengembang memiliki akses ke framework API (Application Programming Interface) yang sama dengan yang dapat diakses oleh aplikasi – aplikasi inti dari android. (Developer Android, 2012)

3. Libraries

Libraries merupakan kumpulan kode yang dapat digunakan oleh komponen atau program lain. Penulisan kode pada Libraries ditulis menggunakan bahasa pemrograman C/C++.

Beberapa Libraries yang terdapat pada Android yaitu :

1. System C Library, implementasi BSD yang diturunkan dari sistem library standar C (libc).
2. Media Libraries, berdasarkan OpenCORE PacketVideo; library dapat

mendukung pemutaran dan perekaman audio dan format video, serta file gambar, termasuk MPEG4, H.264, MP3, AAC, AMR, JPG, dan PNG.

3. Surface Manager, library yang mengelola penggambaran dalam bentuk komposisi 2D, grafis 3D.
4. LibWebCore
5. SGL(Scalable Graphics Library), library yang menangani pengelolaan grafis 2D.
6. 3D libraries, library yang menangani pengelolaan grafis 3D.
7. FreeType, library yang menangani pengelolaan rendering font.
8. SQLite, digunakan untuk pengelolaan database.

(Developer Android, 2012).

4. Android Runtime

Android Runtime memiliki dua bagian utama, yaitu:

- a. CoreLibraries
- b. Dalvik Virtual Machine(DVM)

Aplikasi Android ditulis dengan menggunakan bahasa pemrograman Java. Dalvik Virtual Machine (DVM) adalah sebuah virtual machine yang digunakan untuk menterjemahkan instruksi –instruksi program Java ke dalam instruksi yang dimengerti oleh sistem operasi. Namun dalam platform Android virtual machine yang digunakan bukan Java Virtual Machine (JVM) tetapi Dalvik Virtual Machine (DVM). Dalvik Virtual Machine adalah sebuah virtual machine yang dioptimasi untuk perangkat yang memiliki memori kecil, sumber daya terbatas, dan kemampuan proses yang kecil. Dalvik Virtual Machine mengeksekusi file dalam bentuk Dalvik executable (. Dex).(Andry, 2011).

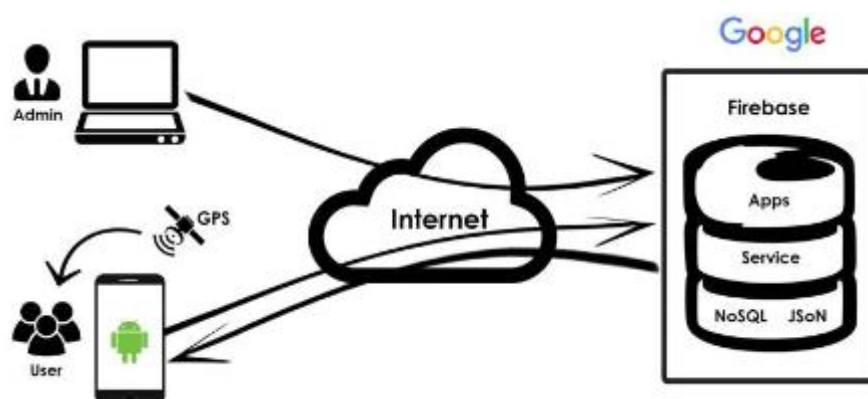
5. Linux Kernel

Dalam bagian ini android menggunakan modifikasi dari Linux Kernel versi 2.6. Bagian ini bertanggung jawab untuk mengelola dan berkomunikasi dengan perangkat keras dimana android berjalan. Pemilihan Linux Kernel sebagai inti dari android adalah karena dukungan dan kestabilannya terhadap berbagai macam komponen perangkat keras. Pada bagian ini disediakan driver (program pengendali) perangkat keras, pengelolaan memori, pengelolaan proses, pengelolaan jaringan, dan keamanan. (Developer Android, 2012).

2.9 Firebase

Firebase memiliki produk utama, yaitu menyediakan database realtime dan backend sebagai layanan (Backend as a Service). Layanan ini menyediakan pengembang aplikasi API yang memungkinkan aplikasi data yang akan disinkronisasi di klien dan disimpan di cloud Firebase ini. Firebase menyediakan library untuk berbagai client platform yang memungkinkan integrasi dengan Android, iOS, JavaScript, Java, Objective-C dan Node aplikasi Js dan dapat juga disebut sebagai layanan DbaaS (Database as a Service) dengan konsep realtime. Firebase digunakan untuk mempermudah dalam penambahan fitur-fitur yang akan dibangun oleh developer [14].

Dalam Gambar 2.12 ditunjukkan contoh arsitektur sistem Firebase dengan Android.



Gambar 2.15 Arsitektur Sistem Firebase

Semua data Firebase Realtime Database disimpan sebagai objek JSON. Bisa dianggap basis data sebagai JSON tree yang di-host di awan. Tidak seperti basis data SQL, tidak ada tabel atau rekaman. Ketika ditambahkan ke JSON tree, data akan menjadi simpul dalam struktur JSON yang ada. Meskipun basis data menggunakan JSON tree, data yang tersimpan dalam basis data bisa diwakili sebagai tipe bawaan tertentu yang sesuai dengan tipe JSON yang tersedia untuk membantu Anda menulis lebih banyak kode yang bisa dipertahankan [14].

2.10 Bahasa C

C secara umum merupakan bahasa pemrograman terstruktur. Instruksi - instruksinya terdiri dari istilah ekspresi aljabar, ditambah dengan kata kunci bahasa Inggris tertentu seperti *if*, *else*, *for*, *do* dan *while*. Dalam hal ini C menyerupai bahasa tingkat tinggi lainnya seperti pascal dan fortran. Bahasa pemrograman C juga mengandung fitur tambahan tertentu yang memungkinkannya digunakan pada level yang lebih rendah, lebih dekat ke bahasa mesin komputer. C pertama kali dikembangkan oleh Dennis Ritchie dan Brian Kerighan di Bell Telephone Laboratories Inc (sekarang AT&T Bel Laboratories) pada 1970-an. C sebagian besar dikembangkan pada Bell Labs hingga 1978, ketika Brian Kerighan dan Dennis Ritchie menerbitkan deskripsi definitif bahasa, "*The C Programming Language, Prentice-Hall*" tahun 1978. Versi C yang sesuai dengan standar ANSI dikenal sebagai ANSI C. Tetapi kebanyakan kompiler C yang kompatibel dengan ANSI juga memiliki fitur-fitur khusus yang ditambahkan, berbeda dengan yang tidak kompatibel dengan ANSI. C memiliki set instruksi yang relatif kecil, 32 kata kunci dalam versi bahasa yang paling umum, tetapi hal ini bisa menjadi komentar dengan fungsi *library* yang luas yang dapat dibangun oleh programmer. Salah satu fitur menarik dari bahasa C adalah bahwa C dapat dikompilasi untuk menghasilkan kode yang dapat dieksekusi dengan sangat cepat dan saling terhubung. Hal ini memungkinkannya digunakan untuk memprogram mikrokontroler, yang biasanya memiliki sedikit memori [15].

2.11 Java

Menurut Budi Raharjo, Imam Heryanto, Arif haryono (Mudah Belajar Java 2010) java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis Java umumnya dikompilasi ke dalam p-code (bytecode) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (general purpose), secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan Java mampu berjalan di beberapa platform sistem operasi yang berbeda, Java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun" [16].

2.12 Metode Pengujian

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean [16]. Sejumlah aturan yang berfungsi sebagai sasaran pengujian pada perangkat lunak adalah :

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.
2. Test case yang baik adalah test case yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.
3. Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.

Karakteristik umum dari pengujian perangkat lunak adalah sebagai berikut :

1. Pengujian dimulai pada level modul dan bekerja keluar kearah integrasi pada sistem berbasiskan komputer.

2. Teknik pengujian yang berbeda sesuai dengan poin-poin yang berbeda pada waktunya.
3. Pengujian diadakan oleh software developer dan untuk proyek yang besar oleh group testing yang independent.
4. Testing dan Debugging adalah aktivitas yang berbeda tetapi debugging harus diakomodasikan pada setiap strategi testing [17].

Metode pengujian perangkat lunak ada 3 jenis, yaitu :

1. *White Box/Glass Box* - pengujian operasi.
2. *Black Box* - untuk menguji sistem.
3. *Use case* - untuk membuat input dalam perancangan black box dan pengujian *statebased* [17].

2.12.1 White Box Testing

Pengujian *white box* adalah pengujian yang meramalkan cara kerja perangkat lunak secara rinci, karenanya logikal *path* (jalur logika) perangkat lunak akan di-test dengan menyediakan test case yang akan mengerjakan kumpulan kondisi atau pengulangan secara spesifik. Secara sekilas dapat diambil kesimpulan *white box* testing merupakan petunjuk untuk mendapatkan program yang benar secara 100%. Menurut Pressman [10], pengujian *White box* atau *Glass box* adalah metode *test case* desain yang menggunakan struktur kontrol desain *procedural* untuk memperoleh *test-case*. Dengan menggunakan metode pengujian *white box*, analisis sistem akan dapat memperoleh *test case* yang:

- a. Memberikan jaminan bahwa semua jalur *independent* pada suatu modul telah digunakan paling tidak satu kali.
- b. Menggunakan semua keputusan logis dari sisi *true* dan *false*.
- c. Mengeksekusi semua batas fungsi *loops* dan batas operasionalnya.
- d. Menggunakan struktur *internal* untuk menjamin validitasnya.

Ujicoba basis *path* adalah teknik uji coba *white box* yang diusulkan Tom McCabe. Metode ini memungkinkan perancang *test case* mendapatkan ukuran kekompleksan logikal dari perancangan prosedural dan menggunakan ukuran ini sebagai petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. *Test case*

yang didapat digunakan untuk mengerjakan basis set yang menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.

Terdapat beberapa proses yang harus di lakukan dalam uji coba basis *path* yaitu diantaranya :

1. Notasi Diagram Alir

Sebelum metode basis *path* diperkenalkan, terlebih dahulu akan dijelaskan mengenai notasi sederhana dalam bentuk diagram alir (grafik alir). Diagram alir menggambarkan aliran kontrol logika yang menggunakan notasi.

2. Kompleksitas Siklomatis

Kompleksitas siklomatis adalah metriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program. Bila metriks ini digunakan dalam konteks metode pengujian basis *path*, maka nilai yang terhitung untuk kompleksitas siklomatis menentukan jumlah jalur independen dalam basis set suatu pemrograman memberi batas atas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua statemen telah dieksekusi sedikitnya satu kali. Jalur independen adalah jalur yang memlalui progarm yang mengintroduksi sedikitnya satu rangkaian statemen proses baru atau suatu kondisi baru. Bila dinyatakan dengan terminologi grafik alir, jalur independen harus bergerak sepanjang paling tidak satu *edge* yang tidak dilewatkan sebelum jalur tersebut ditentukan.

3. Melakukan Test Case

Metode uji coba basis *path* juga dapat diterapkan pada perancangan prosedural rinci atau program sumber. Pada bagian ini akan dijelaskan langkah-langkah uji coba basis *path*.

4. Matriks Grafis

Prosedur untuk mendapatkan grafik alir dan menentukan serangkaian basis *path*, cocok dengan mekanisasi. Untuk mengembangkan peranti perangkat lunak yang membantu pengujian basis *path*, struktur data yang disebut matriks grafis dapat sangat berguna.

2.12.2 Black Box Testing

Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian *black box* atau bisa disebut pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan menggunakan perangkat lunak, serta dilihat apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian *black box testing* harus dibuat dengan kasus benar dan kasus salah. *Black box* testing juga disebut pengujian tingkah laku, memusat pada kebutuhan fungsional perangkat lunak. Teknik pengujian *black box* memungkinkan memperoleh serangkaian kondisi masukan yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Beberapa jenis kesalahan yang dapat diidentifikasi adalah fungsi tidak benar atau hilang, kesalahan antar muka, kesalahan pada struktur data (pengaksesan basis data), kesalahan performasi, kesalahan inisialisasi dan akhir program. *Equivalence Partitioning* merupakan metode *black box* testing yang membagi domain masukan dari program kedalam kelas-kelas sehingga *test case* dapat diperoleh. *Equivalence Partitioning* berusaha untuk mendefinisikan kasus uji yang menemukan sejumlah jenis kesalahan, dan mengurangi jumlah kasus uji yang harus dibuat. Kasus uji yang didesain untuk *equivalence partitioning* berdasarkan pada evaluasi dari kelas ekuivalensi untuk kondisi masukan yang menggambarkan kumpulan keadaan yang valid atau tidak. Kondisi masukan dapat berupa spesifikasi nilai numerik, kisaran nilai, kumpulan nilai yang berhubungan atau kondisi boolean [16].

Pengujian *black box* berusaha menemukan kesalahan dalam kategori :

1. Fungsi – fungsi yang tidak benar atau hilang.
2. Kesalahan *interface*.
3. Kesalahan dalam struktur data atau akses database eksternal.
4. Kesalahan kinerja.
5. Inisialisasi dan kesalahan terminasi [17].