BAB 2
LANDASAN TEORI

2.1 *Al-Qur'an*

Al-Quran berasal dari bahasa Arab yang berarti bacaan atau sesuatu yang dibaca berulang-ulang. Umat islam percaya bahwa *Al-Qur'an* merupakan puncak dan penutup wahyu Allah yang diperuntukkan bagi manusia, dan bagian dari rukun iman, yang disampaikan kepada Nabi Muhammad *Shallallahu 'alaihi wa sallam*, melalui perantaraan Malaikat Jibril. Maka setiap muslim (beragama islam) diwajibkan bisa untuk mempelajari serta mengamalkan *Al-Quran*, yang di dalamnya terkandung bagaimana menjalani hidup di dunia dan mendapatkan surga (selamat dunia akhirat). Di indonesia sendiri terdapat beberapa metode pembelajaran *Al-Quran.* Salah satunya dengan melatih menulis huruf – huruf Al-Quran yaitu huruf hijaiyah [8].

2.1.1 Huruf *Hijaiyah*

Huruf *hijaiyah* adalah huruf yang digunakan dalam *Al-Qur'an*, terdiri dari dua puluh sembilan huruf, terbagi menjadi dua bagian diantaranya *Al-qomariyah* yang berarti jelas dan *Al-syamsiyah* yang berarti tidak jelas (samar) [6]. Banyak metode pelatihan atau pembelajaran membaca serta menulis huruf hijaiyah, salah satunya adalah *iqra*.

خ	ح	ج	ث	ت	ب	ا
kha	haa	jiim	thaa	taa	baa	alif
ص	ش	س	ز	ر	ذ	د
saad	shiin	siin	zaay	raa	thaal	daal
ق	ف	غ	ع	ظ	ط	ض
qaaf	faa	ghayn	ayn	thaa	taa	daad
ي	و	ه	ن	م	ل	ك
yaa	waaw	ha	nuun	miim	laam	kaaf

Gambar 2.1 Huruf hijaiyah [7]

2.1.2 Menulis Huruf *Hijaiyah*

Pelatihan atau pembelajaran baca-tulis *al-Qur'an* adalah kegiatan pembelajaran membaca dan menulis yang ditekankan pada upaya memahami informasi, tetapi ada pada tahap menghafalkan (melesankan) lambang-lambang dan mengadakan pembiasaan dalam melafadkannya serta cara menuliskannya. Adapun tujuan dari pelatihan atau pembelajaran huruf hijaiyah yaitu huruf yang digunakan dalam kitab suci al-Qur'an ini adalah agar dapat membaca katakata dengan kalimat sederhana dengan lancar dan tertib serta dapat menulis huruf dan lambang-lambang arab dengan rapi, lancar dan benar.

2.1.3 *Khat*

Definisi Kaligrafi dan Khat - Kaligrafi berasal dari bahasa Yunani. (kallos) berarti indah dan (graphe) yang artinya tulisan. Syaikh Syamsuddin Al Akhfani (Dalam Irsyad Al Qoshid, 2000) Kaligrafi adalah suatu ilmu yang memperkenalkan beberapa bentuk-bentuk huruf tunggal, letak-letaknya, dan beberapa cara merangkainya sehingga menjadi sebuah kalimat tersusun. Atau apa-apa yang ditulis di atas garis-garis, bagaimana cara menulisnya dan menentukan mana yang tidak perlu ditulis, mengubah ejaan yang perlu diubah dan menentukan cara bagaimana untuk mengubahnya.

Khat ialah perkataan Arab yang bermaksud garisan. Oleh itu, seni khat bermaksud garisan indah yang membentuk tulisan. Khat juga bermaksud tulisan-tulisan (kitabah) yang terikat dengan peraturan dan kaedah yang telah dikaji dan ditentukan oleh mereka yang terlibat dengan kemajuan seni. Tulisan-tulisan Arab pula mempunyai nilai dan kaedah tertentu yang mempunyai estetika yang tinggi. etelah kehadiran Islam, penulisan Arab telah memasuki tahap perkembangan yang begitu cepat. Pada abad pertama dan kedua Hijrah, khat merupakan salah satu ciri untuk memperindah sesuatu penulisan. Melalui khat, sesuatu maksud dapat diungkapkan. Khat turut menjadi unsur penting daripada cabang-cabang kesenian yang masih terpelihara hingga ke hari ini.

Banyak jenis-jenis gaya *khat* Salah satunya adalah *diwani* salah satu gaya khat yang diciptakan oleh masyarakat Turki Usmani. Peletak dasar-dasar kaidah dan ukuran huruf-hurufnya adalah Ibarhim Munif. Tulisan ini mulai popular setelah penaklukan kota Konstantinopel oleh Sultan Muhammad al-Fatih 875 H. Penamaan Diwani karena dinisbahkan kepada kantor-kantor pemerintahan dimana tulisan tersebut digunakan dan dewan-dewan pemerintah itulah khat ini menyebar keseluruh kalangan masyarakat. Karakter Diwani terkenal dengan putaranya sehingga tidak satu pun huruf yang tidak mempunyai lengkungan. Goresannya yang lentur dan lembut memudahkan Diwani beradaptasi dengan tulisan apapun. Diwani memiliki 3 macam bentuk.

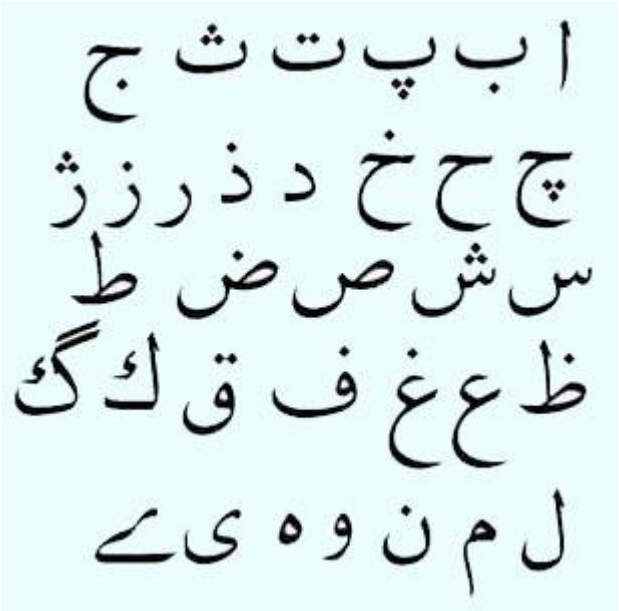
1. *Khat Diwani ‘Adi*

Diwani ‘Adi merupakan gaya khat yang tampil biasa(‘adi) sesuai struktur tulisan sehingga mudah dibaca. Cirri tampilannya tampak pada kali-kali tulisan yang umumnya berbaris datar dengan pucuk-pucuk huruf bergelombang dinamis.

2. *Khat Diwani Murabbit*

Gaya ini merupakan Diwani yang huruf-huruf dan rangkaian katanya saling menjalin atau bersilangan (murabbit) satu sama lain. Besar kemungkinan pola semacam ini merupakan hasil pengaruh khat Musalsal ciptaan Ibnu Bawab.

Dalam jenis Khat Diwani Murabbit ini, kaligrafer modern Gazian Bek dari Mesir merupakan tokohnya.



Gambar 2.2 Seni *Khat*

3. *Khat Diwani Jali*

Khat Diwani jali diciptakan oleh Syahlan Pasha dari Turki dan merupakan pengembangan dari Diwani ‘Adi. Jali artinya Jelas. Kejelasan tersebut tampak pada detail syakal dan hiasan yang penuh didalamnya. Tujuan diciptakan Diwani jali ialah untuk menuliskan peraturan-peraturan kesultanan dan surat-surat keluar negeri.

2.1.4 Pengenalan *Gesture touch*

Gestures touch biasanya digunakan dari gerakan atau pergeseran jari pada layar sentuh sensitif pada Android, yang diproses menggunakan package android.gesture . Gerakan yang paling umum adalah *tap*, yang seperti klik mouse pada komputer. Gesture tertentu juga terdeteksi melalui antarmuka View.OnClickListener. Gerakan berikutnya adalah *drag* atau seret (seperti menyeret dengan mouse pada komputer), sebagai contoh ketika pertama kali menyalakan android umumnya melakukan slider untuk membuka kunci (*key lock*)

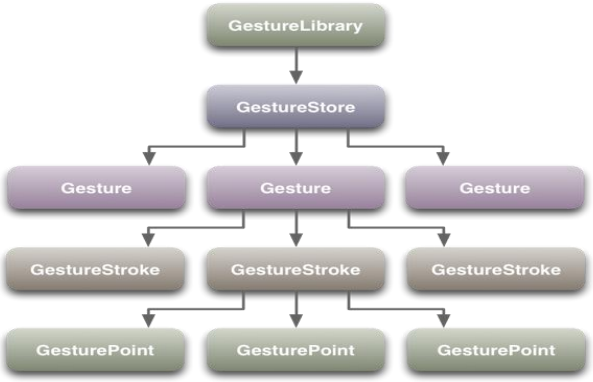
pada Android dengan melakukan gerakan *drag* atau seret. contoh sederhananya ketika *Gesture package* akan melaporkan urutan gerakan jari pada layar sentuh secara penuh dan memungkinkan aplikasi untuk mendeteksi , misalnya, ketika membuat gerakan busur melengkung , ketika membuat gerakan searah jarum jam atau berlawanan dan jika gerakan itu lambat ataupun cepat . Aplikasi harus bisa mengkonversi titik koordinat X/Y dan waktu untuk mencari tahu apa gerakannya, gerakan – gerakan tersebut bisa manfaatkan pada aplikasi - aplikasi android,seperti pada game,widget,aplikasi menggambar dan sebagainya.

2.1.4.1 Pengenalan *Gesture Builder*

Gesture Builder adalah sebuah *tool* yang digunakan untuk membuat suatu *library* dari gesture. Ketika aplikasi menjalankan sebuah gesture, aplikasi akan mencocokan gerakan tersebut ke *GestureLibrary* dan menentukan apakah gesture itu cocok dengan salah satu gesture yang sudah ada di *library*.

2.1.4.2 Pengenalan *Gesture Library*

Gesture Libary dibuat menggunakan *tool* yang di sediakan pengembang yaitu *gesture builder*, *gesture library* disimpan satu paket dengan aplikasi dan dipanggil dari *raw resource* dimana *gesture library* itu disimpan, dibawah ini adalah struktur dari *gesture library.*



Gambar 2.3 Struktur *Gesture Library*[8]

2.2 Aplikasi

Aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar media. Beberapa aplikasi yang digabung bersama menjadi suatu paket kadang disebut suatu paket atau suite aplikasi (*Application Suite*). Contohnya adalah *Microsoft Office* dan *OpenOffice*, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja, serta beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Seringkali, mereka memiliki kemapuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah.

2.3 Android



Gambar 2.4 Logo Android [8]

Android adalah sistem operasi yang berbasis *Linux* untuk telepon seluler seperti telepon pintar (*Smartphone*) dan komputer tablet. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti begerak. Awalnya, *Google Inc.* membeli *Android Inc.*, pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan *Android*, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan piranti keras, piranti lunak dan telekomunikasi, termasuk *Google, HTC, Intel, Motorola, Qualcomm, T-Mobile* dan *Nvidia.*

Bebrapa pengertian lain dari *Android*, yaitu :

1. Merupakan *platform* terbuka (*Open Source*) bagi para programmer untuk membuat atau mengembangkan aplikasi.
2. Merupakan sistem operasi yang dibeli *Google Inc* dari *Android Inc*.
3. Bukan bahasa pemrograman, akan tetapi hanya menyediakan lingkungan hidup atau *runtime environment* yang disebut DVM (*Dalvik Virtual Machine*) yang telah dioptimasi untuk *device* dengan sistem memori yang kecil.

2.3.1 Sejarah Android

Pada Juli 2000, *Google* bekerjasama dengan *Android Inc*., perusahaan yang berada di Palo Alto, California, Amerika Serikat. Para pendiri *Android Inc*. bekerja pada *Google,* diantaranya Andy Rubin, Rich Miner, Nick Sears dan Chris White. Saat itu banyak yang menganggap fungsi *Android Inc.* hanyalah sebagai perangkat lunak pada telepon seluler. Sejak saat itu muncul rumor bahwa *Google* hendak memasuki pasar telepon seluler. Di perusahaan *Google*, tim yang dipimpin Rubin bertugas mengembangkan program perangkat seluler yang didukung oleh *Kernel Linux*. Hal ini menunjukan indikasi bahwa *Google* sedang bersiap menghadapi persaingan dalam pasara telepon seluler. Versi *android* terbaru yaitu versi 5.0. (*KitKat*). *Android* juga sudah bergabung dengan beberapa *smart mobile* seperti *LG, Samsung, Sony Ericsson, HTC, Motorola* dan lainnya.

Sekitar September 2007 sebuah studi melaporkan bahwa *Google* mengajukan hak paten aplikasi telepon seluler (akhirnya *Google* mengenalkan *Nexus One,* salah satu jenis telepon pintar GSM yang menggunakan *Android* pada sistem operasinya. Telepon seluler ini diproduksi oleh *HTC Corporotaion* dan tersedia di pasaran pada 5 Januari 2010).

Pada Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja *Android ARM Holdings, Atheros Communications,* diproduksi oleh *Asustek Computer Inc. Garmin Ltd, Softbank, Sony Ericsson Toshiba Corp* dan *Vodafone Group Plc.* Seiring pembentukan *Open Handset Alliance,* OHA mengumumkan produk perdana mereka, *Android,* perangkat bergerak (*Mobile*) yang merupakan modifikasi *kernel Linux* 2.6. sejak *Android* dirilis telah dilakukan berbagai pembaruan berupa perbaikan *bug* dan penambahan fitur baru.

2.3.2 Versi Android

Banyak *smartphone* dan PC Tablet menggunakan sistem operasi dengan versi yang berbeda. Semakin tinggi versi, fiturnya semakin canggih dan banyak. Telepon pertama yang memakai sistem operasi *Android* adalah HTC Dream yang dirilis pada tanggal 22 Oktober 2008. Beberapa uraian versi *android* seperti dibawah ini :

1. ***Android* versi 1.1**

   Pada 9 Maret 2009, *Google* merilis *Android* versi 1.1. *Android* versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan Email.

2. ***Android* versi 1.5 (*Cupcake*)**

   Pada pertengahan Mei 2009, *Google* kembali merilis telepon seluler dengan menggunakan *Android* dan SDK (*Software Development Kit*) dengan versi 1.5 (*Cupcake*). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video

ke *Youtube* dan gambar ke Picasa langsung dari telepon, dukungan *Bluetooth* A2DP, kemampuan terhubung secara otomatis ke *headset bluetooth*, animasi layar dan *keyboard* pada layar yang dapat disesuaikan dengan sistem.

3. ***Android* versi 1.6 (*Donut*)**

   Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol *applet* VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus pada kamera, *camcorder* dan galeri yang diintegrasikan pada CDMA / EVDO, 802.1x, VPN, *Gestures*, dan *Text-to-spech engine*. Kemampuan dial kontak teknologi *text to change speech* (tidak tersedia pada semua ponsel).

4. ***Android* versi 2.0/2.1 (*Eclair*)**

   Pada 3 Desember 2009 kembali diluncurkan ponsel *Android* dengan versi 2.0/2.1 (*Eclair*), perubahan yang dilakukan adalah mengoptimalkan *hardware*, peningkatan Google Maps 3.1.2, perubahan UI dengan browser baru dan dukungan HTML 5, daftar kontak yang baru, dukungan *flash* untuk kamera 3.2 MP, *digital zoom*, dan *Bluetooth* 2.1.

   Untuk bergerak cepat dalam persaingan perangkat generasi berikut, Google melakukan investasi dengan mengadakan kompetisi aplikasi *mobile* terbaik (*killer apps*-aplikasi unggulan). Kompetisi ini berhadiah $25,000 bagi setiap pengembang aplikasi terpilih. Kompetisi diadakan selama dua tahap yang tiap tahapnya dipilih 50 aplikasi terbaik.

   Dengan semakin berkembangnya dan semakin bertambahnya jumlah *handset Android*, semakin banyak pihak ketiga yang berminat untuk menyalurkan aplikasi mereka kepada sistem operasi *Android*. Aplikasi terkenal yang diubah ke dalam sistem operasi *Android* adalah Shazam, *Backgrounds* dan *WeatherBug*.

   Sistem operasi *Android* dalam situs internet juga dianggap penting untuk menciptakan aplikasi *Android* asli, contohnya oleh MySpace dan Facebook.

5. ***Android* versi 2.2 (*Froyo* : *Frozen Yoghurt*)**

   Pada 20 Mei 2010, *Android* versi 2.2 (Froyo) diluncurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, integrasi dengan V8 JavaScript *engine* yang dipakai Google Chrome yang mempercepat kemampuan rendering pada browser, pemasangan aplikasi dalam SD Card, kemampuan WiFi Hotspot portabel dan kemampuan *auto update* dalam aplikasi Android Market.

6. ***Android* versi 2.3 (*Gingerbread*)**

   Pada 6 Desember 2010, Android versi 2.3 (*Gingerbread*) diluncurkan. Perubahan-perubahan umum yang didapat dari versi ini antara lain peningkatan kemampuan permainan (*gaming*), peningkatan fungsi *copy paste*, layar antarmuka (*User Interface*) didesain ulang, dukungan format video VP8 dan WebM, efek audio baru (*reverb*, *equalization*, *headphone virtualization*, dan *bass boost*), dukungan kemampuan *Near Field Communication* (NFC) dan dukungan jumlah kamera yang lebih dari satu.

7. ***Android* versi 3.1 (*Honeycomb*)**

   Android Honeycomb dirancang khusus untuk tablet. *Android* versi ini mendukung ukuran layar yang lebih besar. *User Interface* pada *Honeycomb* juga berbeda karena sudah didesain untuk tablet. *Honeycomb* juga mendukung multi prosesor dan juga akselerasi perangkat keras (*hardware*) untuk grafis. Tablet peratam yang dibuat dengan menjalankan *Honeycomb* adalah Motorola Xoom.

8. ***Android* versi 4.0 (*Ice Cream Sandwich*)**

   Diumumkan pada tanggal 19 Oktober 2011, membawa fitur *Honeycomb* untuk *smartphone* dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari email secara *offline* dan berbagi informasi dengan menggunakan NFC.

9. ***Android* versi 4.1 (*Jelly Bean*)**

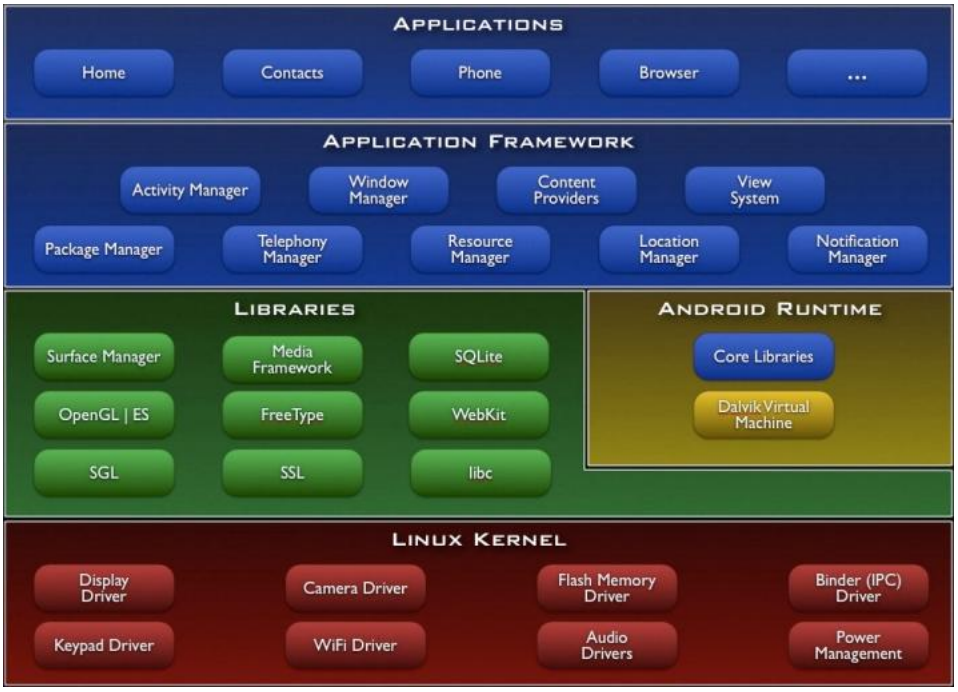
   Pada tanggal 27 Juni 2012 Google mengumumkan *Android* versi 4.1 (*Jelly Bean*) dalam konferensi Google I/O berdasarkan kernel Linux 3.0.31. *Jelly Bean* adalah pembaruan penting yang bertujuan untuk meningkatkan fungsi dan kinerja antarmuak pengguna (*User Interface*). Pembaruan ini diwujudkan dalam proyek *Butter*, perbaikan ini termasuk antisipasi sentuh, *triple buffering*, perpanjangan waktu vsync dan peningkatan *frame rate* hingga 60fps untuk menciptakan UI yang lebih halus. Perangkat pertama yang menggunakan sistem operasi ini adalah tablet Nexus 7, yang dirilis pada 13 Juli 2012.

10. ***Android* versi 4.4 (KitKat)**

    Android versi 4.4 (KitKat) direncanakan akan dirilis pada bulan Oktober 2013. Setelah sebelumnya beredar rumor bahwa Android versi berikutnya setelah *Jelly Bean* diperkirakan akan diberi nomor 5.0 dan dinamai Key Lime Pie.

2.3.3 Anatomi Aplikasi *Android*

Dalam paket sistem operasi Android tediri dari beberapa unsur seperti tampak pada gambar 2.1. Secara sederhana arsitektur Android merupakan sebuah kernel Linux dan sekumpulan pustaka C / C++ dalam suatu *Framework* yang menyediakan dan mengatur alur proses aplikasi.



Gambar 2.5 Detail Anatomi Android [9]

2.3.3.1 *Linux Kernel*

Android dibangun di atas kernel Linux 2.6. Namun secara keseluruhan Android bukanlah linux, karena dalam Android tidak terdapat paket standar yang dimiliki oleh linux lainnya. Linux merupakan sistem operasi terbuka yang handal dalam manajemen memori dan proses. Oleh karenanya pada Android hanya terdapat beberapa pelayanan yang diperlukan seperti keamanan, manajemen memori, manajemen proses, jaringan dan *driver*. Kernel linux menyediakan *driver* layar, kamera, *keypad*, *WiFi*, *Flash Memory*, audio, dan IPC (*Interprocess Communication*) untuk mengatur aplikasi dan lubang keamanan.

2.3.3.2 *Libraries*

Android menggunakan beberapa paket pustaka yang terdapat pada C/C++ dengan standar *Berkeley Software Distribution* (BSD) hanya setengah dari yang aslinya untuk tertanam pada kernel Linux. Beberapa pustaka diantaranya:

1. *Media Library* untuk memutar dan merekam berbagai macam format audio dan video.
2. *Surface Manager* untuk mengatur hak akses layer dari berbagai aplikasi.

3. *Graphic Library* termasuk didalamnya *SGL* dan *OpenGL*, untuk tampilan 2D dan 3D.
4. SQLite untuk mengatur relasi database yang digunakan pada aplikasi.
5. *SSl* dan *WebKit* untuk browser dan keamanan *internet*.

Pustaka-pustaka tersebut bukanlah aplikasi yang berjalan sendiri, namun hanya dapat digunakan oleh program yang berada di level atasnya. Sejak versi Android 1.5, pengembang dapat membuat dan menggunakan pustaka sendiri menggunakan *Native Development Toolkit* (NDK).

2.3.3.3 *Android Runtime*

Pada Android tertanam paket pustaka inti yang menyediakan sebagian besar fungsi Android. Inilah yang membedakan Android dibandingkan dengan sistem operasi lain yang juga mengimplementasikan Linux. Android *Runtime* merupakan mesin virtual yang membuat aplikasi Android menjadi lebih tangguh dengan paket pustaka yang telah ada. Dalam Android Runtime terdapat 2 bagian utama, diantaranya :

1. Pustaka Inti, Android dikembangkan melalui bahasa pemrograman Java, tapi Android *Runtime* bukanlah mesin virtual Java. Pustaka inti Android menyediakan hampir semua fungsi yang terdapat pada pustaka Java serta beberapa pustaka khusus Android.
2. Mesin Virtual Dalvik, Dalvik merupakan sebuah mesin virtual yang dikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah perkampungan yang berada di Iceland. Dalvik hanyalah interpreter mesin virtual yang mengeksekusi file dalam format *Dalvik Executable* (*.dex). Dengan format ini Dalvik akan mengoptimalkan efisiensi penyimpanan dan pengalamatan memori pada file yang dieksekusi. Dalvik berjalan di atas kernel Linux 2.6, dengan fungsi dasar seperti *threading* dan manajemen memori yang terbatas.

2.3.3.4 *Application Framework*

Kerangka aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi Android. Selain itu, juga menyediakan abstraksi generik untuk mengakses perangkat, serta mengatur tampilan *User Interface* dan sumber daya aplikasi. Bagian terpenting dalam kerangka aplikasi Android adalah sebagai berikut:

1. *Activity Manager*, berfungsi untuk mengontrol siklus hidup aplikasi dan menjaga keadaan ”*Backstack*“ untuk navigasi penggunaan.
2. *Content Providers*, berfungsi untuk merangkum data yang memungkinkan digunakan oleh aplikasi lainnya, seperti daftar nama.
3. *Resuource Manager*, untuk mengatur sumber daya yang ada dalam program. Serta menyediakan akses sumber daya diluar kode program, seperti karakter, grafik, dan *file layout*.
4. *Location Manager*, berfungsi untuk memberikan informasi detail mengenai lokasi perangkat Android berada.
5. *Notification Manager*, mencakup berbagai macam peringatan seperti, pesan masuk, janji, dan lain sebagainya yang akan ditampilkan pada *status bar*.

2.3.3.5 Application Layer

Puncak dari diagram arsitektur Android adalah lapisan aplikasi dan *widget*. Lapisan aplikasi merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam Android *runtime* dengan menggunakan kelas dan service yang tersedia pada *Framework* aplikasi.

Lapisan aplikasi Android sangat berbeda dibandingkan dengan sistem operasi lainnya. Pada Android semua aplikasi, baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka API (*Application Programming Interface*) yang sama.

2.3.4 Komponen Aplikasi *Android*

Fitur penting Android adalah bahwa satu aplikasi dapat menggunakan elemen dari aplikasi lain (untuk aplikasi yang memungkinkan). Sebagai contoh, sebuah aplikasi memerlukan fitur *scroller* dan aplikasi lain telah mengembangkan fitur *scroller* yang baik dan memungkinkan aplikasi lain menggunakannya. Maka pengembang tidak perlu lagi mengembangkan hal serupa untuk aplikasinya, cukup menggunakan *scroller* yang telah ada.

Agar fitur tersebut dapat bekerja, sistem harus dapat menjalankan aplikasi ketika setiap bagian aplikasi itu dibutuhkan, dan pemanggilan objek java untuk bagian itu. Oleh karenanya Android berbeda dari sistem-sistem lain, Android tidak memiliki satu tampilan utama program seperti fungsi main() pada aplikasi lain. Sebaliknya, aplikasi memiliki komponen penting yang memungkinkan sistem untuk memanggil dan menjalankan ketika dibutuhkan.

2.3.4.1 Activities

Activity merupakan bagian yang paling penting dalam sebuah aplikasi, karena *Activity* menyajikan tampilan visual program yang sedang digunakan oleh pengguna. Setiap *Activity* dideklarasikan dalam sebuah kelas yang bertugas untuk menampilkan antarmuka pengguna yang terdiri dari *Views* dan respon terhadap *Event*. Setiap aplikasi memiliki sebuah *activity* atau lebih. Biasanya pasti akan ada *activity* yang pertama kali tampil ketika aplikasi dijalankan.

Perpindahan antara *activity* dengan *activity* lainnya diatur melalui sistem, dengan memanfaatkan *activity stack*. Keadaan suatu *activity* ditentukan oleh posisinya dalam tumpukan *acitivity*, LIFO (*Last In First Out*) dari semua aplikasi yang sedang berjalan. Bila suatu *activity* baru dimulai, *activity* yang sebelumnya digunakan maka akan dipindahkan ketumpukan paling atas. Jika pengguna ingin menggunakan *activity* sebelumnya, cukup menekan tombol *Back*, atau menutup *activity* yang sedang digunakan, maka *activity* yang berada diatas akan aktif kembali. *Memory Manager* Android menggunakan tumpukkan ini untuk menentukan prioritas aplikasi berdasarkan *activity*, memutuskan untuk mengakhiri suatu aplikasi dan mengambil sumber daya dari aplikasi tersebut.

Ketika *activity* diambil dan disimpan dalam tumpukkan *activity* terdapat 4 kemungkinan kondisi transisi yang akan terjadi :

1) ***Active***, setiap *activity* yang berada ditumpukan paling atas, maka dia akan terlihat, terfokus, dan menerima masukkan dari pengguna. Android akan berusaha untuk membuat *activity* aplikasi ini untuk untuk tetap hidup dengan segala cara, bahkan akan menghentikan *activity* yang berada dibawah tumpukkannya jika diperlukan. Ketika *activity* sedang aktif, maka yang lainnya akan dihentikan sementara.
2) ***Paused***, dalam beberapa kasus *activity* akan terlihat tapi tidak terfokus pada kondisi inilah disebut *paused*. Keadaan ini terjadi jika *activity* transparan dan tidak *fullscreen* pada layar. Ketika *activity* dalam keadaan *paused*, dia terlihat *active* namun tidak dapat menerima masukkan dari pengguna. Dalam kasus ekstrim, Android akan menghentikan *activity* dalam keadaan *paused* ini, untuk menunjang sumber daya bagi *activity* yang sedang aktif.
3) ***Stopped***, ketika sebuah *activity* tidak terlihat, maka itulah yang disebut *stopped*. *Activity* akan tetap berada dalam memori dengan semua keadaan dan informasi yang ada. Namun akan menjadi kandidat utama untuk dieksekusi oleh sistem ketika membutuhkan sumberdaya lebih. Oleh karenanya ketika suatu *activity* dalam kondisi *stopped* maka perlu disimpan data dan kondisi antarmuka saat itu. Karena ketika *activity* telah keluar atau ditutup, maka dia akan menjadi *inactive*.
4) ***Inactive***, kondisi ketika *activity* telah dihentikan dan sebelum dijalankan. *Inactive activity* telah ditiadakan dari tumpukan *activity* sehingga perlu *restart* ulang agar dapat tampil dan digunakan kembali.

Kondisi transisi ini sepenuhnya ditangani oleh manajer memori Android. Android akan memulai menutup aplikasi yang mengandung *activity inactive*, kemudian *stopped activity*, dan dalam kasus luar biasa *paused activity* juga akan ditutup.

2.3.4.2 *Services*

Suatu *service* tidak memiliki tampilan antarmuka, melainkan berjalan di *background* untuk waktu yang tidak terbatas. Komponen *service* diproses tidak terlihat, memperbarui sumber data dan menampilkan notifikasi. *Service* digunakan untuk melakukan pengolahan data yang perlu terus diproses, bahkan ketika *Activity* tidak aktif atau tidak tampak.

2.3.4.3 *Intents*

Intens merupakan sebuah mekanisme untuk menggambarkan tindakan tertentu, seperti memilih foto, menampilkan halaman *web*, dan lain sebagainya. *Intents* tidak selalu dimulai dengan menjalankan aplikasi, namun juga digunakan oleh sistem untuk memberitahukan ke aplikasi bila terjadi suatu hal, misal pesan masuk. *Intents* dapat eksplisit atau implisit, contohnya jika suatu aplikasi ingin menampilkan *URL*, sistem akan menentukan komponen apa yang dibutuhkan oleh *Intents* tersebut.

2.3.4.4 *Broadcast Receiver*

Broadcast Receivers merupakan komponen yang sebenarnya tidak melakukan apa-apa kecuali menerima dan bereaksi menyampaikan pemberitahuan. Sebagian besar *Broadcast* berasal dari sistem misalnya, Batre sudah hampir habis, informasi zona waktu telah berubah, atau pengguna telah merubah bahasa *default* pada perangkat. Sama halnya dengan *service*, *Broadcast Receivers* tidak menampilkan antarmuka pengguna. Namun, *Broadcast Receivers* dapat menggunakan *Notification Manager* untuk memberitahukan sesuatu kepada pengguna.

2.3.4.5 *Content Providers*

Content Providers digunakan untuk mengelola dan berbagi *database*. Data dapat disimpan dalam file sistem, dalam database *SQLite*, atau dengan cara lain yang pada prinsipnya sama. Dengan adanya *Content Provider* memungkinkan antar aplikasi untuk saling berbagi data. Komponen ini sangat berguna ketika

sebuah aplikasi membutuhkan data dari aplikasi lain, sehingga mudah dalam penerapannya.

2.3.5 Tipe Aplikasi *Android*

Terdapat tiga kategori aplikasi pada Android :

1. ***Foreground Activity***

   Aplikasi yang hanya dapat dijalankan jika tampil pada layar dan tetap efektif walaupun tidak terlihat. Aplikasi dengan tipe ini pasti mempertimbangkan siklus hidup *activity*, sehingga perpindahan antar *activity* dapat berlangsung dengan lancar.

2. ***Background Service***

   Aplikasi yang memiliki interaksi terbatas dengan user, selain dari pengaturan konfigurasi, semua dari prosesnya tidak tidak tampak pada layar. Contohnya aplikasi penyaringan panggilan atau sms *auto respon*.

3. ***Intermittent Activity***

   Aplikasi yang masih membutuhkan beberapa masukkan dari pengguna, namun sebagian sangat efektif jika dijalankan di *background* dan jika diperlukan akan memberi tahu pengguna tentang kondisi tertentu. Contohnya pemutar musik.

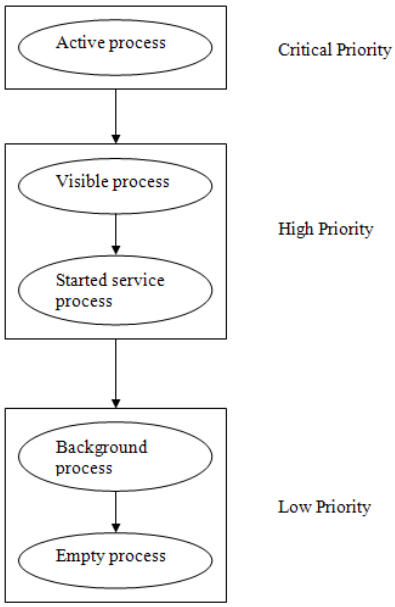
Untuk aplikasi yang kompleks akan sulit untuk menentukan kategori aplikasi tersebut apalagi aplikasi memiliki ciri-ciri dari semua kategori. Oleh karenanya perlu pertimbangan bagaimana aplikasi tersebut digunakan dan menentukan kategori aplikasi yang sesuai.

2.3.6 Siklus Hidup *Android*

Siklus hidup aplikasi Android dikelola oleh sistem, berdasarkan kebutuhan pengguna, sumberdaya yang tersedia, dan sebagainya. Misalnya Pengguna ingin menjalankan *browser web*, pada akhirnya sistem yang akan menentukan menjalankan aplikasi. Sistem sangat berperan dalam menentukan apakah aplikasi dijalankan, dihentikan sementara, atau dihentikan sama sekali. Jika pengguna ketika itu sedang menjalankan sebuah *Activity*, maka sistem akan memberikan

perioritas utama untuk aplikasi yang tersebut. Sebaliknya, jika suatu *Activity* tidak terlihat dan sistem membutuhkan sumber daya yang lebih, maka *Activity* yang prioritas rendah akan ditutup.

Android menjalankan setiap aplikasi dalam proses secara terpisah, yang masing-masing memiliki mesin *virtual* pengolah sendiri, dengan ini melindungi penggunaan memori pada aplikasi. Selain itu juga Android dapat mengontrol aplikasi mana yang layak menjadi prioritas utama. Karenanya Android sangat sensitive dengan siklus hidup aplikasi dan komponen-komponennya. Perlu adanya penanganan terhadap setiap kondisi agar aplikasi menjadi stabil. Gambar 2.2 menunjukkan prioritas dari aplikasi.



Gambar 2.6 Prioritas Aplikasi Berdasarkan *Activity*

2.3.7 Kelebihan *Android*

Sudah banyak platform untuk perangkat selular saat ini, termasuk didalamnya Symbian, iPhone, Windows Mobile, BlackBerry, Java Mobile Edition, Linux Mobile (LiM), dan banyak lagi. Namun ada beberapa hal yang menjadi kelebihan Android. Walaupun beberapa fitur-fitur yang ada telah muncul sebelumnya pada *platform* lain, Android adalah yang pertama menggabungkan hal seperti berikut :

1. Keterbukaan, Bebas pengembangan tanpa dikenakan biaya terhadap sistem karena berbasiskan Linux dan *open source*. Pembuat perangkat menyukai hal ini karena dapat membangun *platform* yang sesuai yang diinginkan tanpa harus membayar royality. Sementara pengembang *software* menyukai karena Android dapat digunakan diperangkat manapun dan tanpa terikat oleh *vendor* manapun.
2. Arsitektur komponen dasar Android terinspirasi dari teknologi *internet Mashup*. Bagian dalam sebuah aplikasi dapat digunakan oleh aplikasi lainnya, bahkan dapat diganti dengan komponen lain yang sesuai dengan aplikasi yang dikembangkan.
3. Banyak dukungan *service*, kemudahan dalam menggunakan berbagai macam layanan pada aplikasi seperti penggunaan layanan pencarian lokasi, *database* SQL, *browser* dan penggunaan peta. Semua itu sudah tertanam pada Android sehingga memudahkan dalam pengembangan aplikasi.
4. Siklus hidup aplikasi diatur secara otomatis, setiap program terjaga antara satu sama lain oleh berbagai lapisan keamanan, sehingga kerja sistem menjadi lebih stabil. Pengguna tak perlu kawatir dalam menggunakan aplikasi pada perangkat yang memorinya terbatas.
5. Dukungan grafis dan suara terbaik, dengan adanya dukungan 2D grafis dan animasi yang diilhami oleh Flash menyatu dalam 3D menggunakan OpenGL memungkinkan membuat aplikasi maupun *game* yang berbeda.
6. *Portabilitas* aplikasi, aplikasi dapat digunakan pada perangkat yang ada saat ini maupun yang akan datang. Semua program ditulis dengan menggunakan bahas pemrograman Java dan dieksekusi oleh mesin *virtual Dalvik,* sehingga kode program *portable* antara ARM, X86, dan arsitektur lainnya. Sama halnya dengan dukungan masukan seperti penggunaan *Keyboard,* layar sentuh, *trackball* dan resolusi layar semua dapat disesuaikan dengan program.

2.4 Perancangan Berorientasi Objek

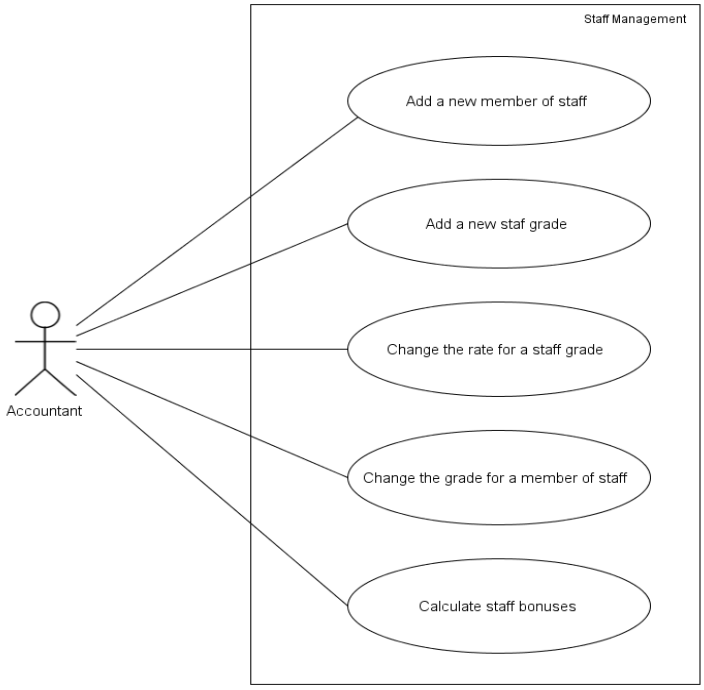
Teknologi objek menganalogikan sistem aplikasi seperti kehidupan nyata yang didominasi oleh objek. Didalam membangun sistem berorientasi objek akan menjadi lebih baik apabila langkah awalnya didahului dengan proses analisis dan perancangan yang berorientasi objek. Tujuannya adalah mempermudah *programmer* didalam mendesain program dalam bentuk objek-objek dan hubungan antar objek tersebut untuk kemudian dimodelkan dalam sistem nyata. Suatu perusahaan *software* yaitu *Rational Software,* telah membentuk konsarium dengan berbagai organisasi untuk meresmikan pemakaian *Unified Modelling Language* (UML) sebagai bahasa standar dalam *Object Oriented Analysist Design* (OOAD).

2.4.1 Unified Modelling Language (UML)

UML dalam sebuah bahasa untuk menentukan visualisasi, konstruksi, dan mendokumentasikan artifacts dari sistem *software*, untuk memodelkan bisnis, dan sistem *non-software* lainnya. UML merupakan sistem arsitektur yang bekerja dalam OOAD dengan satu bahasa yang konsisten untuk menentukan, visualisasi, konstruksi dan mendokumentasikan *artifact* yang terdapat dalam sistem. *Artifact* adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa *software*. *Artifact* dapat berupa model, deskripsi atau *software*.

2.4.1.1 Use Case Diagram

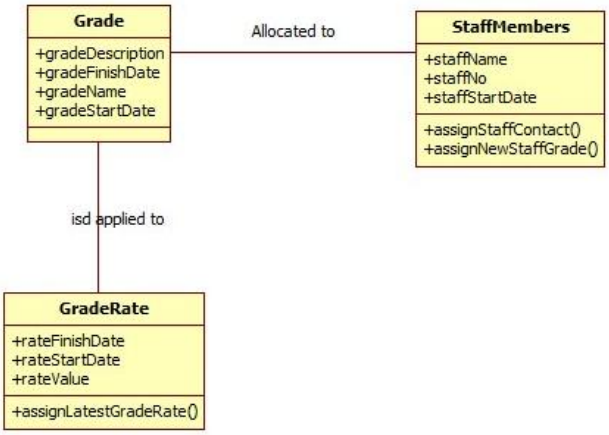
Use Case Diagram menjelakan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (Aktor). Diagram ini menunjukan fungsionalitas suatu sistem yang berinteraksi dengan dunia luar. *Use Case Diagram* dapat digunakan selama proses analisis untuk menagkap *requitment* sistem dan untuk memahami bagaimana sistem bekerja.



Gambar 2.7 *Use Case Diagram*

2.4.1.2 Class Diagram

Class Diagram menjelaskan dalam visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. *Class Diagram* memperlihatakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas dalam model desain dari suatu sistem. Selama proses analisis, *class diagram* memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan prilaku sistem. Selama tahap desain, *class diagram* berperan dalam menagkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.



Gambar 2.8 *Class Diagram*

2.4.1.3 Behavior Diagram

Behavior diagram dapat dikelompokan menjadi tiga diagram, yaitu :

1. *Statechart Diagram*

   Statechart Diagram berfungsi untuk memodelkan prilaku dinamis satu kelas satu objek.

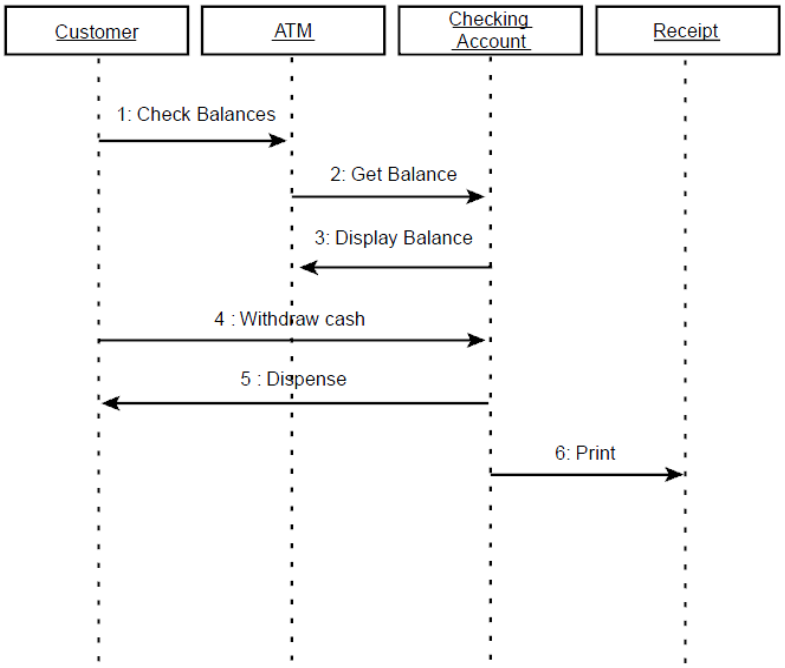
2. *Activity Diagram*

   Activity Diagram memodelkan alur kerja (work flow) sebuah proses bisnis dan urutan aktifitas dalam suatu proses.

3. *Interaction Diagram*

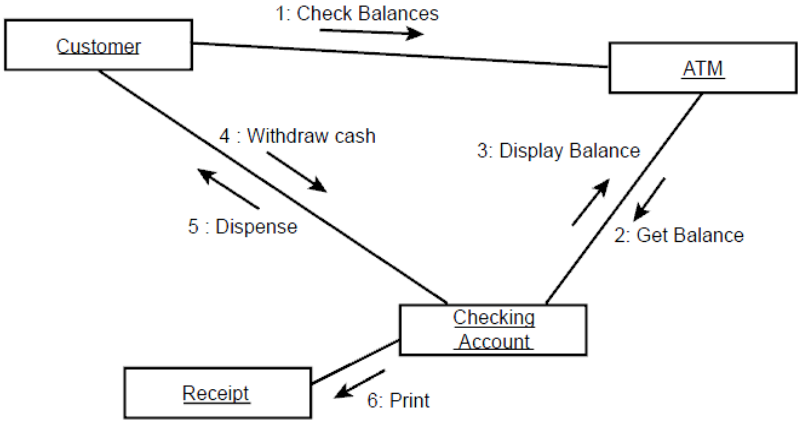
   Interaction Diagram dibagi menjadi dua model diagram yaitu :

   1. *Sequence Diagram* menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus bersosialisasi dengan *use case*. Sequence diagram, memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu dalam *use case*.



Gambar 2.9 *Sequence Diagram*

2. *Collaboration Diagram* melihat pada interaksi dan hubungan terstruktur antar objek. Tipe diagram ini menekankan pada hubungan (*relationship*) antar objek, sedangkan *sequence diagram* menekankan pada urutan kejadian. Dalam *collaboration diagram* terdapat beberapa objek, *link*, dan *message*.



Gambar 2.10 *Collaboration Diagram*

2.4.1.4 Implementation Diagram

Implementation diagram dibagi menjadi dua diagram, yaitu :

1. *Component Diagram* menggambarkan alokasi semua kelas dan objek kedalam komponen-komponen dalam desain fisik sistem *software*. Diagram ini memperlihatkan pengaturan dan kebergantungan antara komponen-komponen *software*, seperti *source code*, *binary code*, dan komponen tereksekusi (*execute components*)

   Deployment Diagram memperlihatkan pemetaan *software* kepada *hardware*. Dimana akan berjalan (di *server/multitier*, *standalone* atau lainnya), dan menggambarkan model koneksi dan kemampuan jaringan dan hal lainya yang bersifat fisik.

2.5 Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*). Berikut ini adalah sifat dari Eclipse:

1. *Multi-platform*: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
2. *Multi-language*: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.
3. *Multi-role*: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya popular adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

2.5.1 Sejarah Eclipse

Eclipse awalnya dikembangkan oleh IBM untuk menggantikan perangkat lunak IBM *Visual Age for Java* 4.0. Produk ini diluncurkan oleh IBM pada tahun 2001. Pada tahun 2003 platform Eclipse tumbuh dengan cepat sebagai aplikasi *open source* dan *commercial extensions*, sehingga menjadi pengembang yang terkenal. Pada tahun 2004 Eclipse *Foundation* yang merupakan sebuah organisasi nonprofit dengan keprofesionalannya serta *independent* dapat mengambil alih penuh platform Eclipse. Eclipse adalah pengembang java terkemuka, karena unik dan arsitektur yang dapat diperluas serta dapat digunakan untuk berbagai macam bahasa pemrograman.

2.5.2 Arsitektur Eclipse

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah *kernel*, yang mengangkat *plug-in*. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah diinstal. Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform* (RCP). Berikut ini adalah komponen yang membentuk RCP:

1. *Core platform*
2. OSGi
3. SWT (*Standard Widget Toolkit*)
4. JFace
5. Eclipse *Workbench*

Secara standar Eclipse selalu dilengkapi dengan JDT (*Java Development Tools*), *plug-in* yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (*Plug-in Development Environment*) untuk mengembangkan *plug-in* baru. Eclipse beserta *plug-in*-nya diimplementasikan dalam bahasa pemrograman Java.

Konsep Eclipse adalah IDE yang terbuka (*open*), mudah diperluas (*extensible*) untuk apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, Eclipse tidak saja untuk mengembangkan program Java, akan tetapi dapat digunakan untuk berbagai macam keperluan, cukup dengan menginstal *plug-in* yang

dibutuhkan. Apabila ingin mengembangkan program C/C++ terdapat *plug-in* CDT (C/C++ *Development Tools*). Selain itu, pengembangan secara visual bukan hal yang tidak mungkin oleh Eclipse, *plug-in* UML2 tersedia untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat *plug-in* sesuai dengan keinginannya.

2.5.3 Perkembangan Eclipse

Sejak tahun 2006, Eclipse Foundation mengkoordinasikan peluncuran Eclipse secara rutin dan simultan yang dikenal dengan nama *Simultaneous Release*. Setiap versi peluncuran terdiri dari Eclipse Platform dan juga sejumlah proyek yang terlibat dalam proyek Eclipse. Tujuan dari sistem ini adalah untuk menyediakan distribusi Eclipse dengan fitur-fitur dan versi yang terstandarisasi. Hal ini juga dimaksudkan untuk mempermudah deployment dan maintenance untuk sistem enterprise. Adapun versi Eclipse yang telah diluncurkan yaitu :

1. Eclipse 3.0
2. Eclipse 3.1
3. Callisto
4. Europa
5. Ganymede
6. Galileo
7. Helios
8. Indigo
9. Juno
10. Kepler

2.6 Software Develpment Kit (SDK)

Software Development Kit (SDK) adalah suatu kit atau library dari bahasa pemrograman untuk pengembangan atau pembangunan suatu perangkat lunak dan biasanya SDK terdiri dari kumpulan tools yang dibutuhkan. Misalnya bahasa pemrograman java, mempunyai SDK yang berisi suatu *library* yang dapat digunakan untuk membuat suatu aplikasi berbasis java.

2.7 Java Development Kit (JDK)

Java Development Kit (JDK) adalah sekumpulan perangkat lunak yang dapat kamu gunakan untuk mengembangkan perangkat lunak yang berbasis Java, Sedangkan JRE adalah sebuah implementasi dari Java Virtual Machine yang benar-benar digunakan untuk menjalankan program java.Biasanya, setiap JDK berisi satu atau lebih JRE dan berbagai alat pengembangan lain seperti sumber kompiler java, bundling, debuggers, development libraries dan lain sebagainya. Perbedaan JDK dengan SDK *(Software Development Kit)* yaitu JDK adalah sebuah SDK tetapi sebuah SDK tidak harus menjadi sebuah JDK.

2.8 Android Development Tools (ADT)

Android Development Tools (*ADT*) adalah *plugin* untuk Eclipse *Intergrated Development Environment* (*IDE*) yang dirancang untuk memberikan lingkungan yang terpadu di mana untuk membangun aplikasi Android.

ADT memperluas kemampuan Eclipse untuk membiarkan para *developer* lebih cepat dalam membuat proyek baru Android, membuat aplikasi UI, menambahkan komponen berdasarkan Android *Framework* API, *debug* aplikasi dalam pengunaan Android SDK, dan membuat file APK untuk mendistribusikan aplikasi.

Mengembangkan aplikasi di Eclipse dengan ADT sangat dianjurkan dan merupakan cara tercepat untuk memulai membuat aplikasi Android, karena banyak kemudahan-kemudahan sebagai tools yang terintegrasi seperti, custom XML editor, dan *debug panel ouput*. Selain itu ADT memberikan dorongan luar biasa dalam mengembangkan aplikasi Android.