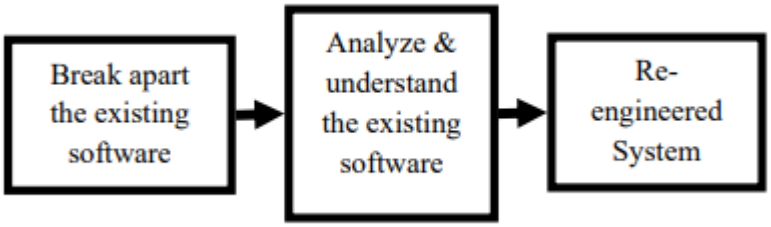
BAB 2
LANDASAN TEORI

2.1 Software Re-engineering

Software Re-engineering adalah proses memodifikasi perangkat lunak yang sudah ada menjadi perangkat lunak baru yang mudah dalam pemeliharaannya. Tetapi pada dasarnya proses dari *Software Re-engineering* adalah memeriksa kembali dan menganalisis isinya, mengubah konten yang diperlukan sesuai dengan kebutuhan yang nantinya akan digabungkan menjadi suatu produk yang baru[7]. Proses dari Software Re-engineering dapat dilihat pada **Gambar 2.1.**



Gambar 2.1 Software Re-engineering

2.1.1 Taksonomi Software Re-engineering

Berikut beberapa sub bagian dari taksonomi dan ruang lingkup *Software Re-engineering*[7].

1. Forward Engineering

Forward Engineering sebagai pengembang *software* dengan *code*, tapi dengan beberapa proses penyaringan baru. Untuk mengubah sebuah fitur yang ada dalam sistem, pekerjaan dilakukan di level abstraksi, dimana informasi tentang karakteristik yang disajikan dengan jelas[7].

2. Reverse Engineering

Reverse Engineering adalah proses menganalisis suatu subjek sistem untuk mengidentifikasi komponen sistem dan menciptakan representasi baru dari sistem dalam bentuk lain atau pada tingkat abstraksi yang lebih tinggi[7].

3. Re-documentation

Re-documentation adalah proses dimana merevisi atau membuat representasi semantik yang ekuivalen dalam level abstraksi yang sama[8].

4. Reverse Design or Design Recovery

Reverse Design or Design Recovery adalah proses dimana menciptakan kembali abstraksi dari kombinasi kode, desain yang sebelumnya (jika ada), pengalaman pribadi, domain aplikasi, dan pengetahuan umum tentang masalah yang terjadi[8].

5. Restructuring

Restructuring adalah proses transformasi satu bentuk representasi ke bentuk lain pada tingkat abstraksi yang relatif sama[8].

6. Recode

Recode proses terdiri dari mengubah implementasi karakteristik program. Merestrukturisasi *Control Flow* dan translasi bahasa adalah perubahan tingkat program[8].

7. Redesign

Redesign adalah proses memodifikasi karakteristik dari desain. Kemungkinan perubahan yang dilakukan berupa peningkatan algoritma, merestrukturisasi arsitektur desain, mengubah model data sistem yang ada dalam DB *(database)* atau dalam struktur data[8].

8. Re-specify

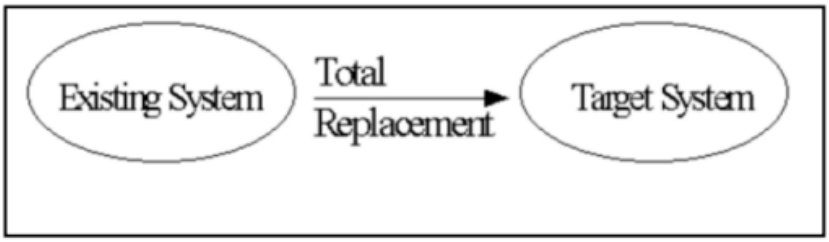
Re-specify proses terdiri dari memodifikasi karakteristik persyaratan. Jenis perubahan ini dapat merujuk pada perubahan persyaratan yang ada saja[8].

2.1.2 Pendekatan Software Re-engineering

1. Pendekatan *Big Bang*

Big Bang atau lebih dikenal dengan pendekatan *"Lump Sum"*, pada suatu saat seluruh sistem diganti. Pendekatan ini sering digunakan ketika membutuhkan pemecahan masalah sistem segera. Misalnya, memigrasikan sistem yang ada ke arsitektur yang berbeda. Keuntungan dari pendekatan ini adalah memungkinkan sistem untuk beroperasi di lingkungan yang berbeda tanpa mempengaruhi integrasi antarmuka. Namun pendekatan ini juga memiliki kelemahan yaitu tidak cocok

untuk proyek besar, memakan banyak waktu dan sumber daya sebelum sistem target ada[7]. Pendekatan ini diilustrasikan pada **Gambar 2.2**.



Gambar 2.2 Pendekatan Big Bang

2.2 Enhanced Reengineering

Enhanced Reengineering adalah proses rekayasa ulang perangkat lunak yang memanfaatkan banyak metode dan level abstraksi, untuk mengubah sistem perangkat lunak yang sudah ada menjadi sistem perangkat lunak yang baru. Pada metode enhanced reengineering ini menggunakan sistem *forward engineering* dan *reverse engineering*. Pada awalnya metode enhanced reengineering ini melakukan studi kelayakan untuk menguji kompatibilitas sistem, dan kemudian membangun komponen yang diperlukan. Kemudian setelah mengumpulkan persyaratan, masuk ke fase kedua yaitu pemetaan restrukturisasi spesifikasi persyaratan perangkat lunak (SRS) untuk menyelesaikan dokumen desain. Dokumen yang dirancang ulang hasil dari fase kedua. Pada tahap ketiga, bagian pemrograman disesuaikan dengan perubahan yang dilakukan pada dokumen yang didesain ulang, pada tahap ini dimungkinkan untuk memutar kembali ke tahap sebelumnya. Kemudian uji ulang dan integrasikan kembali modul perangkat lunak yang ada untuk menjalankan fungsi tertentu. Selama fase ini, membandingkan kinerja sistem perangkat lunak yang ada dengan perangkat lunak baru. Hasil yang diperoleh konsisten, algoritma yang lebih baik akan ditukar dengan sistem yang ada. Manfaat dari proses ini adalah mengurangi kompleksitas dan meningkatkan kualitas perangkat lunak baru. Setelah menyelesaikan berbagai integrasi unit, sistem yang dimodifikasi akan dikerahkan untuk mendapatkan sistem target sesuai dengan kebutuhan pengguna[7].

Berikut ini adalah beberapa tahap yang dilakukan pada *Enhanced Reengineering* dan penjelasannya:

a. Studi kelayakan dan kebutuhan

Pada tahap ini dilakukan studi kelayakan dan perlu dilakukan verifikasi konfigurasi dan kompatibilitas sistem komputer. Setelah studi kelayakan selesai, kebutuhan sistem didefinisikan ulang sesuai dengan keinginan pengguna. SRS memiliki semua persyaratan yang tertulis dalam dokumen resmi[7].

b. Restrukturisasi spesifikasi kebutuhan sistem.

Tahap ini menggambarkan secara rinci proses SRS yang direstrukturisasi. Dokumentasi adalah atribut penting dalam proses pengembangan perangkat lunak karena memproduksi komponen dari keseluruhan proses rekayasa ulang dan berfungsi sebagai perencana untuk produk akhir. Pada tahap ini, para ahli membandingkan kebutuhan sistem yang ada dengan mekanisme sistem yang baru. SRS digunakan untuk mengintegrasikan SRS baru dengan SRS sudah yang ada[7].

c. Design to Code

Langkah ini memberikan informasi rinci tentang desain proses kode. Pada tahap ini sudah tepat untuk mendesain ulang kode-kode terdokumentasi yang telah dilakukan oleh programmer. Biasanya, algoritma dari sistem lama akan diimplementasikan dengan cara pengembangan tradisional dan bahasa dengan fitur yang ada akan ditulis ulang dalam fitur baru[7].

d. Evaluasi Performa Sistem baru dengan yang sudah ada

Tahap ini memberikan rincian tentang proses pengujian ulang. Untuk pengujian ulang, terlebih dahulu ambil aplikasi perangkat lunak yang sudah ada maupun yang baru. Kemudian membandingkan kinerja fungsionalitas dari aplikasi yang ada dengan fungsionalitas dari aplikasi yang baru[7].

e. Implementasi

Tahap ini adalah tahap akhir dari proses mekanisme *Enhanced Re-engineering.* Sesuai dengan tahapan hasil dari rekayasa ulang sebelumnya, implementasi suatu perangkat lunak dapat dilakukan. Dalam implementasinya, semua bagian di migrasi dari sistem yang sebelumnya ke sistem yang baru, berdasarkan empat tahap sebelumnya[7].

2.3 *Model View Controller* (MVC)

Pola desain *Model-View-Controller* adalah salah satu pola desain terpenting di ilmu komputer. Sementara sebagian besar pola mengatasi masalah tertentu, *model-view-controller* (MVC) menggambarkan arsitektur sistem objek. MVC dapat diterapkan pada sub sistem yang terisolasi atau seluruh aplikasi. Berikut penjelasan dari masing - masing komponen MVC[9]:

1. Model

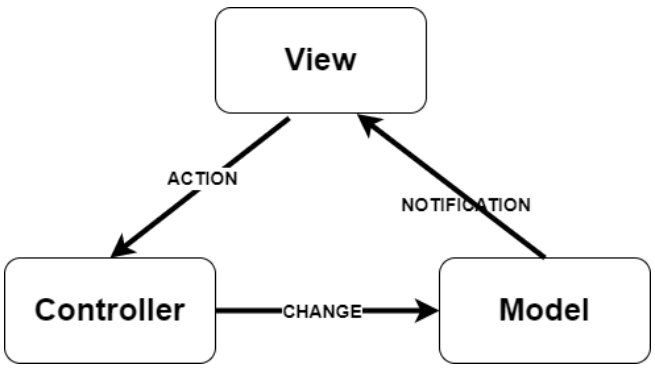
   Objek data model menyimpan, merangkum, dan mengabstraksi data. Objek data model tidak boleh berisi metode atau logika khusus untuk membuat aplikasi.

2. View

   Objek view berfungsi menampilkan informasi dalam model data kepada pengguna.

3. Controller

   Objek *controller* mengimplementasikan tindakan aplikasi anda. Tindakan biasanya dimulai oleh objek tampilan sebagai *response* terhadap *event* pengguna. Pola desain MVC dapat dilihat pada **Gambar 2.3.**



Gambar 2.3 Hubungan antar komponen MVC

2.4 Flutter

Flutter merupakan kerangka kerja buat menciptakan pelaksanaan *cross-platform* yang bertujuan buat berbagi pelaksanaan dalam perangkat seluler yang berperforma tinggi tinggi. *Flutter* dirilis secara *public* dalam tahun 2016 oleh google. Aplikasi *flutter* tidak hanya bisa berjalan pada Android & IOS, namun juga *Fuschia*, sistem operasi berikutnya berdasarkan Google, menentukan *flutter* menjadi framework buat level aplikasinya[10].

Flutter merender setiap komponen dari tampilan menggunakan mesin rendering performa yang tinggi. Sifat ini memberikam kemungkinan untuk membangun aplikasi yang berkinerja tinggi. Dari segi arsitektur, kode C/C++ mesin dikompilasi dengan *android* NDK dan LLVM untuk di *ios,* dan kode Dart apa pun dikompilasi dengan AOT ke dalam kode asli selama kompilasi[10].

Flutter mendukung *hot-reload stateful* saat mengembangkan, yang dianggap sebagai faktor utama untuk meningkatkan siklus pengembangan. *Hot-reload statful* pada dasarnya diimplementasikan dengan menyuntikkan kode sumber yang diperbarui ke dalam Dart Vm yang sedang berjalan tanpa mengubah struktur bagian dalam dari aplikasi, sehingga semua transisi dan tindakan aplikasi akan dipertahankan *hot-reload* dilakukan[10].

2.5 *State Management GET X*

GetX adalah solusi managemen keadaan reaktif yang disederhanakan. Hal ini dibuktikan dengan solusi ekstra ringan dan kuat, yang terdiri dari tiga elemen utama[11]:

1. Performa manajemen status yang tinggi
2. Injeksi ketergantungan yang cerdas
3. Cepat
4. Manajemen rute

GetX adalah solusi manajemen status yang sangat populer. Faktanya bahwa memiliki komunitas aktif yang besar dan memosisikan dirinya sebagai kerangka

kerja yang memecahkan masalah utama yang sulit dari pengembangan *flutter* (tidak hanya *state management*, tetapi dua lainnya yang disebutkan sebelumnya).

2.6 Maintainability

Maintainability didefinisikan sebagai kemampuan produk perangkat lunak untuk di modifikasi. Modifikasi yang dimaksud disini ialah koreksi, perbaikan atau adaptasi perangkat lunak terhadap perubahan lingkungan dan dalam persyaratan dan spesifikasi fungsional[12].

Ada beberapa upaya yang dilakukan untuk mengukur tingkat pemeliharaan suatu perangkat lunak yang biasa disebut dengan metrik. Metrik yang paling banyak digunakan untuk mengukur tingkat pemelihara pada suatu perangkat lunak dikenal dengan *Maintainability Index* (MI)[12].

2.6.1 Maintainability Index (MI)

Maintainability Index (MI) terdiri dari ekspresi *polynomial* dan menghasilkan angka yang menunjukkan tingkat kemampuan pemeliharaan perangkat lunak secara keseluruhan. Dalam buku *Khan dkk* mendefinisikan MI sebagai kombinasi dari metrik perangkat lunak yaitu *Cyclomatic Complexity* (CC), *Halstead`s Volume* (V) dan *Lines of Code* (LOC) yang mempengaruhi dalam pemeliharaan suatu perangkat lunak[12]. Untuk formula dari *Maintainability Index* dapat dilihat seperti berikut:

$$MI = \frac{171 - 5.2 * \log(HV) - 0.23 * CC - 16.2 * \ln(LOC)}{171} * 100$$

Dimana:

MI = *Maintainability Index*

HV = *Halstead Volume*

CC= Cyclomatic Complexity per *module*

LOC = Line of Code per *module.*

Tabel 2.1 Rentang Nilai Maintainability Index

Rentang	Keterangan
$20 \leq MI \leq 100$	Dapat dipelihara dengan baik
$10 \leq MI \leq 20$	Cukup untuk dipelihara dengan baik
$0 \leq MI \leq 10$	Sulit untuk dipelihara

2.6.2 *Halstead Volume*

Halstead Volume adalah salah satu metrik yang dikenal dalam ilmu perangkat lunak. Adapun formula pada metrik *Halstead Volume* sebagai berikut:

n1 = jumlah operator yang berbeda

n2 = jumlah operan yang berbeda

N1 = jumlah total operator yang berbeda

N2 = jumlah total operan yang berbeda

Kalkulasi *Halstead Vocabulary* (n) dan *length* (N)

n = n1 + n2

N = N1 + N2

Program volume (V)

V = N * 2log(n).

2.6.3 *Cyclomatic Complexity*

V (g) = e - n + p

Dimana:

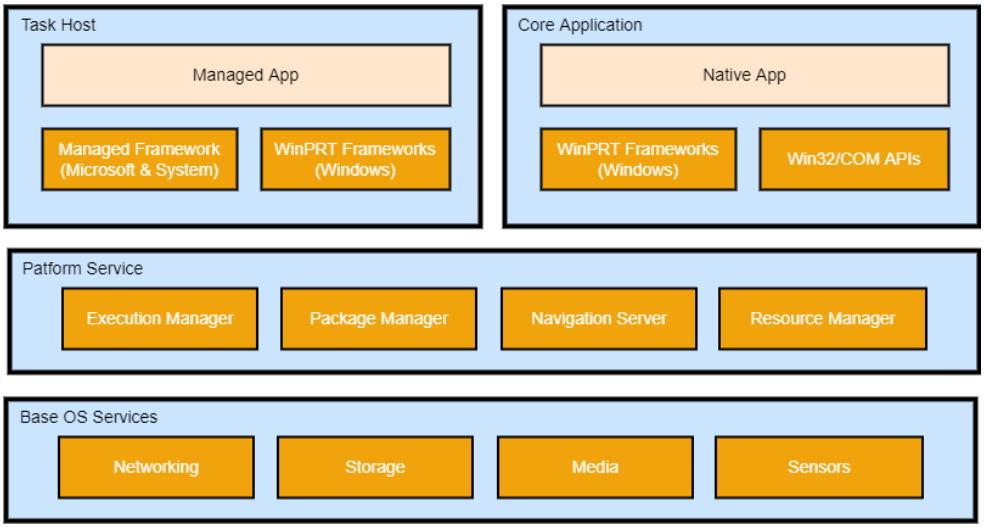
e = jumlah rusuk dalam suatu graf

n = jumlah node dalam suatu graf

p = jumlah komponen penghubung

2.7 Windows Phone

Windows Phone adalah serangkaian sistem operasi yang dikembangkan oleh Microsoft dan merupakan penerus dari *platform Windows Mobile*. Sistem operasi ini diluncurkan bulan Oktober 2010 dan diluncurkan di Asia pada awal tahun 2011. Versi terakhir dari sistem operasi ini adalah Windows Phone 8.1 yang diluncurkan bulan April 2014. Microsoft menciptakan *user interface* baru pada Windows Phone dengan bahasa desain yang disebut *modern design language*[13]. Dalam Windows Phone terdapat arsitektur seperti pada **Gambar 2 – 4.**



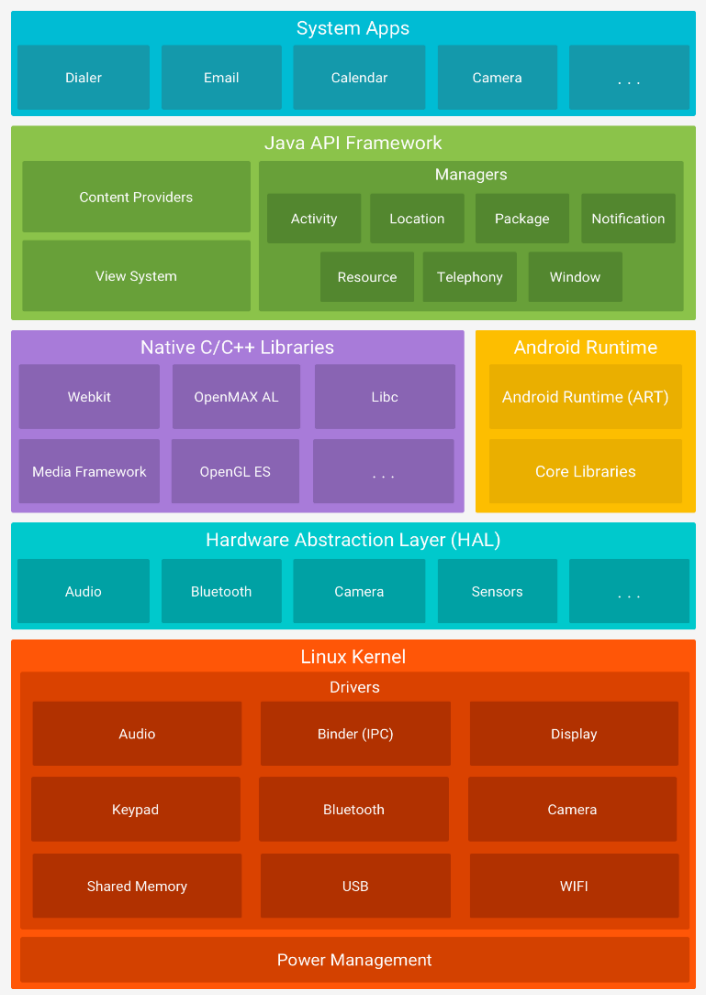
Gambar 2.4 Arsitektur Platform Windows Phone

Di bagian atas dari *stack* terdapat dua model aplikasi yang berbeda. Kotak yang berlabel *"Task Host"* mewakili model aplikasi XAML, yang telah menjadi model utama sejak peluncuruan Windows Phone 7. Kotak yang berlabel *"Core Application"* merupakan model aplikasi baru untuk Windows Phone yang merupakan bagian dari model aplikasi Windows 8. Saat Windows Phone 8 rilis, model aplikasi ini hanya mendukung *native* aplikasi yang menggunakan Direct3D[13].

Kedua model aplikasi yang bergantung pada sebuah *shared set* dari layanan *core platform.* Sebagian besari dari aplikasi *store* hanya pernah menampilkan layanan ini secara tidak langsung[13].

2.8 Android

Android adalah salah satu sistem operasi seluler yang paling banyak digunakan saat ini. Sistem operasi seluler android didasarkan pada kernel linux dan di kembangkan oleh google. Sistem operasi android awalnya dirancang untuk smartphone dan tablet. Sistem operasi android bersifat open source, karena sifatnya yang terbuka itu menjadi favorit bagi banyak konsumen dan pengembangan. Selain itu pengembang perangkat lunak dapat dengan mudah memodifikasi dan menambahkan fitur yang disempurnakan didalamnya untuk memenuhi persyaratan terbaru dari teknologi seluler[14]. Untuk arsitektur dari sistem android dapat dilihat pada **Gambar 2.5.**



Gambar 2.5 Arsitektur Platform Android

Pada gambar arsitektur plaform android diatas terdapat beberapa kotak dimana setiap kotak memiliki fungsi atau kegunaan masing-masing, Adapun penjelasan dari setiap kotak sebagai berikut :

1. Linux Kernel

Linux kernel adalah fondasi platform android. Sebagai contoh, *Android Runtime* (ART) bergantung pada kernel linux untuk fungsionalitas dasar seperti threading dan pengelolaan memori tingkat rendah. Kernel linux bertanggung jawab untuk mengelola memori virtual, jaringan, driver, dan manajemen daya[14].

2. Hardware Abstraction Layer (HAL)

Memberikan antarmuka standar yang mengungkap kemampuan perangkat keras ke kerangka kerja API Java yang lebih tinggi. Hal terdiri atas beberapa modul pustaka, masing-masing menerapkan antarmuka untuk komponen perangkat keras tertentu, seperti modul kamera atau *Bluetooth*. Ketika API kerangka kerja melakukan panggilan untuk mengakses perangkat keras, sistem Android memuat modul pustaka untuk komponen perangkat keras[14].

3. Android *Runtime* (ART)

Android Runtime terdiri dari mesin virtual Dalvik dan pustaka inti Java. Itu terletak ditingkat yang sama dengan lapisan *Native C/C++ Libraries.* Mesin virtual Dalvik adalah Java Virtual Machine yang digunakan untuk menjalankan aplikasi pada perangkat android[14].

4. Native C/C++ Libraries

Lapisan ini memungkinkan untuk menangani jenis data yang berbeda. Data khusus untuk perangkat keras. Semua *libararies* ditulis dalam bahasa C atau C++. *Libraries* ini disebut melalui antarmuka java. Beberapa *library* yang penting adalah[14] :

a. Surface Manager : digunakan untuk mengelola tampilan perangkat.

b. SQLite : digunakan sebagai penyimpanan database di android secara local.

c. Webkit : adalah mesin browser yang digunakan untuk menampilkan HTML.

5. Java API Framework

Lapisan kerangka kerja aplikasi ini menyediakan banyak layanan tingkat tinggi atau API utama untuk aplikasi dalam bentuk kelas Java. *Developer* diizinkan untuk

menggunakan layanan ini dalam aplikasi mereka. Ini adalah blok dimana pengembangan dapat berinteraksi langsung dengan aplikasi yang sedang di bangun. Blok penting dari kerangka kerja aplikasi sebagai berikut[14]:

1. *Activity Manager* : ini mengelola siklus hidup pada aplikasi.
2. *Content Provider* : digunakan untuk mengelola pembagian data antar aplikasi, mengelola cara mengakses data dari aplikasi lain.
3. *Telephony Manager* : digunakan untuk mengelola semua fungsi terkait panggilan suara.
4. *Location Manager* : Digunakan untuk manajemen lokasi, menggunakan GPS.
5. *Resource Manager* : mengelola berbagai jenis sumber daya yang digunakan dalam aplikasi.

Android dilengkapi dengan serangkaian aplikasi inti untuk email, per pesanan SMS, kalender *browser*, kontak, dll. Aplikasi yang disertai dengan platform tidak memiliki status khusus pada aplikasi yang pengguna ingin install. Jadi, aplikasi pihak ketiga dapat menjadi browser web utama, pengelola pesan SMS atau bahkan *keyboard* utama (beberapa pengecualian berlaku, seperti aplikasi setting sistem).

2.8.1 Android Version

Perkembangan versi android dari awal perilisan hingga sampai sekarang, diantaranya:

1. 1.0 - 1.1 (September 2018)
2. Cupcake: 1.5 (30 april 2009)
3. Donut: 1.6 (15 September 2009)
4. Eclair: 2.0 - 2.1.X (26 Oktober 2009)
5. Froyo: 2.2.X (20 May 2010)
6. Gingerbread: 2.3 - 2.3.4 (6 Desember 2010)
7. Honeycomb: 3.0.X - 3.2 (22 Februari 2011)
8. Ice Cream Sandwich: 4.0 - 4.0.4 (Desember 2011)
9. Jelly Bean: 4.1 - 4.3 (9 Juli 2012)
10. Kitkat: 4.4 - 4.4W (4 September 2013)

11. Lollipop: 5.0 - 5.1 (25 Juni 2014)
12. Marshmallow: 6.0 (Oktober 2015)
13. Nougat: 7.0 - 7.1 (22 Agustus 2016)
14. Oreo: 8.0 - 8.1 (Agustus 2017)
15. Pie: 9 (6 Agustus 2018)
16. Android Q: 10 (3 September 2019)
17. Android 11 (8 September 2020)
18. Android 12 (4 Oktober 2021)

2.8.2 *Android Market*

Saat ini teknologi seluler sedang banyak digunakan diseluruh dunia. Sejak tahun 2008 pengguna teknologi mobile berkembang sangat pesat. Banyak ponsel tersedia dengan banyak sistem operasi. Android adalah sistem operasi seluler open *source* yang tersedia di banyak smartphone. Menurut Google 1,3 juta, Perangkat android diaktifkan setiap hari. Menurut laporan Gartner, Android Google menangkap total 82% pasar pada tahun 2016. Pada total kuartal terakhir 2016, 352 juta smartphone terjual habis[15].

Pada juni 2017, Google Play Mobile App Store telah merilis lebih dari 3.000.000 aplikasi Android, dan lebih dari 80 miliar aplikasi telah diunduh. Pada tahun 2017, perusahaan Google I/O menemukan bahwa mereka memiliki lebih dari 2 miliar pengguna aktif Android per bulan, tidak seperti tahun sebelumnya, dengan angka sekitar 1,5 miliar pengguna aktif[9]. Pada tahun 2018 pangsa pasar android meningkat lebih dari 80%[16].

2.9 Analisis dan Desain Berorientasi Objek

OOAD adalah metode analisis yang memeriksa *requirement* dari sudut pandang kelas – kelas dan *object* yang ditemui dalam ruang lingkup permasalahan yang mengarahkan arsitektur *software* yang didasarkan pada manipulasi objek-objek sistem atau sub-sistem. OOAD merupakan cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata[17].

Selanjutnya, bertujuan untuk menghilangkan kompleksitas transisi antar fase pengembangan perangkat lunak, karena dalam pendekatan berorientasi objek, notasi yang digunakan pada tahap desain dan analisis yang sebenarnya sekarang relatif sama, tidak seperti pendekatan klasik, karena simbol yang digunakan pada tahap analisis berbeda. Ini membuat transisi antar tahap pengembangan menjadi rumit[17].

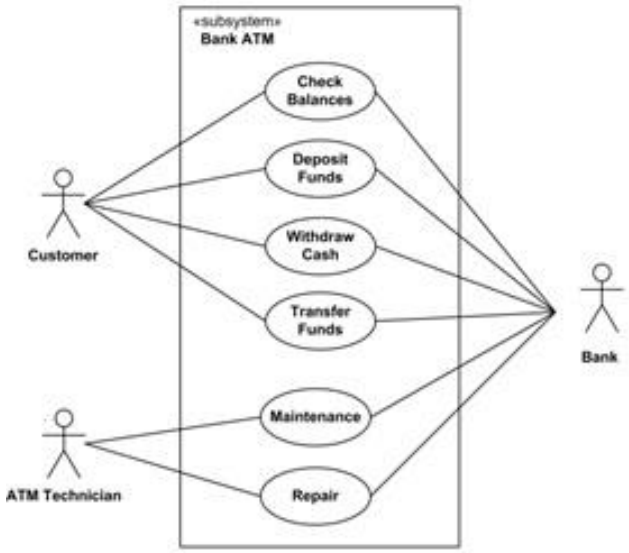
Disamping itu dengan pendekatan objek membawa pengguna kepada abstraksi yang lebih dekat dengan dunia nyata, karena umumnya pada dunia nyata yang sering dilihat oleh pengguna adalah suatu objeknya bukan fungsinya. Berbeda dengan pendekatan terstruktur yang hanya mendukung abstraksi pada level fungsional. Adapun dalam pemrograman berorientasi objek menekankan beberapa konsep seperti berikut ini: *Class, Object, Abstract, Encapsulation, Polymorphism, Inheritance* dan yang terakhir UML (*Unified Modelling Language*)[17].

Unified Modeling Language (UML) adalah bahasa grafis untuk memvisualisasikan, menentukan, membangun, dan mendokumentasikan artefak dari sistem perangkat lunak-intensif. Biasanya UML ditulis dengan *blueprint*, termasuk hal-hal konseptual seperti proses bisnis dan fungsi sistem serta hal-hal yang konkret seperti pernyataan bahasa pemrograman, skema *database*, dan komponen perangkat lunak yang dapat digunakan kembali[17].

Dokumentasi UML menyediakan 14 macam diagram untuk membuat model aplikasi berorientasi objek yang 5 diantaranya sebagai berikut:

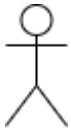
1. *Use Case Diagram*

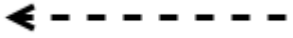
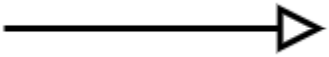
   Use Case Diagram adalah metodologi yang digunakan dalam analisis sistem untuk mengidentifikasi, mengklarifikasi, dan mengatur persyaratan sistem. Diagram UML (*Unified Modeling Language*) digunakan untuk membuat notasi standar untuk pemodelan objek dunia nyata dan sistem[17]. Contoh *use case diagram* dapat dilihat pada **Gambar 2.6.**



Gambar 2.6 Contoh Use Case Diagram

Tabel 2.2 Simbol pada use case diagram

Simbol	Nama Simbol	Keterangan
	Actor	Seseorang atau sekelompok orang diluar sistem yang berperan dalam satu atau lebih interaksi dengan sistem, ini mewakili dari mana informasi berasal dan kemana perginya.
	Use Case	Gambar ini menunjukkan semua fungsi sistem.
	Asosiasi	Menghubungkan keterkaitan antara actor dengan *use case*

Simbol	Nama Simbol	Keterangan
	*Include*	Relasi *use case* dimana proses bersangkutan akan dilanjutkan ke proses yang dituju
	*Extend*	Relasi *Use Case* tambahan ke sebuah *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
	Generalisasi	Sebuah relasi dimana fungsi yang satu adalah fungsi yang umum dari yang lain

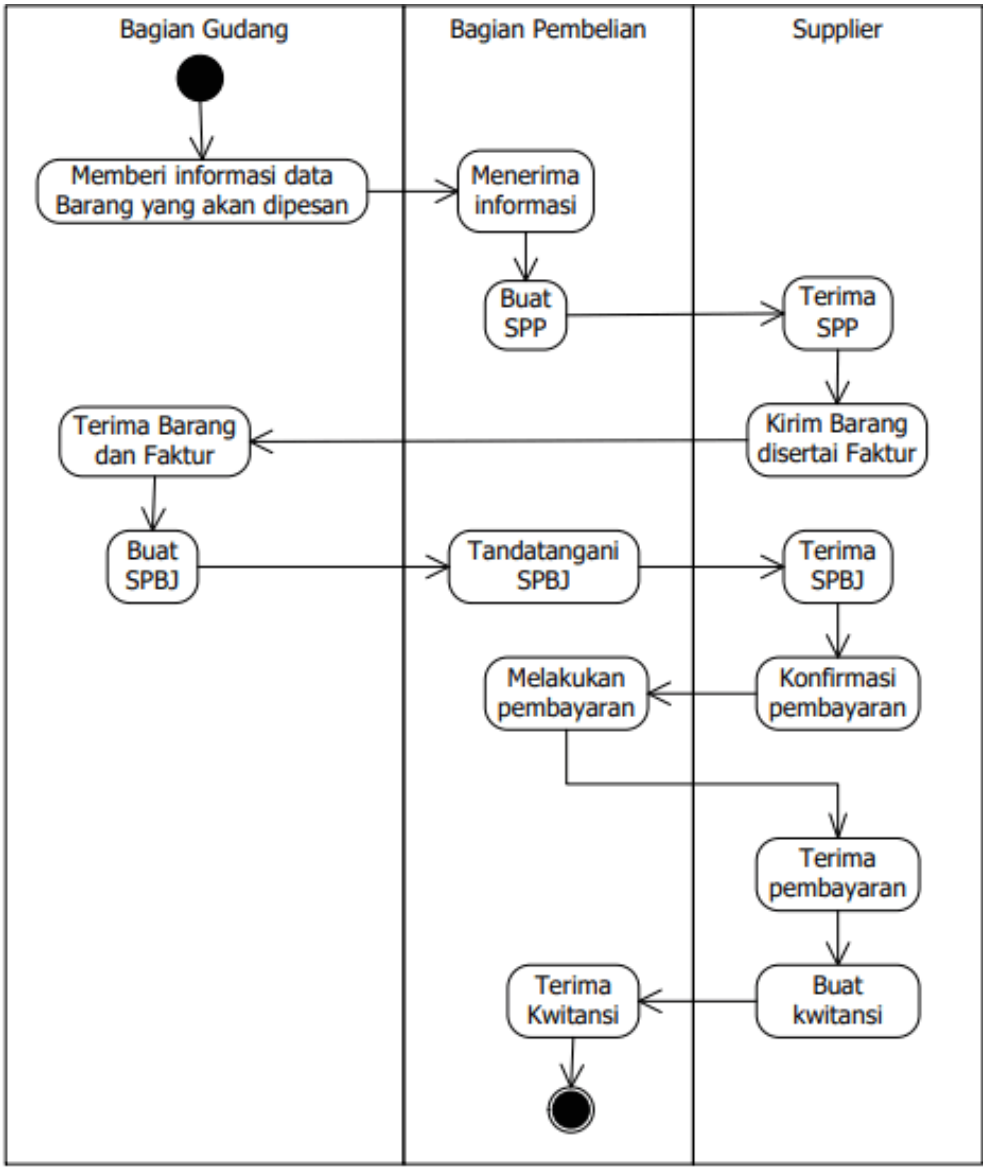
2. *Use Case Scenario*

Use Case Scenario merupakan hasil instansiasi dan penjelajahan dari tiap *use case.* Kompenen *scenario Use Case* dapat dilihat sebagai beriku:

a. *Name*: Memberikan penjelasan singkat tentang nama dari *use case.*
b. *Actors:* Daftar aktor yang dapat mengakses *use case.*
c. *Goals:* Menjelaskan apa yang aktor coba untuk dapatkan dari *use case.*
d. *Preconditions*: Kondisi sistem sebelum *use case* dijalankan.
e. *Summary*: Memberikan penjelasan singkat tentang deskripsi informal dari sesuai *use case.*
f. *Related use cases*: Daftar *use case* yang berhubungan dengan *use case* tersebut.
g. *Steps*: Menjelaskann setiap langkah yang dijalankan pada *use case* tersebut.
h. *Post Conditions*: Kondisi sistem setelah *use case* dijalankan.

3. *Activity Diagram*

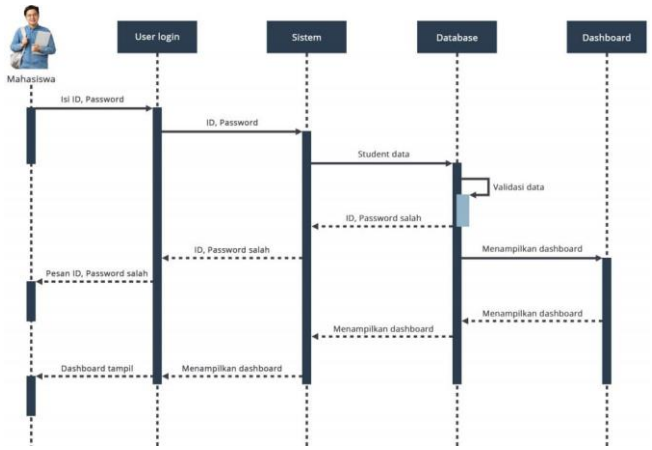
Activity Diagram menggambarkan aliran fungsionalitas dalam suatu sistem informasi. Secara lengkap, *activity diagram* mendefinisikan *workflow* dimulai, dimana berhentinya, aktivitas apa yang terjadi selama *workflow*, dan bagaimana urutan kejadian aktivitas tersebut[17]. Contoh *activity diagram* dapat dilihat pada **Gambar 2.7.**



Gambar 2.7 Contoh Activity Diagram

4. *Sequence Diagram*

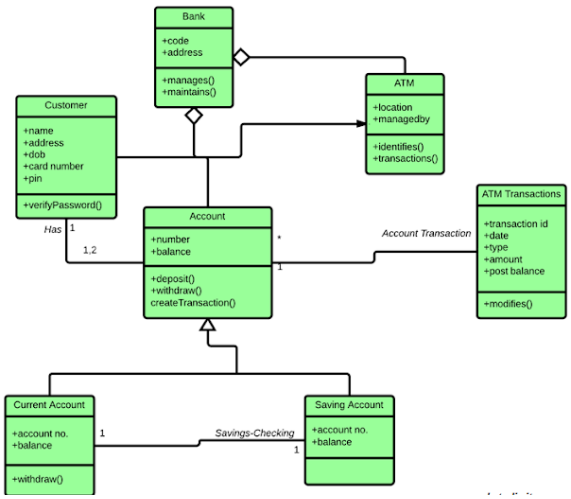
Sequence Diagram alat yang sangat populer dalam pengembangan sistem informasi berorientasi objek untuk menampilkan interaksi antar objek. Selain itu, diagram urutan dapat digunakan sebagai alat desain antarmuka pengguna[17]. Contoh *sequence diagram* dapat dilihat pada **Gmabar 2.8.**



Gambar 2.8 Contoh Squence Diagram

5. *Class Diagram*

Class Diagram adalah jenis diagram struktur statis dalam UML yang menggambarkan struktur sistem dengan menunjukkan sistem *class*, atributnya, metode dan hubungan antar objek[16]. Contoh *class diagram* dapat dilihat pada **Gambar 2.9.**



Gambar 2.9 Contoh Class Diagram