BAB II

TINJAUAN PUSTAKA

2.1 Klasifikasi (*Taksonomi*)

Klasifikasi adalah suatu cara pengelompokan yang didasarkan pada ciri-ciri tertentu. Semua ahli biologi menggunakan suatu sistem klasifikasi untuk mengelompokkan tumbuhan ataupun hewan yang memiliki persamaan struktur, kemudian setiap kelompok tumbuhan ataupun hewan tersebut dipasang-pasangkan dengan kelompok tumbuhan atau hewan lainnya yang memiliki persamaan dalam kategori lain. Hal itu pertama kali diusulkan oleh John Ray yang berasal dari Inggris. Namun ide itu disempurnakan oleh Carl Von Linne (1707-1778), seorang ahli botani berkebangsaan Swedia yang dikenal pada masa sekarang dengan Carolus Linnaeus.

Sistem klasifikasi Linnaeus tetap digunakan sampai sekarang karena sifatnya yang sederhana dan fleksibel sehingga suatu organisme baru tetap dapat dimasukkan dalam sistem klasifikasi dengan mudah. Nama-nama yang digunakan dalam sistem klasifikasi Linnaeus ditulis dalam bahasa Latin karena pada zaman Linnaeus bahasa Latin adalah bahasa yang dipakai untuk pendidikan resmi. Klasifikasi makhluk hidup didasarkan pada persamaan perbedaan ciri dan manfaat yang dimiliki makhluk hidup. Salah satu contoh klasifikasi pada tumbuhan berdasarkan

manfaatnya yaitu tumbuhan dikelompokkan menjadi tanaman obat-obatan, tanaman sandang, tanaman hias, tanaman pangan dan lain-lain (Soepomo,1987).

2.1.1 Dasar-Dasar Klasifikasi Makhluk Hidup

Soepomo pada buku yang berjudul *Morfologi Tumbuhan* tahun 1987 menjelaskan tantang dasar-dasar klasifikasi makhluk hidup yaitu sebagai berikut.

1. Klasifikasi makhluk hidup berdasarkan persamaan yang dimilikinya, persamaan dari beberapa makhluk hidup yang memiliki ciri-ciri dan pola hidup yang sama sehingga dapat digolongkan dalam jenis yang sama.
2. Klasifikasi makhluk hidup berdasarkan perbedaan yang dimilikinya, selain dari beberapa makhluk hidup memiliki persamaan sehingga dapat digolongkan dalam jenis yang sama, namun terdapat perbedaan antara makhluk hidup tersebut.
3. Klasifikasi Makhluk hidup berdasarkan ciri morfologi dan ciri anatomi, klasifikasi makhluk hidup berdasarkan ciri morfologi dan anatomi maksudnya adalah mengelompokan makhluk hidup berdasarkan persamaan dan perbedaan yang dilihat berdasarkan bentuknya dan susunan tubuhnya.
4. Klasifikasi makhluk hidup berdasarkan ciri biokimia, klasifikasi makhluk hidup berdasarkan ciri biokimia contohnya adalah dapat dilihat dari jenis-jenis enzim, jenis-jenis protein dan jenis-jenis DNA yang menjadi penyusun tubuh makhluk hidup tersebut.

5. Klasifikasi makhluk hidup berdasarkan manfaat, dengan mengelompokan makhluk hidup berdasarkan manfaatnya, kita bisa menentukan langkah-langkah yang tepat dalam memanfaatkan kelebihan tersebut secara lebih optimal.

2.1.2 Tujuan Klasifikasi Makhluk Hidup

Tujuan dari klasifikasi makhluk hidup yaitu :

1. Mengelompokkan makhluk hidup berdasarkan persamaan ciri-ciri yang dimiliki
2. Mengetahui ciri-ciri suatu jenis makhluk hidup untuk membedakannya dengan makhluk hidup dari jenis lain
3. Mengetahui hubungan kekerabatan makhluk hidup
4. Memberi nama makhluk hidup yang belum diketahui namanya atau belum memiliki nama.

Selain memiliki tujuan, klasifikasi memiliki manfaat bagi manusia, antara lain :

1. Klasifikasi memudahkan kita dalam mmpelajari makhluk hidup yang sangat beraneka ragam
2. Klasifikasi membuat kita mengetahui hubungan kekerabatan antarjenis makhluk hidup

2.1.3 Tingkatan Takson

Dalam sistem klasifikasi, makhluk hidup dikelompokkan menjadi suatu kelompok besar kemudian kelompok besar ini dibagi menjadi kelompok-kelompok kecil. Kelompok-kelompok kecil ini kemudian dibagi lagi menjadi kelompok yang lebih kecil lagi sehingga pada akhirnya terbentuk kelompok- kelompok kecil yang beranggotakan hanya satu jenis makhluk hidup. Tingkatan-tingkatan pengelompokan ini disebut *takson*. Taksa (takson) telah distandarisasi di seluruh

dunia berdasarkan *International Code of Botanical Nomenclature* dan *International Committee on Zoological Nomenclature.*

Urutan takson antara lain :

Kingdom
Divisio
Clasis
Order
Famili
Genus
Spesies

Keterangan :

1. *Kingdom*. Kingdom merupakan tingkatan takson tertinggi makhluk hidup. Kebanyakan ahli Biologi sependapat bahwa makhluk hidup di dunia ini dikelompokkan menjadi 5 kingdom (diusulkan oleh Robert Whittaker tahun 1969). Kelima kingdom tersebut antara lain : Monera, Protista, Fungi, Plantae, dan Animalia.
2. *Filum/Divisio (Keluarga Besar).* Nama filum digunakan pada dunia hewan, dan nama division digunakan pada tumbuhan. Filum atau division terdiri atas organisme-organisme yang memiliki satu atau dua persamaan ciri. Nama filum tidak memiliki akhiran yang khas sedangkan nama division umumnya memiliki akhiran khas, antara lain *phyta* dan *mycota.*
3. *Kelas (Classis).* Kelompok takson yang satu tingkat lebih rendah dari filum atau division.
4. *Ordo (Bangsa).* Setiap kelas terdiri dari beberapa ordo. Pada dunia tumbuhan, nama ordo umumnya diberi akhiran *ales*.

5. *Famili.* Famili merupakan tingkatan takson di bawah ordo. Nama famili tumbuhan biasanya diberi akhiran *aceae*, sedangkan untuk hewan biasanya diberi nama *idea*.
6. *Genus (Marga).* Genus adalah takson yang lebih rendah dariada famili. Nama genus terdiri atas satu kata, huruf pertama ditulis dengan huruf *kapital*, dan seluruh huruf dalam kata itu ditulis dengan huruf miring atau dibedakan dari huruf lainnya.
7. *Spesies (Jenis).* Spesies adalah suatu kelompok organisme yang dapat melakukan perkawinan antar sesamanya untuk menghasilkan keturunan yang *fertile* (subur).

2.1.4 Kingdom *Plantae*

Plantae adalah organisme multiseluler yang menghasilkan makanan dengan proses fotosintesis. Kerajaan ini meliputi organisme yang berkisar dari lumut yang kecil hingga pohon raksasa. Semua tumbuhan multiseluler dan eukariotik. Salah satu ciri khas tumbuhan adalah adanya pigmen klorofil seperti a dan b dan karotenoid yang membantu untuk mengubah sinar matahari menjadi energi kimia dengan proses fotosintesis (Soepomo,1987).

- **Ciri-ciri kingdom plantae**

Berikut adalah daftar ciri-ciri kingdom plantae. Ciri-ciri inilah yang membedakan kingdom plantae dengan kingdom fungi dan beberapa jenis alga.

1. Multiseluler (memiliki banyak sel)
2. Terdapat dinding sel yang terbuat dari selulosa

3. Eukariotik
4. Mendapatkan makanan dengan cara fotosintesis yang dibantu dengan cahaya matahari
5. bereproduksi secara seksual (putik dan benang sari) maupun aseksual (cangkok, tunas, setek, dll)
6. Hidup di daratan atau perairan
7. Autrotrof (dapat membuat makanan sendiri)

Selain itu, plantae memiliki organ dan sistem organ. Memiliki daun untuk mengumpulkan sinar matahari yang digunakan untuk membuat glukosa. Memiliki akar untuk memperkokoh tumbuhan dan menyerap air. Alat reproduksi seksualnya adalah bunga (Soepomo,1987).

- **Contoh klasifikasi tumbuhan kingdom plantae yaitu :**

Klasifikasi Tumbuhan Jahe

Kingdom: Plantae (Tumbuhan)

Subkingdom: Tracheobionta (Tumbuhan berpembuluh)

Super Divisi: Spermatophyta (Menghasilkan biji)

Divisi: Magnoliophyta (Tumbuhan berbunga)

Kelas: Liliopsida (berkeping satu / monokotil)

Sub Kelas: Commelinidae

Ordo: Zingiberales

Famili: Zingiberaceae (suku jahe-jahean)

Genus: Zingiber

Spesies: *Zingiber officinale* Rosc.

2.2 Tata Nama dan Aturan Binomial Nomenklatur

Banyak makhluk hidup mempunyai nama lokal. Nama ini bisa berbeda antara satu daerah dan daerah lainnya. Untuk memudahkan komunikasi, makhluk hidup harus diberikan nama yang unik dan dikenal di seluruh dunia. Berdasarkan kesepakatan internasional, digunakanlah metode *Binomial Nomenclature*. Metode binominal nomenclature (tata nama ganda), merupakan metode yang sangat penting dalam pemberian nama dan klasifikasi makhluk hidup. Disebut tata nama ganda karena pemberian nama jenis makhluk hidup selalu menggunakan dua kata yaitu nama *genus* dan *spesies* (Martinus dan Hartono, 2008).

Aturan pemberian nama dalam *Binomial Nomenclature* adalah sebagai berikut :

1. Nama spesies terdiri atas dua kata, kata pertama merupakan nama *genus*, sedangkan kata kedua merupakan penunjuk jenis. Contoh: *Zingiber officinale* adalah nama latin dari jahe, genus: Zingiber dan spesies: Officinale.
2. Huruf pertama nama genus ditulis huruf kapital, sedangkan huruf pertama penunjuk jenis digunakan huruf kecil. Contoh: tanaman jahe dengan genus: Zingiber dan spesies: Officinale, maka ditulis Zingiber officinale.
3. Nama spesies harus ditulis berbeda dengan huruf-huruf lainnya (bisa miring atau garis bawah). Contoh: *Zingiber officinale,* <u>Zingiber</u> <u>officinale</u>
4. Jika nama spesies tumbuhan terdiri atas lebih dari dua kata, kata kedua dan berikutnya harus digabung atau diberi tanda penghubung. Contoh: *Hibiscus rosasinensis* atau *Hibiscus rosa-sinensis*.
5. Jika nama spesies hewan terdiri atas tiga kata, nama tersebut bukan nama spesies, melainkan nama subspesies (anak jenis), yaitu nama takson di bawah

spesies. disebut Trinomial nomenklatur. Contoh: *Felix maniculata domestica* (kucing rumah/piaraan).

6. Nama spesies juga mencantumkan inisial pemberi nama tersebut. Contoh: jagung (*Zea Mays L*.) huruf L tersebut merupakan inisial Linnaeus.

2.3 Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* yang menyertakan *middleware* (virtual *machine*) dan sejumlah aplikasi utama. Android merupakan modifikasi dari kernel Linux (Andry, 2011).

Pada awalnya sistem operasi ini dikembangkan oleh sebuah perusahaan bernama Android, Inc. Dari sinilah awal mula nama Android muncul. Android Inc. Adalah sebuah perusahaan *start-up* kecil yang berlokasi di Palo Alto, California, Amerika Serikat yang didirikan oleh Andy Rubin bersama Rich Miner, Nick Sears, dan Chris White. Pada bulan juli 2005, perusahaan tersebut diakuisisi oleh Google dan para pendirinya bergabung ke Google. Andy Rubin sendiri kemudian diangkat menjadi Wakil Presiden divisi *Mobile* dari Google.

Tujuan pembuatan sistem operasi ini adalah untuk menyediakan *platform* yang terbuka, yang memudahkan orang mengakses Internet menggunakan telepon seluler. Android juga dirancang untuk memudahkan pengembang membuat aplikasi dengan batasan yang minim sehingga kreativitas pengembang menjadi lebih berkembang (Andry, 2011).

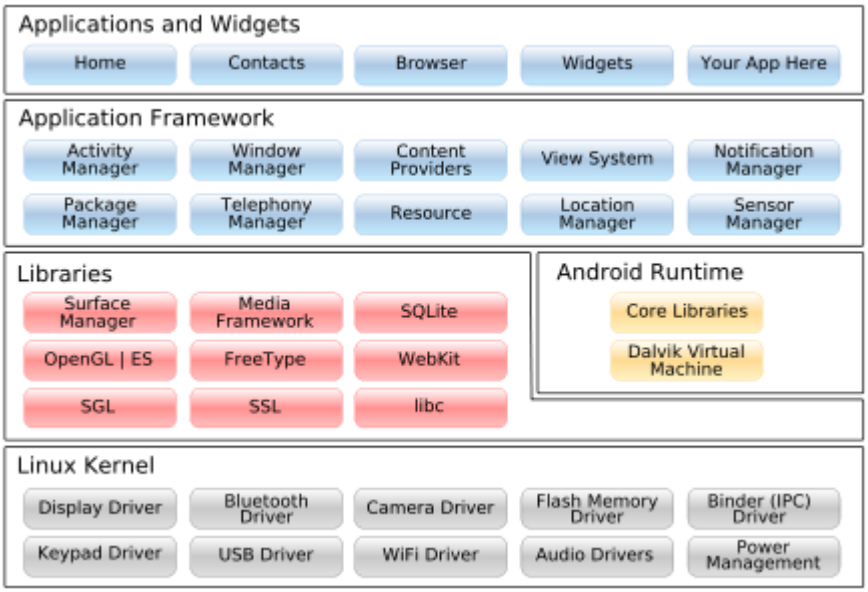
Sebagai *Open Source* dan bebas dalam memodifikasi, di dalam android tidak ada ketentuan yang tetap dalam konfigurasi *Software* dan *Hardware*. Fitur- fitur yang didapat dalam Android antara lain (Lee, 2011) :

1. *Storage* - Menggunakan SQLite, database yang ringan, untuk sebuah penyimpanan data.
2. *Connectivity* - Mendukung GSM/EDGE, IDEN, CDMA, EV-DO, UMTS.
3. *Bluetooth* (termasuk A2DP dan AVRCP), WiFi, LTE, dan WiMax.
4. *Messaging* –Mendukung SMS dan MMS
5. *Web Browser* – Berbasiskan open-*source* WebKit, bersama mesin
6. *Chrome's V8 JavaScript*
7. *Media support* – Termasuk mendukung untuk beberapa media berikut :
8. H.263, H.264 (dalam bentuk 3GP or MP4), MPEG-4 SP, AMR, AMRWB (dalam bentuk 3GP), AAC, HE-AAC (dalam bentuk MP4 atau 3GP), MP3, MIDI, Ogg Vorbis, WAV, JPEG, GIF, dan BMP.
9. *Hardware support* – Sensor akselerasi, Kamera, Kompas Digital, Sensor Kedekatan, GPS.
10. *Multi-touch* –Mendukung *multi-touch screens*
11. *Multi-tasking* – Mendukung aplikasi *multi-tasking*
12. *Flash-support* – Android 2.3 mendukung *Flash* 10.1
13. *Tethering* – Mendukung pembagian dari koneksi Internet sebagai *wired/wireless hotspot*
14. *Play sore* – katalog aplikasi yang dapat di-download dan diinstal pada telepon seluler secara online, tanpa menggunakan PC (Personal Computer).

15. Lingkungan pengembangan yang kaya, termasuk *emulator*, peralatan *debugging*, dan *plugin* untuk Eclipse IDE.

2.3.1 Arsitektur Android

Arsitektur Android dapat digambarkan seperti pada Gambar 2.1 dan secara garis besar Arsitektur Android dapat dijelaskan sebagai berikut (Nazruddin, 2012).



Gambar 2.1 Arsitektur Android (Andry, 2011).

a. Application dan *Widgets*

Application dan *Widgets* ini adalah *layer* dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita download aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di *layer* terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, *browser*, kontak, dan lain-lain. Hampir semua aplikasi ditulis menggunakan bahasa pemrograman Java.

b. Application Frameworks

Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resource*, menjalankan *service background*, mengatur alarm, dan menambah status *notifications*, dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi kategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunakan (*reuse*). Sehingga bisa kita simpulkan *Application Frameworks* ini adalah *layer* dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada *layer* inilah aplikasi dapat dirancang dan dibuat, seperti *content providers* yang berupa sms dan panggilan telepon.

Komponen-komponen yang termasuk di dalam *Application Frameworks* adalah sebagai berikut :

1. Views
2. Content Provider
3. Resource Manager
4. Notification Manager
5. Activity Manager

c. Libraries

Libraries ini adalah *layer* dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *Libraries* untuk menjalankan aplikasinya. Berjalan di atas Kernel, *layer* ini meliputi berbagai library C/C++ inti seperti Libc SSL, serta :

1. *Libraries* media untuk pemutaran media audio dan video
2. *Libraries* untuk manajemen tampilan
3. *Libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D
4. *Libraries SQLite* untuk dukungan *database*
5. *Libraries* SSL dan WebKit terintegrasi dengan *web browser* dan *security*
6. *Libraries* LiveWebcore mencakup modern *web browser* dengan *engine embedded web view*
7. *Libraries* 3D yang mencakup implementasi OpenGL ES1.0 API's.

d. **Android *Run Time***

Layer yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux. Dalvik Virtual Machine (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam Android *Run Time* dibagi menjadi dua bagian yaitu :

1. *Core Libraries*: Aplikasi Android dibangun dalam bahasa Java, sementara Dalvik sebagai virtual mesinnya bukan *Virtual Machine* Java, sehingga diperlukan sebuah *Libraries* yang berfungsi untuk menterjemahkan bahasa Java/C yang ditangani oleh *Core Libraries*.
2. *Dalvik Virtual Machine*: Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat Linux Kernel untuk melakukan *threading* dan manajemen tingkat rendah.

e. **Linux Kernel**

Linux Kernel adalah *layer* dimana inti dari sistem operasi Android itu berada. Berisi file-file sistem yang mengatur sistem *processing*, *memory*, *resource*, *drivers*, dan sistem-sistem operasi Android lainnya. Linux Kernel yang digunakan Android adalah Linux Kernel *release* 2.6 (Nazruddin, 2012).

2.3.2 Versi Android

Sejak pertama kali muncul sampai sekarang, Android telah memiliki sejumlah pembaharuan. Pembaharuan ini dilakukan untuk memperbaiki *bug* dan menambah fitur-fitur yang baru. Versi-versi yang ada pada android yaitu (Developers, 2014) :

1. Android versi 1.1

   Pada tanggal 9 Maret 2009, Google merilis Android versi 1.1.

   Android ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, *voice search*, 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*). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini, yaitu kemampuan merekam dan menonton video dengan 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 sistem.

3. Android versi 1.6 (*Donut*)

   Donut (versi 1.6) dirilis pada September 2009 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; kamera, *camcorder* dan galeri yang dintegrasikan; CDMA / EVDO, 802.1x, VPN, *Gestures*, dan *Text-to-speech engine*; kemampuan dial kontak; teknologi *text to change speech*. (tidak tersedia pada semua ponsel; pengadaan resolusi VWGA.

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 pengoptimalan hardware, peningkatan Google Maps 3.1.2, perubahan UI dengan *browser* baru dan dukungan HTML5, daftar kontak yang baru, dukungan flash untuk kamera 3,2 MP, *digital Zoom*, dan *Bluetooth* 2.1.

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, intergrasi V8 *JavaScript engine* yang dipakai Google Chrome yang mempercepat kemampuan *rendering* pada *browser*, pemasangan aplikasi dalam SD *Card*, kemampuan WiFi *Hotspot* portabel, dan kemampuan pembaruan secara otomatis 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 Android versi ini antara lain peningkatan kemampuan permainan (*gaming*), peningkatan fungsi *copy paste*, desain ulang layar antar muka (*User Interface*), 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 3.0/3.1 (*Honeycomb*)

   Android *Honeycomb* dirancang khusus untuk tablet. Android versi ini mendukung ukuran layar yang lebih besar. *User Interface* pada *Ho*neycomb juga berbeda karena sudah didesain untuk tablet. *Honeycomb* juga mendukung *multiprocessor* dan juga akselerasi perangkat keras (*hardware*) untuk grafis. Tablet pertama yang dibuat dengan menjalankan *Honeycomb* adalah Motorola Xoom. Perangkat tablet dengan platform Android 3.0 telah banyak hadir di Indonesia. Perangkat yang pertama muncul bernama *Eee Pad Transformer* produksi dari Asus yang masuk pasar Indonesia pada Mei 2011.

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

   Diumumkan pada tanggal 19 Oktober 2011, membawa fitur *Ice Cream Sandwich* 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. Ponsel

pertama yang menggunakan sistem operasi ini adalah Samsung Galaxy Nexus (Nazruddin, 2012).

9. Android versi 4.1 (*Jelly Bean*)

   Android Jelly Bean yang diluncurkan pada acara Google I/O lalu membawa sejumlah keunggulan dan fitur baru. Penambahan baru diantaranya meningkatkan *input keyboard*, desain baru fitur pencarian, *user interface* yang baru dan pencarian melalui *voice search* yang lebih cepat.

10. Android versi 4.4 (*Kitkat*)

    Android 4.4 "KitKat" adalah versi dari sistem operasi telepon genggam Android yang dikembangkan oleh Google. Google mengumumkan Android 4.4 KitKat pada tanggal 3 September 2013. memiliki 512 MB RAM sebagai minimum yang disarankan untuk perangkat Android.

11. Android Versi 5.0 (*Lollipop*)

    Pembaruan utama terbaru versi Android adalah Lollipop 5.0, yang dirilis pada 3 November 2014. *Lollipop* adalah update Android paling besar dan ambisius dengan lebih dari 5.000 API baru untuk para *developer*. Perangkat yang menggunakan OS Android L ini akan mampu berintegrasi antar perangkat seperti *smartphone,* tablet berbasis Android (Developers, 2014).

2.3.3 Android SDK

Android SDK adalah tools API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada platform android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci

yang di release oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Sebagai platform aplikasi-netral, android *member* anda kesempatan untuk membuat aplikasi yang kita butuhkan yang bukan merupakan aplikasi bawaan *Handphone* atau *Smartphone* (Developers, 2014).

2.3.4 Eclipse

Eclipse adalah sebuah komunitas bagi individu dan organisasi yang ingin berkolaborasi secara *commercially-friendly* perangkat lunak bersifat *opensource*. Proyek perusahaan terfokus pada membangun sebuah platform pengembangan terbuka terdiri dari *extensible framework*, *tools* dan *runtimes* untuk membangun, menyebarkan dan mengelola perangkat lunak (Eclipse, 2014).

Android dikembangkan menggunakan bahasa pemrograman Java. Telah banyak beredar Java IDE seperti JBuilder dan NetBeans. Namun Open Handset Alliance dan Google telah memilih menggunakan Eclipse sebagai Java IDE dalam pengembangan Android. Berikut ini dijelaskan mengapa Eclipse direkomendasikan sebagai Java IDE untuk aplikasi Android (DiMarzio, 2008) :

1. Sesuai dengan karakteristik Android yang terbuka bagi para pengembang, Eclipse merupakan salah satu yang memiliki fitur lengkap dan gratis dari semua Java IDE yang ada. Eclipse juga sangat mudah digunakan dengan waktu pembelajaran yang minimal.

2. Open Handset Alliance telah merilis *plugin* Android untuk Eclipse sehingga memungkinkan untuk membuat projek Android yang spesifik, melakukan kompilasi, dan menggunakan Android *Emulator* untuk melakukan *debug*.

2.3.5 Android Development Tools (ADT)

Android Development Tools adalah *plugin* yang didesain untuk IDE Eclipse yang memberikan kita kemudahan dalam menggembangkan aplikasi Android. Dengan adanya ADT untuk eclipse akan memudahkan *develop* dalam membuat aplikasi *project* Android, membuat GUI aplikasi, dan menambahkan komponen-komponen yang lainnya, begitu juga kita dapat melakukan running aplikasi menggunakan Android SDK melalui Eclipse. Dengan ADT juga kita dapat membuat *package* Android (.apk) yang digunakan untuk distribusi aplikasi Android yang kita rancang (Developers, 2014).

2.4 Pengertian API

Application Programming Interface (API) bukan hanya satu *set class* dan *method* atau fungsi dan *signature* yang sederhana. API yang bertujuan utama untuk mengatasi "*clueless*" dalam membangun *software* yang berukuran besar, berawal dari sesuatu yang sederhana sampai ke yang kompleks dan merupakan perilaku komponen yang sulit dipahami (Halim, 2011).

2.5 Metodologi Pengembangan Sistem

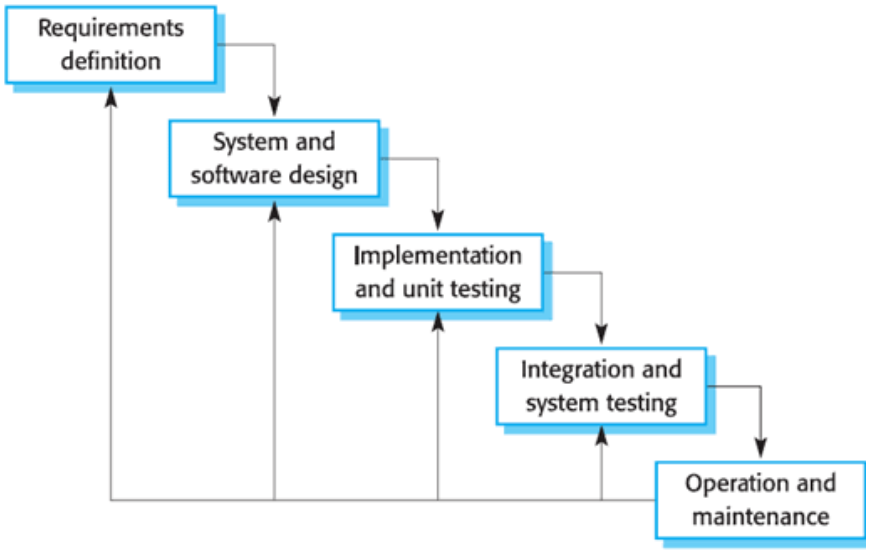
Adapun metodologi yang digunakan dalam Pengembangan Aplikasi Sistem Pembelajaran Klasifikasi (Taksonomi) dan Tata Nama Ilmiah (Binomial

Nomenklatur) pada Kingdom Plantae (Tumbuhan) Berbasis Android ini yaitu: Metode *Waterfall*, dan desain menggunakan *Unified Modeling Language* (UML).

2.5.1 Metode *Waterfall*

Metode *Waterfall* adalah suatu proses pengembangan perangkat lunak berurutan, proses yang berjalan terus mengalir ke bawah (seperti air terjun) melewati fase-fase perencanaan, pemodelan, implementasi (konstruksi), dan pengujian (Pressman, Roger S. 2001).

Tahapan yang dilakukan dalam pengembangan aplikasi ini sesuai dengan Metode *Waterfall* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Metode Waterfall (Pressman, Roger S. 2001)

Tahapan pengembangan sistem dengan Metode *Waterfall* dijelaskan sebagai berikut:

1. **Analisis Kebutuhan Aplikasi (*Requirements Definition*)**

   Analisa kebutuhan sistem atau aplikasi merupakan tahapan pertama yang

menjadi dasar proses pembuatan sistem. Pada tahap ini dilakukan pencarian dan pengumpulan semua kebutuhan yang diperlukan untuk menunjang kelengkapan sistem atau aplikasi, kemudian mendefinisikan semua kebutuhan yang dipenuhi dalam perangkat lunak atau aplikasi yang dibuat.

2. **Desain Aplikasi (*Sistem And Software Design*)**

   Desain aplikasi merupakan tahap perancangan sistem atau aplikasi yang meliputi penyusunan proses, data, aliran proses, dan pemenuhan kebutuhan sesuai dengan hasil analisa kebutuhan. Dokumentasi desain aplikasi yang dihasilkan dari tahapan ini adalah *Use Case Diagram* dan *Activity Diagram.*

3. **Penerapan Desain dan Penulisan Kode Program**

   Penulisan kode program merupakan tahap penerjemahan desain sistem yang telah dibuat ke dalam bentuk perintah-perintah yang dimengerti komputer dengan menggunakan bahasa pemrograman. Penelitian ini menggunakan bahasa pemrograman Java dan *Eclipse* sebagai *software* pengembangan aplikasinya. Pada tahap ini, penulis menerjemahkan *design* kedalam bahasa pemrograman sehingga didapatkan suatu aplikasi yang diinginkan sesuai yang sudah dirancang. sehingga didapatkan suatu *file installer* dengan ekstensi *apk.*

4. **Pengujian Aplikasi (*Integration and Sistem Testing*)**

   Pengujian aplikasi dilakukan untuk memastikan bahwa sistem yang dibuat telah sesuai dengan desain dan semua fungsi dapat dipergunakan dengan baik tanpa ada kesalahan sesuai dengan kebutuhan pengguna. Pengujian aplikasi ini menggunakan metode *Blackbox Testing*. Pengujian dilakukan secara menyeluruh tanpa melihat struktur internal aplikasi atau komponen yang diuji. *Blackbox Testing* berfokus pada kebutuhan fungsional aplikasi yang

berdasarkan pada spesifikasi kebutuhan aplikasi tersebut.

5. **Penerapan Aplikasi dan Perawatan *(Operational and Maintenance)***

   Pada tahapan ini, aplikasi sudah siap untuk diterapkan pada perangkat *mobile* dan siap digunakan sesuai dengan tujuan dibuatnya aplikasi ini. Perawatan, perbaikan dan pengembangan aplikasi dilakukan untuk menjaga kualitas dan kestabilan aplikasi.

2.5.2 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OO). Definisi ini merupakan definisi yang sederhana.

Unified Modeling Language (UML) merupakan strandar yang relatif terbuka yang dikontrol oleh *Object Management Group* (OMG), sebuah konsorsium terbuka yang terdiri dari banyak perusahaan. OMG dibentuk untuk membuat standar – standar yang mendukung interoperabilitas, khusunya interoperabilitas sistem berorientasi objek. OMG mungkin lebih dikenal dengan standar – standar COBRA (*Common Object Request Broker Architecture*).

UML lahir dari penggabungan banyak bahasa permodelan grafis berorientasi objek yang berkembang pesat pada akhir 1980-an dan awal 1990-an. UML dibuat oleh Grady Booch, James Rumbaugh, dan Ivar Jacobson di bawah bendera *Rational Software Corp*. UML menyediakan notasi-notasi yang membantu memodelkan sistem dari berbagai perspektif (Fowler, 2004).

UML dideskripsikan oleh beberapa diagram, yaitu sebagai berikut.

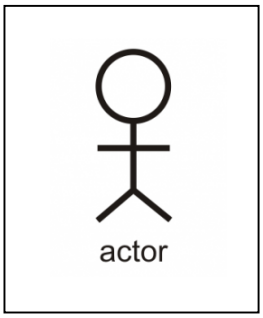
1. *Use case* Diagram

Use case Diagram digunakan untuk menggambarkan sistem dari sudut pandang pengguna sistem tersebut (*user*), sehingga pembuatan *use case* diagram lebih dititikberatkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian. Sebuah *use case* diagram merepresentasikan sebuah interaksi antara aktor dengan sistem yang akan dikembangkan (Fowler, 2004).

Komponen-komponen dalam *use case* diagram (Fowler, 2004) :

a. Aktor

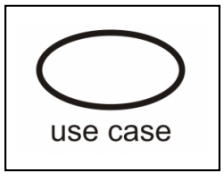
Pada dasarnya aktor bukanlah bagian dari *use case* diagram, namun untuk dapat terciptanya suatu *use case* diagram diperlukan aktor, dimana aktor tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat atau sistem lain) yang berinteraksi dengan sistem yang dibuat. Sebuah aktor mungkin hanya memberikan informasi inputan pada sistem, hanya menerima informasi dari sistem atau keduanya menerima dan memberi informasi pada sistem. Aktor hanya berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. Aktor digambarkan dengan *stick pan* seperti yang terdapat pada Gambar 2.3.



Gambar 2.3 Contoh Aktor (Fowler, 2004).

b. Use Case

Gambaran fungsionalitas dari suatu sistem, sehingga pengguna sistem paham dan mengerti kegunaan sistem yang akan dibangun. Bentuk *use case* dapat terlihat pada Gambar 2.4.



Gambar 2.4 *Use Case* (Fowler, 2004).

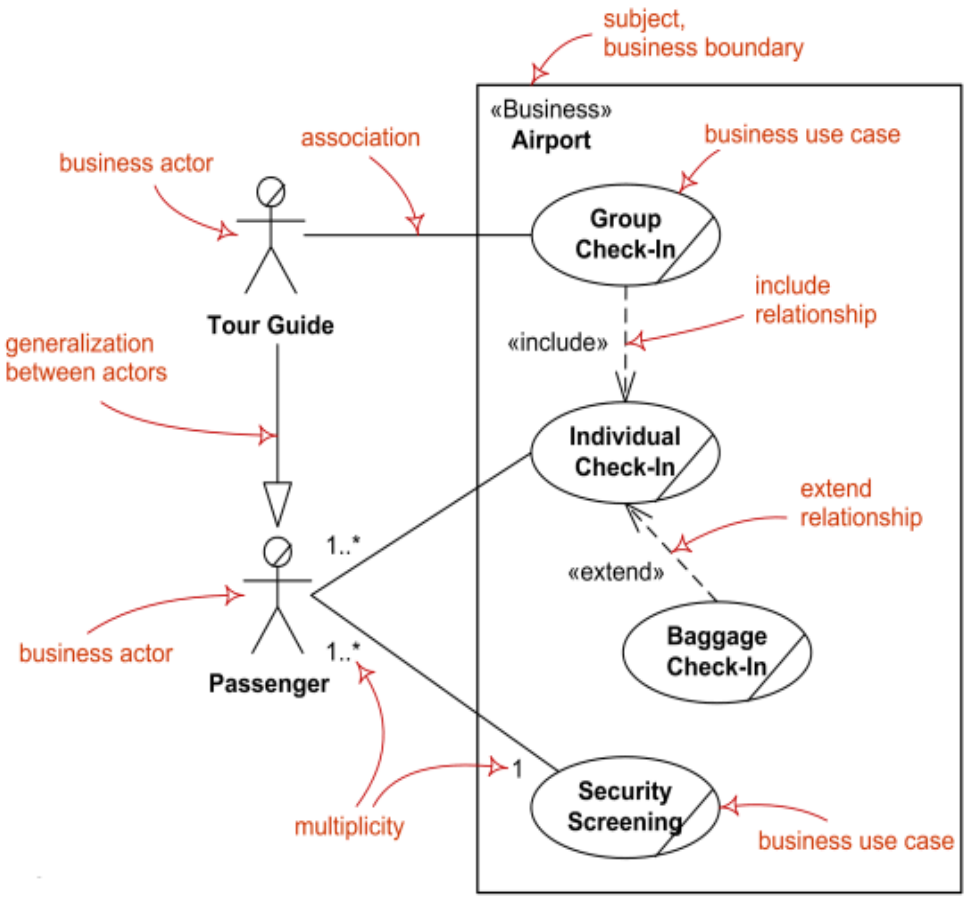
Ada beberapa relasi yang terdapat pada *use case* diagram:

1. *Association*, menghubungkan link antar element.
2. *Generalization*, disebut juga pewarisan (*inheritance*), sebuah elemen dapat merupakan spesialisasi dari elemen lainnya.
3. *Dependency*, sebuah element bergantung dalam beberapa cara ke element lainnya.
4. *Aggregation*, bentuk *association* dimana sebuah elemen berisi elemen lainnya.

Tipe relasi yang mungkin terjadi pada *use case* diagram:

1. <<*include*>>, yaitu kelakuan yang harus terpenuhi agar sebuah *event* dapat terjadi, dimana pada kondisi ini sebuah *use case* adalah bagian dari *use case* lainnya.
2. <<*extends*>>, kelakuan yang hanya berjalan di bawah kondisi tertentu seperti menggerakkan peringatan.
3. <<*communicates*>>, merupakan pilihan selama asosiasi hanya tipe relationship yang dibolehkan antara aktor dan *use case*.

Bentuk dari *use case* diagram dapat terlihat pada Gambar 2.5.



Gambar 2.5 Contoh *Use Case* Diagram (uml-diagrams.org, 2014).

2. *Activity* Diagram

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat digunakan untuk aktifitas lainnya (Fowler, 2004).

Berikut ini adalah tabel Notasi *Activity* Diagram yang diilustrasikan pada Tabel 2.1.

Tabel 2.1 Notasi *Activity Diagram* (Meildy, 2014).

Simbol	Keterangan
	Titik Awal

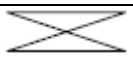
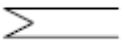
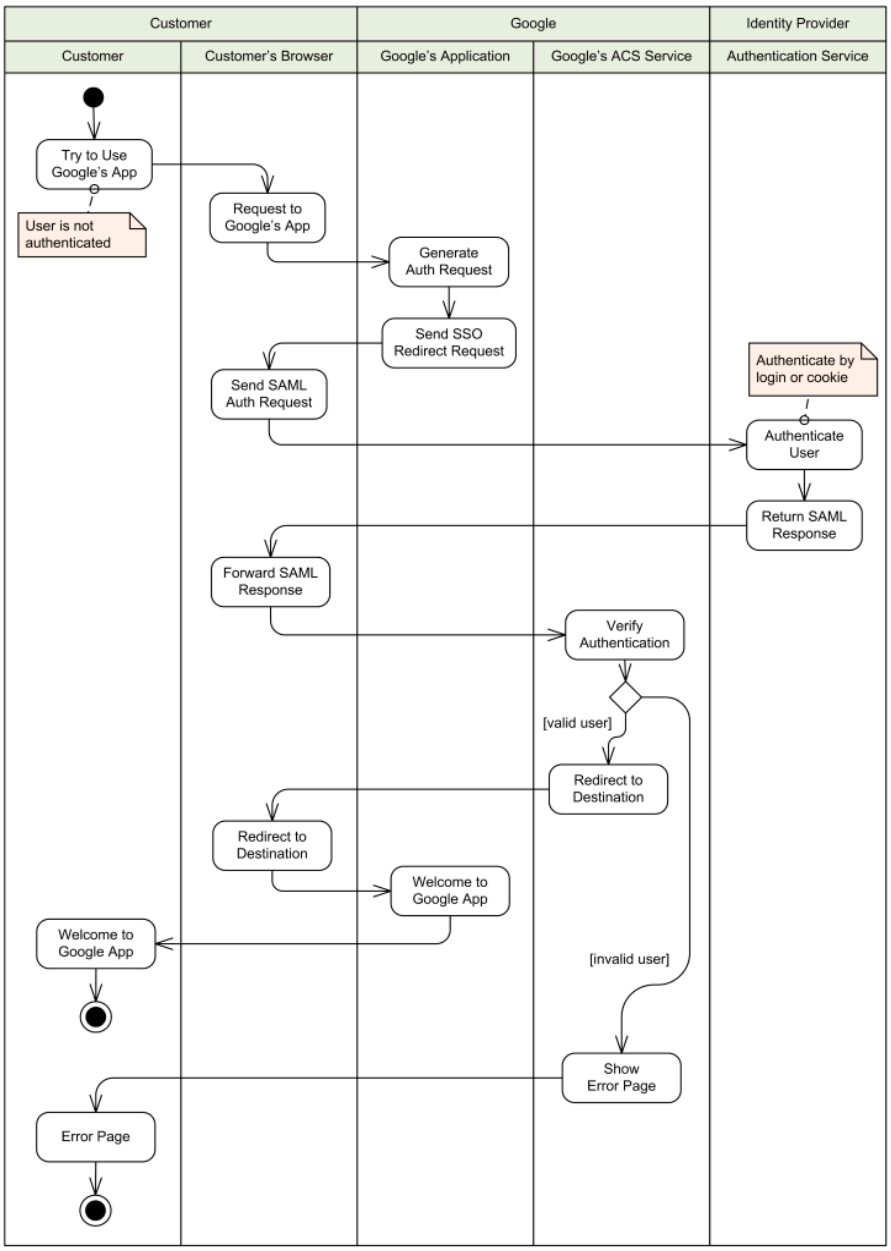
Simbol	Keterangan
	Titik Akhir
	*Activity*
	Pilihan untuk mengambil keputusan
	*Fork*; Digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	*Rake*; Menunjukkan adanya dekomposisi
	Tanda Waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran akhir (*Flow Final*)

Diagram ini sangat mirip dengan *flowchart* karena memodelkan *workflow* dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status. Pembuatan *activity diagram* pada awal pemodelan proses dapat membantu memahami keseluruhan proses. *Activity* diagram juga digunakan untuk menggambarkan interaksi antara beberapa *use case* (Fowler, 2004).

Bentuk dari *activity* diagram dapat terlihat pada Gambar 2.6.



Gambar 2.6 Contoh *Activity* Diagram (uml-diagrams.org, 2014).

3. *Class* Diagram

Class adalah sebuah spesifikasi yang akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode/fungsi). *Class* Diagram

menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain (Fowler, 2004).

Class memiliki tiga area pokok :

1. Nama (*Class Name*)
2. Atribut
3. Metode (*Operations*)

Pada UML, *class* digambarkan dengan segi empat yang dibagi beberapa bagian. Bagian atas merupakan nama dari *class*. Bagian yang tengah merupakan struktur dari *class* (atribut) dan bagian bawah merupakan sifat dari *class* (metode/operasi).

Atribut dan metode dapat memiliki salah satu sifat berikut :

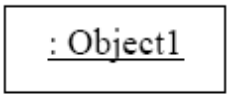
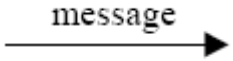
1. *Private* , tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan *class* lain yang mewarisinya.
3. *Public*, dapat dipanggil oleh *class* lain (Fowler, 2004).

4. *Sequence* Diagram

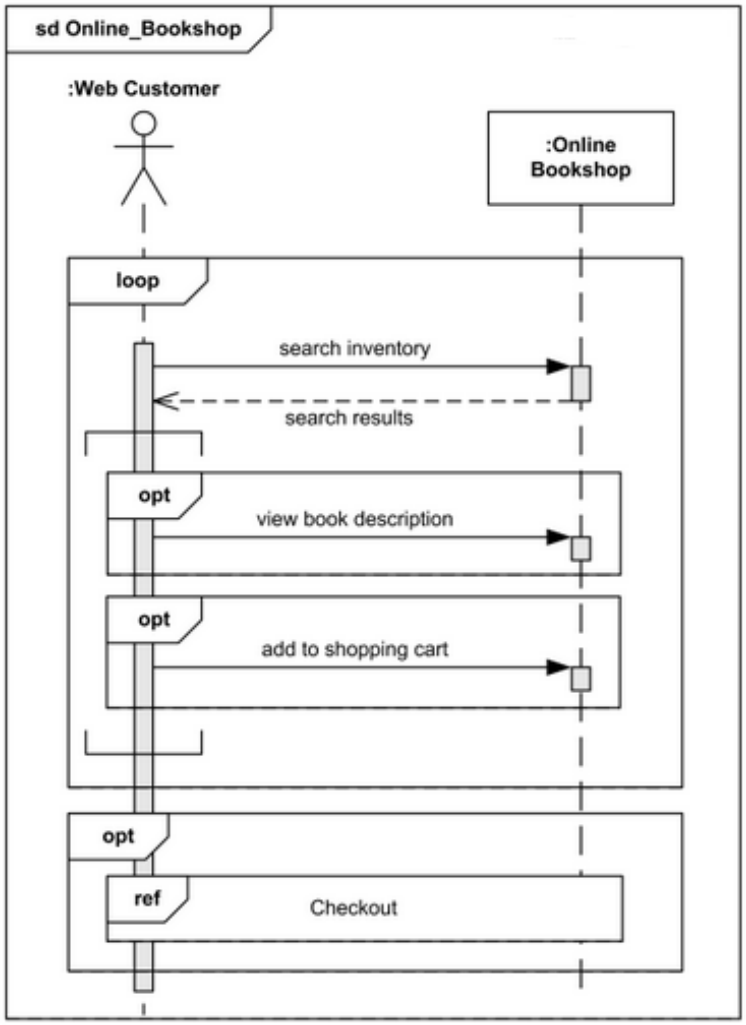
Sequence diagram menggambarkan interaksi antara sejumlah objek dalam urutan waktu. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antar objek yang terjadi pada titik tertentu dalam eksekusi sistem (Fowler, 2014).

Berikut ini adalah Notasi *Sequence* Diagram yang disajikan pada Tabel 2.2.

Tabel 2.2 Notasi *Sequence Diagram* (Meildy, 2014).

Simbol	Nama	Keterangan
	***Object***	*Object* merupakan *instance* dari sebuah *class* dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah *class* (kotak) dengan nama obyek didalamnya yang diawali dengan sebuah titik koma
	***Actor***	*Actor* juga dapat berkomunikasi dengan *object*, maka *actor* juga dapat diurutkan sebagai kolom. Simbol *Actor* sama dengan simbol pada *Actor Use Case* Diagram.
	Lifeline	*Lifeline* mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk *Lifeline* adalah garis putus-putus vertikal yang ditarik dari sebuah obyek.
	***Activation***	*Activation* dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah *lifeline*. *Activation* mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.
	***Message***	*Message*, digambarkan dengan anak panah *horizontal* antara *Activation*. *Message* mengindikasikan komunikasi antara *object-object*.

Contoh dari *squence diagram* dapat dilihat pada Gambar 2.7.



Gambar 2.7 Contoh *Sequence* Diagram (uml-diagrams.org, 2014)

2.5.3 Keunggulan UML

Secara umum UML diterapkan dalam pengembangan sistem/perangkat lunak berorientasi obyek sebab metodologi UML ini umumnya memiliki keunggulan-keunggulan sebagai berikut (Nugroho, 2005) :

a. ***Uniformity***. Dengan metodologi UML, para pengembang cukup menggunakan 1 metodologi dari tahap analisis hingga perancangan. Hal ini tidak bisa dilakukan dalam metodologi pengembangan terstruktur. Dengan

perkembangan masa kini ke arah aplikasi GUI (*Graphical User Interface*), UML juga memungkinkan kita merancang komponen antarmuka pengguna (*user interface*) secara integrasi bersama dengan perancangan perangkat lunak sekaligus dengan perancangan basis data.

b. ***Understandability***. Dengan metodologi ini kode yang dihasilkan dapat diorganisasi ke dalam kelas-kelas yang berhubungan dengan masalah sesungguhnya sehingga lebih mudah dipahami siapapun juga.

c. ***Stability***. Kode program yang dihasilkan relatif stabil ssepanjang waktu sebab sangat mendekati permasalahan sesungguhnya di lapangan.

d. ***Reusability***. Dengan metodologi berorientasi obyek, dimungkinkan penggunaan ulang kode, sehingga pada gilirannya akan sangat mempercepat waktu pengembangan perangkat lunak.

2.6 Teknik Pengujian Perangkat Lunak

Ada dua macam pendekatan kasus uji yaitu *white-box* dan *black-box*. Pendekatan *white-box* adalah pengujian untuk memperlihatkan cara kerja dari produk secara rinci sesuai dengan spesifikasinya (Jiang, 2012). Jalur logika perangkat lunak akan dites dengan menyediakan kasus uji yang akan mengerjakan kumpulan kondisi dan pengulangan secara spesifik. Sehingga melalui penggunaan metode ini akan dapat memperoleh kasus uji yang menjamin bahwa semua jalur independen pada suatu model telah diigunakan minimal satu kali, penggunaan keputusan logis pada sisi benar dan salah, pengeksekusian semua *loop* dalam batasan dan batas operasional perekayasa, serta penggunaan struktur data internal guna menjamin validitasnya (Pressman, 2010).

Pendekatan *black-box* merupakan pendekatan pengujian untuk mengetahui apakah semua fungsi perangkat lunak telah berjalan semestinya sesuai dengan kebutuhan fungsional yang telah didefinisikan (Jiang, 2012). Kasus uji ini bertujuan untuk menunjukkan fungsi perangkat lunak tentang cara beroperasinya. Teknik pengujian ini berfokus pada domain informasi dari perangkat lunak, yaitu melakukan kasus uji dengan mempartisi domain *input* dan *output* program. Metode *black-box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi *input* yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian ini berusaha menemukan kesalahan dalam kategori fungsi-fungsi yang tidak benar atau hilang, kesalahan *interface*, kesalahan dalam struktur data atau akses basis data eksternal, kesalahan kinerja, dan inisialisasi dan kesalahan terminal (Pressman, 2010).

2.6.1 *Equivalence Partitioning*

Equivalence Partitioning (EP) 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* (Pressman, 2001).

2.6.2 Skala *Likert*

Menurut Likert dalam buku Azwar S (2011, p. 139), sikap dapat diukur dengan metode *rating* yang dijumlahkan (*Method of Summated Ratings*). Metode ini merupakan metode penskalaan pernyataan sikap yang menggunakan distribusi *respons* sebagai dasar penentuan nilai skalanya. Nilai skala setiap pernyataan tidak ditentukan oleh derajat *favourable* nya masing-masing akan tetapi ditentukan oleh distribusi *respons* setuju dan tidak setuju dari sekelompok responden yang bertindak sebagai kelompok uji coba (*pilot study*) (Azwar, 2011).

Skala Likert, yaitu skala yang berisi lima tingkat preferensi jawaban dengan pilihan sebagai berikut: 1 = sangat tidak setuju; 2 = tidak setuju; 3 = ragu–ragu atau netral; 4 = setuju; 5 = sangat setuju. Selanjutnya, penentuan kategori interval tinggi, sedang, atau rendah digunakan rumus sebagai berikut :

$$I = \frac{NT - NR}{K}$$

Keterangan :

I = Interval;

NT = Total nilai tertinggi;

NR = Total nilai terendah;

K = Kategori jawaban (Yitnosumarto, 2006).

Kriteria penilaian = % Total skor tertinggi – Interval (I)