BAB II

LANDASAN TEORI

2.1 Android

Menurut Safaat (2011), Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi.

Android dipuji sebagai "*platform mobile* pertama yang Lengkap, Terbuka, dan Bebas".

1. Terbuka (*Open Source Platform*)

   Android dibangun untuk benar-benar terbuka sehingga sebuah aplikasi dapat memanggil salah satu fungsi inti ponsel seperti membuat panggilan, mengirim pesan teks, menggunakan kamera, dan lain-lain. Android menggunakan sebuah mesin virtual yang dirancang khusus untuk mengoptimalkan sumber daya memori dan perangkat keras yang terdapat di dalam perangkat. Android merupakan open source, dapat secara bebas diperluas untuk memasukkan teknologi baru yang lebih maju pada saat teknologi tersebut muncul. Platform ini akan terus berkembang untuk membangun aplikasi mobile yang inovatif.

2. Lengkap (*Complete Platform*)

   Para Desainer dapat melakukan pendekatan yang komprehensif ketika mereka sedang mengembangkan *platform* Android. Android merupakan

sistem operasi yang aman dan memungkinkan untuk peluang mengembangkan aplikasi.

3. *Free* (*Free Platform*)

Android adalah *platform* / aplikasi yang bebas untuk develop. Tidak ada lisensi atau biaya royalti untuk dikembangkan pada *platform* Android. Tidak ada biaya keanggotaan diperlukan . Tidak diperlukan biaya pengujian. Tidak ada kontrak yang diperlukan. Aplikasi Android dapat didistribusikan dan diperdagangkan dalam bentuk apa pun.

Android merupakan generasi baru *platform mobile, platform* yang memberikan pengembang untuk melakukan pengembangan sesuai dengan yang diharapkan. Pengembang aplikasi Android diperbolehkan untuk mendistribusikan aplikasi mereka di bawah skema lisensi apa pun yang mereka inginkan.

Pengembang memiliki beberapa pilihan ketika membuat aplikasi yang berbasis Android. Sebagian besar pengembang menggunakan *Eclipse* yang tersedia secara bebas untuk merancang dan mengembangkan aplikasi Android. *Eclipse* adalah IDE yang paling populer untuk pengembangan-pengembangan Android. Selain itu, *Eclipse* juga mendapat dukungan langsung dari Google untuk menjadi IDE pengembangan aplikasi Android, ini terbukti dengan adanya penambahan plugins untuk *Eclipse* untuk membuat *project* android dimana *source software* langsung dari situs resminya Google. Akan tetapi, hal diatas tidak menutup kemungkinan untuk menggunakan IDE yang lain seperti Netbeans untuk melakukan pengembangan android.

Aplikasi Android dapat dikembangkan pada sistem operasi berikut :

1. Windows XP, Vista/Seven
2. Mac OS X (Mac OS X 10.4.8 atau lebih baru).
3. Linux

2.2 Kamus Elektronik

Dalam penelitian disini, kamus elektronik adalah kamus yang berupa piranti lunak dan bisa diinstal ke komputer (Cook,2001). Hal ini dikarenakan karena kamus tersebut menggunakan alat tertentu yang harganya cukup mahal dan tidak bisa diinstal di komputer,sehingga penggunanya sangat terbatas. Terdapat dua versi kamus elektronik *bilingual* Inggris-Indonesia yamg cukup luas tersedia, yaitu *Linguist version 1.0* dan *Indict Version 2.0*.

2.3 Bahasa Korea

Bahasa Korea (한국어/조선말) adalah bahasa yang paling luas digunakan di Korea, dan merupakan bahasa resmi Korea Selatan dan Korea Utara. Bahasa ini juga dituturkan secara luas di Yanbian di Cina timur laut. Secara keseluruhan terdapat sekitar 78 juta penutur bahasa Korea di seluruh dunia termasuk kelompok-kelompok besar di Uni Soviet, AS, Kanada dan Jepang. Klasifikasi resmi bahasa Korea masih belum disetujui secara universal, namun dianggap oleh banyak orang sebagai bahasa isolat. Beberapa ahli bahasa memasukkannya ke

dalam kelompok bahasa Altaik. Bahasa Korea juga banyak mirip dengan bahasa Jepang yang status kekerabatannya juga kurang jelas.

2.3.1 Macam – macam huruf Hangeul

Alfabet Hangeul terdiri dari 14 konsonan tunggal dan 10 huruf hidup (vokal), 5 konsonan rangkap, 11 vokal rangkap, 7 bunyi konsonan akhir. Informasi ini ditemukan di http://world.kbs.co.kr/, (19 Juni 2013).

2.3.1.1 14 Konsonan Tunggal

Tabel 2.1 14 Konsonan Tunggal

No.	Huruf	Nama	*Bunyi* Awal	*Bunyi* Tengah	*Bunyi* Akhir
1.	ㄱ	Kiyeok	K/G	G	K
2.	ㄴ	Nieun	N	N	N
3.	ㄷ	Tigeut	T/D	D	T
4.	ㄹ	Rieul	R/L	R/L	L
5.	ㅁ	Mieum	M	M	M
6.	ㅂ	Pieup	P/B	B	P
7.	ㅅ	Siot	S	S	T
8.	ㅇ	Ieung	-	-	Ng
9.	ㅈ	Chieut	Ch	J	T
10.	ㅊ	Ch'ieut	Ch'	Ch	T
11.	ㅋ	Khieuk	Kh	Kh	K
12.	ㅌ	Thieut	Th	Th	T
13.	ㅍ	Phieup	Ph	Ph	P
14.	ㅎ	Hieut	H	H	T

2.3.1.2 Vokal Tunggal

Untuk menulis vokal tunggal yang berbunyi sendiri harus digabung dengan konsonan “ o ” (lung).

Tabel 2.2 Vokal Tunggal

No.	Huruf	Bunyi
1.	아	a
2.	야	ya
3.	어	eo
4.	여	yeo
5.	오	o
6.	요	yo
7.	우	u
8.	유	yu
9.	으	eu
10.	이	i

2.3.1.3 Konsonan Rangkap

Konsonan rangkap terjadi dari dua penggabungan konsonan tunggal

Tabel 2.3 Konsonan Rangkap

No.	Huruf	Nama	*Bunyi* Awal	*Bunyi* Tengah	*Bunyi* Akhir
1.	ㄲ	Sang Kiyeok	kk	kk	k
2.	ㄸ	Sang Tigeut	tt	tt	-
3.	ㅃ	Sang Pieup	pp	pp	-
4.	ㅆ	Sang Siot	ss	ss	t
5.	ㅉ	Sang Ch'ieut	cc	cc	-

2.3.1.4 Vokal Rangkap

Vokal rangkap terjadi dari dua penggabungan vokal tunggal, dan untuk menulis vokal rangkap yang berbunyi sendiri, harus digabungkan dengan konsonan “ o ” (lung).

Tabel 2.4 Vokal Rangkap

No.	Huruf	Bunyi	Proses
1.	애	Ae	아 + 이
2.	얘	Yae	야 + 이
3.	에	E	어 + 이
4.	예	Ye	여 + 이
5.	외	Oe	오 + 이
6.	와	Wa	오 + 아
7.	왜	Wae	오 + 애
8.	워	Wo	우 + 어
9.	웨	We	우 + 에
10.	위	Wi	우 + 이
11.	의	Eui	으 + 이

2.3.1.5 Bunyi Konsonan Akhir

Bunyi konsonan di dalam bahasa Korea ditempatkan di bawah konsonan hidup. Kecuali bunyi konsonan akhir ‘S’, ‘R’, ‘D’

Tabel 2.5 Bunyi Konsonan Akhir

No.	Huruf	Bunyi
1.	ㄱ, ㅋ, ㄲ	k
2.	ㄴ	n
3.	ㄷ, ㅅ, ㅈ, ㅊ, ㅌ, ㅎ, ㅆ	t
4.	ㄹ	l
5.	ㅁ	m
6.	ㅂ, ㅍ	p

No	Huruf	Bunyi
7.	ㅇ	ng

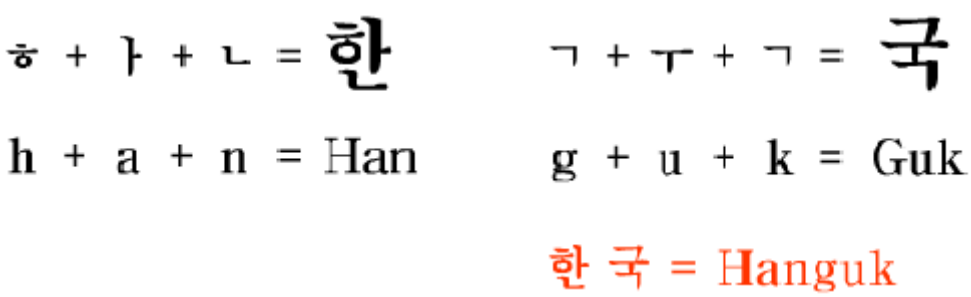
2.3.1.6 Konsonan Akhir Komplek

Konsonan akhir kompleks ini terdiri dari dua huruf konsonan yang ditempatkan dibawah konsonan hidup. Akan tetapi hanya satu huruf konsonan saja yang dibaca/diucapkan.

Gambar 2.1 Huruf Konsonan Akhir Komplek

2.3.2 Penulisan Korea Hangeul

Cara penulisan Bahasa Korea Hangul dilakukan setelah menggabungkan huruf vokal dan konsonan untuk membentuk satu suku kata seperti bahasa asing lainnya, dan dituliskan dari kiri ke kanan atau dari atas ke bawah.



Gambar 2.2 Contoh Penulisan Huruf Hangul

2.4 Metode Pencarian Biner

Binary Search adalah teknik yang diterapkan hanya pada elemen yang telah terurut (*sorted)*. Pencarian beruntun memiliki satu kekurangan, yaitu dalam kasus terburuk (elemen yang dicari berada pada posisi terakhir) makan pencarian harus dilakukan sepanjang larik. Semakin banyak elemen makan semakin lama pencarian harus dilakukan (Suarga,2012).

Pencarian pada data yang terurut menunjukan kinerja yang lebih baik daripada pencarian pada data yang belum terurut. Hal ini, sudah kita bicarakan pada metode pencarian beruntun untuk data yang sudah terurut. Data yang terurut banyak ditemukan didalam kehidupan kita sehari-hari. Data nomor telepon di dalam buku telepon misalnya, sudah terurut berdasarkan nama/instansi pelanggan telepon dari A sampai Z. Data karyawan dirurut berdasarkan nomor induknya dari nomor kecil ke nomor besar. Data mahasiswa diurut berdasarkan NIM (Nomor Induk Mahasiswa), kata-kata (entry) di dalam kamus bahasa Inggris/Indonesia telah diurut dari A sampai Z, dan sebagainya.

Menurut (Munir,2011) Terdapat algoritma pencarian pada data terurut yang paling mangkus (*efficient*), yaitu algoritma pencarian bagidua atau pencarian biner

(*binary search*). Algoritma ini digunakan untuk kebutuhan pencarian dengan waktu yang cepat. Sebenarnya, dalam kehidupan sehari-hari, sebenarnya kita juga sering menggunakan pencarian biner. Misalnya saat ingin mencari suatu kata dalam kamus Prinsip dari pencarian biner dapat dijelaskan sebagai berikut. Proses yang terjadi pada pencarian dengan metode ini adalah sebagai berikut :

1. Membaca Array data

Tabel 2.6 Contoh Data Belum Terurut

0	1	2	3	4
Kamu	Aku	Saya	Dia	kalian

2. Apabila array belum terurut maka array diurutkan terlebih dahulu

Tabel 2.7 Contoh Data Terurut

0	1	2	3	4
Aku	Dia	Kalian	Kamu	Saya

3. Menentukan data yang akan dicari

   Contoh kata yang dicari adalah Kamu

4. Menentukan elemen tengah dari array atau menentukan indeks tengah.

   Cara menentukan posisi tengah dengan rumus (posisi awal+posisi tengah)/2

Tabel 2.8 Contoh Hasil Pencarian Pertama

0	1	2	3	4
Aku	Dia	Kalian	Kamu	Saya
Awal		Tengah		Akhir

5. Jika nilai elemen tengah sama dengan data yang dicari,maka pencarian berhenti.

   Posisi tengah ditemukan kata kalian sedangkan kata yang dicari adalah kamu. Maka data belum ditemukan

6. Jika elemen tengah tidak sama dengan data yang dicari maka :

   a. Jika nilai elemen tengah > (lebih dari) data yang dicari maka pencarian dilakukan pada setengah array pertama -1.

   b. Jika nilai elemen tengah < (kurang dari) pada data yang dicari maka pencarian pada setengah array berikutnya +1.

   Dikarenakan kalian < (kurang dari) kamu maka dilanjutkan ke pencarian pada setengah array berikutnya. Dengan mencari indeks tengah dan membandingkan dengan kata yang dicari.

Tabel 2.9 Contoh Hasil Pencarian Kedua

0	1	2	3	4
Aku	Dia	Kalian	Kamu	Saya
			Awal=tengah	akhir

Setelah melakukan pencarian kembali, dilakukan pembandingan dengan data tengah. Pada indeks tengah kali ini ditemukan kata “Kamu” sedangkan kata yang dicari adalah “Kamu”, berarti pencarian kata sudah ditemukan pada indeks ke 3.

2.5 SQLite

Menurut Android SDK Docs (2010), Android menyediakan dukungan penuh untuk SQLite database. Setiap database yang anda buat akan dapat diakses dengan nama untuk tiap class dalam aplikasi, tapi bukan di luar aplikasi. Android SDK berisi sqlite3 *database tools* yang memungkinkan dapat menelusuri isi tabel,

menjalankan perintah SQL, dan melakukan fungsi-fungsi berguna lainnya pada database SQLite. Menurut Jay A. Kreibich (2010) adapun fitur dari SQLite yakni :

a. *Serverless*

SQLite memerlukan proses server atau sistem yang terpisah untuk mengoperasikannya. SQLite *library* mengakses databasenya secara langsung.

b. *Zero Configuration*

Tidak ada *server* berarti tidak ada pengaturan. Membuat sebuah *instance* database semudah membuka file.

c. *Cross-Platform*

Database pada SQLite berada dalam file *cross-platform* tunggal yang tidak memerlukan administrasi. Sehingga dapat digunakan di berbagai sistem operasi.

d. *Self-Contained*

Sebuah *library* berisi seluruh sistem database yang terintegrasi langsung ke *application-host*.

e. *Small Runtime Footprint*

Menggunakan sedikit memory untuk *library* databasenya.

f. *Transactional*

SQLite *transaction* memperbolehkan proses penyimpanan melalui beberapa proses *thread*

g. Full-featured

SQLite mendukung penggunaan bahasa SQL standar.

h. Highly Reliable

Dalam sistem android memiliki beberapa teknik untuk melakukan penyimpanan data.

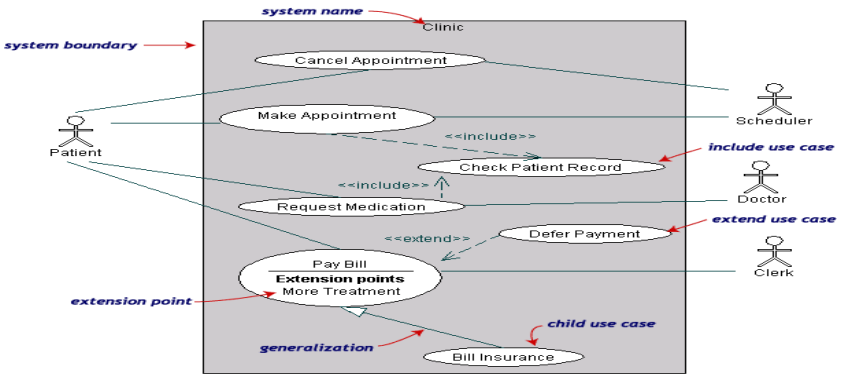
2.6 UML

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah meanjadi standar dalam industri untuk menentukan, visualisasi, merancang dan mendokumentasikan artifact dari sistem software, untuk memodelkan bisnis dan sistem non software lainnya (Fowler,2004).

Untuk membuat suatu model, UML memiliki diagram grafis sebagai berikut (Fowler,2004) :

1. Use case diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu.



Gambar 2.3 Contoh *use case diagram*

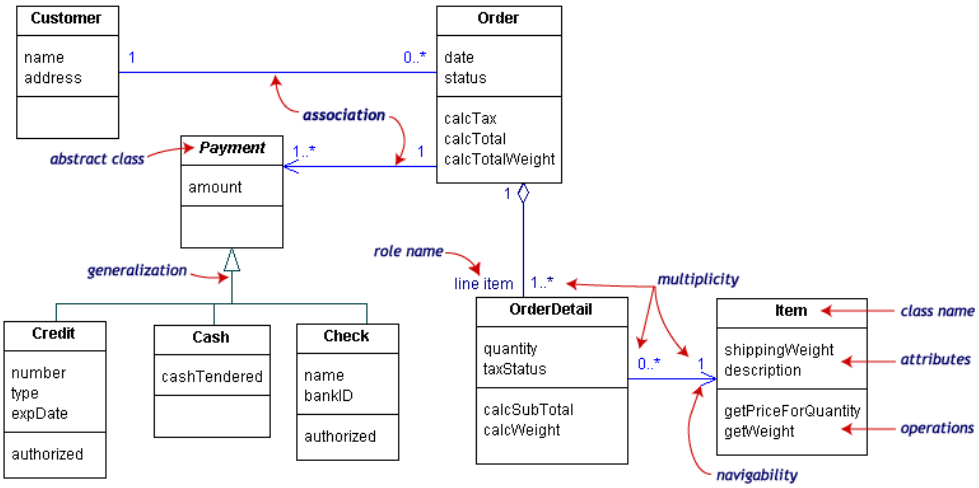
2. Class diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi 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 (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda



Gambar 2.4 Contoh *Class Diagram*

3. *Statechart diagram*

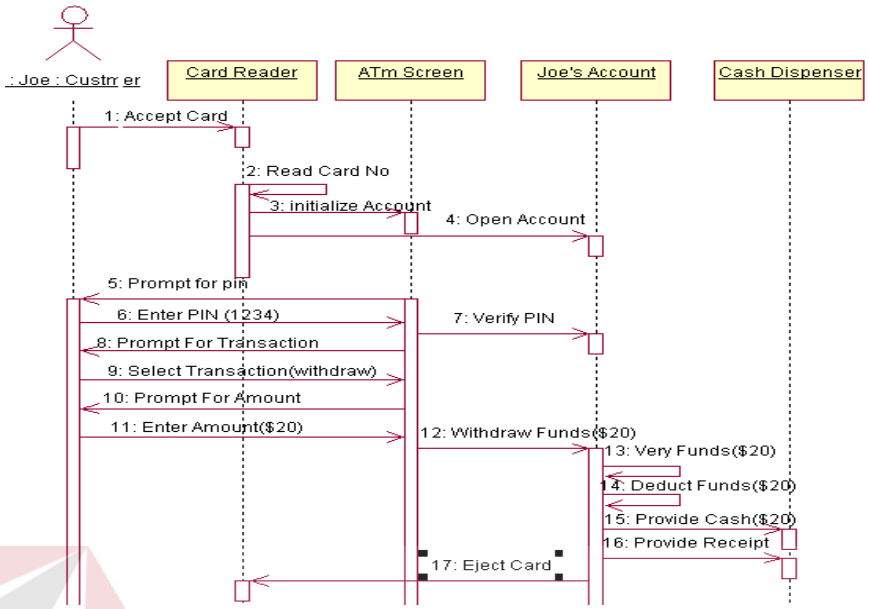
Menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

4. Activity diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir.

5. Sequence diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).



Gambar 2.5 Contoh *Sequence Diagram*

6. *Collaboration diagram*

Menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

7. *Component diagram*

Menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya.

8. Deployment diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada

mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal

Diagram-diagram tersebut diberi nama berdasarkan sudut pandang yang berbeda-beda terhadap sistem dalam proses analis atau rekayasa.

Dibuatnya berbagai jenis diagram diatas karena (Fowler,2004) :

1. Setiap sistem yang kompleks selalu paling baik yang satu sama lain hampir saling bebas (*independent)*. Sudut pandang tunggal senantiasa tidak mencukupi untuk melihat sistem yang besar dan kompleks.
2. Diagram yang berbeda-beda tersebut dapat menyatakan tingkatan yang berbeda-beda dalam proses rekayasa.
3. Diagram-diagram tersebut dibuat agar model yang dibuat semakin mendekati realitas.

2.7 Testing

Menurut Bill Hetzel 1983 (Revisi): Testing adalah tiap aktivitas yang digunakan untuk dapat melakukan evaluasi suatu atribut atau kemampuan dari program atau sistem dan menentukan apakah telah memenuhi kebutuhan atau hasil yang diharapkan

Test case merupakan suatu tes yang dilakukan berdasarkan pada suatu inisialisasi, masukan, kondisi ataupun hasil yang telah ditentukan sebelumnya.

Adapun kegunaan dari *test case* ini, adalah sebagai berikut:

1. Untuk melakukan testing kesesuaian suatu komponen terhadap spesifikasi – *Black Box Testing*.
2. Untuk melakukan testing kesesuaian suatu komponen terhadap disain – *White Box Testing.*

2.7.1 White Box Testing

Kadang disebut juga *glass box testing* atau *clear box testing*, adalah suatu metode disain *test case* yang menggunakan struktur kendali dari disain prosedural.

Metode disain *test case* ini dapat menjamin:

1. Semua jalur (*path*) yang independen / terpisah dapat dites setidaknya sekali tes.
2. Semua logika keputusan dapat dites dengan jalur yang salah dan atau jalur yang benar.
3. Semua *loop* dapat dites terhadap batasannya dan ikatan operasionalnya.
4. Semua struktur internal data dapat dites untuk memastikan validitasnya.

2.7.2 Black Box testing

Black box testing, dilakukan tanpa pengetahuan detil struktur internal dari sistem atau komponen yang dites. juga disebut sebagai *behavioral testing*, *specification-based testing, input/output* testing atau functional testing.

Black box testing berfokus pada kebutuhan fungsional pada *software*, berdasarkan pada spesifikasi kebutuhan dari *software*. Dengan adanya *black box testing*, perekayasa *software* dapat menggunakan sekumpulan kondisi masukan

yang dapat secara penuh memeriksa keseluruhan kebutuhan fungsional pada suatu program.

Black box testing bukan teknik alternatif daripada *white box testing*. Lebih daripada itu, ia merupakan pendekatan pelengkap dalam mencakup *error* dengan kelas yang berbeda dari metode *white box testing*.

Kategori error yang akan diketahui melalui black box testing:

1. Fungsi yang hilang atau tak benar
2. Error dari antar-muka
3. Error dari struktur data atau akses eksternal database
4. Error dari kinerja atau tingkah laku
5. Error dari inisialisasi dan terminasi