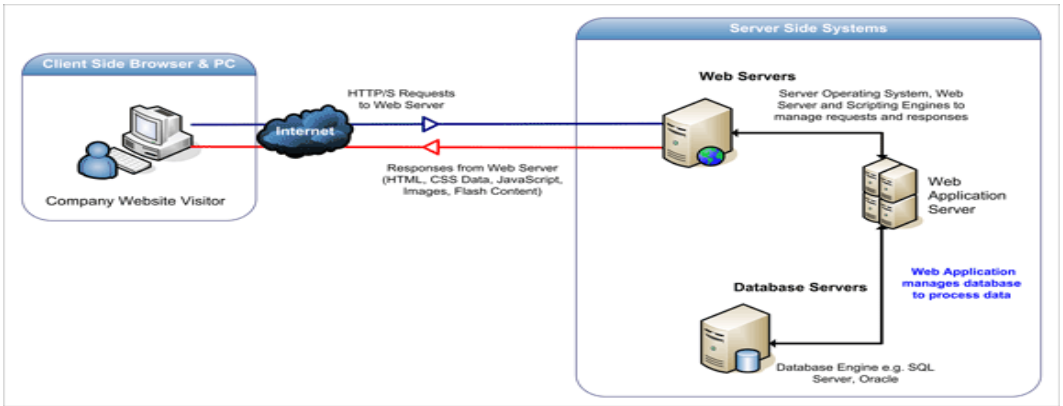
BAB II
TINJAUAN PUSTAKA

2.1 *Web Applications* (*Web Apps*)

Web Applications (*Web Apps*) adalah sebuah aplikasi yang diakses oleh pengguna melalui jaringan seperti internet atau intranet (Apers, 2010). *Web Apps* juga dapat diartikan sebagai sebuah *software* yang dikodekan dengan pemograman yang dapat terintegrasi dengan browser, biasanya pemograman yang digunakan adalah JavaScript yang dikombinasikan dengan HTML.

Sebuah *Web Apps* pada dasarnya memiliki 3 *layer*. Pada *layer* pertama berada di sisi *client* yang memiliki sistem *browser* dasar. Pada *layer* kedua terdapat *dynamic content generation tool* seperti JavaScript dan php. Pada *layer* ketiga terdapat penyimpanan data dan terdiri dari *back end* dari *database* seperti Mysql atau oracle. Untuk ilustrasi cara kerja *web apps* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Cara Kerja *Web Apps*
Sumber: Anonim, 2013.http://www.acunetix.com/websitesecurity/web-applications.htm

2.2 JQuery Mobile

JQuery Mobile adalah *touch-optimized web framework* (sebuah *library* dari JavaScript atau *mobile framework*). JQuery Mobile ini berfokus pada menciptakan sebuah *framework* yang kompatibel dengan berbagai macam *smartphone* dan komputer *tablet*, diciptakan untuk pembuatan aplikasi yang dapat berjalan di *smartphone* ataupun tablet seperti *web apps*. (JQuery Documentation, 2013)

Beberapa kelebihan Jquery Mobile ini antara lain:

1. Kompatibel dengan semua *mobile platform* termasuk iOS, Android, Blackberry, WebOS, Symbian, Windows Phone 7 dan lainnya.
2. Dibuat berdasarkan jQuery
3. *Theming framework* yang memungkinkan membuat tema *custom*.
4. Ukuran yang ringan dan mengutamakan kecepatan.
5. *Coding* yang sama pada setiap layar dan terintegrasi ke setiap layar.
6. Kemampuan navigasi Ajax dengan transisi animasi halaman yang memberikan kemampuan untuk membersihkan URL melalui *pushState*.
7. Berbasiskan HTML5.
8. Pengoptimalan pada *widget-UI* terutama pada *touch-optimized* dan *platform-agnostic*.

2.3 Android

Sistem Operasi Android adalah sebuah sistem operasi yang berbasis linux yang dikembangkan untuk perangkat *mobile* seperti *smartphone* dan *tablet*, oleh *Open Handset Alliance* yang dipimpin *Google*. Android adalah sebuah sistem operasi yang *open-source*, sehingga saat ini banyak sekali jenis dari sistem operasi

yang dikembangkan dan muncul ke publik. Selain pengembang sistem operasi, banyak juga pengembang yang mengembangkan aplikasi atau *apps* untuk sistem operasi ini. Saat ini, versi terbaru dari Android adalah versi 4.1 (*Jelly Bean*), yang saat ini baru muncul pada perangkat *tablet*. Untuk beberapa *smartphone,* versi terbaru dari sistem operasinya adalah versi 4.0 atau *Ice Cream Sandwitch* (Google Code, 2013).

2.4 iOS

iOS (sebelumnya iPhone OS) adalah sistem operasi perangkat bergerak yang dikembangkan dan didistribusikan oleh Apple Inc. Sistem operasi ini pertama diluncurkan tahun 2007 untuk iPhone dan iPod Touch, dan telah dikembangkan untuk mendukung perangkat Apple lainnya seperti iPad dan Apple TV. Tidak seperti Windows Phone (Windows CE) Microsoft dan Android Google, Apple tidak melisensikan iOS untuk diinstal di perangkat keras non-Apple. iOS versi terbaru adalah iOS 6.0.2. (Apple Documentation, 2013)

2.5 *Web Server*

Web server merupakan *server* internet yang mampu melayani koneksi *transfer data* dalam *protocol* HTTP. *Web server* merupakan hal yang terpenting dari *server* di internet dibandingkan *server* lainnya seperti *e-mail server, ftp server* ataupun *news server*. Hal ini di sebabkan *web server* telah dirancang untuk dapat melayani beragam jenis data, dari text sampai grafis 3 dimensi. Kemampuan ini telah menyebabkan berbagai institusi seperti universitas maupun perusahaan dapat menerima kehadirannya dan juga sekaligus menggunakannya sebagai sarana di internet. *Web server* juga dapat digabungkan dengan dunia *mobile wireless*

internet atau yang sering di sebut sebagai WAP (*Wireless Access Protocol*) yang banyak digunakan sebagai sarana *handphone* yang memiliki fitur WAP. Dalam kondisi ini, *web server* tidak lagi melayani data *file* HTML tetapi telah melayani WML (*wireless Markup Language*) (Kuo, 2007).

2.6 Sistem Antrian (*Queue*)

Antrian timbul disebabkan oleh kebutuhan akan layanan melebihi kemampuan (kapasitas) pelayanan atau fasilitas layanan, sehingga pengguna fasilitas yang tiba tidak bisa segera mendapat layanan disebabkan kesibukan layanan. Pada banyak hal, tambahan fasilitas pelayanan dapat diberikan untuk mengurangi antrian atau untuk mencegah timbulnya antrian. Akan tetapi biaya karena memberikan pelayanan tambahan, akan menimbulkan pengurangan keuntungan mungkin sampai di bawah tingkat yang dapat diterima. Sebaliknya, sering timbulnya antrian yang panjang akan mengakibatkan hilangnya pelanggan / nasabah.

Menurut Hiller dan Lieberman (1967), ada tiga komponen dalam sistem antrian yaitu:

1. Kedatangan Populasi yang akan Dilayani (*calling population*)

   Karakteristik dari populasi yang akan dilayani (*calling population*) dapat dilihat menurut ukurannya, pola kedatangan, serta perilaku dari populasi yang akan dilayani. Menurut ukurannya, populasi yang akan dilayani bisa terbatas (*finite*) bisa juga tidak terbatas (*infinite*). Pola kedatangan bisa teratur, bisa juga acak (*random*). Kedatangan yang teratur sering kita jumpai pada proses pembuatan/ pengemasan produk yang sudah distandardisasi.

Pada proses semacam ini, kedatangan produk untuk diproses pada bagian selanjutnya biasanya sudah ditentukan waktunya. Sedangkan pola kedatangan yang sifatnya acak (*random*) banyak kita jumpai misalnya kedatangan nasabah di bank. Pola kedatangan yang sifatnya acak dapat digambarkan dengan distribusi statistik dan dapat ditentukan dua cara yaitu kedatangan per satuan waktu dan distribusi waktu antar kedatangan.

2. Antrian

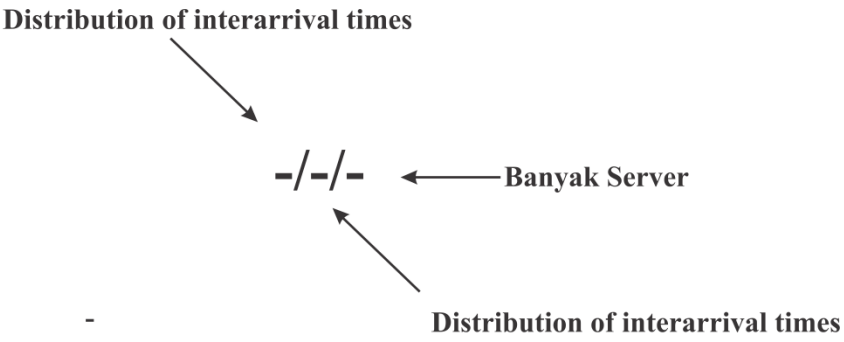
Batasan panjang antrian bisa terbatas (*limited*) bisa juga tidak terbatas (*unlimited*). Sebagai contoh antrian di jalan tol masuk dalam kategori panjang antrian yang tidak terbatas. Sementara antrian di rumah makan, masuk kategori panjang antrian yang terbatas karena keterbatasan tempat. Dalam kasus batasan panjang antrian yang tertentu (*definite line-length*) dapat menyebabkan penundaan kedatangan antrian bila batasan telah tercapai. Contohnya, apabila jumlah pesawat pada landasan telah melebihi suatu kapasitas bandara, maka kedatangan pesawat yang baru dialihkan ke bandara yang lain.

3. Fasilitas Pelayanan

Karakteristik fasilitas pelayanan dapat dilihat dari tiga hal, yaitu tata letak (*lay out*) secara fisik dari sistem antrian, disiplin antrian, waktu pelayanan.

Banyak model antrian yang mengasumsikan bahwa semua waktu kedatangan dan waktu pelayanan adalah berdiri sendiri dan memiliki distribusi yang dapat dikenali.

Semua bentuk model antrian mengikuti aturan:



Dimana

***M* = *exponential/poisson distribution* (Markovian)**
***D* = *degenerate distribution* (constant time)**
E_k = *erlang distribution* (shape parameter = k)
G = *General distribution*

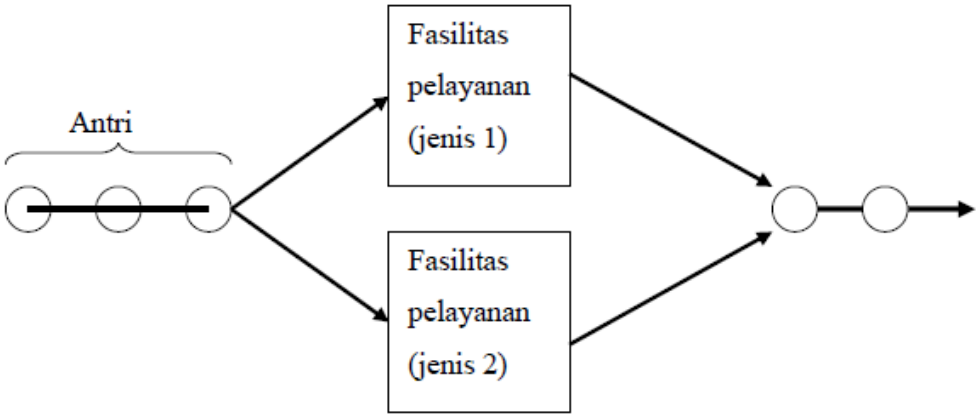
Untuk analisis dan model struktur antrian yang digunakan dalam penelitian ini adalah model *Multi Channel, Single Phase*(M/M/S).

2.7 *Multi Channel, Single Phase* (M/M/S)

Multi Channel Single Phase terjadi kapan saja ketika dua atau lebih fasilitas pelayanan dialiri oleh antrian tunggal. Misalnya pada pembelian tiket yang dilayani oleh lebih dari satu loket, pelayanan nasabah di bank dan lain lain. Contoh sistem antrian dapat dilihat pada Gambar 2.2. Arti dari (M/M/S) ini dalam *Multi Channel Model* adalah terdiri dari :

1. M yang pertama adalah rata-rata kedatangan yang mengikuti distribusi probabilitas Poisson.
2. M yang kedua adalah tingkat pelayanan yang mengikuti distribusi probabilitas eksponensial

3. S adalah jumlah fasilitas pelayanan dalam sistem atau lebih dari satu saluran.



Gambar 2.2 Model Antrian *Multi Channel, Single Server*

Multi Channel Model ini mempunyai ciri-ciri berikut:

1. *Layout* : Ganda (*Multi Channel*)
2. Fasilitas Pelayanan : Tunggal (*Single Phase*)
3. Populasi : Tidak Terbatas
4. Pola Kedatangan : Distribusi Poisson
5. Disiplin antrian : FIFO (*First In Fisrt Out*)
6. Pola Pelayanan : Eksponensial
7. Panjang antrian : Tidak Terbatas

Menurut Heizer(2010), Cara kerja model antrian M/M/s *(Multiple Channel Model)* menggunakan rumus sebagai berikut:

a. Probabilitas terdapat 0 orang dalam sistem

$$P_0 = \frac{1}{\left[\sum_{n=0}^{M-1} \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n\right] + \frac{1}{M!}\left(\frac{\lambda}{\mu}\right)^M \frac{M\mu}{M\mu - \lambda}} \qquad \text{for } M\mu > \lambda$$

b. Jumlah pelanggan rata-rata dalam sistem

$$L_s = \frac{\lambda\mu(\lambda/\mu)^M}{(M-1)!(M\mu-\lambda)^2} P_0 + \frac{\lambda}{\mu}$$

c. Waktu rata-rata yang dihabiskan pelanggan dalam sistem

$$W_s = \frac{\mu(\lambda/\mu)^M}{(M-1)!(M\mu-\lambda)^2} P_0 + \frac{1}{\mu} = \frac{L_s}{\lambda}$$

d. Jumlah orang rata-rata menunggu dalam antrian

$$L_q = L_s - \frac{\lambda}{\mu}$$

e. Waktu rata-rata yang dihabiskan pelanggan menunggu dalam antrian

$$W_q = W_s - \frac{1}{\mu} = \frac{L_q}{\lambda}$$

f. Probabilitas kapasitas orang dalam sistem

$$P_n = \frac{\left[\frac{\lambda}{\mu}\right]^1}{n!} P_0$$

Dengan

μ = tingkat pelayanan rata-rata
λ =tingkat kedatangan rata-rata
P_o = probabilitas tidak ada individu dalam sistem
Pn = Probabilitas kapasitas orang dalam sistem
Ls = jumlah pelanggan rata-rata dalam sistem
Lq = jumlah pelanggan rata-rata dalam antrian
Ws = waktu rata-rata yang dibutuhkan pelanggan dalam sistem
Wq = waktu rata-rata yang dibutuhkan pelanggan dalam antrian

2.8 *Multilevel Feedback Queue*

Multilevel feedback queue adalah salah satu algoritma yang berdasar pada model antrian *Multi Channel Single Server*. Kelebihan mendasar yang dimiliki oleh *multilevel feedback queue* adalah kemungkinan adanya suatu proses berpindah dari satu antrian ke antrian lainnya, misalnya dengan prioritas yang lebih rendah ataupun lebih tinggi (Dusseau, 2012).

Algoritma ini didefinisikan melalui beberapa parameter, antara lain :

a. Jumlah antrian.
b. Algoritma penjadwalan tiap antrian.
c. Kapan menaikkan proses ke antrian yang lebih tinggi.
d. Kapan menurunkan proses ke antrian yang lebih rendah.
e. Antrian mana yang akan dimasuki proses yang membutuhkan.

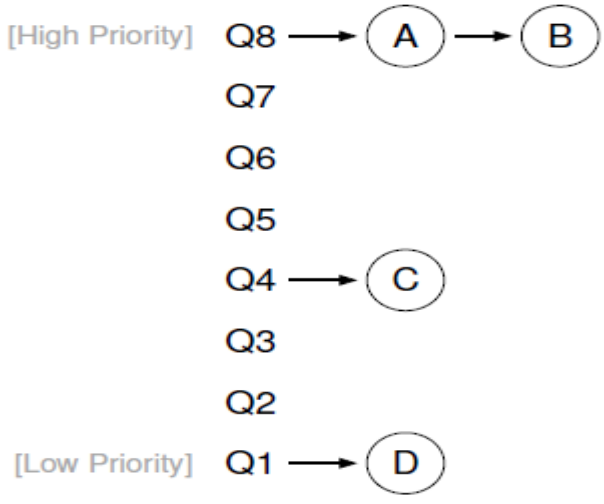
Algoritma ini biasa digunakan dalam pengembangan *Operating System.* Terutama dalam penjadwalan CPU. Jika suatu proses menyita CPU terlalu lama, maka proses itu akan dipindahkan ke antrian yang lebih rendah. Hal ini menguntungkan proses interaksi karena proses ini hanya memakai waktu CPU yang sedikit. Demikian pula dengan proses yang menunggu terlalu lama. Proses ini akan dinaikkan tingkatannya.

Ada tiga jenis kategori yang digunakan dalam *multilevel feedback queue* ini, yaitu *High Priority, Normal Priority* dan *Low Priority.* Suatu antrian masuk ke dalam prioritas *high priority* apabila memerlukan proses yang banyak dan penggunaan CPU yang tinggi , untuk antrian yang memerlukan proses pengerjaan dan CPU yang normal maka termasuk *normal priority* dan *low priority* adalah yang memerlukan waktu dan penggunaan CPU yang sedikit.

Pada *Multilevel feedback queue* ada beberapa aturan penting yaitu:

- Jika Prioritas (A) > Prioritas (B), Maka A dieksekusi (B tidak).
- Jika Prioritas (A) = Prioritas (B), A & B dijalankan pada algoritma FSFC (*First Come, First Serve*).
- Jika pekerjaan masuk kedalam sistem, maka pekerjaan tersebut masuk ke dalam kategori *highest priority (the topmost queue)*.
- Jika pekerjaan memerlukan waktu tambahan ketika berjalan, prioritasnya diturunkan (contoh, perpindahan satu antrian).
- Jika pekerjaan selesai sebelum waktunya , pekerjaan tetap pada level yang sama.

Ilustrasi *Multilevel feedback queue* ini dapat dilihat pada Gambar 2.3. Pada Gambar ada 2 pekerjaan yaitu (Pekerjaan A dan B) memiliki level prioritas paling tinggi, dan pekerjaan C dan D berada pada level prioritas rendah. Maka pekerjaan yang akan dikerjakan oleh CPU adalah Pekerjaan A → B → C → D.



Gambar 2.3 Contoh *Multilevel feedback queue*

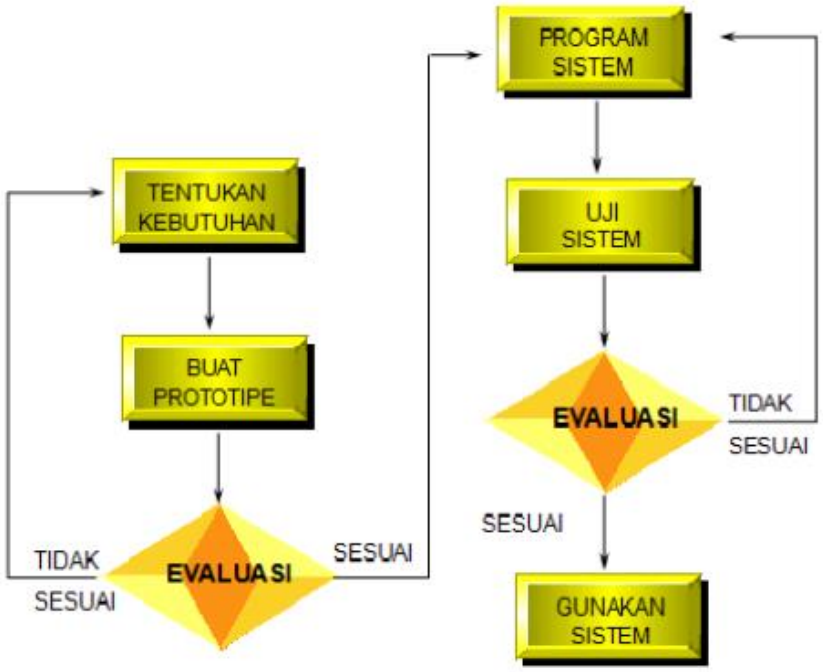
2.9 Metode *Prototyping*

Prototyping merupakan salah satu metode pengembangan perangkat lunak yang banyak digunakan. Dengan metode *prototyping* ini pengembang dan pelanggan dapat saling berinteraksi selama proses pembuatan sistem. Sering terjadi seorang pelanggan hanya mendefinisikan secara umum apa yang dikehendakinya tanpa menyebutkan secara detail *output* apa saja yang dibutuhkan, pemrosesan dan data-data apa saja yang dibutuhkan. Sebaliknya disisi pengembang kurang memperhatikan efesiensi algoritma, kemampuan sistem operasi dan *interface* yang menghubungkan manusia dan komputer. Untuk mengatasi ketidakserasian antara pelanggan dan pengembang, maka harus dibutuhkan kerjasama yang baik diantara keduanya sehingga pengembang akan mengetahui dengan benar apa yang diinginkan pelanggan dengan tidak mengesampingkan segi-segi teknis dan pelanggan akan mengetahui proses-proses dalam menyelesaikan sistem yang diinginkan. Dengan demikian akan menghasilkan sistem sesuai dengan jadwal waktu penyelesaian yang telah ditentukan (Mcleod, 1995).

Kunci agar model *Prototyping* ini berhasil dengan baik adalah dengan mendefinisikan aturan-aturan main pada saat awal, yaitu pelanggan dan pengembang harus setuju bahwa *Prototyping* dibangun untuk mendefinisikan kebutuhan. *Prototyping* akan dihilangkan sebagian atau seluruhnya dan perangkat lunak aktual direkayasa dengan kualitas dan implementasi yang sudah ditentukan.

2.10 Tahapan *Prototyping*

Tahapan-tahapan dalam *Prototyping* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Tahapan *Prototyping*

Sumber : McLeod Jr, 2008. *Sistem Informasi Manajemen*.

1. Pengumpulan kebutuhan

   Pelanggan dan pengembang bersama-sama mendefinisikan format seluruh perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.

2. Membangun *prototyping*

   Membangun *prototyping* dengan membuat perancangan sementara yang berfokus pada penyajian kepada pelanggan (misalnya dengan membuat *input* dan format *output*).

3. Evaluasi *protoptyping*

   Evaluasi ini dilakukan oleh pelanggan apakah *prototyping* yang sudah dibangun sudah sesuai dengan keinginan pelanggan. Jika sudah sesuai maka langkah 4 akan diambil. Jika tidak *prototyping* direvisi dengan mengulangi langkah 1, 2 , dan 3.

4. Mengkodekan sistem

   Dalam tahap ini *prototyping* yang sudah di sepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.

5. Menguji sistem

   Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus dites dahulu sebelum digunakan.

6. Evaluasi Sistem

   Pelanggan mengevaluasi apakah sistem yang sudah jadi sudah sesuai dengan yang diharapkan. Jika ya, langkah 7 dilakukan; jika tidak, ulangi langkah 4 dan 5.

7. Menggunakan sistem

   Perangkat lunak yang telah diuji dan diterima pelanggan siap untuk digunakan.

2.11 *Blackbox Testing*

Metode ujicoba *blackbox* memfokuskan pada keperluan fungsional dari *software*. Karena itu ujicoba *blackbox* memungkinkan pengembang *software* untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program. Ujicoba *blackbox* bukan merupakan alternatif dari

ujicoba *whitebox*, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode *whitebox*. (Beizer, 1995)

Ujicoba *blackbox* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan performa
5. kesalahan inisialisasi dan terminasi

Tidak seperti metode *whitebox* yang dilaksanakan diawal proses, ujicoba *blackbox* diaplikasikan dibeberapa tahapan berikutnya. Karena ujicoba *blackbox* dengan sengaja mengabaikan struktur kontrol, sehingga perhatiannya difokuskan pada informasi *domain*. Ujicoba didesain untuk dapat menjawab pertanyaan – pertanyaan berikut :

1. Bagaimana validitas fungsionalnya diuji?
2. Jenis input seperti apa yang akan menghasilkan kasus uji yang baik ?
2. Apakah sistem secara khusus sensitif terhadap nilai input tertentu ?
3. Bagaimana batasan-batasan kelas data diisolasi?
4. Berapa rasio data dan jumlah data yang dapat ditoleransi oleh sistem?
5. Apa akibat yang akan timbul dari kombinasi spesifik data pada operasi sistem?

Dengan mengaplikasikan ujicoba *blackbox*, diharapkan dapat menghasilkan sekumpulan kasus uji yang memenuhi kriteria berikut :

1. Kasus uji yang berkurang, jika jumlahnya lebih dari 1, maka jumlah dari uji kasus tambahan harus didesain untuk mencapai ujicoba yang cukup beralasan.
2. Kasus uji yang memberitahukan sesuatu tentang keberadaan atau tidaknya suatu jenis kesalahan, daripada kesalahan yang terhubung hanya dengan suatu uji coba yang spesifik.

Perbandingan pengujian antara *blackbox* dan *whitebox* dapat dilihat pada Tabel 2.1.

Tabel 2.1 Perbandingan *blackbox testing* dan *whitebox testing*

Testing Type	Specification	General Scope	Opacity	Who generally does it?
Unit	Low-Level Design Actual Code Structure	Small unit of code no larger than a class	White Box	Programmer who wrote code
Integration	Low-Level Design High-Level Design	Multiple classes	White Box Black Box	Programmers who wrote code
Functional	High Level Design	Whole product	Black Box	Independent tester
System	Requirements Analysis	Whole product in representative environments	Black Box	Independent tester
Acceptance	Requirements Analysis	Whole product in customer's environment	Black Box	Customer
Beta	Ad hoc	Whole product in customer's environment	Black box	Customer
Regression	Changed Documentation High-Level Design	Any of the above	Black Box White Box	Programmer(s) or independent testers

Sumber: Beizer, 1995. *Black Box Testing*